Decoloniality, Digital-coloniality and Computer Programming Education

HANLI GEYSER, University of the Witwatersrand, Johannesburg, South Africa

Like digital technologies themselves, programming education is embedded in the colonial matrix of power, and access to programming knowledge demands immersion in the epistemologies of the Global North. While there is a growing body of work exploring ways to decolonise programming education, far more needs to be done. Current research focuses on the language of instruction and contextual curricula; outward-facing engagements with decolonisation. However, to move towards digital-decoloniality involves scrutinising how programming knowledge is recontextualised within curricula. Part of the project should be equipping both educators and students with the tools to recontextualise programming itself. To dismantle the colonial logic embedded in programming education, attention must be given to the knowledge formation of the discipline to identify moments of disruption. One such moment is the difficulty students face when recontextualising their mental models of computing, from programming skills to programming concepts. This occurs at the moment of reading, tracing and writing code. Programming requires one to refocus computational thinking and engage with a specific semiotic system, translating the authors' intention into an executable computational process. Disrupting this moment using the strategies of critical literacies opens computer programming and its resulting code to critical examination, allowing an inward-facing decolonial engagement with the discipline.

CCS Concepts: • **Social and professional topics** → **Computational thinking**; **Computing literacy**; **Computational science and engineering education**;

Additional Key Words and Phrases: Introductory programming education, decoloniality, digital-colonialism, digital literacy, critical literacy, language

ACM Reference format:
Hanli Geyser. 2024. Decoloniality, Digital-coloniality and Computer Programming Education. *ACM Trans. Comput. Educ.* 24, 4, Article 54 (December 2024), 30 pages.
https://doi.org/10.1145/3702332

1 Introduction

1.1 Aim

Software development is deeply embedded in the episteme of the Global North and exclusionary of minorities and the subaltern [1–7]. The need for decoloniality in software development in both industry and education is well established; however, there is still a need to extend research into possible approaches and practices [8]. Most decolonial studies engage at the level of implemented technologies; software observed in the wild rather than the interrelationships of the software creation process. Moreover, a critical examination of the connection between the implemented

Author's Contact Information: Hanli Geyser (corresponding author), University of the Witwatersrand, Johannesburg, South Africa; e-mail: hanli.geyser@wits.ac.za.

ACM 1946-6226/2024/12-ART54
https://doi.org/10.1145/3702332

technologies and the code underpinning them is situated in highly specialised disciplines inaccessible to most developers and students [9]. As the reach of the digital sphere extends, access to and participation in software creation is essential. International emphasis on STEM fields, including teaching people programming skills, has led to the proliferation of programming courses in higher education and sectoral training. While skills are desperately needed to enable access to the global knowledge economy, educators need to remain aware of how uncritical adoption of international technocratic mythologies and curricula embeds us in the systems of the technological centre.

The research has emerged through my teaching practice within the Department of Digital Arts at the University of the Witwatersrand in South Africa, where I teach both programming and digital art theory. We need to actively engage with decoloniality in our courses that introduce computer programming. Digital-coloniality is raised in the arts and theory-based courses but has not been sufficiently linked to programming as this is primarily taught as a functional skill. This lack leads students to see programming as 'neutral', existing outside of socio-cultural influence, and limits engagement with systemic digital-coloniality. As demonstrated later in this article, this is not a unique problem; it can be seen across programming curricula in various disciplines internationally.

A barrier to entry for computer programming educators wanting to approach decoloniality in their curricula is accessing the vast body of knowledge. To provide a starting point, this article aims to set out working parameters for a discussion of decoloniality in computer programming education. I identify intersections in the literature on decoloniality, digital-coloniality and programming education and propose an additional avenue for investigation: leveraging critical literacies pedagogy to allow moments of contestation in programming curricula and challenge the pervasive view of technology's axiological neutrality. I argue that opening computer programming to scrutiny in our curricula enables us to deconstruct the digital-colonial power structures embedded in it and foster a critical understanding of how these are reinforced in the code we write.

1.2 Structure

This article sets out each concern separately, briefly reviews the literature in each area and raises intersectional moments, leading to the argument for deploying critical code literacies to subvert the myth of neutrality perpetuating digital-colonialism. Section 2, Digital-(De)coloniality, establishes what we mean by decoloniality, how this extends into the digital realm, and identifies the colonial apparatus present in the smallest units of digital meaning-making: the myth of axiological neutrality in the code we write. Section 3, Decolonial Curricula, situates the curriculum as a construct within the rationality/modernity/coloniality matrices of power through its selection of valid knowledge. It considers the balancing act we face in our curricula between decolonial engagement and global relevance. It concludes by arguing for inward-facing decolonial engagement through reading the discipline contrapuntally. Section 4, Programming Education, offers a high-level overview of the paradigms and common approaches underpinning programming education. It identifies the difficulty of remapping conceptual frameworks when learning to program as a possible moment of disruption when students engage with the reading, tracing and writing of code. Section 5, Decolonial Approaches, sets out some common decolonial approaches to programming education. It breaks these down into two broad categories: contextualisation (including localisation, ethnocomputing and indigenous knowledge) and language (including the language of instruction, the language teaching approach in computer programming education and translanguaging). It argues for an additional approach focused on knowledge production in programming itself, in the moment of reading, tracing and writing code, to challenge the myth of neutrality. These strands are brought together in Section 6, Critical (Code) Literacies. The article examines the possibilities of leveraging critical computing, critical code studies and critical literacies to disrupt the myth of axiological

neutrality and open code to decolonial engagement. Section 7 concludes the article and offers avenues for further research.

1.3 Positionality

As a white South African woman and a lecturer at a research-based university, I am inextricably complicit in coloniality. Therefore, I position my work as concerned *with* decoloniality rather than claiming it as decolonising or decolonised. As an educator, this translates into a conscious focus on finding moments of disruption to critique colonial constructs and raise decoloniality in my teaching and curricula. I use hooks' framing of 'supporting' or 'moving towards' to access the labour of decoloniality without appropriation [10]. While the relationship between conscientisation and decoloniality is problematised, they exist in productive tension [10–16]. hooks articulates this when reflecting on the links between conscientisation and decolonisation: a starting point, not an end goal '…that historical moment when one begins to think critically about the self and identity in relation to one's political circumstance' [10, p. 47].

2 Digital-(De)coloniality

2.1 What Is Decoloniality?

Decoloniality relies on and revels in plurality; it is a state and a disposition, a philosophic action alive with contradictions [17–20]. Decoloniality is as diverse as the forms of coloniality itself. It is accessed in a myriad of different ways by different theorists, depending on their fields, concerns and circumstances. In their comprehensive literature review of decolonisation in curriculum and pedagogy, Shahjahan et al. identify that '…disciplinary reflexivity, student movements, and indigenous policy initiatives have created conflicting conceptualizations and practices of decolonization, signalling the importance of teasing out these various meanings' [8, pp. 74–75]. Still, all conceptions address the power relations embedded in the rationality/modernity/coloniality complex [21–24].

'Decolonisation' and 'decoloniality' are often used interchangeably; however, like colonialism and coloniality, there is a distinction. A vast oversimplification would be that one is more commonly associated with concerns of colonial domination, focusing on the territorial, political and cultural, and the other with redressing ideological and epistemological systems of power and subjugation [11, 17, 25]. While 'decolonisation' can be more directly linked to settler colonialism, 'decoloniality' refers to an epistemic delinking from the rationality/modernity/coloniality matrix [26]. Decoloniality is concerned with the intersection between coloniality of power, knowledge and being. As Quijano states, it requires interrogating ideological and epistemological matrices of power inherent in colonial systems 'to liberate the production of knowledge, reflection, and communication from the pitfalls of European rationality/modernity' [22, p. 177]. Mignolo traces the development of the term and states that by the early 2000s:

> De-coloniality became the common expression paired with the concept of coloniality and the extension of coloniality of power (economic and political) to coloniality of knowledge and of being (gender, sexuality, subjectivity and knowledge), [and was] incorporated into the basic vocabulary among members of the research project. [26, p. 457]

In their highly influential article, 'Decolonization is not a metaphor', Tuck and Yang call for resistance to the use of 'decolonisation' as a term to refer to critical, anti-colonial, postcolonial and social justice concerns and refer to its use in these contexts as a 'settler move to innocence', outlining ways in which the term has been appropriated and equivocated [11]. Their critique has been adopted in much of the literature on decolonising pedagogy. However, the review by

Shahjahan et al. indicates that rather than the term being appropriated and equivocated, it simply means different things at different times to different people. They note that:

> Overall, the literature on DCP [Decolonizing Curriculum and Pedagogy] suggested three meanings of decolonization: (a) recognizing constraints, (b) disrupting, and (c) making room for alternatives. All three meanings, while not exclusive, fed on each other. However, as we demonstrated, the third meaning and how it manifested varied across different regions. It is unsurprising that the regional variety in meanings is connected to varying relationships with geopolitical power centers. Geographical, disciplinary, economic, and/or political orientation to the metropole influenced decolonizing vantage. [1, p. 85]

This aligns with Tuck and Yang's argument, which insists on specificity and context. As Tuck and Yang make clear in their text and as elaborated on by Zembylas [12] and Garba and Sorentino [27], their argument addresses settler colonialism as the focal point. They emphasise this when they state that the question:

> 'What is colonization?' must be answered specifically, with attention to the colonial apparatus that is assembled to order the relationships between particular peoples, lands, the 'natural world', and 'civilization'. Colonialism is marked by its specializations… Decolonization likewise must be thought through in these particularities. [11, p. 21]

To further this discussion, it is therefore necessary to examine the specificities of the colonial apparatus in question. What is digital-colonialism, where is it situated and what are the particularities of decoloniality in this context?

2.2 What Is Digital-Coloniality?

Horvath defines colonialism as a 'form of domination—the control by individuals or groups over the territory and/or behaviour of other individuals or groups' [28, p. 46]. Colonialism takes many forms, interlinked in a complex web, built on and in each other. A distinguishing feature of all constructions of colonialism is that they are embedded in the mechanisms of the colonial apparatus, the colonialities of power, knowledge and being. Lüthi et al. broadly reconfigure colonialisms under classical colonies, settler colonies, colonies at the margins and colonialism without colonies [29]. Their categorisation aims to decentre perspectives of colonialism to include colonialities in the margins. However, they warn that 'like all such designations, these different terms … provide the abstract poles of a continuum rather than paradigms or precise descriptive categories' [29, p. 4]. One colonial system cannot exist outside of the others. They argue that approaching colonial formations in this way allows one to trace the 'structural continuities of a colonial matrix' despite their 'engagements and interdependencies' [29, p. 1]. Digital-colonialism emerges from the continuation of colonialism, through neo-colonialism, to techno-colonialism.

Neo-colonialism, coined by Sartre, emerged during the 1960s and was originally theorised by African scholar Kwame Nkrumah [30]. In 1961, the All-African Peoples' Conference released a *Resolution on Neo-Colonialism*, defining it as 'an indirect and subtle form of domination by political, economic, social, military or technical…' means [31, np]. Neo-colonialism continues the rationality/modernity/coloniality complex by extending the colonial myth of linear progress, linking access to 'modernity' to subjugation to economic and cultural imperialism, as well as capitalism, globalisation and colonial aid [22, 32–34]. Techno-colonialism indicates the ways in which technology is embedded in the rationality/modernity/coloniality complex through the logic of progress, and digital-colonialism extends this into the digital technology sphere. While techno-colonialism refers to the domination resulting from access and control over all technologies, for example, medical or industrial, the terms 'techno' and 'digital' are currently used almost

interchangeably in the literature to refer to digital and computational technologies. However, it does not exist only within that framing as part of a continuum; it also becomes a new form of colonialism in and of itself [35, 36].

Situating the colonial apparatus present in digital technologies requires thinking through the coloniser/colonised dichotomy. Where do we situate the colony in the ever-shifting digital sphere? A productive springboard is the colonial tensions between the Global South and the Global North. While the terms have been criticised for following an 'asymmetrical relation' of colonial dichotomies [37, p. 167], they offer a fluidity and instability that resonates with the reach of digital technologies. Using Global South and Global North follows the work of Grosfoguel [38], Comaroff and Comaroff [39], and Connell [35], but the terms are particularly useful as articulated by Kloß [40] and Amrute [36]. Kloß describes the Global South as active, a process and a practice that constantly reforms global networks of power. It is '…a liminal space of transition in which a phase of anti-structure enables the re-organization of, for example, social and epistemological power relations, and which creates a new model of social, economic, and political interactions that relies on egalitarian principles' [40, p. 8]. Amrute deploys South/North specifically in relation to techno-colonialism as being permeable and dispersed, they write 'we need to treat the South as both dispersed across the globe and as very particular sites that first developed new ways of coping, refusing, and revisioning these relationships' [36, p. 8].

With the reach of the digital, its permeable, unstable nature is self-replicating. Even as it spreads, it continues to reproduce the epistemologies of the technological centre, the Global North. It is so pervasive that it enforces ways of thinking and being across the globe, extending the rationality/modernity/coloniality complex and reproducing historical inequities. The digital realm, in its data and algorithms, shapes our social interactions and media consumption [41–43], our labour practices [44, 45], our language [46, 47], our histories and our geographies [48–51] and governs access to fundamental resources like healthcare and education [52]. Birhane outlines how corporate control over the algorithms that govern networked interaction and 'AI-driven solutions' constitutes an 'algorithmic invasion [that] simultaneously impoverishes development of local products while also leaving the continent dependent on Western software and infrastructure' [5, p. 389]. Couldry and Mejias examine the greed for data as a refashioning of colonialism, mirroring its 'function within the development of economies on a global scale, its normalization of resource appropriation, and its redefinition of social relations so that dispossession came to seem natural' [53, p. 5].

Kwet sets out a conceptual framework for techno-colonialism and traces how the USA recreates imperial structures of economic dominance and exclusion through control of the digital ecosystem [3]. Jandrić and Kuzmanić present a nuanced analysis of techno-colonialism in their article 'The Wretched of the Network Society: Techno-Education and Colonisation of the Digital', in which they argue for exploring digital technologies through the critical lens of postcolonialism [2]. In 'Tech Colonialism Today', Amrute outlines the (re)production of colonial relationships as they play out in digital technologies [36]. They discuss digital technologies as hierarchical, extractive, exploitative, resulting in uneven consequences, and displaying malevolent paternalism, giving clear examples of each instance. But Amrute takes this further and suggests that perhaps '[o]ur theory of colonialism needs to be amended to account for the complicated territory brought into being in the current moment' [36, p. 7].

Software development is overwhelmingly credited to the North, while labour from the South is exploited [44, 45]. Geographically, access to technologies is restricted by a lingering digital divide [48–51]. Attention has also increasingly been drawn to the discriminatory impacts of algorithmic technologies on marginalised communities [54, 55]. When combined, one can see how peoples from the South are constructed as being passive consumers of technologies rather than active agents in the technologies produced, and, as Smith points out,

> ...the production of these materials is heavily regulated by a technocracy—a cadre of scientists and hobbyists who have particular technological proclivities that make their entrance into the 'discourse of coding' possible while concurrently limiting the types of people who can join the discourse community... [9, p. 149]

Broussard frames this as techno-chauvinism, where the producers are a small and relatively homogenous group (overwhelmingly white men from the Global North), leaving the Global South as 'subjects' to the technologies they create [42]. This construct of being 'subject to' digital technology is not a new problem. Writing from India in 1989, Mohan critiques the ideologies of technology as they observed it playing out in local policy and discourse. A key concern they raise is the impact of the discourse on positioning 'development' in the 'third world'. They write, '... we end up measuring our own future prospects purely in terms of our ability to cope with these 'emerging' technologies and our perceptions of their role in the "Information Age"' [56, p. 1815]. Therefore, the drive for relevance in a global knowledge economy seems dependent on adoption and assimilation into a discourse of knowledge production that continually (re)enforces the paradigms of the North.

This implicates both digital knowledge production and its recontextualisation in education. It is a self-reinforcing cycle, as Birhane and Guest argue: 'The present [computational science education] ecosystem sustains itself by rewarding work that reinforces its conservative structure. Anything and anyone seen as challenging the status quo faces systemic rejection, resistance, and exclusion' [57, p. 61]. 'Worthwhile' or 'valid' knowledge in the digital space is dictated by digital-colonial power, and gaining access to it requires one to be immersed in ways of thinking and doing of the Global North. Kroeze argues that digital technologies 'often continue carrying forth the colonial values of the past in an unobtrusive way... In fact, western logic (esp. Wittgenstein's binary philosophical logic) lies at the core of the digital computer' [58, p. 42].

2.3 Code and the Myth of Neutrality

But as educators, where do we begin? Pinar reminds us, '[t]he first task of thought in our era is to think what technology is' [59, p. 3]. The technology sector is vast, with many different subsets, one of which is software development, the core skill set of which is computer programming. Programming can be extended further into computational thinking and the syntax of programming languages. 'Code' is the artefact of programming as an exercise: It is the textual, executable body of work that denotes computational thinking through the syntax of a programming language. I argue that deconstructing the colonial logic of progress embedded in digital technologies requires us to drill down to the smallest units of meaning in digital structures and, therefore, to open computer programming and its resultant code to scrutiny.

In industry and academia, critical and decolonial engagement with computer programming stalls when faced with the overwhelming perception that, in its algorithms and underlying code, it is a mathematical, universal 'truth' [5, 10, 41, 60–64]. This is a refrain heard in widespread opposition to the decolonial critique of many sciences [65–70]. Stemming from the rationality/coloniality/modernity matrix of power, this argument follows the colonial logic of progress.

This perception of programming is hard to shift, as it is embedded in the first introduction, where the focus is on the mastery of computational thinking, abstraction, logic and problem-solving [71–73]. It is taught as clear, neutral machine instructions, leaving the semiotic complexity and sociocultural entanglements overlooked [63, 64, 74, 75]. Smith states that a failure to interrogate the production of digital tools allows '...technology and colonialism continue to entrench/insert themselves within a milieu wherein technological spread and colonization are left unquestioned' [9, p. 159]. Therefore, moving towards decoloniality requires investigating how programming

knowledge, encoded into the code we write, is recontextualised into curricula and how that works to reinforce digital-colonial systems.

3 Decolonial Curricula

3.1 Situating the Curriculum

In an overview of definitions of the curriculum, Marsh notes the constant use of metaphors of movement [76]. They highlight metaphors of flight, rivers, streams, roads and highways. They also show how these are linked to maps, borders and terrain. Writing on Indigenous Education, Madden explores the metaphor of pedagogic pathways:

> Consider a 'hiking trail' formed by the relationships among communities of animals, trees, rocks, streams, and earth; trail markings; a specified distance and level of difficulty described on a website; and the promise of a spectacular view. Similarly, assumptions about education and teaching, associated purposes and goals, central themes, and pedagogical methods comprise a pedagogical pathway that shapes, but does not determine, the learning journey. Some elements of the pathway remain constant while others fluctuate, and the journey is continuously contextual, distinct, relational, and unforeseeable. [77, p. 2]

Building on these metaphors, making and living a curriculum is the attempt to establish movement through an entangled terrain of winding paths, obstacles and promised reveals. In all of these conceptualisations, the curriculum is a liminal space. Always between, always in tension, revealed through the intra-action between the who and the how, as much as the what and why. Curricula exist in the visible (formal knowledge) and the hidden (the values and epistemes embedded in the knowledge structures). Shahjahan et al. situate their discussion by conceiving of the 'curriculum (material content and purpose, which imply what counts as knowledge), [as] manifesting inside/outside of classrooms, such as in a course, program, discipline/profession, institution, and/or minoritized community setting' [8, p. 77].

Seeing the curriculum as existing in tension arises from the traditions of critical theory in pedagogy and curriculum studies [78–82]. These tensions are intensified and problematised in discussions of decolonising the curriculum [9, 12, 83–86]. These engagements address the relationships between knowledge, education and power in different ways, but some refrains arise: The creation and enactment of a curriculum is an act of power; it constructs a disciplinary knowledge that is complicit in silencing and erasure; slippages in the processes of knowledge production, recontextualisation and reproduction open discursive gaps as sites of contestation; teaching and learning is a relational, participatory, social practice. With these understandings, Boughey and McKenna suggest that curriculum research is approached through three guiding questions:

(1) What knowledge is legitimated by the curriculum?
(2) Which knowers are legitimated by the curriculum?
(3) How are these knowledges and knowers legitimated in the curriculum? [87, p. 83].

In education, the selection of 'valid' knowledge represented in curricula is repressive to subaltern knowledges. Education constructs a dichotomy between those 'inside' and those 'outside', the 'knowers' and the 'not knowers' [25]. This hierarchical relationship extends from the selection of content to pedagogic modes and assessment methodologies, and the institutions of higher education are complicit in maintaining the *status quo*.

No curriculum is neutral; it is inherently ideological as it governs access to knowledge and knowing. Bernstein's seminal work on the pedagogic device engages with ways of knowing inherent in the curriculum [78, 88]. Bernstein traces the translation of communication from the

production of discourse, its recontextualisation within the curriculum and reproduction in teaching, learning and assessment. He explores the ways in which this is an ideological process as well as a disciplinary one and part of a system of symbolic control [88, 89]. Within this translation, he identifies 'discursive gaps' controlled by the curriculum in an act of power and authority, always favouring the *status quo*. Knowledge cannot bridge these gaps without ideology at play. However, Clarence argues that these gaps also open powerful moments for disruption and change [85]. These moments of disruption allow us to consider how students come to know and what students come to know as an act of decoloniality. For academics to identify productive moments of disruption, they first need to articulate 'what constitutes knowledge in their disciplines', 'the knowledge structure of their disciplines', and 'the knowledge creation process of their disciplines' [84, np]. Quinn and Vorster link the idea of specialised disciplinary knowledge to 'powerful knowledge' and argue that what is done with that knowledge is often the domain of the North, alienating students from gaining epistemic access [84]. Section 4 of this article addresses how this plays out in computer programming education.

3.2 Outward- and Inward-Facing Approaches

The system of knowledge production is cyclical, with that from the South reflecting and reproducing colonial epistemologies to achieve 'validity'. When examining knowledge production from a decolonial lens, we need to recognise that it is constructed in multiple forms not always designated as 'valid' and recognise these in our work [25]. However, part of the project is also to deconstruct and reconstruct colonial, 'valid' knowledge systems. Shahjahan et al. identify this as an inward-facing decolonial approach that is prominent in decolonial research from Africa and Asia. They find that:

> Similarly, for some, decolonization meant making room for synthesis of knowledge within disciplines, and not necessarily 'destroying' or completely replacing 'Eurocentrism'. … Here, disrupting to decenter, not complete removal, was a salient idea among scholars focused on African decolonization of knowledge. [8, p. 85]

This inward-facing approach to decoloniality is also disciplinary, occurring more frequently in the social sciences and humanities. Shahjahan et al. note that within these fields, 'several authors thus described altering the curriculum or pedagogy in ways still within, yet pushed the boundaries of their disciplinary areas' [8, p. 90]. They observe that more outward-facing strategies occur in applied fields where 'scholars articulated strategies focused on students critically examining their training and/or their relationship to populations one would engage' [8, p. 91].

The long arm of the global knowledge economy has extended to include computer programming as a 'valid' applied knowledge underlying digital-colonialism. In *New Digital Worlds*, Risam argues that:

> Within colonised and formerly colonised nations and for people outside of dominant cultures, access to the means of digital knowledge production is essential for reshaping the dynamics of cultural power and claiming the humanity that has been denied by the history of colonialism [62].

Access to digital knowledge production, embodied in programming and code, is essential, and true access requires us to surface the embedded epistemologies and engage with them productively from within the Global South [9, 58, 90]. The difficulty of this project is exacerbated by the stringent curricula required to meet international professional standards essential for uptake into a global economy. This extends to many applied fields, as Shahjahan et al. point out, '… the professional culture informing the curriculum posed a challenge [to decolonisation] … In short, the paradoxes of balancing professional knowledge, socialization, and subjugated knowledge added to the complexity of decolonizing pedagogy and curriculum' [8, p. 97]. The response to these complexities is often

to refocus outward, to the engagement of the curriculum with the community, and to overlook the intrinsic coloniality of the disciplines. As demonstrated in the literature reviewed in Section 5, programming education has primarily focused on outward-facing strategies.

Like Shahjahan et al., Quinn and Vorster, and Monnapula-Mapesela et al. note the tensions inherent in higher education to address the demands to equip graduates for employment in a global knowledge economy and still engage with the need for decolonisation [8, 83, 84]. Dalvit et al. explore this tension in relation to computer science in Africa, where they state that the field is 'potentially empowering both economically and in terms of global access' but caution that it remains deeply embedded in Western epistemologies [91, p. 287]. This balancing act does not require us to abandon the one in favour of the other but rather to deconstruct the curriculum content and pedagogy in ways that unsettle the epistemologies inherent in approaches from the Global North and engage in construction and reconstruction within a decolonial framework [8, 83]. I argue that an inward-facing approach to critiquing and decentring the core knowledge formation of the discipline from within is also warranted.

3.3 Reading the Discipline Contrapuntally

Zembylas argues that a decolonial curriculum embraces 'antiessentialism, contrapuntal readings and ethical solidarity' [92, p. 11]. Decolonial programming education would require a concerted effort to approach the content, teaching people to code, from this framework. While writing from a completely different context (Human Rights Education), Zembylas outlines a concern similar to the one faced in programming education: the need to equip educators and learners with tools to recontextualise and deconstruct [92]. A decolonising pedagogy needs to be informed by a 'theoretical heteroglossia that strategically utilizes theorizations and understandings from various fields and conceptual frameworks to unmask the logics, workings, and effects of [...] colonial domination, oppression, and exploitation in our contemporary contexts' [92, p. 9].

The greatest challenge in programming curricula is to find ways to examine the content, learning to code, through the lens of antiessentialism and ethical solidarity. Smith explains this challenge as being due to the extent to which code is obfuscated and resistant to the critique that other pedagogic material is subjected to [9]. Their argument challenges how we engage with pedagogies, deconstruct textbooks and histories, but read around the code. Our eyes slide over it. In teaching programming, both educators and students lack the tools to recontextualise programming itself. To borrow from Zembylas, for us to engage with digital-coloniality, we need to read programming knowledge contrapuntally.

4 Programming Education

4.1 Paradigms of Computing Education

Computer programming education has primarily been theorised in computer science and software engineering. Eden situates computer science research as primarily drawing from three paradigms: the rationalist, the technocratic and the scientific [93]. Their article traces the field's historical progression and the subsequent philosophical disputes among researchers. It argues vehemently against the dominance of the technocratic paradigm, specifically as it relates to undergraduate curricula. They state that the technocratic turn led to 'courses focusing on technological trends teaching software design methodologies, software modelling notations... programming platforms, and component-based software engineering technologies' at the expense of the theoretical foundations of the discipline and assert that the scientific paradigm provides the most productive path forward [93, p. 24]. However, there has been an increasing interest in further broadening the paradigms from which research is conducted.

In computer science education, Thota et al. reason for pluralism and adopting methodological eclecticism through a pragmatic paradigm [94]. In another study, Clear argues for the adoption of critical enquiry [95]. Couldry and Mejias extend the use of critical theory to address a 'decolonial turn' explicitly [61], an approach echoed by Ricaurte [60]. Mejia et al. argue that 'traditional [engineering] scholarships have been normed by epistemological perspectives that have failed to examine structures of domination and oppression in educational settings' and call for critical paradigms to be adopted [96, p. 2]. Their study, 'Critical Theoretical Frameworks in Engineering Education: An Anti-Deficit and Liberative Approach', adopts Freirean models of pedagogy and utilises Critical Discourse Analysis as a method. This is echoed in the study by Cristaldi et al. of the impact of social science education on computing education, which also highlights Freirean models [97]. This broadening of paradigms and methodologies is essential in allowing disparate voices to emerge and to attain the theoretic heteroglossia that demands of decolonial curricula [92].

4.2 Approaches to Teaching Programming

The challenges of teaching code are well documented in a substantial body of literature on pedagogies for programming [71–73, 98, 99]. The bulk of the literature exists within the discourses of computer science and software engineering, as well as that of sectoral training. These all approach programming to facilitate different aims and outcomes and have disparate ways of knowing, doing and making embedded. How programming is taught exists within these broader discourses to serve a purpose as a stepping-stone into disciplinary knowledge or as an entry to employment in the software sector. Computer programming is most often a foundation that must be laid to enable deeper disciplinary engagement with technology fields down the line [100, 101]. The difficulties in teaching programming, combined with the foundational disciplinary nature, mean that introductory programming courses tend to be utilitarian and focus on access; introducing computational thinking, problem-solving and language divorced from context [100]. This simplified, clear focus is necessary for responding to the difficulty of the field. There are several reasons identified for the difficulty of learning programming; these include 'an inaccurate understanding of how a computational model works; an inability to master reading, tracing, and writing code; and an inability to understand high-level concepts such as design' [99].

Various overviews and meta-analyses of programming pedagogic approaches have identified similar challenges [72, 73, 83, 99]. A recurring theme in the literature is that students lack sufficient mental models to approach programming: They struggle with the 'strictness' of programming languages, unnecessarily overemphasise syntax and semantics and fail to map these to computational logic and problem-solving. In his paper 'Programming Pedagogy—A Psychological Overview', Winslow articulates this by stating that '[g]iven a new, unfamiliar language, the syntax is not the problem, learning how to use and combine the statements to achieve the desired effect is difficult' [71, np]. Winslow identifies the pedagogic chain as building from syntax and semantics to combination through design, patterns, planning and testing, and finally to general problem-solving skills [71]. Lau and Yuen's review of programming pedagogy literature also highlights a common three-phase cognitive structure: syntactic, conceptual and strategic [72]. In 'Relationships: Computational Thinking, Pedagogy of Programming, and Bloom's Taxonomy', Selby traces the typical order of teaching programming as follows:

(1) constructs, facts, types
(2) how individual constructs work
(3) use programming constructs in contrived contexts
(4) discriminate, decompose, abstract

(5) create programs, algorithm design
(6) test, evaluate [99, np].

Aligning these to Bloom's Taxonomy Cognitive Domain, '[e]valuation is assigned to the evaluation level; algorithm design is assigned to the synthesis level; abstraction and decomposition are assigned to the analysis level; generalisation is assigned to the application level', and constructs, facts and types (syntax and semantics) are assigned to the comprehension and knowledge levels [99, np]. They then continue by arranging these in order of perceived difficulty, with one being the least difficult and six being the most difficult:

(1) evaluation
(2) algorithm design
(3) generalisation
(4) abstraction of functionality
(5) abstraction of data
(6) decomposition [99, np].

They, therefore, note that the perceived difficulty of computational thinking skills above the levels of knowledge and comprehension (constructs, facts and types) is a reversal of their mapping to Bloom's Taxonomy Cognitive Domain. While they emphasise the need for further research, the observation is useful to help structure a pedagogic approach. In application, one concern identified by Butler and Morgan is that students receive far more and more detailed feedback for simpler concepts than for more complex concepts [102]. Teaching patterns, including sequencing, domain modelling, feedback and assessment need to be carefully considered to ensure that students are receiving support in the areas that challenge them most.

Lau and Yuen identified seven common pedagogic approaches from their literature review: the Structure Programming approach, the Problem-Solving approach, the Software Development approach, the Small Programming approach, the Language Teaching approach, the Learning Theory approach and the nebulous category of 'Other' approaches [72]. In their work, they differentiate between programming skills and programming concepts—broadly syntax and semantics, and problem-solving. When mapped against Selby's reading of the skills in line with Bloom, programming skills are located at the comprehension, knowledge and application levels, while programming concepts are located at analysis, synthesis and evaluation [99].

The approaches outlined here barely scratch the surface of available literature on programming pedagogies. In application, pedagogic strategies are selected based on the structures of knowledge in the individual disciplines where introductory programming is taught and are often adopted as convention. Disciplinary progression and training guide the conceptualisation of curriculum, and, as Clarence warns, negotiation between these and decolonisation is complex [85].

4.3 Identifying Moments of Disruption

Boughey and McKenna, and Quinn and Vorster task educators to identify what knowledge is legitimated in the curriculum and how that knowledge is structured and produced in the discipline [84, 87]. The core skill set of learning to program is commonly approached through a three-phase cognitive structure: syntactic, conceptual and strategic [72, 99, 102]. This structure makes sense from within the epistemic framework of the Global North, where the colonial matrix of power underlies logic but is fundamental to accessing a global discipline [8, 58]. So, the task is to identify where slippages occur in the recontextualisation of knowledge into the curriculum and to deploy those as moments of disruption [84, 86, 89]. Based on the overviews presented, the moment that poses the greatest challenge to students learning to program is remapping conceptual frameworks

of computational models to cross from programming skills to programming concepts, working from syntax and semantics through reading, tracing and writing code, to problem-solving and high-level concepts [72, 99, 102]. This bottleneck can be reframed as a moment of disruption, where the literacy moves from alphabetised to interpretive and applied.

5 Decolonial Approaches

5.1 Contextualised Curricula

5.1.1 Localisation. As the need to train software developers to compete in the global knowledge economy grows, courses teaching programming have proliferated in all education sectors. This has led to the uncritical adoption of international 'best practice' approaches. As in many scientific and technical fields, the content and the skills required to progress and enter a global economy are defined by curricula from international bodies operating within the Global North. When the core difficulties in teaching programming arise from a mismatch of mental models enabling knowledge progression, shoehorning students from the Global South into the ways of knowing and being embedded in curricula from the Global North sets them up for failure. In the case of Tanzania, Apiola and Tedre identify that:

> The curricula are often copied directly from western institutions, and they sometimes hold context-dependent views about content, pedagogy, organisation, interaction, and processes, which may hinder results. Implementing western pedagogical solutions, such as problem-based learning (PBL) and learner-centred practices, into the developing world has faced challenges in various contexts. [98, p. 287]

Apiola and Tedre present an extensive case study of a BSc program in Information Technology at Tumaini University, Tanzania [98]. The paper highlights the failures of a copy-paste curricular approach and argues that 'standard curricula, such as the ACM/IEEE IT curriculum (2005), are not sufficient for that particular socio-cultural and economic context' [98, p. 286]. A study by Sutinen and Vesisenaho similarly finds that:

> Nearly everyone who works in Computing Education Research (CER) uses the universal ACM/IEEE Computing Curricula (ACM & IEEE, 2001) to anchor her/his understanding of what students should learn, and on which basis learning outcomes should be measured. Although such an approach might be justified on purely conceptual grounds, it might attract a researcher to ignore the realities of the learners' background in those cases where the cultural assumptions of learners are radically different from those of learners who have grown up in so-called Western cultures where ICTs are more commonly an integral and accepted part of everyday life. [100, p. 240]

It has been extensively argued that emulating curricula from the North disregards our local context and perpetuates the marginalisation already evident internationally [84]. While it may be expedient, adopting international 'best practice' curricula and pedagogies in preparing students to enter a notoriously exclusionary and adversarial field is complicit in upholding the *status quo*. Kroeze points out that 'although the international guidelines for IS syllabuses leave room for the cultivation of intercultural skills during undergraduate and postgraduate programmes… the dominating effect of western textbooks may leave little room for deep integration of indigenous inputs into these curricula' [58, p. 44]. This is expanded on by Ayalwe et al. in a review of students' performance in a first-year programming course in Botswana [101]. Similar concerns were noted in Nigerian programming instruction, where the authors specifically link their concerns and possible solutions to the need to approach programming as a linguistic skill and consider a range of

pragmatic interventions, including access to infrastructure, exercises and assessment, and tutoring structures [103].

5.1.2 Ethnocomputing and Indigenous Knowledges. In addition to concerns with the structure and delivery of curricula and the access to infrastructure, localisation and Africanisation are also expressed as strategies drawing from translanguaging and Indigenous knowledges [58, 90, 91]. van der Poll et al. argue that this approach may alleviate the alienation that occurs when 'computing scientists emphasize the modernity of computing education and often position it in opposition to traditional knowledge' [90, p. 145]. They argue for courses that emphasise students' contextual circumstances and community needs.

The seminal work on ethnocomputing by Eglash et al. is perhaps the best-known decolonial approach, as it focuses on incorporating disparate knowledge systems into computational thinking and conceptualises software development as a cultural construct [104–106]. Dalvit et al. provide the following explanation:

> Ethnocomputing emphasises the importance of integrating cultural elements into software design and the teaching of Computer Science in developing countries [11]. Since computers were invented in the West, they tend to reflect Western values and cultural traits, thus promoting dependency. To counter dependency, the use and teaching of computers must integrate indigenous knowledge and respond to local problems, making technology more relevant and more accessible at the same time. [91, p. 291]

This finds expression in several interrelated approaches, including Indigenous or culturally situated design [105–107], socially responsible computing [108], justice-centred computing [109], ancestral computing [110], **culturally responsive computing (CRC)** [107, 111, 112], counter-hegemonic computing [104], liberatory computing [113] and intercultural computing [114]. While each approaches the problem from a different angle, the core focus in all of them is on a reciprocal relationship with the community through situated practice drawing from culturally specific or indigenous knowledges.

Ryoo et al. advocate for socially responsible computing, defining it as an approach that:

> … challenges the notion that CS is neutral, objective, or apolitical by making visible the relationships between technological innovation, its creators, and the larger sociocultural and political contexts in which both exist.
> … acknowledges that computing is a form of power in today's society by critically examining how new technologies potentially reflect and reproduce existing inequities.
> … centres social impact and ethics throughout all computing design processes. [108, p. 1]

The focus here is to enable students to develop a critical consciousness in their interactions with computing and take this forward in their own community activism. This addresses the myth of neutrality that permeates technology, our understanding of the digital realm, and computing. It is about contextualising computing within a broader socio-political framework, emphasising that 'these tech-related forms of education "can't pretend to be apolitical"' [108, p. 7].

Similarly, CRC emphasises working with community organisers and empowers students to approach computing from within their own personal narratives to benefit their own communities. Yan et al. describe a process where culturally specific modules were added, and teachers became the locus for 'incorporating and demonstrating the combination of CS [Computer Science] learning and culture' [112, p. 204]. One concern, as Lachney et al. point out, is that '[t]he majority of CRC research tends to report on out-of-school or after-school contexts', and they argue that 'without more attention to CRC in formal contexts the current state of underrepresentation is unlikely

to improve' [107, p. 463]. Another related approach is Ancestral computing, which is guided by centring Indigenous epistemologies and articulates an Ancestral Paradigm. 'This Ancestral paradigm has several guiding principles: a. Embracing Ancestral Knowledge Systems; b. Relational Accountability; c. Computing as a life-asserting and preserving the body of knowledge; and d. Research as a praxis of healing' [110, p. 437].

Efforts to integrate Indigenous knowledges and cultural contexts in computing education are rich and nuanced and are situated in the specificities of each instance. However, two main strategies can be seen: cultural practice and Indigenous knowledges as a lens through which to approach the digital realm, and cultural practice and Indigenous knowledges as a means to communicate computing principles. This simplification effaces the richness of emerging approaches but serves as an entry point to accessing the body of research. The first concerns developing critical consciousness and connecting computing to real-world cultural engagement, adding modules or assignments that address local cultural knowledge and engaging with local community leaders and 'cultural experts' to benefit the community. The second shares the same starting point but continues to draw from Indigenous knowledge and cultural practices, incorporating them in projects that translate the principles into computing. Good examples of this can be seen in the 'Cornrow Curves' project discussed by Lachney et al. [107] and the 'Anishinaabe Arcs' project discussed by Eglash et al. [105].

Eglash et al. locate their discussion as Culturally Situated Design tools. They emphasise 'respectful contextualisation' and emic, insider engagement with the culture. For them, 'the technology interface design process inhabits a "contact zone"', and they stress the development of design agency, where 'student learners are not merely simulating older designs, but discovering "heritage algorithms"... blending of localized knowledge and STEM to develop new community-relevant innovations' [105, p. 1572]. 'Cornrow Curves' also draws from heritage algorithms, again seeing the technology interface as a contact zone, a 'type of meeting point for CS curriculum and local cultural knowledge to connect and interact conceptually and materially' [107, p. 480]. Both projects use a block-based interface to coding, enabling the identification of computing concepts through the culturally situated practice.

5.1.3 The Drawbacks of Hypercontextual Curricula. Contextual curricula address the question of which knowers are legitimated in the curricula [84, 87]. It affirms their knowledges, their communities and their everyday struggles as part of the pedagogic process. However, Lachney et al. warn that these approaches must be deeply rooted in the community, as it is 'apparent that there are always risks of reproducing shallow culture-computing connections or assimilationist logics in CS education' [107, p. 480]. There are two drawbacks to hypercontextualised curricula. The first is the amount of affective labour involved in creating each new curriculum to avoid shallow connections and assimilation. The second is student resistance [8], as the curricula do not always readily translate to skill development for industry. There is still a need to address the complexities of decoloniality while providing students access to the global economy.

5.2 The Language Problem

5.2.1 Context and Programming Language. Internationally, the lack of addressing context in programming curricula is beginning to shift. In the 2021 article 'Programmers' Affinity to Languages', Neto et al. establish their study through the lens of student context. They draw from a range of established pedagogic approaches to investigate the relationship between the programming languages chosen for instruction and the contextual background of the students. They argue that there needs to be a contextual affinity to the programming language selected. They reason that for the selection of an introductory programming language, the key characteristics of the language be identified and that these should then be weighed against the students' contextual background,

notably including the socioeconomic context, previous experience and English language proficiency [115]. Duke et al. also set out a range of recommendations for selecting the programming language used in instruction, but these are largely built on technical and disciplinary needs and do not adequately engage with student context [116]. More focus on the selection of introductory programming language is required to allow us to align with student needs. Careful selection can also provide more opportunities for laying bare the moments of reading, tracing and writing code as a possibility for disruption.

5.2.2 Language of Instruction. The need to equip students for a global economy and the associated need for English proficiency and professional practice is in constant tension with the increased accessibility of learning in a native language [90, 91]. English itself is a site of struggle that has been approached from a broad range of views, stretching from the perspectives of ethical solidarity of multilingualism to incorporating ecocentric worldviews tied to Indigenous communities' relationships to the land [117–120]. As programming languages exist for people and are primarily written for English speakers, English language proficiency and the medium of instruction may form a large part of student access to programming. Lau and Yuen put forward a comprehensive study of the impact of the medium of instruction from the context of programming courses in Hong Kong [121]. While mitigated by many complicating factors, the study suggests that instruction in the student's first language yields better results and that, somewhat predictably, students with middle to low English proficiency are at high risk in English medium instruction. However, in the study of success factors in introductory programming courses in Botswana by Ayalew et al., they found no correlation between students' performance and English as a second language skills [101], neither did Soosai Raj et al. in their study of India [122]. These results require further investigation but may have far-reaching implications in multilingual societies and for institutions with complex language policies. However, with international studies showing contrary results, it needs to be considered from within the context of individual students and institutions.

Additionally, students often challenge decolonial approaches to computer programming education that centre on native language instruction. Shahjahan et al. note that a common challenge for decolonisation in applied fields is student resistance to moving away from 'valid' systems that grant access to the global economy [8]. van der Poll et al. encounter this when addressing the dominance of English in computing curricula in an African context. They attribute it to two primary factors: limited indigenous terminologies in the discipline and English constituting the 'universal language of science' in the South African institutions they investigate. They add that 'computing departments face pressure to compete in the global industry, which can reinforce the idea that English is superior and that African languages cannot make significant contributions with regard to knowledge building in the global arena' [90, p. 145]. The cyclical search for global 'validity' is, therefore, undermining the process from within.

5.2.3 The Language Teaching Approach. One way to address this is to turn to the language-teaching pedagogic approach outlined by Lau and Yuen [72]. This approach maps programming knowledge to that of second natural language acquisition. It suggests that 'the value of reading programs before writing, the use of authentic programs, the study of the cultural milieu of programs, and so forth' needs closer examination [72]. Early proponents of this approach include Schou and Nord [123], Robertson and Lee [124], Baldwin and Macredie [125], and Deek and Friedman [126]. In more recent work, Cunningham et al. investigate the ways in which **second language acquisition (SLA)** theories and pedagogies can strengthen support programming education [127]. In his article 'The Introductory Computer Programming Course Is First and Foremost a Language Course', Portnoff takes this further, neurologically linking the comprehension of computer programs to the 'same regions of the brain that process natural languages' and argues that introductory

programming education often fails as instructors have not taken into account the linguistic aspects of programming [128]. One of the most prolific writers on the use of SLA theory in teaching programming is Lulu Sun. Working with a range of other researchers and co-authors across various studies, they trace the implementation of SLA in programming courses [129–131] but also study the possibility of using SLA techniques to increase student motivation and interest [132].

5.2.4 Translanguaging. Another approach to the problem, in line with the SLA approach to programming pedagogy, is translanguaging. Translanguaging offers an exciting perspective within multilingual contexts. García simplifies translanguaging as '… the act performed by bilinguals of accessing different linguistic features or various modes of what are described as autonomous languages, in order to maximise communicative potential' [133]. García and Kleifgen trace the use of translanguaging theories in literacy studies and education [134]. They focus on a holistic meaning-making repertoire beyond language, multilingualism and literacies. The article also presents case studies of the approach in practice. Ndlangamandla and Chaka build extensive theoretical frameworks for translanguaging and multilingualism as decolonial practice [118, 120, 135]. Jacob et al. and Vogel et al. expand this position to include Computer Science education, leveraging translanguaging in their literacies approach to programming [136, 137]. A video by the Participating in Literacies and Computer Science project is particularly useful. It demonstrates how translanguaging is already used in classrooms and their pedagogy [138]. Dalvit et al., Kroeze, and van der Poll et al. all incorporate translanguaging into their decolonial strategies [58, 90, 91]. Mbirimi-Hungwe and Hungwe also argue for a translanguaging approach to facilitate multilingual students' understanding of computer science concepts through two case studies [139, 140]. The data they present show promising results and fascinating examples, but, as they point out, far more research is still needed to expand these concepts.

5.3 Turning Inward, Reading Contrapuntally

As decoloniality must be specific, each instance is unique, resulting in a wealth of decolonial approaches. This overview only scratches the surface of the work towards counter-narratives in computing education. Most of the approaches outlined here turn to the knowers, their context, their community, their language, but do not yet describe efforts to turn inwards, facing the discipline itself and reading the core knowledge contrapuntally. The interest in this article is programming education, specifically teaching students to code. Projects teaching computational concepts through and with cultural and indigenous knowledges often employ block-type code, solidifying concepts but not teaching market-relevant skills. As Shahjahan et al. point out, this could lead to student resistance, as the pressure to participate in the global economy is keenly felt [8]. They also rely heavily on community engagement, co-creation of curricula and insider voices, which is challenging to implement at scale. How do we overcome student resistance, limited institutional capacity and large classes?

One possibility is to shift towards an inward-facing approach, interrogating the discipline and identifying moments of disruption in knowledge formation. We should shift our understanding of what it means to teach programming and what knowledge is embedded in the discipline itself. The first act of digital coloniality is to mask itself behind the myth of neutrality. To address this, in its specificity, we need to turn to how programming is taught in our curricula.

6 Critical (Code) Literacies

6.1 The Critical and Decoloniality

The relationship between decoloniality and the critical (be it paradigm, theory, consciousness, pedagogy or literacies) is contested [11–14]. Critical theory relies on a deeply colonial discourse,

embedding the human subject in the rationality/modernity/coloniality complex. Bhambra proposes decolonising critical theory through explicit acknowledgement of colonial histories in the construction of the colonial/modern in relation to historical progress [15]. Darder investigates approaches to decolonising interpretivist research [16] and the challenges to Freire's pedagogy when faced with decolonial identity [14]. Zembylas reflects on the necessity to reinvent critical pedagogy [12]. Darlaston-Jones et al., on the other hand, argue for conscientisation over indigenisation in decolonising the curriculum [13]. Mhandu and Ojong evoke the need for a Freirean humanising pedagogy as decolonial praxis [141], and hooks describes how, for her, conscientisation and decolonisation are inextricably linked [10]. Zembylas argues that:

> On the one hand, there are important commonalities in the political project of a 'critical' and a 'decolonising' [education] that ought to be kept in mind; on the other hand, to acknowledge Tuck and Yang's (2012) warning, critical theory and pedagogy may not be always appropriate for making sense of the colonial condition… [12, p. 6]

Despite this tumultuous relationship, strands of critical theory inhabit the decolonial approaches outlined, be it in SLA, translanguaging or contextualisation. Opening this avenue to explicit engagement provides additional tools for decolonial curricula as it allows us to focus on the underlying mythologies of digital-coloniality.

6.2 Finding Moments of Disruption through Reading Programming Knowledge Contrapuntally

Some possible alternatives arise when considering the intersections of moments of disruption with the decolonial strategies of multi- or translanguaging and indigenisation or contextualisation. If we return to the points raised in Sections 2 and 3, we need to follow Pinar and Zembylas and ask ourselves what programming is and how we read programming knowledge contrapunctually [59, 92].

While the foregrounding of context and language alleviates the alienation of students' lived reality from the practice, and develops a critical consciousness of computation, it still re-enacts the perception of 'coding' as something separate, axiologically neutral and hegemonic [142]. This is insufficient to prepare students for real engagement with an adversarial technology sector and for furthering decoloniality.

An alternative approach is to build on the construction of programming as a literacy, both functional and critical. This draws from the Language Teaching approach to programming and shares the concerns of translanguaging and context in curricula. Regarding computational literacy, which she situates as a comprehensive functional and critical engagement, Vee states that it 'enables us to more critically engage with our software because it highlights the people who write it as well as the historical patterns that precede it' [75, 86]. This is foreshadowed as early as 1988 when Schou and Nord motivated for the application of Literary Criticism techniques to the teaching of programming, arguing that 'programming instructors might find it useful to examine program texts in the light of other critical approaches' [123]. Understanding our pedagogy as an introduction to critical literacies offers an entry point into a transformative and decolonial understanding of programming that aligns with a moment of disruption in programming education: the recontextualisation from programming skills to programming concepts through reading, tracing and writing code [72, 99, 116].

6.3 Critical Engagement with Computation

Explicit critical engagement with computation has been proposed for decades under various names, each with a distinct focus but all inextricably interlinked. Many of the decolonial and social justice

approaches outlined earlier speak to the traditions of critical studies. The approach is most certainly not new yet; while more work is actively being produced, it remains on the fringes of common discourse on programming.

Tissenbaum et al. call for developing Critical Computational Literacy, digital literacies that are integrated with students' contexts and identities, leading to empowerment [143]. They define digital literacy 'as the ability to share ideas through digital mediums' and identify that a core challenge to how digital literacy is approached is the separation of the computational from the social. Drawing from Brennan and Resnick, they break computational thinking into 'concepts, the key constructs and ideas that are central to most forms of computing; practices, the activities people engage in when creating computational projects; and perspectives, the ways in which individuals see themselves as computational thinkers [5]' [143, p. 1]. For them, Critical Computational Literacy combines computational thinking and critical pedagogy and 'accounts for complex analytical and interpretive practices that go well beyond the mechanics of learning to code' [143, p. 3]. They point out that,

> Most introductory coding courses and tutorials (e.g., codeacademy.org) aim to teach students a particular programming language, focusing on teaching students 'how to code', rather than encouraging them to learn to think computationally. If computational thinking is going to have the transformational effect across all disciplines predicted by Wing [16], and if today's youth are going to drive this change, we need to radically rethink the contexts we provide students to think computationally with, both socially and programmatically. [143, p. 4]

To address this, their research focuses on situated connections between the student's projects and the socio-cultural contexts they enhance. They work with the MIT App Inventor, a block-based programming environment, to enable students to translate between ideas and applications quickly 'without the need to understand or wrestle with complicated syntax' [143, p. 3]. This emphasises the connection between critical thinking and computational thinking but does not yet address the core concern of many students enrolling in programming education: how to code to gain access to the workforce.

Lee and Soep invert the question, and rather than asking 'how to code' or 'code for all', they ask 'code for what' [144]? The long-running Youth Radio Interactive project they describe teaches youth to code but constantly questions '[w]hat investigative, imaginative, critical, and practical problem-solving abilities do young people need in using code to transform institutions that too often fail them and their communities?' [145, p. 11]. Like Smith [9], they warn against the failure to interrogate the production of digital tools and insist that these tools 'must be critically examined with the same rigor as literary texts' [145, p. 481]. However, they go further, arguing that youth should not only be prepared to critique these tools but also 'to create and produce their own interactive platforms that support counter-narratives to existing dominant ideologies. Only through the production of these digital tools will youth develop the agency required to make the changes they want to see' [145, p. 481]. The Youth Radio Interactive project is expansive. Running outside of formal education, it is able to inhabit a co-production space where discussion is foregrounded and projects are selected to have real-world impacts. The curriculum explicitly questions dominant ideologies through the production of digital multimodal, transmedial products where 'participants learn design and coding not as ends in themselves, but as tools that allow our youth colleagues to make media that matters to them and makes a difference in their social and civic worlds' [145, p. 482].

6.4 Critical Engagement with Code

But what of the code itself? Smith argues that engaging with decoloniality, '[w]hile understanding the outcomes or consequences of tools and techniques is indeed necessary when dealing with

technology, one must also consider how those tools are produced' [9, p. 150]. In the field of Composition Studies, Eyman and Ball trace 'three critical practices for composition that accommodate the many media, modes, and delivery mechanics in use today: rhetoric, design, and code' [146, p. 114]. Like Bogost [147], they argue that rhetorical function is inseparable from the code that underpins it; 'that is, the rhetorical functions enacted at the level of code that promote certain user activity over other possibilities. As such, it is equally important for authors of digital texts to understand and engage with the coding aspects of a webtext with as much rigor as the rhetorical and design aspects' [146, p. 116]. Much of the work engaged specifically with critical reading of code is loosely collected under the banner of Software Studies. Marino traces the history of work gesturing towards this approach, starting with Kittler in 1995 [63, 148]. The Critical Code Studies approach moves analysis from the point of view of the effects of the software as lived in the world to an understanding of the 'situation more reciprocally: to think about the relationship between the audience's experience and the system's internal operations' [149, p. 11].

Marino argues that:

> A person writing what to them is ordinary, functional code is making meaning already. Critically reading code does not depend on the discovery of hidden secrets or unexpected turns, but rather examining encoded structures, models, and formulations; explicating connotations and denotations of specific coding choices; and exploring traces of the code's development that are not apparent in the functioning of the software alone. [63, p. 17]

This is a challenging transition to make. Reading code in this manner requires not only skill but also the conceptual willingness to explore this option. If we do not prepare students to view code as a system of signification, as practitioners, they will not be equipped to see the meanings they create in ordinary day-to-day software development. When approached from a base of purely functional code literacy, these complexities are obfuscated and, therefore, dismissed [9].

6.5 Literacies

Luke notes that 'Definitions of literacy have expanded to include engagement with texts in a range of semiotic forms: visual, aural, and digital multimodal texts…' [150, p. 8]. To begin applying the concepts of literacy to yet another semiotic system, code, it is helpful to first briefly set out how it functions in traditional understanding [74]. Literacy theory is vast and often contested, with a multitude of frameworks and approaches. Following Bailey and Flower, the Cambridge Assessment report on literacy argues that it is historically contingent and offers the following insights: Literacy is an action, not an ability; it is a discursive practice; it is dependent on social convention; it starts with expressive and rhetorical practices; and it allows for metacognitive and social awareness [151]. McLaren breaks literacy into three forms: functional, cultural and critical [152]. I have drawn on the terms functional and critical literacy.

Functional literacy as a term in itself is deployed in multiple ways, making it hard to define briefly [153]. Levine outlines functional literacy as an assertion that there is a standard of literacy competence that is fundamental to individual and collective interaction [154], while McLaren narrows it further to mastery of literacy skills to the level of decoding simple texts [152]. Both McLaren and Levine highlight that, while literacy has often been reduced to functional literacy, a technical discourse necessary for entry into the workforce, understandings of the field have broadened significantly to emphasise critical literacies [152, 154].

Paulo Freire's seminal work *Pedagogy of the Oppressed* (1968) [79] outlines the pedagogic act as ideological and demonstrates it as an imposition on, and overwriting of, subaltern knowledges by the schooling class. In 1987, Freire and Macedo put forward a collection that expands this understanding, touching on literacies, by arguing that 'reading the word' cannot be independent of

'reading the world' and that literacy relies on engaging with context, culture, ideology and building new ways of knowing [155]. Luke concretises this when she explains that technical mastery should be a means to an end, used to 'analyse, critique and transform the norms, rule systems and practices governing the social fields of everyday life' [150, p. 2]. A rich and vast body of theory has emanated from this in the project of critical literacy. Critical literacy is overtly political, engaging with, decoding and critiquing the dominant ideological and epistemological frameworks [150, 156, 157]. 'A critical literacy situates itself in the intersection of language, culture, power, and history—the nexus in which the subjectivities of students are formed through incorporation, accommodation, and contestation' [152, p. 229].

6.6 Critical Code Literacy

But how do we situate code as a literacy within this framework? Vee puts forward an extensive argument for the need to view code as a literacy in its own right [75]. Their work draws on various definitions of literacy and measures code against them. Vee gives a detailed historical account of the parallels between the development, adoption and promotion of 'traditional' literacy and code literacy. They trace the use of the terms in conjunction to the 1960s but demonstrate that this was less about applying literacy as a concept and more as a strategic move to leverage the importance of literacy to promote computing and the computational sciences overall [74].

Authors like Marc Prensky draw on the broad concepts of literacy to emphasise the pervasiveness of engagement with computing and the need for programming to become as fundamental as reading and writing in society [158]. However, as can be seen in Prensky's article, Vee notes that 'unfortunately, when "literacy" is connected to programming, it is often in unsophisticated ways: literacy as limited to reading and writing text; literacy divorced from social or historical context; literacy as an unmitigated form of progress' [74, p. 43]. It is, therefore, deployed in the sense of a functional literacy and, like other functional literacies, connected to the ability to engage in the workplace.

This reductive view of computational literacy disregards the social, cultural and ideological frameworks in which texts and code function. Vee notes that code literacy has often been discussed as a subset under other terms, including procedural literacy and computational literacy. They articulate this as:

> ... the constellation of abilities to break a complex process down into small procedures and then express— or 'write'—those procedures using the technology of code that may be 'read' by a non-human entity such as a computer. In order to write code, a person must be able to express a process in terms and procedures that can be evaluated by recourse to explicit rules. In order to read code, a person must be able to translate those hyper-explicit directions into a working model of what the computer is doing. [74, p. 47]

The act of coding is then to refocus computational thinking and engage with a specific semiotic system. Code is the artefact created to translate the authors' intention to an executable computational process through the deployment of signs; signifier (the language and syntax) and signified (the procedures executed). Code is often introduced as a set of instructions that you give a computer, requiring precise encoding within the parameters of specific rules set out by the programming language [73, 99]. While this is true, it is also reductive. Vee argues, 'We might think of the fallacy of right-or-wrong code as similar to that of literacy's mechanistic misrepresentation—that reading and writing are simply a matter of proper grammar and accurate decoding' [74, p. 56]. This focus on code as a set of instructions supports functional code literacy, an alphabetised approach leading students to read and write code as an unambiguous, utilitarian system removed from social systems and free of ideology [74]. This myth, the axiological neutrality of technology that programming

buys into, is built on the ideology of linear progress from the rationality/coloniality/modernity matrix of power. Smith explains, '... the ways that the code acts to cryptically represent knowledge ensures that the (re)production of particular forms of representations remain without critique at the level of its production' [9, p. 147].

Programming languages are higher-level abstractions; the programming idioms are not machine instructions; they are converted from that language *into* machine instructions. The languages chosen and the relationships constructed between them in tech stacks dictate the kinds of operations that can be instructed and executed. Because of this, programming languages need to signify broadly as well as specifically. Vee reminds us that code is written for translation by computers but also for other people; it has a dual audience [74]. While code may be functional, decoded and enacted by the computer, the complexity and expressiveness of programming languages and idioms exist for the benefit of the author and are born out in its human reception. This manifests as readability (e.g., the perceptions of 'good' vs. 'bad' code), or aesthetics (e.g., 'elegant' vs. 'messy' code), which is context and audience dependant. While it is true that the computer requires precise expression, the conventions guiding that expression are social:

> Strictures such as how to control the program flow, how to name variables, how long functions should be, and how much code to write per line are established socially to help programmers work together, especially in very large teams, but they matter little to the computer. In other words, there are ways of organizing code that the computer understands perfectly well, but that are eschewed by certain human value systems in programming. [74, p. 56]

Cleaner, more precise code may perform more effectively, but at each level, decisions are being made as to *which* performance is being measured. These decisions are made based on the system requirements and architecture, defining what is 'important' to the system being created, what constitutes 'valid' behaviour, and impact the code being written. But, as code is expressive, *what* is written and *how* it is written to meet these requirements reflects back into the system. The selection, therefore, occurs at all levels of the pipeline: from the implemented technology through the software architecture to the level of code. As at all levels, these are human decisions made in a social framework, enacted in a semiotic system; they are ideological. When image recognition software misrecognises race and gender, causing lived harm [54, 55, 159–161], it may not be intentional. It may be an artefact of the algorithm and training data, and it may be a difficult problem to solve, but this does not make it neutral: It stems from an interconnected system of small decisions, a selection of what is 'valid' and what is 'not valid', each replicating the decision maker's ways of knowing and being in the world, cumulatively reinforcing dominant ideologies. As this interplay is so complex, the significance of the small moments is lost. They are obfuscated by only being perceivable at extreme levels of specialisation in the field or, for most, the level of the lived application of the technology and what it does [9]. Marino argues that:

> ... it is not enough to understand what code does without fully considering what it means. ... Like other systems of signification, code does not signify in any transparent or reducible way. And because code has so many interoperating systems, human and machine-based, meaning proliferates in code. [63, p. 4]

The proliferation of meaning, the opacity of the underlying systems, combined with the fallacy that you are coding for a computer and the lingering myth of neutrality, means that these subtleties can be difficult for students, or even professionals in the field, to engage with. This stems from how practitioners are inducted into the space from the ground up, reinforced by how programming is taught as a functional literacy. This approach does not sufficiently prepare students to critically assess the complex systems of signification, leading to practitioners who are sceptical of, or even

aggressively resistant to, critiques of the neutrality of code [63]. Well before complex analyses can take place, students need to understand that code *means*, that it *represents*. Without that foundation, critical engagement with systems of power embedded in the code cannot occur, hampering critical reflective practice. Students need to be made aware of the moment-to-moment meanings, the multiplicity, in the everyday code they write through a continuous reinforcement that, as a programmer, what you say and how you say it in your code matters.

7 Conclusion

This research grew from concerns identified in my own teaching practice. Teaching both programming and digital art criticism to students revealed a disconnect, and students struggled to see the relationship between them. Engagement with critical studies and digital-coloniality cannot be separated from the nitty gritty experience of teaching students how to code. Even if the same students are learning the 'technical' skills in parallel to the 'critical' skills, not enough was done to bring these together. Efforts to redress this revealed the scope of the problem, not only in my own courses but also internationally, in research on computer programming education, digital-coloniality and decolonial computing education. The problem of bridging this gap persists. This article presents a broad review of the problem space to identify areas of intersection. It identifies a moment of disruption in programming curricula and argues for a possible approach to the problem, teaching computer programming as a critical literacy.

The literature demonstrates that critical and decolonial engagement with computer programming encounters a significant obstacle in both industry and academia: the prevailing perception that science, technology and, by extension, programming are axiologically neutral. This aligns with the colonial logic of progress. Altering this perception of programming proves challenging. From its initial introduction in many curricula, programming emphasises mastery of computational thinking, abstraction, logic and problem-solving. This is often taught as clear, neutral machine instructions, overlooking the semiotic complexity and sociocultural entanglements inherent in code. I argue that this creates a blindspot at the level of code itself, masking the digital-colonial apparatus.

Programming education has a wealth of research to draw from; due to the scale of the field, I primarily engaged with meta-analyses. Programming is taught across various disciplines, including computer science, software engineering, sectoral training and digital arts, albeit differently and to varying ends. However, a moment of disruption presents itself in one of the most commonly identified bottlenecks: the remapping of computational models to cross from programming skills to programming concepts. This occurs when moving from syntax and semantics to problem-solving and high-level concepts through reading, tracing and writing code. From the literature review on programming pedagogies, the Language Teaching approach may offer opportunities to challenge the myth of neutrality surrounding programming practice. Language Teaching incorporates both the SLA and Literacies models of programming pedagogy.

Decoloniality in programming education has also been addressed in various forms but can be roughly broken into approaches focusing on contextualisation (including localisation, ethnocomputing and indigenous knowledge) and ones focusing on language (including the language of instruction, the language teaching approach, and translanguaging). Ethnocomputing and Culturally Situated Design are perhaps the best-known decolonial approaches, as they focus on incorporating disparate knowledge systems into computational thinking and conceptualise software development as a cultural construct. Most approaches focus on an outward-facing view, engaging students with communities, cultures and indigenous knowledge to critically engage with their own lived experiences. As decoloniality exists in the specificities of the instance, developing curricula like these in formal educational settings is challenging. It requires community engagement, affective

labour and emic knowledge, which is not readily accessible given the constraints of large, traditional higher education. It also faces student resistance, as the perception is that these curricula do not always readily translate to skill development for industry.

An alternative, inward-facing approach to support decoloniality in programming education is to challenge the pervasive myth of axiological neutrality in programming practice. To open computer programming to scrutiny enables us to deconstruct the digital-colonial power structures embedded in it and foster a critical understanding of how these are reinforced in the code we write. To deconstruct the myth of neutrality, criticality should be incorporated from the outset in programming education. To investigate this further, I turn to literacies. Code literacies can be broken into two interrelated areas: functional and critical code literacies. If functional code literacies are the ability to read and write in a programming language and construct programmatic logic, critical code literacies would be the ability to situate the act of programming as a system of signification and power in a broader sociocultural framework.

Research on programming as a literacy primarily adopts a functional perspective, but the use of critical literacies in programming pedagogy has also been explored. This article argues that deploying critical literacy pedagogy opens new avenues to explore a decolonial engagement with programming.

Access to critical literacy as powerful knowledge follows the logic of conscientisation. While conscientisation, critical pedagogy and critical theory are in tension with decoloniality and often problematised, the relationships are complex and entangled. Rather than reading these against each other, if we read programming education, critical literacies and digital-coloniality through, with, and in relation to each other, it opens avenues to hold space for decolonial praxis.

Further research is needed into applying critical literacy pedagogies in computer programming education. While work in this vein exists, a more concerted focus on decoloniality and the colonial underpinnings of digital knowledge production is necessary. The first step is for us, as programming educators, to provide students with the tools to deconstruct the code they write. This requires criticality in how we model and teach students to program and how they read the resultant code. I argue that opening computer programming to scrutiny in our curricula enables us to deconstruct the digital-colonial power structures embedded in it and foster a critical understanding of how these are reinforced in the code we write.

References

[1] P. Jandrić and A. Kuzmanić. 2015. Digital Postcolonialism. *IADIS International Journal on WWW/Internet* 13, 2 (2015), 18.

[2] P. Jandrić and A. Kuzmanić. 2017. The Wretched of the Network Society: Techno-Education and Colonisation of the Digital. In *Out of the Ruins: The Emergence of Radical Informal Learning Spaces*. R. H. Haworth and J. M. Elmore (Eds.), PM Press, Oakland, 86–104.

[3] M. Kwet. 2019. Digital Colonialism: US Empire and the New Imperialism in the Global South. *Race & Class* 60, 4 (2019), 3–26. DOI: https://doi.org/10.1177/0306396818823172

[4] M. Madianou. 2019. Technocolonialism: Digital Innovation and Data Practices in the Humanitarian Response to Refugee Crises. *Social Media + Society* 5, 3 (Apr. 2019), 2056305119863146. DOI: https://doi.org/10.1177/2056305119863146

[5] A. Birhane. 2020. Algorithmic Colonization of Africa. *SCRIPT-ed* 17, 2 (Aug. 2020), 389–409. DOI: https://doi.org/10.2966/scrip.170220.389

[6] G. Verdi. 2021. The Road to Technocolonialism. *Institute for Internet & the Just Society*. Retrieved May 4, 2021 from http://www.internetjustsociety.org/the-road-to-techno-colonialism

[7] A. Bon, F. Dittoh, G. Lô, M. Pini, R. Bwana, C. WaiShiang, N. Kulathuramaiyer, and A. Baart. 2022. Decolonizing Technology and Society: A Perspective from the Global South. In *Perspectives on Digital Humanism*. H. Werthner, E. Prem, E. A. Lee, and C. Ghezzi (Eds.), Springer International Publishing, Cham, 61–68. DOI: https://doi.org/10.1007/978-3-030-86144-5

[8] R. A. Shahjahan, A. L. Estera, K. L. Surla, and K. T. Edwards. 2022. 'Decolonizing' Curriculum and Pedagogy: A Comparative Review across Disciplines and Global Higher Education Contexts. *Review of Educational Research* 92, 1 (Feb. 2022), 73–113 pages. DOI: https://doi.org/10.3102/00346543211042423
[9] B. Smith. 2016. Mobile Applications and Decolonization: Cautionary Notes about the Curriculum of Code. *Journal of Curriculum and Pedagogy* 13, 2, (May 2016), 144–163. DOI: https://doi.org/10.1080/15505170.2016.1196274
[10] Hooks. 1994. *Teaching to Transgress: Education as the Practice of Freedom*. Routledge, New York.
[11] E. Tuck and K. W. Yang. 2012. Decolonization Is Not a Metaphor. *Decolonization: Indigeneity. Education & Society* 1, 1, Article 1 (Sep. 2012), 1–40. Retrieved September 1, 2022 from https://jps.library.utoronto.ca/index.php/des/article/view/18630
[12] M. Zembylas. 2018. Reinventing Critical Pedagogy as Decolonizing Pedagogy: The Education of Empathy. *Review of Education, Pedagogy, and Cultural Studies* 40, 5 (Oct. 2018), 404–421. DOI: https://doi.org/10.1080/10714413.2019.1570794
[13] D. Darlaston-Jones, J. Herbert, K. Ryan, W. Darlaston-Jones, J. Harris, and P. Dudgeon. 2014. Are We Asking the Right Questions? Why We Should Have a Decolonizing Discourse Based on Conscientization Rather Than Indigenizing the Curriculum. *Canadian Journal of Native Education* 37, 1, Article 1 (2014), 86–104. DOI: https://doi.org/10.14288/cjne.v37i1.196570
[14] A. Darder. 2015. Paulo Freire and the Continuing Struggle to Decolonize Education. *Counterpoints* 500 (2015), 39–54.
[15] G. K. Bhambra. 2021. Decolonizing Critical Theory? Epistemological Justice, Progress, Reparations. *Critical Times* 4, 1 (2021), 73–89. DOI: https://doi.org/10.1215/26410478-8855227
[16] A. Darder. 2015. Decolonizing Interpretive Research: A Critical Bicultural Methodology for Social Change. *The International Education Journal: Comparative Perspectives* 14, 2 (2015), 63–77.
[17] W. D. Mignolo and C. E. Walsh. 2018. *On Decoloniality: Concepts, Analytics, Praxis*. Duke University Press Books.
[18] S. Martinot. [2011]. The Coloniality of Power: Notes toward De-Colonization.*Unpublished Paper, San Francisco State University*. Retrieved 12 November, 2024 from https://d3n8a8pro7vhmx.cloudfront.net/globaljusticecenter/pages/55/attachments/original/1403191569/imp7.pdf?1403191569
[19] N. Maldonado-Torres. 2007. On the Coloniality of Being. *Cultural Studies* 21, 2–3 (Mar. 2007), 240–270. DOI: https://doi.org/10.1080/09502380601162548
[20] W. Mignolo. 2011. Epistemic Disobedience and the Decolonial Option: A Manifesto. *TRANSMODERNITY: Journal of Peripheral Cultural Production of the Luso-Hispanic World* 1, 2 (2011), 44–66. DOI: https://doi.org/10.5070/T412011807
[21] A. Quijano and M. Ennis. 2000. Coloniality of Power, Eurocentrism, and Latin America. *Nepantla: Views from South* 1, 3 (2000), 533–580.
[22] A. Quijano. 2007. Coloniality and Modernity/Rationality. *Cultural Studies* 21, 2–3 (Mar. 2007), 168–178. DOI: https://doi.org/10.1080/09502380601164353
[23] A. Stojnić. 2017. Power, Knowledge, and Epistemic Delinking. *AM Journal of Art and Media Studies* 14 (Oct. 2017), Article 14, 105–111. DOI: https://doi.org/10.25038/am.v0i14.218
[24] E. Restrepo. 2018. Coloniality of Power. In *The International Encyclopedia of Anthropology* (1st ed.). H. Callan (Ed.), Wiley, 1–6. DOI: https://doi.org/10.1002/9781118924396.wbiea2118
[25] B. L. Hall and R. Tandon. 2017. Decolonization of Knowledge, Epistemicide, Participatory Research and Higher Education. *Research for All* 1, 1 (Jan. 2017), 6–19. DOI: https://doi.org/10.18546/RFA.01.1.02
[26] W. D. Mignolo. 2007. DELINKING: The Rhetoric of Modernity, the Logic of Coloniality and the Grammar of De-Coloniality. *Cultural Studies* 21, 2–3 (Mar. 2007), 449–514. DOI: https://doi.org/10.1080/09502380601162647
[27] T. Garba and S.-M. Sorentino. 2020. Slavery Is a Metaphor: A Critical Commentary on Eve Tuck and K. Wayne Yang's 'Decolonization Is Not a Metaphor'. *Antipode* 52, 3 (2020), 764–782. DOI: https://doi.org/10.1111/anti.12615
[28] R. J. Horvath. 1972. A Definition of Colonialism. *Current Anthropology* 13, 1 (Feb. 1972), 45–57. DOI: https://doi.org/10.1086/201248
[29] B. Lüthi, F. Falk, and P. Purtschert. 2016. Colonialism without Colonies: Examining Blank Spaces in Colonial Studies. *National Identities* 18 (Jan. 2016), 1–9. DOI: https://doi.org/10.1080/14608944.2016.1107178
[30] K. Nkrumah. 1966. *Neo-Colonialism: The Last Stage of Imperialism,* Later (printing ed.). International Publishers, New York.
[31] All-African Peoples' Conference. 1961. Resolution on Neocolonialism. In *Conference Statement*, Cairo: Republished by Pambazuka News, 1961. Retrieved November 12, 2024 from https://www.pambazuka.org/global-south/africa-all-african-peoples-conference-statement-neocolonialism
[32] O. T. Afisi. 2022. Neocolonialism. *The Internet Encyclopedia of Philosophy*. Retrieved November 12, 2024 from https://iep.utm.edu/neocolon/
[33] T. Obadina. 2022. The Myth of Neo-Colonialism. *Africa Economic Analysis*, African Economic Analysis, Retrieved November 12, 2024 from https://www.africabib.org/rec.php?RID=P00042674&DB=p

[34] D. Haag. 2012. Mechanisms of Neo-Colonialism: Current French and British Influence in Cameroon and Ghana. *International Catalan Institute for Peace, Working Paper No. 2011/6.* DOI: https://doi.org/10.2139/ssrn.2033138
[35] R. Connell. 2020. *Southern Theory: The Global Dynamics of Knowledge in Social Science*. Routledge, London. DOI: https://doi.org/10.4324/9781003117346
[36] S. Amrute. 2019. *Tech Colonialism Today*. *EPIC2019.* Rhode Island, Nov. 10, 2019. Retrieved from https://medium.com/datasociety-points/tech-colonialism-today-9633a9cb00ad
[37] S. Swartz, A. Nyamnjoh, and A. Mahali. 2020. Decolonising the Social Sciences Curriculum in the University Classroom: A Pragmatic-Realism Approach. *Alternation*, Vol. SP36, 165–187. DOI: https://doi.org/10/gsqzrp
[38] R. Grosfoguel. 2002. Colonial Difference, Geopolitics of Knowledge, and Global Coloniality in the Modern/Colonial Capitalist World-System. *Review (Fernand Braudel Center)* 25, 3 (2002), 203–224.
[39] J. Comaroff and J. L. Comaroff. 2012. Theory from the South: Or, How Euro-America Is Evolving Toward Africa. *Anthropological Forum* 22, 2 (Jul. 2012), 113–131.
[40] S. T. Kloß. 2017. The Global South as Subversive Practice: Challenges and Potentials of a Heuristic Concept. *The Global South* 11, 2 (2017), 1–17. DOI: https://doi.org/10.2979/globalsouth.11.2.01
[41] S. U. Noble and B. M. Tynes. 2016. *The Intersectional Internet: Race, Sex, Class, and Culture Online* (Illustrated ed.). Peter Lang Inc., International Academic Publishers, New York.
[42] M. Broussard. 2018. *Artificial Unintelligence: How Computers Misunderstand the World*. Mit Pr, Cambridge, Massachusetts.
[43] C. O'Neil. 2017. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy* (Reprint ed.). Broadway Books, New York.
[44] S. Amrute. 2016. *Encoding Race, Encoding Class: Indian IT Workers in Berlin*. Duke University Press Books.
[45] M. L. Gray and S. Suri. 2019. *Ghost Work: How to Stop Silicon Valley from Building a New Global Underclass* (Illustrated ed.). Harper Business, Boston.
[46] W. Nekoto. [n.d.]. Participatory Research for Low-Resourced Machine Translation: A Case Study in African Languages. arXiv:2010.02353. Retrieved May 5, 2021 from http://arxiv.org/abs/2010.02353
[47] P. Chonka, S. Diepeveen, and Y. Haile. 2022. Algorithmic Power and African Indigenous Languages: Search Engine Autocomplete and the Global Multilingual Internet. *Media, Culture & Society* 45, 2 (Jun. 2022) 246–265. DOI: https://doi.org/10.1177/01634437221104705
[48] M. Graham. 2014. Internet geographies. In *Society and the Internet: How Networks of Information and Communication Are Changing Our Lives*. M. Graham and W. H. Dutton (Eds.), Oxford University Press, 99–116. DOI: https://doi.org/10.1093/acprof:oso/9780199661992.001.0001
[49] M. Ragnedda. 2017. *The Third Digital Divide: A Weberian Approach to Digital Inequalities*. Routledge, Taylor & Francis Group, London, New York.
[50] M. Ragnedda and G. W. 2017. *Muschert*. Routledge, London, New York.
[51] M. Graham, S. Sabbata, and M. A. Zook. 2015. Towards a Study of Information Geographies: (Im)Mutable Augmentations and a Mapping of the Geographies of Information. *Geography and Environment* 2, 1 (Jun. 2015), 88–105. DOI: https://doi.org/10.1002/geo2.8
[52] M. Madianou. 2020. A Second-Order Disaster? Digital Technologies during the COVID-19 Pandemic. *Social Media + Society* 6, 3 (Jul. 2020), 2056305120948168. DOI: https://doi.org/10.1177/2056305120948168
[53] N. Couldry and U. A. Mejias. 2019. Data Colonialism: Rethinking Big Data's Relation to the Contemporary Subject. *Television & New Media* 20, 4 (May 2019), 336–349. DOI: https://doi.org/10.1177/1527476418796632
[54] J. Buolamwini and T. Gebru. 2018. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, PMLR. 77–91. Retrieved from https://proceedings.mlr.press/v81/buolamwini18a.html
[55] I. D. Raji, T. Gebru, M. Mitchell, J. Buolamwini, J. Lee, and E. Denton. 2020. Saving Face: Investigating the Ethical Concerns of Facial Recognition Auditing. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES '20).* ACM, New York, NY, 145–151. DOI: https://doi.org/10.1145/3375627.3375820
[56] D. Mohan. 1989. Promotion of 'Modern' Technology: A New Tool for Neo-Colonialism. *Economic and Political Weekly* 24, 32 (1989), 1815–1818.
[57] A. Birhane and O. Guest. 2020. Towards decolonising computational sciences. arXiv:2009.14258. Retrieved September 7, 2023 from http://arxiv.org/abs/2009.14258
[58] J. H. Kroeze. 2019. A Framework for the Africanisation of the Information Systems Discipline. *Alternation* 28 (Dec. 2019) 38–65. DOI: https://doi.org/10/gs7z3d
[59] W. F. Pinar. 2013. Modernity, Technology, Nationality. *Critical Literacy: Theories and Practices* 7, 2 (2013), 3– 19.
[60] P. Ricaurte. 2019. Data Epistemologies, the Coloniality of Power, and Resistance. *Television & New Media* 20, 4 (May 2019), 350–365. DOI: https://doi.org/10.1177/1527476419831640

[61] N. Couldry and U. A. Mejias. 2021. The Decolonial Turn in Data and Technology Research: What Is at Stake and Where Is It Heading? *Information Communication & Society* (Nov. 2021), 1–17. DOI: https://doi.org/10.1080/1369118X.2021.1986102
[62] R. Risam. 2018. *New Digital Worlds: Postcolonial Digital Humanities in Theory, Praxis, and Pedagogy*. Northwestern University Press.
[63] M. C. Marino. 2020. *Critical Code Studies*. The MIT Press. DOI: https://doi.org/10.7551/mitpress/12122.001.0001
[64] N. Montfort. 2021. *Exploratory Programming for the Arts and Humanities* (2nd. ed.). The MIT Press.
[65] C. C. Cruz. 2021. Decolonizing Philosophy of Technology: Learning from Bottom-Up and Top-Down Approaches to Decolonial Technical Design. *Philosophy & Technology* 34, 4 (Dec. 2021), 1847–1881. DOI: https://doi.org/10.1007/s13347-021-00489-w
[66] I. Kakoma. 2003. *Science and Technology in Africa*. Africa World Press.
[67] G. Sica. 2005. *What Mathematics from Africa?* Polimetrica s.a.s.
[68] L. J. F. Green. 2008. 'Indigenous Knowledge' and 'Science': Reframing the Debate on Knowledge Diversity. *Arch* 4, 1 (2008), 144–163. DOI:https://doi.org/10/dgq6c8
[69] L. J. Green. 2012. Beyond South Africa's Indigenous Knowledge-Science Wars. *South African Journal of Science* 108, 7–8 (2012), 44–54.
[70] D. Hess. 2011. Science in an era of globalization: Alternative pathways. In *The Postcolonial Science and Technology Studies Reader*. S. Harding (Ed.), Duke University Press, 419–438.
[71] L. E. Winslow. 1996. Programming Pedagogy—A Psychological Overview. *SIGCSE Bulletin* 28, 3 (Sep. 1996), 17–22. DOI: https://doi.org/10.1145/234867.234872
[72] W. Lau and H. Yuen. 2009. Toward a framework of programming pedagogy. In *Encyclopedia of Information Science and Technology* (2nd. ed.). M. Khosrow-Pour (Ed.), IGI Global, Hershey, PA, 3772–3777. DOI: https://doi.org/10.4018/978-1-60566-026-4.ch601
[73] S. Xinogalos. 2016. Designing and Deploying Programming Courses: Strategies, Tools, Difficulties and Pedagogy. *Education and Information Technologies* 21, 3 (May 2016), 559–588. DOI: https://doi.org/10.1007/s10639-014-9341-9
[74] A. Vee. 2013. Understanding Computer Programming as a Literacy. *Literacy in Composition Studies* 1 (Oct. 2013), 42–64.
[75] A. Vee. 2017. *Coding Literacy: How Computer Programming Is Changing Writing*. MIT Press.
[76] C. Marsh. 2008. *Key Concepts for Understanding Curriculum* (4th. ed.). Routledge, London. DOI: https://doi.org/10.4324/9780203870457
[77] B. Madden. 2015. Pedagogical Pathways for Indigenous Education with/in Teacher Education. *Teaching and Teacher Education* 51 (Oct. 2015), 1–15. DOI: https://doi.org/10.1016/j.tate.2015.05.005
[78] B. Bernstein and J. Solomon. 1999. 'Pedagogy, Identity and the Construction of a Theory of Symbolic Control': Basil Bernstein Questioned by Joseph Solomon. *British Journal of Sociology of Education* 20, 2 (Jun. 1999), 265–279. DOI: https://doi.org/10.1080/01425699995443
[79] P. Freire. 1996. *Pedagogy of the Oppressed*. Penguin, London.
[80] M. W. Apple. 2004. *Ideology and Curriculum*. Routledge.
[81] M. W. Apple. 2013. *Education and Power*. Routledge.
[82] W. F. Pinar. 2005. The Problem with Curriculum and Pedagogy. *Journal of Curriculum & Pedagogy* 2, 1 (2005), 67–82. DOI: https://doi.org/10.1080/15505170.2005.10411529
[83] M. Monnapula-Mapesela, N. Malebo, and I. Ntshoe. 2019. Re-imagining curriculum development and the role of academic developers in a university of technology in the post-colonial setting. In *Re-Imagining Curriculum: Spaces for Disruption*. L. Quinn (Ed.), AFRICAN SUN MeDIA.
[84] L. Quinn and J.-A. Vorster. 2019. Why the focus on 'curriculum'? Why now? In *Re-Imagining Curriculum: Spaces for Disruption*. L. Quinn (Ed.), AFRICAN SUN MeDIA.
[85] S. Clarence. 2019. Re-imagining knowledge in the curriculum: Creating critical spaces for alternative possibilities in curriculum design. In *Re-imagining Curriculum: Spaces for disruption*. L. Quinn (Ed.), AFRICAN SUN MeDIA.
[86] S. Morreira, K. Luckett, S. H. Kumalo, and M. Ramgotra. 2020. Confronting the Complexities of Decolonising Curricula and Pedagogy in Higher Education. *Third World Thematics: A TWQ Journal* 5, 1–2 (Mar. 2020), 1–18. DOI: https://doi.org/10.1080/23802014.2020.1798278
[87] C. Boughey and S. McKenna. 2021. *Understanding Higher Education: Alternative Perspectives*, African Minds, Cape Town, South Africa (2021). DOI: https://doi.org/10.47622/9781928502210
[88] B. Bernstein. 1981. Codes, Modalities, and the Process of Cultural Reproduction: A Model. *Language in Society* 10, 3 (1981), 327–363.
[89] B. B. Bernstein. 2003. *Class, Codes, and Control*. Routledge, London, New York.

[90] A. Poll, I. Zyl, and J. Kroeze. 2020. Towards Decolonisation and Africanisation of Computing Education in South Africa. *Communications of the Association for Information Systems* 47, 1 (2020), 140–164. DOI: https://doi.org/10.17705/1CAIS.04707

[91] L. Dalvit, S. Murray, and A. Terzoli. 2008. The role of indigenous knowledge in computer education in Africa. In *Learning to Live in the Knowledge Society*. M. Kendall and B. Samways (Eds.), Springer US, Boston, MA, 287–294. DOI: https://doi.org/10/fjfdbd

[92] M. Zembylas. 2017. Re-Contextualising Human Rights Education: Some Decolonial Strategies and Pedagogical/Curricular Possibilities. *Pedagogy, Culture & Society* 25, 4 (Oct. 2017), 487–499. DOI: https://doi.org/10.1080/14681366.2017.1281834

[93] A. H. Eden. 2007. Three Paradigms of Computer Science. *Minds & Machines* 17, 2 (Aug. 2007), 135–167. DOI: https://doi.org/10.1007/s11023-007-9060-8

[94] N. Thota, A. Berglund, and T. Clear. 2012. Illustration of Paradigm Pluralism in Computing Education Research. In *Proceedings of the Fourteenth Australasian Computing Education Conference in ACE '12*. Australian Computer Society, Inc., Vol. 123, 103–112.

[95] T. Clear. 2005. Critical enquiry in computer science education. In *Computer Science Education Research*. S. Fincher and M. Petre (Eds.), CRC Press, 105–125.

[96] Joel Alejandro Mejia, Renata A. Revelo, Idalis Villanueva, and Janice Mejia. 2018. Critical Theoretical Frameworks in Engineering Education: An Anti-Deficit and Liberative Approach. *Education Sciences* 8, 4 (2018), 158. DOI: https://doi.org/10.3390/educsci8040158

[97] G. Cristaldi, K. Quille, A. P. Csizmadia, C. Riedesel, G. M. Richards, and F. Maiorana. 2022. The Intervention, Intersection and Impact of Social Sciences Theories upon Computing Education. In *Proceedings of the 2022 IEEE Global Engineering Education Conference (EDUCON '22)*. IEEE, Tunis, Tunisia, 1561–1570.

[98] M. Apiola and M. Tedre. 2012. New Perspectives on the Pedagogy of Programming in a Developing Country Context. *Computer Science Education* 22, 3 (Sep. 2012), 285–313. DOI: https://doi.org/10.1080/08993408.2012.726871

[99] C. C. Selby. 2015. Relationships: Computational Thinking, Pedagogy of Programming, and Bloom's Taxonomy. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, 80–87. DOI: https://doi.org/10.1145/2818314.2818315

[100] E. Sutinen and M. Vesisenaho. 2006. Ethnocomputing in Tanzania: Design and Analysis of a Contextualized ICT Course. *Research and Practice in Technology Enhanced Learning* 01, 03 (Nov. 2006), 239–267.

[101] Y. Ayalew, E. Tshukudu, and M. Lefoane. 2018. Factors Affecting Programming Performance of First Year Students at a University in Botswana. *African Journal of Research in Mathematics, Science and Technology Education* 22, 3 (Sep. 2018), 363–373. DOI: https://doi.org/10.1080/18117295.2018.1540169

[102] M. Butler and M. Morgan. 2007. Learning Challenges Faced by Novice Programming Students Studying High Level and Low Feedback Concepts. In *Proceedings Ascilite Singapore*. Singapore, 99–107.

[103] M. A. Sokunbi, A. Yekini N., A. F. Akinsola, and P. E. Ishola. 2015. Pragmatic Approaches to Improve Computer Programming Pedagogy in Tertiary Institutions. In *Presented at the Conference: 3rd Conference of Computer Educators Association of Nigeria (CEAN)*. University of Nigeria Nsukka. Unpublished. DOI: https://doi.org/10.13140/RG.2.1.3899.8802

[104] R. Eglash, A. Bennett, L. Cooke, W. Babbitt, and M. Lachney. 2021. Counter-Hegemonic Computing: Toward Computer Science Education for Value Generation and Emancipation. *ACM Transactions on Computing Education* 21, 4 (Dec. 2021), 1–30.

[105] R. Eglash, M. Lachney, W. Babbitt, A. Bennett, M. Reinhardt, and J. Davis. 2020. Decolonizing Education with Anishinaabe Arcs: Generative STEM as a Path to Indigenous Futurity. *Education Technology Research and Development* 68, 3 (Jun. 2020), 1569–1593.

[106] R. Eglash, A. Bennett, C. O'Donnell, S. Jennings, and M. Cintorino. 2008. Culturally Situated Design Tools: Ethnocomputing from Field Site to Classroom. *American Anthropologist* 108, 2 (Jan. 2008), 347–362. DOI: https://doi.org/10.1525/aa.2006.108.2.347

[107] M. Lachney, A. G. Bennett, R. Eglash, A. Yadav, and S. Moudgalya. 2021. Teaching in an Open Village: A Case Study on Culturally Responsive Computing in Compulsory Education. *Computer Science Education* 31, 4 (Oct. 2021), 462–488.

[108] J. J. Ryoo, A. Morris, and J. Margolis. 2021. 'What Happens to the Raspado Man in a Cash-Free Society?': Teaching and Learning Socially Responsible Computing. *ACM Transactions on Computing Education* 21, 4 (Oct. 2021), 31:1–31:28.

[109] G. Jayathirtha, G. Chapman, and J. Goode. 2024. Holding a Safe Space with Mutual Respect and Politicized Trust: Essentials to Co-Designing a Justice-Oriented High School Curricular Program with Teachers. In *Proceedings of the 2024 on RESPECT Annual Conference*. ACM, Atlanta, GA, 215–223. DOI: https://doi.org/10.1145/3653666.3656090

[110] A. López-Quiñones, M. Martinez-Lopez, C. D. Moreno Sandoval, J. Carroll-Miranda, A. E. Lindala, M. C. Chatman, J. Fleming, E. T. Shockley, D. Cadeau, and E. Flores-Reyes. 2023. Ancestral Computing for Sustainability:

Centering Indigenous Epistemologies in Researching Computer Science Education. *TechTrends* 67, 3 (May 2023), 435–445.

[111] L. Cooke, S. Vogel, M. Lachney, and R. Santo. 2019. Culturally Responsive Computing: Supporting Diverse Justice Projects in/as Computer Science Education. In *Proceedings of the 2019 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT '19)*, 1–2. DOI: https://doi.org/10.1109/RESPECT46404.2019.8985928

[112] W. Yan, J. A. Hovermill, P. Prescott, and A. Amresh. 2024. Teaching Computing in Indigenous Schools: An Early Experience Report. In *Proceedings of the 2024 on RESPECT Annual Conference*. ACM, Atlanta, GA, 201–205. DOI: https://doi.org/doi:CrossRef][10.1145/3653666.3656104

[113] R. Walker, O. Dias, M. Taylor, and C. Breazeal. 2024. Alleviating the Danger of a Single Story through Liberatory Computing Education. In *Proceedings of the 2024 on RESPECT Annual Conference*. ACM, Atlanta, GA, 169–178.

[114] I. Arawjo and A. Mogos. 2021. Intercultural Computing Education: Toward Justice across Difference. *ACM Transactions on Computing Education* 21, 4 (Dec. 2021), 1–33. DOI: https://doi.org/10.1145/3458037

[115] A. C. Neto, C. Araújo, M. J. V. Pereira, and P. R. Henriques. 2021. Programmers' Affinity to Languages. In *Proceedings of the 2nd International Computer Programming Education Conference*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 7.

[116] R. Duke, E. Salzman, J. Burmeister, J. Poon, and L. Murray. 2000. Teaching Programming to Beginners - Choosing the Language Is Just the First Step. In *Proceedings of the Australasian Conference on Computing Education (ACSE '00)*. ACM, Melbourne, Australia, 79–86. DOI: https://doi.org/10.1145/359369.359381

[117] b. hooks. 1995. This is the oppressor's language/yet I need it to talk to you: Language, a place of struggle. In *Between Languages and Cultures: Translation and Cross-Cultural Texts.* A. D. Needham and C. Maier (Eds.), University of Pittsburgh Press, Pittsburg, PA, 295–301.

[118] C. Chaka. 2020. Translanguaging, Decoloniality, and the Global South: An Integrative Review Study *Scrutiny2* 25, 1 (Jan. 2020), 6–42. DOI: https://doi.org/10.1080/18125441.2020.1802617

[119] P. J. Meighan. 2021. Decolonizing English: A Proposal for Implementing Alternative Ways of Knowing and Being in Education. *Diaspora, Indigenous, and Minority Education* 15, 2 (2021), 77–83. DOI: https://doi.org/10/gtkvwm

[120] S. C. Ndlangamandla. 2024. The Coloniality of English Proficiency and EMI: Decolonization, Language Equity, and Epistemic (in) Justice. *International Journal of Language Studies* 18 (2024), 105–130.

[121] W. W. Lau and A. H. Yuen. 2011. The Impact of the Medium of Instruction: The Case of Teaching and Learning of Computer Programming. *Education and Information Technologies* 16, 2 (Jun. 2011), 183–201.

[122] A. G.Soosai Raj, K. Ketsuriyonk, J. M. Patel, and R. Halverson. 2018. Does Native Language Play a Role in Learning a Programming Language? In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, 417–422. DOI: https://doi.org/10.1145/3159450.3159531

[123] C. D. Schou and R. Nord. 1988. Literary Criticism and Programming Pedagogy. In *Proceedings of the 1988 ACM 16th Annual Conference on Computer Science (CSC '88)*. ACM, New York, NY, 67–71. DOI: https://doi.org/10.1145/322609.322617

[124] S. A. Robertson and M. P. Lee. 1995. The Application of Second Natural Language Acquisition Pedagogy to the Teaching of Programming Languages: A Research Agenda. *SIGCSE Bulletin* 27, 4 (Dec. 1995), 9–12. DOI: https://doi.org/10.1145/216511.216517

[125] L. P. Baldwin and R. D. Macredie. 1999. Beginners and Programming: Insights from Second Language Learning and Teaching. *Education and Information Technologies* 4, 2 (Oct. 1999), 167–179. DOI: https://doi.org/10.1023/A:1009652001566

[126] F. P. Deek and R. S. Friedman. 2001. *Computing and Composition. Common Skills, Common Process.* JCSE Online.

[127] R. Cunningham, P. S. Espejo, C. Frederick, L. Sun, and L. Ding. 2016. A Second Language Acquisition Approach to Learning Programming Languages, Embry-Riddle Aeronautical University Publications, 1–10, Retrieved November 12, 2024 from https://commons.erau.edu/publication/173

[128] S. R. Portnoff. 2018. The introductory computer programming course is first and foremost a language course. *ACM Inroads* 9, 2 (Apr 2018), 34–52. DOI: https://doi.org/10.1145/3152433

[129] L. Sun and C. Frederick. 2015. Applying Second Language Acquisition to Facilitate a Blended Learning of Programming Languages. Publications. Retrieved from https://commons.erau.edu/publication/172

[130] L. Sun, C. Frederick, P. Sanjuan Espejo, and R. Cunningham. 2016. Can We Teach a Programming Language as a Second Language? In *Proceedings of the 2016 ASEE Annual Conference & Exposition Proceedings*. ASEE Conferences, New Orleans, Louisiana, 26434. DOI: https://doi.org/10.18260/p.26434

[131] L. Sun, C. Frederick, L. Ding, and R. Rohmeyer. 2017. The Application of Second Language Acquisition to Programming Language Study. Retrieved from https://commons.erau.edu/publication/575

[132] L. Sun. 2018. Motivating Students to Learn a Programming Language: Applying a Second Language Acquisition Approach in a Blended Learning Environment. Retrieved from https://peer.asee.org/motivating-students-to-learn-a-programming-language-applying-a-second-language-acquisition-approach-in-a-blended-learning-environment
[133] O. García. 2009. Chapter 8: Education, multilingualism and translanguaging in the 21st century. In *Social Justice through Multilingual Education*. T. Skutnabb-Kangas, R. Phillipson, A. K. Mohanty, and M. Panda (Eds.), Multilingual Matters,140–158. DOI: https://doi.org/10.21832/9781847691910-011
[134] O. García and J. A. Kleifgen. 2020. Translanguaging and Literacies. *Reading Research Quarterly* 55, 4 (2020), 553–571. DOI: https://doi.org/10.1002/rrq.286
[135] S. C. Ndlangamandla and C. Chaka. 2020. The Intersection between Multilingualism, Translanguaging, and Decoloniality in the Global South. *Scrutiny2* 25, 1 (Jan. 2020), 1–5. DOI: https://doi.org/10.1080/18125441.2020.1832328
[136] S. Vogel, C. Hoadley, A. R. Castillo, and L. Ascenzi-Moreno. 2020. Languages, Literacies and Literate Programming: Can We Use the Latest Theories on How Bilingual People Learn to Help Us Teach Computational Literacies? *Computer Science Education* 30, 4 (Oct. 2020), 420–443. DOI: https://doi.org/10.1080/08993408.2020.1751525
[137] S. R. Jacob, S. Vogel, R. K. Pozos, P. O. Franco, and J. Ryoo. 2021. Leveraging Multilingual Students' Resources for Equitable Computer Science Instruction. In *Proceedings of the 2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT '21)*. IEEE, Philadelphia, PA, 1–2.
[138] *Translanguaging Pedagogy in CS Ed.* 2020. [Online Video]. Retrieved October 7, 2021 from http://archive.nyu.edu/handle/2451/61986
[139] V. Mbirimi-Hungwe and T. Hungwe. 2018. Translanguaging for Epistemic access to Computer Science Concepts: A Call for Change. *Per Linguam: A Journal of Language Learning = Per Linguam: Tydskrif vir Taalaanleer* 34, 2 (Dec. 2018), 97–111. DOI: https://doi.org/10.5785/34-2-771
[140] V. Mbirimi-Hungwe and T. Hungwe. 2020. The Use of Translanguaging among Speakers of Mutually Intelligible Languages to Understand Computer Science Concepts: A Case of 'Sepitori' in South Africa. In *Emerging Perspectives on Translanguaging in Multilingual University Classrooms*. V. Mbirimi-Hungwe, T. Hungwe, and S. M. Seeletse (Eds.), Cambridge Scholars Publishing, 1–16.
[141] J. Mhandu and V. B. Ojong. 2020. Rethinking the Complexities of Decolonising Curricula and Humanising Pedagogy in South Africa's Higher Education. *Alternation*, Vol. SP36, 148–138.
[142] L. Strate. 2012. If It's Neutral, It's Not Technology. *Educational Technology* 52, 1 (2012), 6–9.
[143] M. Tissenbaum, J. Sheldon, L. Seop, C. H. Lee, and N. Lao. 2017. Critical Computational Empowerment: Engaging Youth as Shapers of the Digital Future. In *Proceedings of the 2017 IEEE Global Engineering Education Conference (EDUCON '17)*. IEEE, Athens, Greece, 1705–1708.
[144] C. H. Lee and E. Soep. 2016. None but Ourselves Can Free Our Minds: Critical Computational Literacy as a Pedagogy of Resistance. *Equity & Excellence in Education* 49, 4 (Oct. 2016), 480–492.
[145] E. Soep. 2018. Beyond Coding: Using Critical Computational Literacy to Transform Tech. *Texas Education Review* 6, 1 (2018), 10–16. DOI: https://doi.org/10.15781/T24J0BF37
[146] D. Eyman and C. Ball. 2014. Composing for Digital Publication: Rhetoric, Design, Code. *Composition Studies* 42, 1 (2014), 114–117.
[147] I. Bogost. 2007. *Persuasive Games: The Expressive Power of Videogames*. MIT Press, Cambridge, MA.
[148] F. Kittler and J. Johnston, 1997. There is No Software. In *Literature, Media, Information Systems*. Routledge, 147–155.
[149] N. Wardrip-Fruin. 2009. *Expressive Processing*. MIT Press.
[150] A. Luke. 2012. Critical Literacy: Foundational Notes. *Theory into Practice* 51, 1 (Jan. 2012), 4–11. DOI: https://doi.org/10.1080/00405841.2012.636324
[151] Cambridge Assessment. 2013. What Is literacy? An Investigation into Definitions of English as a Subject and the Relationship between English, Literacy and 'Being Literate'. *Cambridge Assessment, Research Report*. Accessed May 6, 2021. [Online].
[152] P. McLaren. 2011. Culture or Canon? Critical Pedagogy and the Politics of Literacy. *Harvard Educational Review* 58, 26 (Jan. 2011), 213–235. DOI: https://doi.org/10.17763/haer.58.2.n106615465585220
[153] O. Chigisheva. 2018. Functional Literacy: Terminological Ambiguity in the Worldwide Educational Context. *Astra Salvensis - revista de istorie si cultura* VI, Special Issue (018), 963–970.
[154] K. Levine. 1994. Functional Literacy in a Changing World. In *Functional Literacy: Theoretical Issues and Educational Implications*. L. T. Verhoeven (Ed.), John Benjamins Publishing.
[155] P. Freire and D. P. Macedo. 1987. *Literacy: Reading the Word & the World*. Bergin & Garvey Publishers.
[156] H. Janks. 2009. *Literacy and Power* (1st. ed.). Routledge, New York.
[157] A. Dison and L. Hess-April. 2019. Integrating academic literacies into the curriculum in occupational therapy. In *Re-Imagining Curriculum: Spaces for Disruption*. L. Quinn (Ed.), AFRICAN SUN MeDIA.
[158] M. Prensky. 2008. *Programming: The New Literacy*. Edutopia.org. The George Lucas Education Foundation.

[159] I. D. Raji. 2020. Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT '20)*. ACM, New York, NY, 33–44. DOI: https://doi.org/10.1145/3351095.3372873

[160] T. Simonite. 2018. When It Comes to Gorillas, Google Photos Remains Blind. *Wired*. Retrieved September 14, 2023 from https://www.wired.com/story/when-it-comes-to-gorillas-google-photos-remains-blind/

[161] D. P. Williams. 2020. Fitting the Description: Historical and Sociotechnical Elements of Facial Recognition and Anti-Black Surveillance. *Journal of Responsible Innovation* 7, Suppl. 1 (Dec. 2020), 74–83. DOI: https://doi.org/10.1080/23299460.2020.1831365

Received 8 March 2024; revised 30 September 2024; accepted 10 October 2024