Materiality Matters in Computing Education: A Duoethnography of Two Digital Logic Educators

Eric J. Mayhew[*]
McGill University
Montréal, Québec, Canada
eric.mayhew@mail.mcgill.ca

Elizabeth Patitsas[*]
McGill University
Montréal, Québec, Canada
elizabeth.patitsas@mcgill.ca

ABSTRACT

Computer science needs to be sustainable, and CS educators have an important role to play in changing the discipline. Recent efforts have emerged to teach CS in ways that apply computing to mitigate climate change, but this alone is insufficient: we must also change what it means to *do* computing. We use duoethnography to interrogate our practices as CS educators to support the goal of integrating sustainability into CS. Despite being invested in these goals, we each realized that we had been nevertheless reinforcing the cultural norms that underpin the environmental and social damage caused by computing. We found five themes in the qualitative analysis of our reflections: (1) A lack of materiality in CS classes makes it difficult for computer scientists to scrutinize the environmental costs of hardware, (2) The discourse on “greenness” in computing neglects the role of embodied emissions, (3) The lack of context in CS education teaches students to perceive the status quo as “natural", (4) Those who have bought into the dominant ideology of CS can be resistant to innovating CS education, and (5) Materiality helps with teaching computing. We illustrate how changing CS to become more sustainable requires deeper thought than “add sustainability and stir” to the curriculum, and insights toward addressing the root ideology of CS education.

KEYWORDS

computer science education, computing education, sustainability, digital logic, climate crisis, e-Waste

Reference Format:
Eric J. Mayhew and Elizabeth Patitsas. 2021. Materiality Matters in Computing Education: A Duoethnography of Two Digital Logic Educators. In *LIMITS ’21: Workshop on Computing within Limits, June 14–15, 2021*.

[*]Both authors contributed equally to this research, and are listed in alphabetical order.

1 INTRODUCTION

Computer science (CS) has an important role to play in mitigating the climate crisis: CS can be applied to issues of climate modelling, and in providing tools for political organizing, citizen science, resource optimization, and changing consumer behaviour. But CS also has an environmental footprint that must be reckoned with. This includes ever growing e-waste [70], operational energy costs (e.g. [3]), and environmental damage from resource extraction [9].

Yet sustainability and the climate crisis are seldom talked about in computing classrooms. Many of us are working to incorporate sustainability into CS education, such as through supporting teachers [18, 50] and students in integrating sustainability [35, 37, 77, 78, 84]. These efforts are complicated by the contested definitions of sustainability, such as the environmental, social, and economic framings of sustainability [19, 53]. Regardless of the definition of sustainability, many computing educators claim they had not considered incorporating sustainability in their teaching, and others don’t see the connection between sustainability and CS concepts like algorithms [51].

The LIMITS community, which is concerned with the ecological and social limits of computing, has long recognized the importance of education as a leverage point for improving the sustainability of CS [42], with one paper going so far as to state that “Educational approaches will often be the easiest to implement” [15].

We illustrate how changing CS education is hardly easy; it is not just a matter of persuading more educators to add sustainability to their curricula. Through our duoethnography we demonstrate how, even for educators who approach CS teaching with the goals of instilling ethical practice, there are many ways that the disciplinary norms of CS can undermine these intentions.

In particular, we discuss how the immateriality of CS contributes to a status quo that underestimates and separates CS’s responsibility in climate change. Both of us have extensive experience teaching digital logic and were both proud of the innovations we had each made to teaching it — only to realize upon learning about the environmental costs of integrated circuit (IC) production that neither of us had addressed this when teaching with ICs.

Through our reflections, we contribute to the LIMITS community some insights as education researchers into how to teach computing within limits [42], pitfalls that interested educators may encounter, and disciplinary norms of CS that must change to address the climate crisis.

To connect our experiences to documented sociological phenomena, we draw on literature from science and technology studies (STS), the interdisciplinary field of study that examines the reciprocal relationships between science, technology and society. Feminist STS has a rich literature looking at how science and technology are socially constructed as disciplines, including how education plays a vital role in establishing the boundaries [4, 43], epistemic cultures [20, 38], excluded/included demographics [68, 83] and cultural norms/values [5, 13, 54, 80] of different technoscientific disciplines.

2 METHODS

2.1 What is a duoethnography?

Norris [57] describes a duoethnography as "a collaborative research methodology in which two or more researchers of difference juxtapose their life histories to provide multiple understandings of the world". This centres the researchers as the research site, as the authors "interrogate and reinscribe previously held beliefs" [57]. We chose duoethnography as our method because it allowed us to speak from and about our personal experiences as CS educators.

Duoethnography (also known as co-autoethnography) was first popularized in education research (e.g. [59, 75, 81]), gender studies, and critical ethnic studies [33]; and has since grown in use in HCI research (for a review see [33]), allowing for generative views of sociotechnical systems and exposition of "the ways that contexts connect and even conflict" [41]. For examples, see [22, 23, 41, 88].

Norris [56] has outlined four precepts of this research genre:

(1) It is dialogic, where the narratives of the researchers are positioned in juxtaposition to each other. It intentionally disrupts the typical nature of metanarratives that can emerge from solitary writing (including collaborative writing synthesized within a single authorial voice).
(2) It facilitates reconceptualization of past experiences and stories by allowing the other researcher to challenge the adequacy of the frames held in interpretation. It is not authoritative in stance, but rather positions the reader as an active Other in meaning making, and therefore as an implicit co-author.
(3) It positions differences as crucial to exploring a larger shared experience. Duoethnographies position differences between the writers' points of view as a strength and an opportunity to explore different meanings to a shared phenomenon.
(4) The methodology must remain open and flexible, to avoid becoming prescriptive. Duoethnographies do not need to adhere to a set procedure.

This process of narrating and reflecting on our experiences and beliefs are to encourage you, the reader, to recall your experiences and consider how the materiality of computing might connect to your own context. Indeed, this is a common goal of qualitative research, and is known as *reader generalization* (a.k.a. transferability) [71]. To support reader generalization, we provide *thick description*: rich, thorough descriptive information, to give the reader the "raw" experience that they too can interpret [71].

Reader generalization, along with statistical generalization and analytic generalization, comprise the three types of generalization in research methodology [71]. Statistical generalization refers to the practice of sampling from a population to discern information about the population as a whole; larger sample sizes and random sampling are associated with a better representation the population in question. Duoethnography does not attempt to provide statistically generalizable findings.

Analytic generalization *is* a major goal of duoethnography. Analytic generalization is when a researcher generalizes from the particular to broader theoretical constructs. In our paper, we will be linking our experiences to relevant sociological theories to demonstrate both how our experiences exemplify these sociological phenomena, and how these sociological phenomena manifest themselves in this particular context.

2.2 What was our process?

Our duoethnography was incited by reading Gabrys' *Silicon Elephants* chapter of *Digital Rubbish: A Natural History of Electronics* [32]. This chapter sheds light on all the social and physical "inputs" of ICs that are often omitted from common descriptions of IC input/outputs, such as the social and environmental costs of ICs. We read this as part of our lab's weekly reading group. For our reading group, members often mark up the same PDF with thoughts and reactions. We (the authors) noticed many similarities in the annotations. Specifically, many CS students in the group shared similar reactions about how they felt their CS education was very abstract and disconnected from society. During the discussion of the reading, we (the authors) spoke of how this paper made us reflect more critically on our teaching, often cringing at what we felt we did wrong. Considering how transformative this paper was for us, Elizabeth suggested turning our reflections as educators into a formal publication. Our intent in writing this paper was to critically analyze our experience teaching digital logic in a material way, to help support others to do the same, and to generate what we consider much-needed conversation around the environmental costs of computing.

We began writing this by meeting and doing several "streams of consciousness" writing sessions. These sessions involved writing for 20 minutes about our reactions to the *Silicon Elephants* chapter and how it made us think differently about our teaching. After each 20 minute writing block, we shared our writings with each other and exchanged ideas, reactions, and compared our experiences. This in turn generated a whole new set of ideas and questions to reflect on, which initiated our next writing session. We did this five times until we had several pages (about 10k words) of initial reflection.

We then each qualitatively analyzed our own reflective writings, looking for general themes and topics that emerged. For each theme or idea we spotted, we created a sticky note on a shared virtual whiteboard. We then began spatially organizing the sticky notes so that similar ideas were closer to each other (affinity mapping). From there, we saw a few general themes from our writings. Then we began writing the paper around these themes, and iteratively conducting additional stream-of-consciousness sessions as new themes emerged.

Our raw streams of consciousnesses naturally included a number of sociological concepts given ours background in STS and sociology. When we analyzed our reflections, we discussed how different papers and theoretical frameworks emerged. Feminist STS, particularly the works of Haraway and Breslin were the most common frameworks we invoked[1]. We also had references to neo-Marxist sociology and sociocultural education. In discussing the themes and concepts in our reflections, we realized that Breslin's work tied everything together parsimoniously, since Breslin explicitly drew on Haraway, Gramsci, and sociocultural education in her work.

2.3 Theoretical basis

Our work builds on anthropologist Samantha Breslin's *The Making of Computer Scientists* [17], an ethnographic analysis of how an undergraduate CS education builds a particular type of social identity which contributes to the problematic aspects of computing as a discipline. This happens through an education that frames CS knowledge as though it is objective and as something that is disconnected from the social world.

2.3.1 Feminist Epistemology and The God Trick. Using the work of Donna Haraway, Breslin shows how students are taught by authority that CS is objective, in contrast to many contemporary understandings of epistemology which assert that all knowledge produced by humans is situated in the context it was produced in and limited by the perspective of the knower [39]. Indeed, for feminist philosophies of science like Haraway, embracing the partiality of scientific knowledge is necessary for science to live up to its ideals of transparency and reproducability [39, 80]. Denying the partiality of science leads to distrust of science, especially by those who are actively harmed by its false claims of objectivity [61].

Haraway coined the notion of the "God Trick" to refer to the myth that the scientific process yields so-called "objective" truths about the world, creating knowledge that "see everything from nowhere". However, since all knowledge is situated in the context of its origin, "objectivity" only masks a particular standpoints (generally Western, male, abled, white, etc) as purportedly "universal truths".

Indeed, feminist approaches to scientific methodology stress situating one's self in one's research, and being continually reflexive about how one's standpoint affects the knowledge one produces [40]. Our act of writing in the first-person for this paper is in line with this feminist methodology, reinforcing how research is an activity carried out by humans.

2.3.2 Cultural Hegemony. Breslin also draws on Antonio Gramsci's notion of *cultural hegemony* to demonstrate how the enculturation of CS students leads them to buy in to dominant-group ideology. To Gramsci, cultural hegemony "is a social condition in which all aspects of social reality are dominated by or supportive of a single class" [29]; through culture, the dominating class can manipulate other groups to buy into the dominant group's beliefs and interests [34].

For example, "in capitalist society, journalists, the media in general, and professors create the sense that capitalism is efficient, egalitarian, natural, and so on [...]", making difficult or laughable to envision a world without it [34].

2.3.3 Rendering Technical. Finally, Breslin uses Tania Murray Li's work on hegemonic processes in global development [44]. Li defines "rendering natural" as a hegemonic process in which a dominant group's practices are seen as "natural" by other groups — such as how exploitation, capitalism, environmental destruction, extractivism, and neocolonialism are presented in the global development community as natural and unquestionable parts of human society and progress.

Li also defines "rendering technical", which is the hegemonic process in which development and environmental issues have their contexts stripped away and turned into technical "problems" in need of technical solutions. Technical solutionism acts a hegemonic norm in many environmental spaces where local context and political action are neglected in favour of shiny technological solutions [12, 46].

Breslin documents how a CS education teaches students to render the world technical: to strip away all context of "real world" problems, and see the world *only* in terms of abstract computational problems, ready for computational solutions. This is in contrast to context-appreciative design philosophies such as Soft Systems Methodology in which a modeller starts from fully immersing themself in the local context, abstracts it to make a model, and then de-abstracts it to meaningfully integrate it into its local situation [21].

Rendering technical serves to socially construct CS as something that is separate from the "real" world that it interacts with [17, 36]. The separation of CS from society and the environment is in turn a hegemonic norm in the discipline [5, 10, 10, 13, 24, 48]. Rendering technical also serves to assist capitalist notions of production [11, 45, 55]. Together these build the "I'm just an engineer" phenomenon [31, 60] that in turn leads to computer scientists viewing sustainability and social justice as outside their roles. And indeed, Mann et al. found through surveying CS students about their moral development that 79% percent of CS students agreed with at least one of these statements: "Computing is largely theoretical or technical - with little consequence", "There is no room in business for soft things like ethics, if your competitor does it then you can", and "Business is a special case; the ethics are different to personal life". [49]

2.4 Who are we?

2.4.1 Elizabeth. I am currently an assistant professor at McGill University, specializing in CS education. I bring to this duoethnography over a decade of experience with teaching CS at large, public, research-intensive universities (namely,

[1]For the purposes of this research, we will also work with an environmental interpretation of sustainability, as that is what emerged from our data.

the University of British Columbia, University of Toronto, and McGill University), along with what I have now come to realize is a relatively unusual experience of a CS undergraduate that exposed me to physical electronics, in part through a minor in physics. My background situates me well for appreciating the use of electronics in CS education.

I am a disabled, queer, white settler. Throughout my CS teaching career I've emphasized applications and social context, and taken pride in sharing my teaching materials with others. I have published multiple papers sharing experiences and research on my teaching of introductory programming (CS1) [35, 63], digital logic [62, 65–67], and algorithms & data structures [64, 69].

2.4.2 Eric. I am currently a master's student in CS education at McGill with Elizabeth as my supervisor. Like Elizabeth, I bring to this duoethnography my experience of being a CS educator and student dedicated to bringing social justice into my practice. Unlike Elizabeth, I learned CS mostly through informal settings like YouTube videos and my social network, and most of my CS teaching has also been in informal settings.

I identify as a gay, white, disabled man. I ran a student-led 24-lesson workshop on computing, with a large focus on digital logic with my boyfriend. I also have experience teaching from the several internships I did as requirements for my B.Ed. Because I am relatively new to CS, my perspective also illuminates the contemporary learning of CS.

2.5 A Word on Style

This paper will include coloured text boxes containing excerpts from our stream of consciousness writings (lightly edited for clarity), surrounded by analysis.

As discussed in subsection 2.1, this is to give you (the reader) a sense of the data we produced so that you can interpret for yourself if the data is transferable (*reader generalizability*) and to illustrate the richness of the text.

We begin this paper with our initial reactions to *Silicon Elephants*, and from there focuses on five main themes that emerged from our reflections.

3 BACKGROUND: SILICON ELEPHANTS

We begin with our reactions to reading *Silicon Elephants* [32]. This chapter illustrates how ICs are tied up in complicated social, political, environmental systems that are often forgotten when one considers the "input" and "outputs" of ICs. Indeed, ICs have significant costs to the environment and people who make them, yet this is rarely considered part of ICs' inputs and outputs.

Elizabeth:

It wasn't until I got to the second page that really my eyes started to widen [at the cold, gruesome details of cost and consumption in IC manufacturing]:

> "From design to manufacture, the typical microchip (as produced at Intel) requires more than 200 workers, two years, and considerable material and chemical inputs [...] generally speaking, the input of chemicals, gas, light, and other materials can require up to 300 phases to reach a complete chip. Many of these material inputs are not reflected in the end electronic product but are instead discarded as part of the hidden resource flows that contribute to electronics. In fact, microchips require far more resources than these miniature devices imply. To produce a two-gram memory microchip, 1.3 kilograms of fossil fuels and materials are required [...] as much as 99 percent of materials used discarded during the production process.
> [...]
> The clean rooms within fabrication facilities (or "fabs") where microchips are assembled are zones specifically designed to be free of dust, as even the smallest impurity may ruin the minute transistors. Workers ... don uniforms ... not so much to protect themselves from the chemicals but to protect the microchips from the dirt and debris that workers bring into the clean rooms. An uncanny inversion of waste occurs with microchip production, where clean rooms ensure the purity of electronics while simultaneously contributing to the contamination of workers' bodies, many of whom are low-paid immigrants and women of color."

Reading about the actual details of how microchips were made and their social and environmental costs was horrifying. I had known they were bad but it was still something to have a 26-page paper lay it all out there.

For my student Hana [a McGill undergrad in CS] it was all new to her, and she shared [via comments on the PDF]:

> "I don't know if other people had this, but this reminds me of learning about circuits in a way that was totally removed from physical circuits (I don't even know if they used the word circuit, I think it was just presented as logic) — they also showed images of physical representations like a door latch to explain latches but no images of ICs — just presented as abstract math and diagrams"

It got me to thinking about my own experience of learning about circuits. I actually *did* wind up doing labs with ICs thanks to CPSC 121 [digital logic] at UBC. Obviously at McGill it wasn't the same. And it did remind me of CPSC 313 [operating systems] at UBC where they talked about transistors but only showed them as logic diagrams; I don't even remember photos of physical transistors.

The discussion based on Hana's comment in reading group made me **realize that it was unusual that I even handled ICs at all in my CS undergrad**, that the entirety of a McGill undergrad in CS was like my experience of [operating systems].

Unfortunately, Eric, like Hana, had a very different introduction to CS hardware that did not involve learning with the actual hardware.

Eric:

I originally didn't know ICs were still a thing you could buy until my boyfriend told me! When running my workshop for fellow students not in CS, I really felt like my workshop was unique *because* it incorporated ICs. **Just the act of teaching hardware in a material way was such a departure from how I saw ICs being taught** in YouTube videos and textbooks which all used logic gate diagrams.

That begin said, having taught intro to hardware using ICs as a pedagogical tool, I thought that I would encounter this kind of teaching when I took my first hardware class at university. I was sorely mistaken...

Reactions to the paper also connected uncomfortable dots for both Eric and Elizabeth with respect to their teaching of hardware, teaching they were each proud of:

Elizabeth:

It got me to thinking about how I taught 121 and the scholarship I produced with my UBC collaborators about 121 [digital logic]. **I'd been so proud that we had realized the importance of giving students actual circuits to build, to enjoy actually playing with circuits and making the LEDs light up. But I realized with shame that I'd never connected the physicality of circuits with the many costs of manufacturing.**

Eric:

This paper was a real eye opener for me. My first thought was back to when I taught hardware to fellow students. I used ICs all throughout the lessons, and presented ICs with a sense of reverence and awe towards their power, compactness, and innovation but never talked about what they cost. I wanted others to understand that ICs were *good*:, efficient, powerful, revolutionary – and being so caught up with this goal I never considered how ICs could be harmful. **I thought I had done *such* a good job in my teaching, especially for using and playing with ICs as part of learning.** But this paper made me think more critically about my teaching and what kind of image of ICs I passed on to my peers, and of course, that's uncomfortable, especially since I was so proud of my teaching!

Reading *Silicon Elephants* brought out uncomfortable yet rich reactions from Elizabeth and Eric, both students and educators. Interrogating our reactions allowed us to better understand how educators dedicated to sustainable CS can be completely unaware of these painful realities.

4 THEME 1: HEGEMONIC IMMATERIALITY

Our first theme surrounds how the materiality of computing is obscured in CS education. By *material*, we mean what is made of matter; and by *materiality* we mean relatedness to material aspects (as opposed to conceptual/mathematical). This absence of materiality (*immateriality*) manifests itself in many ways through Eric's and Elizabeth's experiences teaching and learning CS, and actively resist efforts to incorporate materiality in CS education.

4.1 Rendering Technical and Externalization

In Breslin's ethnography, she contributes an analysis of hegemonic identity in CS. One hegemonic facet she documents is how a CS education teaches students to "render the world technical" [17]. Students are taught to take real-world problems and render them into purely technical problems. The sociopolitical contexts are stripped away and seen as separate and unrelated to CS. With time, students become so accustomed to this rendering that they see the whole world this way [17]. Both of us could relate:

Elizabeth:

I certainly [rendered technical] when I took [CS1] as an undergrad. I started seeing everything in terms of programming problems and I felt SO POWERFUL as a result.

One of the arguments I see (and have previously used) for widespread computing literacy (which often really just means coding) is that it is empowering. **But we don't really question where that sense of power comes from** and let's be real, it's a power that comes from treating basically everything else as an externality.

The idea of rendering technical was a common theme lying beneath the surface of both Eric's and Elizabeth's experiences learning and teaching CS, especially when it came to the materiality of CS. Rendering the world technical includes the process of stripping the environmental and social context in which computers are produced.

Both our reflections include pushing against (or being pushed into) a technical rendition of computing that marginalizes the material, environmental, and human cost of computers. Our reflections of being ignorant (and at times resistant) to considering the materiality of computing points to a larger hegemony in CS where immaterial conceptions of computing is the norm.

4.2 Rendering Magical: Coding as "Wizardry"

The sense that learning computing instills magic power shows up frequently in materials used to motivate people to learn about computing [85]. Indeed, Eric noted it in recollecting on how computers were magical to him until learning about digital logic:

Eric:

I remember before learning about computers, being so confused by the phrase "computers are just 1s and 0s". I knew what binary was, kind of, it's just a way to represent things using two digits. But when I look at computers and see video games, browsing the internet, chatting and video calling with friends, it didn't make sense to me how all this was "1s and 0s". In particular, I didn't understand how a computer can "remember" a 1 or 0. It's electricity, shouldn't it just dissipate?

It was only when I watched that a YouTube video that showed how to a simple circuit can store an electrical charge (a "1") did things really click for me. I remember that moment so well, when the way computers remembered made sense to me – **it was such an empowering feeling to turn something magical into something *I* could do**. This motivated me to find out the "trick" behind all the magic I thought computers did. This was certainly a catalyst that helped motivate me to learn more about CS.

While it is common for people to describe learning as empowering, we noted how the particular empowering feelings of learning are commonly described in terms of magic tricks and wizardry, unlike other forms of knowledge-based empowerment. To Eric, he had a sense that he was learning about things had been opaque to him in a society that obscures computing behind black boxes.

For Elizabeth, the discourse of "creating something out of nothing" differentiated the empowering feeling from computing knowledge from the empowering feelings she got from other previously-obscured domains:

Elizabeth:

There's this sense that computing is "creating something out of nothing" that I see tossed around all the time when talking about how computing is a magic power.

I used to believe that it was creating something out of nothing. It felt different from when I created solutions in the chem lab or built physical equipment for my physics labs. Sure, those made me feel smart and handy. But in computing, **since it seemed that I wasn't using any solid, visible material objects other than my computer to make a thing it /did/ seem like I was creating out of nothing.** So that powerful feeling felt magical, rather than handy.

But now I realize that this was predicated on a lie. It comes from how the actual inputs have been hidden – the machine built with conflict minerals, an electric grid that burns coal, the undersea internet cables that reflect British colonialism, and so on.

The sense of power that both Eric and Elizabeth describe when learning CS can be attributed to the way that the hegemonic immateriality of CS obscures and complicates CS's relationship to its impacts and origins in the world. The persistent *dis-origining* [52] of computing from its worldly context results in CS knowledge seeming out of this world.

In her Foucaultian analysis of code.org videos, Sara Vogel dissected the association between coding and magic:

> This discourse on personal agency, individualism, omniscience, and magic merges with neoliberal notions about competition, as crystallized in comments such as one uttered by Gabe Newell, the founder of the gaming company, Valve: "The programmers of tomorrow are the wizards of the future. You know you are going to look like you have magic powers compared to everybody else." [85]

Vogel's paper came to mind for Elizabeth in thinking about how learning computing is motivated:

Elizabeth:

Now thinking of how [colleague] shows the code.org propaganda [at the start of term in an introductory CS course] and Sara Vogel's takedown of it. There's so much about that propaganda that's bad that Vogel takes apart. BUT one thing Vogel doesn't get into is how it presents code as creating something *out of nothing*.

The idea that code is "magic" and the creation of something out nothing is a huge theme in intro coding propaganda and it sets up this narrative that computing is immaterial. Super convenient how it means you don't have to think about the ugly reality of how computing's materiality is destroying the planet and debilitating people in the Global South.

The "out of this world" framing of CS is highlighted in Amrute's ethnography of migrant IT workers in Germany. Amrute uses Saïd's notion of Orientalism in her work, which refers to how cultures from what Europe traditionally saw as "The Orient" (the Middle East, Northern Africa, South India, etc) have been othered in a way that treated these lands as exotic, magical, and barbaric [76]. Amrute uses this to discuss how South Asians are perceived by white Germans as better at IT work *because* of Orientalist notions that Indians are Otherworldly [6].

4.3 Rejecting the Partiality of Knowledge

A second sense of "magic" showed up in our reflections, in reference to how computing would reject the partiality of CS knowledge.

By stripping away the environmental and social contexts of computing, what is left behind can be mistaken for "magic" by those who do not know better. While any education will necessarily be incomplete, what we realized was problematic was how CS educators, instead of being open and honest about the partiality of CS knowledge, instead presented its limits as "magical".

For Elizabeth, it was telling that in CPSC 121 [Models of Computation], a course she took in first year undergrad and later went on to TA as lab coordinator, the breadboard kit used to teach circuitry used was literally called "The Magic Box". While it usefully blackboxed aspects of circuitry (voltage, wave generation, oscilloscopy, etc) that were outside the learning goals of the course, it normalized a notion that circuitry is "magic", rather than something that could be learned through physics or electrical engineering coursework.

4.4 The Obscuring Effects of Immateriality

Anusas and Ingold [7] point out in their discussion of modern, Western design practice that it is shaped by an "imperative to cast the material world in the guise of objects" [7]. The material world is not limited to solid objects, and includes *non-object* materiality such as electricity, fluids, gases, and other physical phenomena. In this framing of materiality, the touchable, enclosing surfaces of solid objects (like computer cases or plastered walls) hides "all manner of electrical, chemical, and mechanical workings; their parts, structures, and conduits; and the energies, gases, and fluids they carry" [7]. The result is a greater alienation from the natural world, as black boxing further obscures an individual's understanding and relationship to the inner workings of the objects around them [7]. CS goes a step further than this by casting the world into abstractions of objects.

For Elizabeth, abstractions were more difficult to learn without the underlying physics. For example, learning about analogue-to-digital conversion and issues of power supply made it clear to her why digital logic uses a binary system. But in her reflection, she realized that her students just take it as magic that circuits would be this way.

Not only does the immateriality of CS result in an even greater disconnection from the natural world, it makes it harder for computer scientists to wrap their minds around non-object materiality. While non-object materiality was an easy sell for Elizabeth (physics background), it was alien to Eric:

Eric:

I was at first reluctant to this definition. Firstly, I thought, material must mean it's touchable, physical, really just an object – **what**

is material is an object that I can hold or touch. I found it hard to consider things that weren't physical objects to be material.

I initially explained to Elizabeth, "well people learn about electricity in the sense their computers use electricity, and binary is essentially electricity, so CS students are learning electricity." But she pointed out how CS education rarely talks more deeply about electricity beyond binary. And honestly, she was right – even I remember trying to program my breadboard with NO knowledge of resistors, capacitors, measuring electricity or anything. I know from experience just playing with the breadboard did not teach me about electrical engineering.

And as Eric reflected more, he came to realize his understanding of object-centric materiality came from the hegemonic ideas about what *is* CS and what *is not* CS:

Eric:

Upon reflection, I see **my reaction came from being afraid of losing a clear definition of CS**. If we include electricity in CS, and environmentalism, and social justice, and physics/chemistry (ex. as they relate to ICs), and this and that... what would be left that would make CS CS? Then it dawned on me - Why am I defending the status quo definition of CS? I realized I was just preferring the status quo over change using slippery slope arguments to back up my resistance.

The second motivation that made my resistance to considering electricity as material was coming from a scarcity mentality. Having taught informally and formally, I am unfortunately well aware of the scarcity mentality instilled in teachers, like the meme that says "math teachers on the first day of class: OK everyone we're already way behind schedule we have to move fast". And this issue is very real in my experience teaching and in my conversation with my teacher friends. Although it's true that teachers are scarce on resources like time, that has nothing to do with whether electricity is material or not.

The fear of changing what it means to do computer science can lead to closing down important discussions about what the discipline *should* be. There is a wealth of STS literature on how the boundaries of scientific disciplines are socially constructed [43]. The question of CS's boundaries have important implications for broadening participation in CS, as computing has shifted boundaries in the past to exclude subfields that become more female-typed [4, 68, 83]. The hegemonic idea that computing itself does not need to change in order to become more demographically diverse has been implicated in why CS has had so little success in increasing its diversity despite wide efforts [68, 79].

4.5 Instilled Reverence

Hardware, especially sophisticated devices and components like ICs and CPUs are awe-inspiring marvels of technological progress. Eric reflected on how he was presented ICs as something deserving of reverence and went on to propagate this view of hardware in his teaching:

Eric:

When I first learned about IC production through the YouTube videos that introduced me to computer science, I was honestly very confused. Makes sense, as the production of ICs are very complicated, with the many phases of production, all involving complex chemical concoctions and processes. Just the sheer complexity of the production, and how these little ICs could replace refrigerator sized computers really **instilled a sense of reverence towards ICs in me – this idea that ICs are complicated and hard to make, and for that reason, they are just *good*.**

[When I taught my workshop on computer hardware] I wanted people to understand that ICs were just great because that's how I had learned about them. I watched YouTube videos that used pretty animations and flashy graphics that really instilled a sense of awe in me – just look at how *complicated* and *advanced* these are, look at all the sophisticated chemical processes used to create these tiny yet power things! What is there to criticize?

ICs are certainly incredible in what they are able to do. However, this reverence towards ICs can result in elevating ICs to a position where these electronic components are untouchable by critique. This undermines and stalls efforts to understand the full environmental cost of computing.

5 THEME 2: GREENNESS AS IMMATERIAL

This next theme we extracted from our narratives revolved around how "greenness" in a computing context was socially constructed in a way that was immaterial. "Green" computing generally refers to reducing the *operational* footprint of computing — without considering the *embodied* costs.

Embodied cost is defined as the sum cost of all the greenhouse gases released from cradle to site of use, including material extraction, transportation, refinement, processing, and assembly, whereas operational cost is the emissions released through use of the product [1]. There are additional sources of emissions from computing, such as how computing is frequently applied to induce demand for consumer goods, prospecting fossil fuels [74], and from the processing of e-Waste. For brevity, we have focused on the carbon emission aspect of embodied cost, though there are other costs such as pollution, that *Silicon Elephants* discusses in detail.

5.1 Greenwashing Through Focusing on Operational Costs

A theme that emerged in our reflections is how greenness moves CS away from materiality by focusing primarily on the operation cost of computing, often excluding the embodied cost all together. Indeed, writing this paper was Eric's first introduction to the very term "embedded cost":

Eric:

Sadly, only through sharing our experiences and reflecting on them with Elizabeth while writing this paper did I even learn what embodied cost was. I'm a bit surprised and disappointed to know that **despite my consistent effort to think critically about computing, I didn't encounter this really important part of "greenness" in computing**.

Eric's experience points clearly to how "greenness" in computing very frequently leaves out an important source of carbon emissions in computing production: the embodied cost. And while Elizabeth learned about embodied costs in graduate school through having a PhD advisor who specializes in climate change and computing, even she found herself falling prey to greenwashing in computing:

Elizabeth:

Last time I was searching for a web hosting company **I totally fell for the "green energy" thing and went with a company that promised 100% green energy.** So much of how greenness in computing is framed is around energy consumption, without considering embedded emissions.

Although it is easy to focus on just the operational cost of computing, Eric's reflections in researching the embodied cost of computers reveals how much greater the embodied cost of electronics can be compared to their operational cost:

Eric:

Upon learning about the embodied cost of computing, I began to notice how little the embodied cost of computing is discussed. I recently read an article in the Guardian analyzing and criticizing the environmental cost of bitcoin [3]. What struck me about this article was it doesn't mention at all the embodied cost of bitcoin mining in their analysis of bitcoin's absurd use of energy. When I read this, I couldn't help but feel like their assessment of the cost of bitcoin was so lacking by not mentioning the embodied cost. I also felt a sense of pride and empowerment in what I knew but also sad this is barely talked about.

To assess exactly how important the embodied cost of computing is when considering the "greenness" of computing, I did some research comparing the embodied and operational cost of devices. This lead me to a Microsoft Developers' blog post (of all places) that succinctly and powerfully lays out just how much greater embodied costs are compared to operational costs. The blog post **I read claims that 85 - 95% of emissions from devices like smart phones come from the embodied cost of producing these devices in their first year**. The blog post goes on to say that you'd have to use a surface laptop for about 12 years to match the embodied costs to the operation cost [2].

This reinforces the idea that we cannot rely solely on reducing the use and energy consumption of computers to address sustainability in CS. To move CS towards sustainability, questions must be ask about the unsustainable cost of producing new devices.

5.2 Applying CS vs. Scrutinizing CS

The way greenness in computing is constructed also misguides individuals concerned with climate change to focus on how CS can be applied to move CS towards sustainability, instead of scrutinizing the unsustainable practices of CS and computer manufacturing.

Elizabeth:

Now I'm thinking about teaching [CS1 at McGill] last year. I wanted to bring climate issues into the class. I created a [programming] project on carbon footprints even though I knew they were problematic. I even knew at the time that BP [British Petroleum] had come up with them as a way of shifting responsibility [of fossil fuel emission from BP to individuals]. But a personal carbon footprint calculator was just so much simpler to do compared to a country/corp's calculator and much simpler than a climate model. And it was also personal. Wanted assignments that connected to their lives. I had them put in their own values and get results. Liked that it would be related to lives. But did I do wrong?

Elizabeth, like many CS educators concerned with climate change, found herself teaching how to *apply* computing concepts toward sustainability - rather than teaching how to scrutinize the existing approaches to computing.

Indeed, the literature on CS education for sustainability predominantly focuses on how to apply computing concepts to climate modelling and sustainable logistics [30, 35, 72]. This tendency has also been an issue for the LIMITS community [42]. As Elizabeth notes in her reflection, this advantage is *easier* for a number of reasons: it *fits* with the existing paradigm, it directly addresses curricular goals, and can motivate students.

But as Elizabeth grappled working within the existing paradigm, she found herself reinforcing its hegemonic norms: an individualist (rather than structural) theory of change, a techno-optimistic mindset, and a lack of challenging the system of perpetual consumption that is at the root of the climate crisis [8, 28, 47, 87].

Though Elizabeth did not note it in her reflection, one of the sources of carbon that students calculated in the personal footprint calculator *was* the embodied emissions of all the new devices they had bought in a given year. It was her way of "sneaking in" a way to teach embodied emissions.

The disconnect between sustainability and scrutinizing CS shows up from the other side in Eric's teaching. He intentionally taught students to scrutinize CS with regard to social justice — but until reflecting for this paper, he had not included sustainability in that scrutiny:

Eric:

Something I hadn't considered until talking to Elizabeth about my teaching was how I often left out teaching about the environment as it related to CS. Interestingly though, I know I'm sure to include social justice, inequality, and oppression in my teaching.

In particular, I think back to when I was teaching machine learning (ML) to other students as part of my workshops. During the workshop, we talked about the ways machine learning is used in all sorts of awful and oppressive ways, but I never talked about how ML requires a ton of energy to train models, and more importantly, how training ML models requires a lot of computer hardware like fancy GPUs, which have significant embodied costs. So why didn't I include the environment?

At the time as a new learner of CS and sociology of CS, I saw *a lot* of emphasis on how ML can reproduce and obscure discrimination. But very rarely did I hear or read about the environmental cost of computing. In the few times I did mention the environment when teaching ML, it was often about how ML can help address climate change. In particular, I mentioned during a lesson how advances in computer hardware improves climate modelling simulations, allowing simulations to account for more intense and spontaneous natural disasters worsened by climate change. I know find it telling that I **would talk about how ML can help address climate change**

while ironically not mentioning how it may be exacerbating the crisis.

I see now this idea that discrimination is "more related" to ML than the environment just goes to show how deep I was in the hegemony of "greenness" in computing. The only reason I could entertain or believe arguments that stated ML was "not very related" to the environment are all the levels of abstraction and obfuscation that go into the environmental costs of computing. I don't have to see the landfills piling up with ewaste, or the waste of producing hardware – in fact living where I do, I don't even have to see the wires that transport electricity in and outside my home, or the dams that reshaped Quebec's northern ecosystems and climate patterns to provide me with electricity. Even the literature of the sociology of CS I know, I rarely read about the environmental cost of computing.

For Elizabeth it was interesting to see Eric saying that the sociology of CS rarely talks about the environmental cost of computing, because in her experience of the STS world this is actually a prominent issue (e.g. [38, 45]).

5.3 A Culture of New Hardware All The Time

The immateriality of CS also manifests itself in the values passed on to students in hardware courses. Elizabeth recalls her hardware coursework focusing on valuing fast and fault-tolerant hardware, rather than valuing long-lived hardware:

Elizabeth:

I'm now thinking about grad school, the course I took on storage systems. [The prof]'s talking about how the bottleneck in speed now is in memory access/reading, not processing. All our readings are about making storage faster, more reliable. New ways of managing storage. New hardware. New new new new.

At some point I ask what can be done to retrofit old systems so that we don't have to be always buying new hardware. People in class seem awkward. Somebody was like oh you mean like JBOD ["Just a bunch of Disks"] architecture? I had never heard the term before. We were most of of the way through [the term] and **at no point in the formal curriculum did we ever learn about JBOD or other techniques for making old machines still useful.**

I will say this for the course - we did talk a ton about reliability. Wasn't just speed we cared about. My term project involved doing requirements elicitation of sysadmins in large scientific labs (NASA Goddard, CERN LHC) about what they wanted in storage systems and what they were actually doing with existing storage systems. I don't remember it being surprising when in my interviews it arose they used actual *tape* to store things long term, which makes me think that [the prof] did spend time talking about how to keep information well-stored to avoid data rot and the like.

Even in this context where designers were taking a long-view approach to hardware design — so that data would be reliably accessed for decades in the future — the prevailing ideology remained that we should always be producing *more* hardware and continually switching to the newest hardware.

Indeed, Moore's Law was frequently discussed as something that storage systems researchers had a *duty* to uphold. And in Elizabeth's concurrent HCI coursework, she was presented with techno-optimistic views of a ubiquitous computing future, where having ICs in everything was *The* Utopian Dream.

6 THEME 3: THE GOD TRICK

The process of removing knowledge from the context it is born from is what feminist epistemologist Donna Haraway calls "the God Trick" [39]. She describes the God Trick as the "myth" of "seeing everything from nowhere", pushing against the idea that any one person can hold "objective" knowledge. Instead, she describes how all knowledge is situated and limited by the people and context any knowledge originates from, and we are better off embracing and acknowledging the limitations of our knowledge rather than tricking oneself into believing that they have a god view. Eric and Elizabeth's reflections revealed many instances of the God Trick in their experience learning and teaching CS.

6.1 Rendering Natural: Appropriating Scientific Language

Elizabeth's reflections connect Haraway's "God Trick" to the use of "Laws" in CS when reading *Silicon Elephants*.

Elizabeth:

From *Silicon Elephants*:

> the development of microchips not only depends on chemical compounds to ensure the accurate conductivity of silicon; it further depends on chemical compounds in order to increase conductivity. The terms of constant innovation and doubling of circuit capacity, which are captured by Moore's Law, have a chemical foundation. (Detailed description of the chemical processes used to refine silicon and the growth of the industry to create it...)

This got me to thinking about Moore's Law. Tedre paper talking about how CS dresses up observations as "laws" that don't actually merit it. Moore's Law rendered natural. **Wondering if CS is just so alienated from physical world that we invent natural sounding things to try to make up for it.**

The Tedre and Sutinen [82] paper mentioned above discusses the philosophy of computing, critiquing how CS has framed itself as a "science" without using the appropriate epistemic practices [82]. The paper contains a detailed discussion of how CS researchers have used the term "law" inconsistently and never in line with the way it is used by natural scientists, instead rounding up rules of thumb as "laws" [82]. This is an example of the God Trick as speculative claims about the nature and future of ICs uses scientific language (like "laws") to elevate these claims from speculation to scientific "fact", disconnected the claim from the situated and limited reality from it originates. Perversely, by framing laws like Moore's law as natural, computer scientists may feel compelled to make these laws true – regardless of the material or environmental implications (which are already missing from the purview of computing).

Similarly, Abbate has documented how computing dressed itself up as "computer science" to become eligible for NSF funding in the US [54]. Appropriating scientific language was part of the act [54, 82]. Yet over time it seems CS has forgotten this was an act, rendering these "laws" as natural.

6.2 Lack of History

CS's disconnection from its history was a common theme in both of our experiences learning CS and provides another example how CS knowledge is often removed from the situated context from which it was created. This lack of historical grounding serves to present computing concepts as a given and objective by disconnecting these concepts from their original context:

Elizabeth:

Talking to Eric about the Noyce history stuff [the physicist who designed and popularized the *intentionally* unrecyclable silicon IC], he knew so much more about it and it got me thinking about one of my other pet peeves in computing education and that's how we present it without any history.

Like you learn physics or math or anthropology or sociology and there's a huge focus on what was done in the past and how things have changed with time and how it's a human enterprise and who the big names are etc.

In computing we don't really do that at all. **We just serve the state of knowledge on a platter to students as though it was handed to us by God.**

This serves to reinforce the rendering natural of computing - instead of learning about the people behind it we learn only the concepts as though that's the way the world must be. Don't get to see how it's all a social construction.

For Eric, it was through his STS education, not his CS education, that he learned about CS history:

Eric:

It was actually through McPherson's article *US Operating Systems at Mid-Century* that I was presented a comprehensive review of UNIX philosophy! Prior to this reading I could only rattle off a few catchy "tenets" like KISS (keep it simple, stupid). Kind of funny that one of the most pivotal and important coding philosophies in CS I learned reading STS literature, not through my courses. **That [STS] paper helped me see that these "principles" of coding common today were just a choice by some hippies in the 70s, not some God-given set of commandments and rules.**

Not only does this obscure how computing is a social institution, this also makes it more difficult for students to scrutinize its social processes. Here, the God Trick is employed to render computing natural, by removing computing from the societal context it is situated in.

When Elizabeth broke with the mold to discuss CS's history in her teaching, she found that students found it enlightening and motivating — but she found herself feeling uncomfortable when touching on CS's ugly past:

Elizabeth:

[While teaching CS1 at McGill,] I talked about how climate models were one of the first main applications of civilian computing, but also something the military cared about and had the students brainstorm reasons climate models and weather forecasting are so important. Students did great on this (e.g. planting crops, transportation planning, emergency planning etc). I think this was the only time that term I ever wound up touching on computing's military past though. I made it pretty clear to students that was a huge thing in its history. But I felt uncomfortable while doing this. I'm supposed to be trying to motivate these students to learn CS, and help them feel at ease in a 800-student class, and here I am getting onto topics of war and genocide? **Maybe this is why we hide the history from our students when we teach computing.**

Avoiding discussing CS history also means educators can avoid difficult topics, such as its militaristic origins and its role in genocides (e.g. IBM's role in the holocaust [14]). But it also means that CS is presented as though it does not have history, nor that it is created by people.

7 THEME 4: COMPUTING IS PEOPLE

This theme, true to feminist epistemology [39, 80], discusses how social capital has so clearly shaped our experiences teaching and learning computing. We've made a point of keeping most names in our excerpted reflections to emphasize the social nature of learning and teaching. Indeed, this very paper was incited because one of Elizabeth's other students (Emma McKay) got us to read *Silicon Elephants*.

Every one of the experiences we reflected had people behind them who had taught and/or supported us in improving our teaching. Our raw reflections included over a dozen named individuals who supported us in our experiences. It was apparent to us when we coded our streams of consciousness that our experiences were strongly influenced by people in our lives. CS is not just pieces of knowledge: it is a community of people who create and share knowledge.

7.1 Resistance to Changing Computing

But not all the people we noted in our reflections were supportive. Consistent with other literature on how counter-hegemonic practices in CS receive social pushback [17, 79], we both recounted cases where we encountered resistance to making computing more material and playful.

Elizabeth:

In both of the digital logic courses I've been involved in teaching (CPSC 121 at UBC, CSC 258 at Toronto), TAs could be a major source of pushback for making changes.

For example, when in CPSC 121 when we started having students do more writing/communicating their knowledge, many **TAs pushed back because they felt marking students' writing was "too subjective"**. These TAs were so steeped in a culture of right/wrong based on input/output that grading little paragraphs of student reflections was outside their comfort zones.

Finding qualified TAs for these courses was a perennial issue. So many CS grad students just never had any experience with basic breadboarding — which in hindsight was a sign that my education at UBC was unusual!

One of the really important changes we made in 121 was for me (and my successors) to have weekly meetings with the TAs where the TAs actually did the labs. So many of our TAs were new to actually doing the material that it proved vital to have them learn it a week (or two) before they would be teaching it.

At both schools, we tended to get a lot of TAs from electrical and computer engineering as a result of not having enough qualified CS TAs. Many of them were great. But unfortunately some of them brought some of the worst parts of engineering culture with them - very RTFM [Read The F***ing Manual], very curt with students, heterosexist, rude to me and other women on the staff, etc. Those ones tended to really push back on efforts to make labs more open-ended, creative, and connected to societal applications.

And Eric recounted how the veneration of ICs led to push-back from his peers when he questioned their use:

Eric:

Having learned about the cost of ICs and being more critical of them can actually be kind of isolating. **When I bring up the big cost of IC production, I often am faced with very defensive responses from people in CS, like "well you use computers, don't you?"** or "what are we supposed to do now? It was only natural that ICs are this way", or "*How else*/can you think of a better way to have computers as they are now?". Other times people try to justify themselves to me with responses like "Well I *am* a vegetarian, and you eat meat so who's doing worse for the planet?". I get it this is a big pill to swallow but gee, don't shoot the messenger.

7.2 Student Support for Changing Computing

But while grad students – who have already been indoctrinated in the hegemonic norms in computing – pushed back on being explicit about computing's political dimensions, Elizabeth found in her teaching that first-year undergraduates really appreciated including it:

Elizabeth:

[During the fall 2020 term when I was teaching CS1, there was a international climate strike that fell on one of my teaching days.] I cancelled class the day of the actual climate strike. And the week *before* the actual climate strike I did a class on the Friday [in solidarity with Fridays for Future] about computing and the climate, to motivate why I was cancelling class one week hence.

Anyway, I talked about how computing has a huge environmental cost from both the embodied and operational perspectives. I wagged my finger at Bitcoin. And I also talked about how computing can help with climate crisis mitigation. Logistics optimization and that stuff. Power grid management. Weather disaster planning. And oh also we need to think about revamping all the cables and stuff for sea rise. Btw students the internet is all cables.

I was terrified students would be angry that I made computing "political" since this is a thing I've had pushback for in the past. But **the students were so into it. Many of them came by after class or in subsequent classes to thank me for even acknowledging the climate crisis** which is something that's clearly been weighing on so many young people's minds. In my teaching evals I was praised for showing the ways that computing affects society.

Elizabeth's experiences point to the importance of having people who can teach these topics at formative stages in CS education, before students have accepted the hegemonic norms. It also gives us hope: computing *can* change.

8 THEME 5: ADVANTAGES OF LEARNING WITH HARDWARE

Finally, an important theme that emerged in both of our reflections was that there were pedagogical benefits to learning with hardware: it was fun and hence motivating, but also cognitively made the abstract ideas tangible and more accessible.

8.1 Making Learning Concrete

Both of us wrote in our reflections how working with physical hardware helps students learn the abstract concepts in computing.

Eric:

What [Silicon Elephants] made me realize is that teaching hardware through a very material approach (e.g. using and playing with electronic components in the classroom) begins to address two problems I see in the dominant forms of CS education: **1) The manufacturing process of hardware is obscured to students and 2) CS concepts are often taught in very abstract ways**, making it hard for students (myself included) to grasp these concepts in the first place.

I remember having a very profound moment when I understood how basic logic gates could create a simple memory latch or loop. That blew me away, I understood the connection between the abstract logic gates and the computer I use all the time!

Similarly, when Elizabeth was a student, programming physical microprocessors was how she learned pointers:

Elizabeth:

[My friend who was two years ahead of me in the CS+physics programme] told me to take PHYS 319 [microprocessor lab course] when I took CPSC 313 [operating systems]. The two courses went together really well. CPSC 313 [operating systems] was totally abstract and we only talked about transistors in abstract terms. In PHYS 319 [microprocessors] we actually used transistors in circuits. I don't think I would have understood much in CPSC 313 without PHYS 319 actually making it real for me.

The prereq to CPSC 313 had been CPSC 213 [computer systems] which I felt I'd done poorly in. 213 was when we were introduced to C and Assembly languages and honestly I didn't understand pointers even by the end. I just put *s and &s randomly in the hopes that variables would work, and then brute force the combination when debugging. I lacked a real understanding of how memory was being managed.

Pointers didn't actually make sense to me until I was taught to program an HC12 microprocessor for PHYS 319. 319 was exciting. Not only were things from CS class ACTUALLY making sense for the first time, but I was actually using the knowledge in physical, tangible, cool ways.

Looking back on it the chance to implement things in a physical place was really important for my learning. CS classes just assumed that a mathematical description was sufficient for learning and boy that was insufficient.

That said, Elizabeth did actually get *some* hardware exposure through one of her undergraduate CS courses, but found the exposure insufficient to be very useful:

Elizabeth:

My intro to digital logic came as a first-year undergrad, in CPSC 121 [Models of Computation]. I had to buy a breadboard kit and use it in labs. The course was weird. We had abstract math logic in lecture, and then digital logic in the labs, and there wasn't much of a connection between the two when I took it. I was so excited at the start of term for the circuitry and was so disappointed by how it actually played out. The breadboard kit, called "The Magic Box", was only used in four of the (10? 12?) weekly labs. I wound up resenting purchasing it because it cost like a hundred dollars and we barely used it. When we did use it, it was boring, doing things like plugging in an AND gate and "checking" that it matched its truth table.

Elizabeth later went on to become the lab coordinator for the course — which involved rewriting the labs — and co-writing experience reports about efforts to rethink the course. In [66], one of her co-authors wrote (emphasis ours):

> Roughly four to six of the approximately nine labs in the term involve actual hardware, with students creating their own beginner circuitry using an in-house-created kit called "The Magic Box." Prior to the spring of 2009, students had to purchase and manage their own kits (at a cost of $80 CAD per pair of students). **The intention was that ownership would give students interest and opportunity to experiment with them.** Unfortunately, anecdotal and open ended survey feedback both suggested that students instead saw the boxes as a frustrating and unnecessary expense.

The idea that simply giving students a breadboard kit without providing instruction or scaffolding on how it could be used would magically foster student interest and engagement is something Elizabeth looks at now and names as constructionism run amok. For a description of constructionism and its hegemonic role in reinforcing individualist, masculinist approaches to CS education, see [5].

We include this as a cautionary tale, exemplifying the idea that "add hardware and stir" will not foster student learning on its own — the hardware needs to be explicitly taught and connected to the relevant computational concepts.

8.2 Hardware is just fun

A common theme and source of excitement for both of us was how fun playing with ICs can be (when scaffolded appropriately). Both of us talked often and fondly of getting LEDs to light up using simple circuitry and ICs. Something about the simple satisfaction of getting LEDs to light up with intention made learning about ICs and computer hardware very enjoyable for both of us.

Eric:

I remember being amazed while watching the YouTube crash course videos on circuitry and memory. Something about seeing how a circuit can be configured in such a way that stores memory was so fascinating and exciting to me. The videos used abstract symbols for logic gates to represent the memory latch, and when I learned that it was relatively cheap to get the hardware components that corresponded to these logic symbols, I really wanted to create the memory latch myself, before my very eyes.

I will be honest, it was *a lot* more difficult than the videos made it seem – I guess it's easier when you're using illustrations opposed to "programming" the real thing. It was hard using the resistors and transistors, and I had struggled with a lot of bugs. After successfully implementing the memory latch using transistors, I tried with ICs. This was also very challenging: the documentation for these ICs were hard to find and hard to parse, requiring error prone work like matching the pins of the ICs and to the diagrams. However, I wasn't deterred, indeed **that satisfaction of getting the LED to light up using transistors reminded me the work is worth it**. I remember finally constructing a memory latch and jumping with joy, excitedly showing off my messy breadboard of jumper cables and other discrete components to my friends.

And when we each started teaching digital logic, we both independently brought that emphasis on playful, material learning into our teaching.

Elizabeth:

In the early days of being lab coordinator for 121 [Models of Computation] I often thought of my work as "making the labs not suck". The original labs were very procedural: do X then do Y. And they were primarily on a digital logic simulator called tkgate rather than using the Magic Box [breadboarding kit]. One of my jobs as lab coordinator was to rewrite the labs (with input from the course instructor).

One of the early innovations came from Steve [Wolfman, the course instructor] to change an activity from "plug in a [physical] gate and record its truth table" to us covering the gate's label with white-out and then tasking students with identifying the "mystery chip". Steve set a precedent for me that I could bring more play and creativity into the labs. I started adding activities where students could create their own artistic creations with the LEDs and the clockwave generator, and making circuits that could *do* things, such as hardware-level encryption or error-correction. I had fun creating an activity where students would come up with their own hardware-level pseudorandom number generator.

Making the labs fun and creative led to the students engaging with the labs WAY MORE but also helped engage the TAs in the labs. A lot of the time the old labs could be pretty boring as a TA and it's hard to get students engaged when you yourself find it dull. Having the creative space and instructor support over several terms to create and refine new lab activities that were actually fun and practical was a really special time for me.

The successes of these new labs that Elizabeth worked on translated into two experience reports documenting just how much of an improvement the new labs were [66, 67]. And Eric also found that giving students a chance to play with circuits was also a big hit:

Eric:

In the [24 session] workshop I created with my boyfriend, we deliberately used ICs throughout the hardware module of our workshop. We had activities of translating abstract logic diagrams on breadboards. I remember while trying to refocus the workshop to turn to the next activity, I had a hard time getting participants' attention away from their breadboards. It seems like **they really enjoyed playing around with the circuits!** Participants often went above and beyond the subject material for the lesson and implemented more complicated circuits on their own, or stayed after our scheduled time to continue toying around.

The fun participants had using breadboards was also reflected in their feedback for the workshop. On the first ever iteration of our workshop, the feedback had several mentions of the breadboards and just how enjoyable they were to play with. One participant wrote in their feedback: "Everything was awesome seriously. I partiularly [sic] liked playing around with the breadboard to build circuits such that we can relate how computers work to physical tangible things." A different participant wrote: "The hands on computer hardware section, and how binary memory works [was my favourite part]". In the 11 response we got from our first iteration of the workshop, the majority explicitly mention enjoying using the breadboards.

Both of us noted the difficulty of working with hardware but that the satisfaction of getting it working made it feel "all worth it". Indeed, a great deal of research supports the idea that play helps learners learn [25, 26, 73, 86]. And while this is well-appreciated in elementary education, it is frequently neglected in university pedagogy. Tactility is an important modality for teaching [58], and neglecting it cuts off a useful medium for learning. Removing the tactile element also serves to disconnect the concepts from their material origins [52].

However, similar to the cautionary tale in subsection 8.1, it must be emphasized that just using ICs when learning digital logic is not enough to bring student's awareness to the environmental cost of ICs. Without proper guidance and instruction, the compact qualities of ICs may suggest to students that, given their size, ICs have a relative small environmental impact. Indeed, it was the very gory and uncomfortable truths presented in Silicon Elephants about the costs of ICs that inspired this very paper. Instructors looking to teach ICs in a way that includes this environmental cost must make teaching it a priority.

9 CONCLUSIONS

Coming to grips with the harrowing reality that is the waste and pollution in the production of ICs led us to see a larger pattern of obscuring the troubling relationship between computing and its environmental consequences. Our findings are consistent with similar observations in STS literature which illustrate how the dominant framing of CS sees computing as separate from the "social/political" world. In this framing, CS is purely "technical", conveniently labelling the environmental, social, and political consequences of computing as "not computer science" [10, 17, 24, 48, 49].

Through reflecting on our experiences we found common themes concerned with the immateriality of CS. We both experienced the ways in which computing's consequences in the world are positioned as separate from CS. Similarly, we both learned CS in a way that was removed from its historical context, distancing CS from its humanity. Our reflections also revealed how "greenness" is constructed in computing such that the other costs of computing are obscured. Finally, we share how teaching and learning with concrete, tactile components can help build richer student understanding. We found our experiences, though different, were in line with Breslin's ethnographic work of CS education, and together speak to how the social and environmental harms of CS stem from ideological roots in the discipline.

Having CS education promote values of sustainability and social justice means we must do more as educators than just add sustainability-themed assignments. If we do not address and challenge the hegemonic norms of computing that contribute to its environmental impacts, our students are destined to fail as they search for purely technical solutions to social problems like climate change. Fortunately, educators are key stakeholders in establishing the norms of computing. Given the right resources, professional development, and reflection, educators have the potential to shift the norms of CS to include sustainability and materiality at its core.

So how do we disrupt the hegemonic immateriality in CS? One way is to connect CS to its materiality. For example, bringing physical hardware into the digital logic classroom has many benefits for educators: ICs are fun and provide cognitive advantages when learning CS. Introducing ICs and other physical hardware into the classroom also presents opportunities to discuss and critique the material dimensions of computing. Seeing and touching ICs invites questions that examine the origin and creation of ICs, like where do these objects come from? What goes into making them? These questions segue well into larger (ideally student-led) inquiries about the very social and environmental dimensions of CS that are often lacking when considering the inputs and outputs of computational devices. These inquiries can work to directly challenge the pervasive attitudes that frames CS as magical and begin vital conversations about the partiality of CS. However, it should be noted this is *one* potential intervention that must be part of a larger cultural shift that needs to happen.

Even for educators who are dedicated to bringing environmentalism and social justice to our teaching, our paper demonstrates how there are many pitfalls in learning how to teach CS within the limits of what our ecosystems and social systems can support. This highlights the need for teacher education, education research, and community support for those who want to teach CS in counter-hegemonic ways.

ACKNOWLEDGEMENTS

We thank Aditya Bhargava, Christoph Becker, Alan Borning, Sam Breslin, Jay Chen, Steve Easterbrook, Steve Engels, Anna Ma, Emma McKay, Peter McMahan, Jacob Errington, Andrew Petersen, Barath Raghavan, David Rolnick, and Steve Wolfman for all of their feedback on this paper. We also thank everybody in the SHAPE of Computing Reading Group who participated in the discussion that sparked this paper. This research was funded in part by the Social Sciences and Humanities Research Council of Canada.

REFERENCES

[1] 2020. https://circularecology.com/embodied-carbon.html
[2] 2020. Examining the Carbon Footprint of Devices. https://devblogs.microsoft.com/sustainable-software/examining-the-carbon-footprint-of-devices/
[3] 2021. Electricity needed to mine bitcoin is more than used by 'entire countries'. https://www.theguardian.com/technology/2021/feb/27/bitcoin-mining-electricity-use-environmental-impact

[4] Janet Abbate. 2012. *Recoding gender: Women's changing participation in computing.* Mit Press.
[5] Morgan G Ames. 2018. Hackers, computers, and cooperation: A critical history of Logo and constructionist learning. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–19.
[6] Sareeta Amrute. 2016. *Encoding race, encoding class: Indian IT workers in Berlin.* Duke University Press.
[7] Mike Anusas and Tim Ingold. 2013. Designing environmental relations: From opacity to textility. *Design Issues* 29, 4 (2013), 58–69.
[8] Nicole Aschoff. 2015. *The new prophets of capital.* Verso Trade.
[9] Mehdi Azadi, Stephen A Northey, Saleem H Ali, and Mansour Edraki. 2020. Transparency on greenhouse gas emissions from mining to enable climate change mitigation. *Nature Geoscience* 13, 2 (2020), 100–104.
[10] David Adam Banks and Michael Lachney. 2017. Engineered violence: Confronting the neutrality problem and violence in engineering. *International Journal of Engineering, Social Justice, and Peace* 5 (2017), 1–12.
[11] Wolmet Barendregt, Christoph Becker, EunJeong Cheon, Andrew Clement, Pedro Reynolds-Cuéllar, Douglas Schuler, and Lucy Suchman. 2021. Defund Big Tech, Refund Community. *Tech Otherwise* (2021).
[12] Caleb Behn and Karen Bakker. 2019. Rendering Technical, Rendering Sacred: The Politics of Hydroelectric Development on British Columbia's Saaghii Naachii/Peace River. *Global Environmental Politics* 19, 3 (2019), 98–119.
[13] Ruha Benjamin. 2019. *Race after technology: Abolitionist tools for the new jim code.* John Wiley & Sons.
[14] Edwin Black. 2012. *IBM and the Holocaust: The Strategic Alliance Between Nazi Germany and America's Most Powerful Corporation-Expanded Edition.* Dialog press.
[15] Alan Borning, Batya Friedman, and Deric Gruen. 2018. What pushes back from considering materiality in IT?. In *Proceedings of the 2018 Workshop on Computing within Limits.* 1–6.
[16] Alan Borning, Batya Friedman, and Nick Logler. 2020. The'invisible'materiality of information technology. *Commun. ACM* 63, 6 (2020), 57–64.
[17] Samantha Breslin. 2018. *The making of computer scientists: rendering technical knowledge, gender, and entrepreneurialism in Singapore.* Ph.D. Dissertation. Memorial University of Newfoundland.
[18] Yu Cai. 2010. Integrating Sustainability into Undergraduate Computing Education. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (Milwaukee, Wisconsin, USA) *(SIGCSE '10).* Association for Computing Machinery, New York, NY, USA, 524–528. https://doi.org/10.1145/1734263.1734439
[19] AL Carew and CA Mitchell. 2008. Teaching sustainability as a contested concept: capitalizing on variation in engineering educators' conceptions of environmental, social and economic sustainability. *Journal of cleaner production* 16, 1 (2008), 105–115.
[20] Karin Knorr Cetina. 2009. *Epistemic cultures: How the sciences make knowledge.* Harvard University Press.
[21] Peter Checkland. 2000. Soft systems methodology: a thirty year retrospective. *Systems research and behavioral science* 17, S1 (2000), S11–S58.
[22] Marika Cifor and Patricia Garcia. 2019. Inscribing Gender: A Duoethnographic Examination of Gendered Values and Practices in Fitness Tracker Design. In *Proceedings of the 52nd Hawaii International Conference on System Sciences.*
[23] Marika Cifor and Patricia Garcia. 2020. Gendered by design: A duoethnographic study of personal fitness tracking systems. *ACM Transactions on Social Computing* 2, 4 (2020), 1–22.
[24] Tony Clear. 2004. Critical enquiry in computer science education. *Computer science education research* (2004), 101–125.
[25] Guy Cook. 2000. *Language play, language learning.* Oxford University Press.
[26] Kathleen Coolahan, John Fantuzzo, Julia Mendez, and Paul McDermott. 2000. Preschool peer interactions and readiness to learn: Relationships between classroom peer play and learning behaviors and conduct. *Journal of Educational Psychology* 92, 3 (2000), 458.
[27] Anu Corin et al. 2019. Inside out. Exploring invisible spaces of material flow through a critical design process. (2019).
[28] Eileen Crist. 2007. Beyond the climate crisis: a critique of climate change discourse. *Telos* 141, Winter (2007), 29–55.
[29] Greg Dimitriadis and George Kamberelis. 2006. *Theory for Education: Adapted from Theory for Religious Studies, by William E. Deal and Timothy K. Beal.* Routledge.
[30] Ali Erkan, Tom Pfaff, Jason Hamilton, and Michael Rogers. 2012. Sustainability themed problem solving in data structures and algorithms. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education.* 9–14.
[31] Casey Fiesler, Natalie Garrett, and Nathan Beard. 2020. What Do We Teach When We Teach Tech Ethics? A Syllabi Analysis. In *Symposium on Computer Science Education (SIGCSE'20).*
[32] Jennifer Gabrys. 2013. *Digital rubbish: A natural history of electronics.* University of Michigan Press.
[33] Patricia Garcia and Marika Cifor. 2019. Expanding our reflexive toolbox: Collaborative possibilities for examining socio-technical systems using duoethnography. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–23.
[34] Roberta Garner and Black Hawk Hancock. 2014. *Social theory: continuity and confrontation: A reader.* Vol. 1. University of Toronto Press.
[35] Michael Goldweber, John Barr, Tony Clear, Renzo Davoli, Samuel Mann, Elizabeth Patitsas, and Scott Portnoff. 2012. A framework for enhancing the social good in computing aducation: a values approach. In *Proceedings of the final reports on Innovation and Technology in Computer Science Education 2012 Working Groups* (Haifa, Israel) *(ITiCSE-WGR '12).* ACM, New York, NY, USA, 16–38. https://doi.org/10.1145/2426636.2426639
[36] Horaţiu Halmaghi. 2019. *Learning computer science was hard. Unlearning computer science is harder.* Master's thesis. McGill University.
[37] Margaret Hamilton. 2015. Learning and Teaching Computing Sustainability. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (Vilnius, Lithuania) *(ITiCSE '15).* Association for Computing Machinery, New York, NY, USA, 338. https://doi.org/10.1145/2729094.2754850
[38] Aimi Hamraie and Kelly Fritsch. 2019. Crip technoscience manifesto. *Catalyst: Feminism, Theory, Technoscience* 5, 1 (2019), 1–33.
[39] Donna Haraway. 1988. Situated knowledges: The science question in feminism and the privilege of partial perspective. *Feminist studies* 14, 3 (1988), 575–599.
[40] Sandra Harding. 2016. *Whose science? Whose knowledge?* Cornell University Press.
[41] Megan Hofmann, Devva Kasnitz, Jennifer Mankoff, and Cynthia L Bennett. 2020. Living Disability Theory: Reflections on Access, Research, and Design. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility.* 1–13.
[42] Michelle Kaczmarek, Saguna Shankar, Rodrigo dos Santos, Eric M. Meyers, and Lisa P. Nathan. 2020. Pushing LIMITS: Envisioning beyond the Artifact. In *Proceedings of the 7th International Conference on ICT for Sustainability* (Bristol, United Kingdom) *(ICT4S2020).* Association for Computing Machinery, New York, NY, USA, 255–266. https://doi.org/10.1145/3401335.3401367
[43] Michèle Lamont and Virág Molnár. 2002. The study of boundaries in the social sciences. *Annual review of sociology* 28, 1 (2002), 167–195.
[44] Tania Murray Li. 2007. *The will to improve: Governmentality, development, and the practice of politics.* duke university Press.
[45] Max Liboiron. 2021. *Pollution is colonialism.* Duke University Press.
[46] Heather Lovell and Donald MacKenzie. 2011. Accounting for carbon: the role of accounting professional organisations in governing climate change. *Antipode* 43, 3 (2011), 704–730.
[47] Fred Magdoff and John Bellamy Foster. 2010. What every environmentalist needs to know about capitalism. *Monthly Review* 61, 10 (2010), 1–30.
[48] James W Malazita and Korryn Resetar. 2019. Infrastructures of abstraction: how computer science education produces anti-political subjects. *Digital Creativity* (2019), 1–13.
[49] Samuel Mann, Mike Lopez, Dobrila Lopez, and Nell Smith. 2015. Educating for ICT4S: Unpacking sustainability and ethics of ICT student intakes. In *29th International Conference on Informatics for Environmental Protection and the 3rd International Conference ICT for Sustainability (EnviroInfo & ICT4S 2015).* Atlantis Press, 229–241.
[50] Samuel Mann, Logan Muller, Janet Davis, Claudia Roda, and Alison Young. 2010. Computing and Sustainability: Evaluating Resources for Educators. *SIGCSE Bull.* 41, 4 (Jan. 2010), 144–155.

https://doi.org/10.1145/1709424.1709459

[51] Samuel Mann, Lesley Smith, and Logan Muller. 2008. Computing Education for Sustainability. *SIGCSE Bull.* 40, 4 (Nov. 2008), 183–193. https://doi.org/10.1145/1473195.1473241

[52] Emma McKay. 2021. Dis-origined Materials: The Role and Invisibilization of Extraction in Technoscience. https://doi.org/10.31235/osf.io/8hxj7

[53] Phil McManus. 1996. Contested terrains: Politics, stories and discourses of sustainability. *Environmental politics* 5, 1 (1996), 48–73.

[54] Thomas J Misa. 2016. *Communities of Computing: Computer Science and society in the ACM.* Morgan & Claypool.

[55] Lisa P Nathan, Michelle Kaczmarek, Maggie Castor, Shannon Cheng, and Raquel Mann. 2017. Good for Whom? Unsettling Research Practice. In *Proceedings of the 8th International Conference on Communities and Technologies*. 290–297.

[56] Joe Norris. 2017. Duoethnography. In *The Sage encyclopedia of qualitative research methods*, Lisa M Given (Ed.). Sage publications.

[57] J. Norris, R.D. Sawyer, and D. Lund. 2016. *Duoethnography: Dialogic Methods for Social, Health, and Educational Research.* Taylor & Francis. https://books.google.ca/books?id=YawYDQAAQBAJ

[58] National Academies of Sciences Engineering and Medicine. 2018. *How people learn II: Learners, contexts, and cultures.* National Academies Press.

[59] Jennifer C Park and Sara ED Wilmes. 2019. A critical co/autoethnographic exploration of self: Becoming science education researchers in diverse cultural and linguistic landscapes. In *Critical voices in science education research.* Springer, 141–155.

[60] Shobita Parthasarathy and Jack Stilgoe. 2019. Episode 3: Considering Ethical Responsibility in Science and Technology ft. Nicholas Carr. The Received Wisdom.

[61] Shobita Parthasarathy and Jack Stilgoe. 2021. Equity in Science and Technology Policy and the Promise of Vaccines Ft.Maya Goldenberg.

[62] Elizabeth Patitsas. 2012. Teaching labs on pseudorandom number generation. In *Proceedings of the seventeenth ACM Annual Conference on Innovation and Technology in Computer Science Education* (Haifa, Israel) *(ITiCSE '12)*. ACM, New York, NY, USA, 376–376. https://doi.org/10.1145/2325296.2325392

[63] Elizabeth Patitsas. 2015. A Numpy-First Approach to Teaching CS1 to Natural Science Students. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education.* ACM, 333–333.

[64] Elizabeth Patitsas, Michelle Craig, and Steve Easterbrook. 2013. Comparing and contrasting different algorithms leads to increased student learning. In *Proceedings of the ninth annual international ACM conference on International computing education research* (San Diego, San California, USA) *(ICER '13)*. ACM, New York, NY, USA, 145–152. https://doi.org/10.1145/2493394.2493409

[65] Elizabeth Patitsas, Vanessa Kroeker, Rachel Jordan, and Kimberly Voll. 2012. Teaching CPU architecture: a new way to provide effective scaffolding. In *Proceedings of the twelfth Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '12)*. ACM, New York, NY, USA, 149–150. https://doi.org/10.1145/2401796.2401820

[66] Elizabeth Patitsas, Kimberly Voll, Mark Crowley, and Steven Wolfman. 2010. Circuits and logic in the lab: Toward a coherent picture of computation. In *Proceedings of the fifteenth Western Canadian Conference on Computing Education* (Kelowna, British Columbia, Canada) *(WCCCE '10)*. ACM, New York, NY, USA, Article 7, 5 pages. https://doi.org/10.1145/1806512.1806523

[67] Elizabeth Patitsas and Steven Wolfman. 2012. Effective closed labs in early CS courses: lessons from eight terms of action research. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (Raleigh, North Carolina, USA) *(SIGCSE '12)*. ACM, New York, NY, USA, 637–642. https://doi.org/10.1145/2157136.2157318

[68] Elizabeth Ann Patitsas. 2019. *Explaining Gendered Participation in Computer Science Education.* Ph.D. Dissertation.

[69] Elizabeth Patitsas [90%] and Daniel Levy. 2013. Dr. Horrible's Fork Bomb: A Lab for Teaching Security in CS2. In *Proceedings of the eighteenth ACM Annual Conference on Innovation and Technology in Computer Science Education* (Canterbury, UK) *(ITiCSE '13)*. ACM, New York, NY, USA.

[70] Devin N Perkins, Marie-Noel Brune Drisse, Tapiwa Nxele, and Peter D Sly. 2014. E-waste: a global hazard. *Annals of global health* 80, 4 (2014), 286–295.

[71] Denise F Polit and Cheryl Tatano Beck. 2010. Generalization in quantitative and qualitative research: Myths and strategies. *International journal of nursing studies* 47, 11 (2010), 1451–1458.

[72] Ian Pollock, Bedour Alshaigy, Andrew Bradley, Birgit R Krogstie, Viraj Kumar, Linda Ott, Anne-Kathrin Peters, Charles Riedesel, and Charles Wallace. 2019. 5 Degrees of Separation: Computer Science Education in the Age of the Anthropocene. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education.* 1–25.

[73] Ingrid Pramling Samuelsson and Eva Johansson. 2006. Play and learning—inseparable dimensions in preschool practice. *Early child development and care* 176, 1 (2006), 47–65.

[74] David Rolnick. 2021. Private Communication.

[75] Heath Rose and Anuchaya Montakantiwong. 2018. A tale of two teachers: A duoethnography of the realistic and idealistic successes and failures of teaching English as an international language. *RELC Journal* 49, 1 (2018), 88–101.

[76] Edward Said. 2020. *Orientalism.* Routledge.

[77] Jeffrey A. Stone. 2016. Integrating Sustainability Concepts into Introductory Programming Courses (Abstract Only). In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 685. https://doi.org/10.1145/2839509.2850527

[78] Jeffrey A. Stone. 2019. Assessing the Impact of Sustainability-Themed Programming Assignments. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 1283. https://doi.org/10.1145/3287324.3293764

[79] Susan Michele Sturman. 2009. *'Women in Computing' as Problematic: Gender, Ethics and Identity in University Computer Science Education.* Ph.D. Dissertation. University of Toronto.

[80] Lucy Suchman. 1993. Working relations of technology production and use. *Computer supported cooperative work* 2, 1-2 (1993), 21–39.

[81] Monica Taylor, Emily J Klein, and Linda Abrams. 2014. Tensions of reimagining our roles as teacher educators in a third space: Revisiting a co/autoethnography through a faculty lens. *Studying Teacher Education* 10, 1 (2014), 3–19.

[82] Matti Tedre and Erkki Sutinen. 2008. Three traditions of computing: What educators should know. *Computer Science Education* 18, 3 (2008), 153–170.

[83] K Tijdens et al. 1997. Gender segregation in the IT occupations. *Grundy, AF Women, Work, and Computerization. Berlin, Heidelberg, New York: Springer* (1997), 449–462.

[84] Giovanni Vincenti and Wolf T. Pecher. 2020. Merging Sustainability and Technology in the Classroom: An Experience Report. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 448–453. https://doi.org/10.1145/3328778.3366899

[85] Sara Vogel. 2019. Power, Discourse, and Knowledge in Computer Science Education Advocacy: An Analysis of Popular Code. org Videos. (2019).

[86] Lev S Vygotsky. 1967. Play and its role in the mental development of the child. *Soviet psychology* 5, 3 (1967), 6–18.

[87] Ronald Wright. 2004. *A short history of progress.* House of Anansi.

[88] Anon Ymous, Katta Spiel, Os Keyes, Rua M Williams, Judith Good, Eva Hornecker, and Cynthia L Bennett. 2020. " I am just terrified of my future"—Epistemic Violence in Disability Related Technology Research. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems.* 1–16.