Do Abstractions Have Politics? Toward a More Critical Algorithm Analysis

Kevin Lin
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA, USA
kevinl@cs.uw.edu

***Abstract*—The expansion of computer science (CS) education in K–12 and higher-education in the United States has prompted deeper engagement with equity that moves beyond inclusion toward a more critical CS education. Rather than frame computing as a value-neutral tool, a justice-centered approach to equitable CS education draws on critical pedagogy to ensure the rightful presence of political struggles by emphasizing the development of not only knowledge and skills but also CS disciplinary identities. While recent efforts have integrated ethics into several areas of the undergraduate CS curriculum, critical approaches for teaching data structures and algorithms in particular are undertheorized. Basic Data Structures remains focused on runtime-centered algorithm analysis.**

We argue for affordance analysis, a more critical algorithm analysis based on an affordance account of value embedding. Drawing on critical methods from science and technology studies, philosophy of technology, and human-computer interaction, affordance analysis examines how the design of computational abstractions such as data structures and algorithms embody affordances, which in turn embody values with political consequences. We illustrate 5 case studies of how affordance analysis refutes social determination of technology, foregrounds the limitations of data abstractions, and implicates the design of algorithms in disproportionately distributing benefits and harms to particular social identities within the matrix of domination.

***Keywords*—abstractions, affordances, algorithms, critical pedagogy, computing education, data structures, design justice, ethics, political identity, rightful presence**

Current discourse around CSforAll and broadening participation in computing frame "equity as inclusion" [6, 28], calling for an extension of access to a high-quality computing education as a fundamental right for all students. "However, are we satisfied with everyone learning to code, if the end game is to produce (admittedly more 'diverse') coders who will primarily work to ensure the continued profitability of capitalist start-ups and technology giants?" [9]. Computing education has traditionally positioned computing as value-neutral and more interested in efficiency and business profit than "do[ing] something good" [16, 19, 28, 29]. This approach reinforces the dominant narratives about the apolitical disciplinary identity of computer science and reproduces systems of oppression that dehumanize and invalidate marginalized students' experiences and perspectives [3, 6, 19, 25, 29].

Technologies embody social relations and political power [9, 15, 31]. A more equitable computer science (CS) education thus requires a more critical CS education that "recognizes computing is not an unequivocal social good" [14] and centers the "rightful presence" of political struggles: the "fraught histories" and "concrete injustices" experienced by students in the computing classroom [6, 25]. "To move toward a justice-centered approach to equity, Vakil argues, we must simultaneously attend to at least three features of CS education: the content of curriculum, the design of learning environments, and the politics and purposes of CS education reform." Specifically, Ko *et al.* define three ideas for a more critical CS education: that computing has limits, data has limits, and CS has responsibility.

Recent efforts to design a more critical CS education in higher education include standalone ethics courses [11, 24]; ethics integrated across the undergraduate computing curriculum [8, 13]; and integrated ethics in specific courses such as machine learning [26], human-centered computing [27], and introductory CS [10, 12]. In emphasizing the unjust distribution of benefits and harms caused by algorithmic decision-making systems, these efforts reflect a recent turn toward a structural and systemic analysis of computing injustices that "explore ethics in relation to institutions, societies, ideology, or epistemological perspectives in CS rather than a focus on the 'good' and 'bad' decisions individual actors make in their interactions with technology" [28].

"Both [the machine learning and mechanism design communities] have been heeding the call for attention to values, politics, and 'social good' more generally, holding more than twenty technical workshops and conferences between them on some variation of fairness, bias, discrimination, accountability, and transparency in the last five years" [2]. Similarly, social scientists have begun studying algorithms as opaque "black boxes" that require "a range of methodological strategies in order to bypass these layers of impenetrability and document the inner working of computational systems" [7]. However, it is less clear how these efforts might translate to data structures and algorithms: the study of computational abstractions underpinning such systems. Critically, methods that treat systems as black boxes will not implicate the design of data structures and algorithms toward the system's decision-making, values, and outcomes. Far from studying a black box system, the study of data structures and algorithms is uniquely positioned in the undergraduate CS curriculum to enable critical examination of the inner workings of computational systems in order to address the question, "Do abstractions have politics?"

Asymptotic Analysis

A survey and expert panel study conducted by Porter *et al.* identified the following learning goals for Basic Data Structures.

1) Analyze runtime efficiency of algorithms related to data structure design.
2) Select appropriate abstract data types for use in a given application.
3) Compare data structure tradeoffs to select the appropriate implementation for an abstract data type.
4) Design and modify data structures capable of insertion, deletion, search, and related operations.
5) Trace through and predict the behavior of algorithms (including code) designed to implement data structure operations.
6) Identify and remedy flaws in a data structure implementation that may cause its behavior to differ from the intended design.

These learning goals present a unique challenge for designing a more critical data structures and algorithms course. "Abstract data types were introduced as a way of freeing a programmer from concern about irrelevant details in his use of data abstractions" [17], divorcing information—and the values embodied therein—from the data structures and algorithms that organize them [19]. Rather than examine the applications of impacts data structures and algorithms on society, Basic Data Structures focuses on examining their efficiency: algorithm analysis is defined as "[r]untime analysis and/or space complexity" [23] using asymptotic notation, such as Big O notation.

Algorithm design and implementation is thus a means of realizing a specification or abstract data type without critically questioning the design of the abstraction [9, 19]. Design issues and tradeoffs are narrowly framed in terms of code behavior and program efficiency rather than design justice—"design that is led by marginalized communities and that aims explicitly to challenge, rather than reproduce, structural inequalities" [9]. The rightful presence of political struggles in the data structures and algorithms classroom is predicated on a more critical algorithm analysis that examines the values embodied in data structures and algorithms (and applications thereof).

Affordance Theory

Affordance analysis is an alternative algorithm analysis that draws on science and technology studies, philosophy of technology, and human-computer interaction to examine how computational abstractions such as data structures and algorithms embody affordances. Affordances are relational properties of objects that make "specific outcomes more likely given the circumstances provided that the subject aims to bring about these outcomes" [15]. For example, "a chair *affords* sitting, a doorknob *affords* turning, a mouse *affords* moving the cursor on the screen and clicking at a particular location, and a touchscreen *affords* tapping and swiping" [9].

The *affordance account of value embedding* is a theory for understanding how technological artifacts embody moral values [15]. Affordances are not value-neutral: like the disposition to act in a certain way, affordances enable outcomes that may be valuable. An artifact such as an assault rifle has "negative value because it enables killing in a broad range of circumstances" [15]. In general, the value of an artifact is determined by the "actions or events it affords" and their resulting values [15].

Affordance Analysis

Affordance analysis applies an affordance account of value embedding toward computational abstractions such as data structures and algorithms. If these abstractions embody affordances that in turn embody values, abstractions can produce "consequences logically and temporally *prior* to any of its professed uses" [31]. In other words, abstractions have politics through the values embodied by their affordances.

> For engineers aiming to design for value, the affordance account already indicates a simple recipe: measure which actions a given artefact makes likely given a context (for which we can build on social scientific tools) and then evaluate whether these affordances are legitimately desirable (for which we have normative ethics). [15]

To identify the affordances of a programming abstraction, consider its Application Programming Interface (API), which "lists the affordances that a software entity makes available" [1]. Data structures often implement abstract data types that provide common programming interfaces [17]. In Java, a `class` or `interface` defines public methods that afford certain actions.

To evaluate an affordance according to its effects on social systems and institutions [2, 7, 9, 32], consider Ferreira *et al.*

History and Context When examining a specific technology, what are the historical and cultural circumstances in which it emerged? When was it developed? For what purpose? How has its usage and function changed from then to today?

Power Dynamics and Hegemony Who benefits from this technology? At the expense of whose labor? How is this technology sold and marketed? What are the economic and political interests for the proliferation of this technology?

Developing Effective Long-Term Solutions What solutions are currently being implemented to address this labor/benefit asymmetry? In what ways do they reinforce or challenge the status quo? What are the long- and short-term implications of these solutions and who will benefit from them?

Wong *et al.* organizes these questions under "*infrastructural speculation*, a lens to center and unravel the lifeworlds of speculative designs," presenting 8 design tactics to focus on "the 'background' practices surrounding technologies beyond use, to think about the broad—yet differential—impacts of infrastructures and contend with questions of institutional power." By attending to "the lifeworld of artifacts—the social, perceptual, and political environment in which they exist," [32] affordance analysis implicates the affordances of computational abstractions such as data structures and algorithms to the systems they empower and the social futures they create.

To illustrate affordance analysis, we present 5 case studies for Basic Data Structures. More examples are available online.[1]

Priority Queues for Content Moderation

A priority queue is an abstract data type where elements are retrieved according to their associated priority value. A max-oriented priority queue retrieves highest-priority elements first while a min-oriented priority queue retrieves lowest-priority elements first. Social media platforms rely on algorithms to draw attention to the most engaging user-generated content.

[1]https://kevinl.info/do-abstractions-have-politics/

In order to manage user-generated content, platforms design content moderation systems. A content moderation system might use a priority queue for human moderators to review flagged content by assigning the priority values according to the most toxic (severe, obscene, disrespectful, harmful, or otherwise disengaging) content.

Content moderation plays an understated but integral role in determining the content shown to users. Affordance analysis reveals that content moderation priority ordering embodies value. For users of the platform—particularly the most marginalized users—prioritizing moderation for the most toxic content may not necessarily reduce the most harmful content. Some users might consider personalized harassment or identity attacks as more harmful than toxic content identified by a general-purpose algorithm [18]. For social media hackers who spread misinformation, the particular way in which content is prioritized can introduce loopholes with broad political impacts. Even if our priority values considered misinformation, we might further question whether misinformation should even be in the same priority queue as toxic content. Human moderators review hundreds of submissions everyday, leading to fatigue, mental health issues, and PTSD-like symptoms. Human moderators desensitized by repeated exposure to toxic content might find it harder to flag and remove misinformation.

A priority queue affords access to the highest/lowest priority-valued elements. Applied toward content moderation, priority queues optimize for review of certain content, distributing social power, benefits, and harms according to their priority values. By attending to the design of the priority queue abstraction, affordance analysis refutes *social determination of technology*: the view that, "What matters is not technology itself, but the social or economic system in which it is embedded" [31].

Binary Trees for Hiring Decisions

Binary trees are the foundation for data structures such as binary search trees that implement associative sets and maps as well as binary heaps that implement priority queues. But beyond implementing abstract data types, binary trees can also directly represent hierarchical relationships between elements in the recursive tree structure such as in a decision tree. A hiring algorithm could represent its decision-making process as a binary tree. In this example, each internal node in the binary tree could represent a hiring question with a yes/no answer that corresponds to the left/right children, and each leaf node could represent a final yes/no hiring decision.

A binary tree affords questions that encode hard requirements for the hiring position because those questions can be answered with a yes/no answer, but other characteristics might be harder to represent. It might be hard to say exactly how much prior experience (or what kind of prior experience) is needed for the job beyond the core requirements, especially when there are many candidates with diverse backgrounds. Creating a more nuanced binary tree hiring decision-making system requires acknowledging this design constraint throughout the requirements planning and question design process. A design that affords yes/no question answers can preclude more open-ended questions that allow a broader diversity of ways for a candidate to demonstrate suitability, rather than only the most prevalent or dominant candidate experiences. By narrowly prioritizing design for the dominant candidate experiences while marginalizing others as edge cases, sociotechnical systems risk exacerbating social injustice.

It's possible to design a binary tree that is not limited to yes/no question answers. Just as we can rewrite multiway if/else-if/else conditionals into nested binary if/else conditionals, we can represent any multiway tree as a binary tree. But because binary trees afford binary questions, algorithms that rely on binary trees may tend toward solutions modeled with purely binary questions. Affordance analysis suggests that the decision to represent a hiring algorithm as a binary tree *dis-affords* questions incompatible with binary answers.

Autocomplete for Search Engines

Autocomplete is an application feature that helps a user select valid search results by showing possible inputs as they type. Wayne (in Parlante *et al.*) describe *Autocomplete-Me*, a simple autocompletion API designed as an assignment for Basic Data Structures. The Java API provides two key operations: a constructor that stores the corpus of all possible autocompletion terms and an `allMatches` method that returns all terms that start with the given prefix ordered by descending weight so that the most important terms appear first.

Our evaluation of these affordances might begin along the same lines as in *Priority Queues for Content Moderation* by critiquing the weight (or importance) ranking. But the use of autocomplete in the specific context of search engines also draws attention to more structural issues. In *Algorithms of Oppression*, Noble implicates search engines in reproducing sexist, racist, or misogynistic ideas through their search suggestions and results. In 2013, "[t]he Google Search autosuggestions featured a range of sexist ideas such as the following:

- Women cannot: drive, be bishops, be trusted, speak in church
- Women should not: have rights, vote, work, box
- Women should: stay at home, be slaves, be in the kitchen, not speak in church
- Women need to: be put in their places, know their place, be controlled, be disciplined

These associations are exacerbated at the intersection of social identities. Noble examined the racist and sexist ideas in the top autocompletion and search results for queries including "Black girls", "Latinas", and "Asian girls". With the ubiquity of search engines, such results embody cultural power: its impacts also extend to people who don't directly use Google Search. The value of a computational abstraction is not only in how it directly affects end users, but also how it affects *societal systems and structures* by eroding civil and human rights.

Shortest Paths for Navigation Directions

The single-pair shortest paths problem focuses on finding a shortest path between two nodes in a graph that minimizes the sum of the edge weights. Several well-known algorithms have been invented to solve variants of the shortest paths problem, including Dijkstra's algorithm and the A* search algorithm.

Shortest paths algorithms can be used to compute navigation directions for a mapping application. Consider a road network represented as an edge-weighted graph where junctions are nodes and edges are distances (or travel time) along the road segment between junctions. However, a shortest path is not necessarily the "best" path. It might not take into account mode of travel, road grade, or ability of the user. It might route through local roads not designed to sustain large amounts of traffic safely, increasing risks to the local neighborhood and other drivers. For pedestrian walking directions, especially at night, it might not offer the most well-lit or populated route. To address these problems, we could modify the edge weights to better match desired outcomes. But even so, the use of shortest paths algorithms for navigation directions might mask more structural issues such as public disinvestment municipal infrastructure and public transit.

Affordance analysis surfaces how algorithms can implicitly encode for an imagined default user situated atop the *matrix of domination*: "white, male, abled, English-speaking, middle-class US citizens" [9]. It can make more visible the unequal distribution of benefits and harms in society as well as highlight the role algorithms can play toward either reinforcing or dismantling those relationships [2].

Shortest Paths for Seam Carving

Seam carving is an approach for content-aware image resizing [4]. Hug (in Parlante *et al.*) describe *Seam Carving* as an assignment for Basic Data Structures. Rather than shrinking images by scaling the image or cropping-out the edges, seam carving removes the least-noticeable vertical or horizontal *seam*, or path of pixels from top-to-bottom or left-to-right.

Shortest paths algorithms can find a least-noticeable seam by representing the image as a graph where vertices represent pixels and edge weights represent the visual difference between adjacent pixels as defined by an *energy function*. Since the energy function defines the visual difference between pixels, it determines the seams that are selected by the shortest paths algorithm and then ultimately removed. In their SIGGRAPH presentation video, Avidan *et al.* compared different energy functions on an image of a dark-skinned woman, including 2 functions that erased body parts. Although they recognize how the "results depend on the given image," an evaluation of the role of skin color and gender [5] is notably absent.

Affordance analysis not only attends to the information erased by abstractions as we've seen in the preceding case studies, but also extends critique to the *cultural and epistemic abstractions* of CS education that lead to the production of anti-political subjects that "acknowledge ethical and political dimensions" but "divest them from 'what counts' as [CS]" [19].

Discussion

Affordance analysis problematizes the study of data structures and algorithms by centering ethical dilemmas. Computational abstractions make certain technical solutions more accessible and likely than other solutions. Since these abstractions embody affordances, and affordances embody values, the choice of abstraction can lead to differential consequences. Affordance analysis advances beyond "deep tech ethics" [11] by implicating the design of abstractions in the outcomes that they produce. Peck (in Doore *et al.*) suggests that this experience of design, evaluation, and critique can lead to ethical reflections: "What does it mean to design a fair algorithm? What is the human cost of efficiency? What systemic advantages/disadvantages are our algorithms [likely] to amplify?"

Affordance analysis offers a more critical algorithm analysis, one that centers the rightful presence of political struggles by examining how sociotechnical systems distribute benefits and harms. Each of the 5 case studies highlighted the implicit and explicit ways that algorithms structure and reinforce social relations. By teaching a more critical algorithm analysis, we not only teach the limits of computation and data [16], but also support a greater appreciation, understanding, and responsibility toward equity—a key element of cultural competence [30].

Although affordance analysis considers social relations, it is inherently limited to the algorithmic components of a sociotechnical system. Affordance analysis recognizes and critiques how abstractions embody values, but it provides less direction for redressing design values and limitations. Costanza-Chock defines design justice as a frame for rethinking the "universalizing assumptions behind affordance theory" and asking "questions about how inequality structures affordance perceptibility and availability." To move beyond universalizing assumptions, affordance analysis must be considered in dialogue with ecological concerns over how data structures and algorithms are developed: the design practices, design narratives, design sites, and design pedagogies that create contemporary social conditions [9]. For example, to rethink design practices, rather than identify a 'better' prioritization for content moderation, we might instead collaborate with users (intended beneficiaries), content moderators (local experts), and governments (regulatory institutions) to design a more sustainable and restorative approach to moderating user-generated content. We might expect social media platforms to treat content moderation as a primary rather than tertiary concern [2].

Affordance analysis is not mutually exclusive with traditional learning objectives for Basic Data Structures such as asymptotic analysis [23]. In fact, we consider it important to study both of these framings "in tandem" [14] in order to recognize the various past, present, and future purposes and limits of CS education [28]. Asymptotic analysis can also be a critical algorithm analysis: the demand for efficient algorithms and computation centralizes hegemonic power in technosocial elites. But it is important to recognize that, historically, CS education has sidelined justice-centered framings in favor of cognitive framings focused on developing students' knowledge, skills, and understanding of CS concepts and practices at the cost of developing students' disciplinary identities [3, 28, 29].

Affordance analysis expands the definition of algorithm analysis by foregrounding the affordances embodied within computational abstractions. As a step toward a more critical CS education, affordance analysis makes space for the rightful presence of political struggles in the computing classroom [6].

References

[1] R. Abbott, "The Bit (and Three Other Abstractions) Define the Borderline Between Hardware and Software," *Minds and Machines*, vol. 29, no. 2, pp. 239–285, 2019. DOI: 10.1007/s11023-018-9486-1.

[2] R. Abebe, S. Barocas, J. Kleinberg, K. Levy, M. Raghavan, and D. G. Robinson, "Roles for Computing in Social Change," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, New York, NY, USA: ACM, 2020, pp. 252–260. DOI: 10.1145/3351095.3372871.

[3] P. Agarwal and T. Sengupta-Irving, "Integrating Power to Advance the Study of Connective and Productive Disciplinary Engagement in Mathematics and Science," *Cognition and Instruction*, vol. 37, no. 3, pp. 349–366, 2019. DOI: 10.1080/07370008.2019.1624544.

[4] S. Avidan and A. Shamir, "Seam Carving for Content-Aware Image Resizing," in *ACM SIGGRAPH 2007 Papers (SIGGRAPH '07)*, New York, NY, USA: ACM, 2007, 10–es. DOI: 10.1145/1275808.1276390.

[5] J. Buolamwini and T. Gebru, "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification," in *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, PMLR, vol. 81, 2018, pp. 77–91.

[6] A. Calabrese Barton and E. Tan, "Beyond Equity as Inclusion: A Framework of 'Rightful Presence' for Guiding Justice-Oriented Studies in Teaching and Learning," *Educational Researcher*, vol. 49, no. 6, pp. 433–440, 2020. DOI: 10.3102/0013189X20927363.

[7] A. Christin, "The Ethnographer and the Algorithm: Beyond the Black Box," *Theory and Society*, vol. 49, no. 5, pp. 897–918, 2020. DOI: 10.1007/s11186-020-09411-3.

[8] L. Cohen, H. Precel, H. Triedman, and K. Fisler, "A New Model for Weaving Responsible Computing Into Courses Across the CS Curriculum," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*, New York, NY, USA: ACM, 2021, pp. 858–864. DOI: 10.1145/3351095.3372871.

[9] S. Costanza-Chock, *Design Justice: Community-Led Practices to Build the Worlds We Need*. Cambridge, MA, USA: The MIT Press, 2020.

[10] S. A. Doore, C. Fiesler, M. S. Kirkpatrick, E. Peck, and M. Sahami, "Assignments that Blend Ethics and Technology," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*, New York, NY, USA: ACM, 2020, pp. 475–476. DOI: 10.1145/3328778.3366994.

[11] R. Ferreira and M. Y. Vardi, "Deep Tech Ethics," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*, New York, NY, USA: ACM, 2021, pp. 1041–1047. DOI: 10.1145/3408877.3432449.

[12] C. Fiesler, M. Friske, N. Garrett, F. Muzny, J. J. Smith, and J. Zietz, "Integrating Ethics into Introductory Programming Classes," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*, New York, NY, USA: ACM, 2021, pp. 1027–1033. DOI: 10.1145/3408877.3432510.

[13] B. J. Grosz, D. G. Grant, K. Vredenburgh, J. Behrends, L. Hu, A. Simmons, and J. Waldo, "Embedded EthiCS: Integrating Ethics Across CS Education," *Communications of the ACM*, vol. 62, no. 8, pp. 54–61, 2019. DOI: 10.1145/3330794.

[14] Y. Kafai, C. Proctor, and D. Lui, "From Theory Bias to Theory Dialogue: Embracing Cognitive, Situated, and Critical Framings of Computational Thinking in K-12 CS Education," in *Proceedings of the 2019 ACM Conference on International Computing Education Research (ICER '19)*, New York, NY, USA: ACM, 2019, pp. 101–109. DOI: 10.1145/3291279.3339400.

[15] M. Klenk, "How Do Technological Artefacts Embody Moral Values?" *Philosophy & Technology*, 2020. DOI: 10.1007/s13347-020-00401-y.

[16] A. J. Ko, A. Oleson, N. Ryan, Y. Register, B. Xie, M. Tari, M. Davidson, S. Druga, and D. Loksa, "It Is Time for More Critical CS Education," *Communications of the ACM*, vol. 63, no. 11, pp. 31–33, 2020. DOI: 10.1145/3424000.

[17] B. Liskov and S. Zilles, "Programming with Abstract Data Types," *SIGPLAN Notices*, vol. 9, no. 4, pp. 50–59, 1974. DOI: 10.1145/942572.807045.

[18] K. Mahar, A. X. Zhang, and D. Karger, "Squadbox: A Tool to Combat Email Harassment Using Friendsourced Moderation," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA: ACM, 2018, pp. 1–13. DOI: 10.1145/3173574.3174160.

[19] J. W. Malazita and K. Resetar, "Infrastructures of abstraction: how computer science education produces anti-political subjects," *Digital Creativity*, vol. 30, no. 4, pp. 300–312, 2019.

[20] S. U. Noble, *Algorithms of Oppression: How Search Engines Reinforce Racism*. NYU Press, 2018.

[21] N. Parlante, J. Zelenski, B. Franke, A. Bhusnurmath, K. Her, K. Gee, E. Manley, T. Urness, M. Zhang, B. Hou, J. DeNero, J. Hug, and K. Wayne, "Nifty Assignments," in *Proceedings of the 47th ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '16)*, New York, NY, USA: ACM, 2016, pp. 588–589. DOI: 10.1145/2839509.2844678.

[22] N. Parlante, J. Zelenski, P.-M. Osera, M. Stepp, M. Sherriff, L. Tychonievich, R. Layer, S. J. Matthews, A. Obourn, D. R. Raymond, J. Hug, and S. Reges, "Nifty Assignments," in *Proceedings of the 46th ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '15)*, New York, NY, USA: ACM, 2015, pp. 673–674. DOI: 10.1145/2676723.2677327.

[23] L. Porter, D. Zingaro, C. Lee, C. Taylor, K. C. Webb, and M. Clancy, "Developing Course-Level Learning Goals for Basic Data Structures in CS2," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*, New York, NY, USA: ACM, 2018, pp. 858–863. DOI: 10.1145/3159450.3159457.

[24] R. Reich, M. Sahami, J. M. Weinstein, and H. Cohen, "Teaching Computer Ethics: A Deeply Multidisciplinary Approach," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*, New York, NY, USA: ACM, 2020, pp. 296–302. DOI: 10.1145/3328778.3366951.

[25] J. J. Ryoo, T. Tanksley, C. Estrada, and J. Margolis, "Take space, make space: how students use computer science to disrupt and resist marginalization in schools," *Computer Science Education*, vol. 30, no. 3, pp. 337–361, 2020. DOI: 10.1080/08993408.2020.1805284.

[26] J. Saltz, M. Skirpan, C. Fiesler, M. Gorelick, T. Yeh, R. Heckman, N. Dewar, and N. Beard, "Integrating Ethics within Machine Learning Courses," *ACM Transactions on Computing Education (TOCE)*, vol. 19, no. 4, pp. 1–26, 2019. DOI: 10.1145/3341164.

[27] M. Skirpan, N. Beard, S. Bhaduri, C. Fiesler, and T. Yeh, "Ethics Education in Context: A Case Study of Novel Ethics Activities for the CS Classroom," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*, New York, NY, USA: ACM, 2018, pp. 940–945. DOI: 10.1145/3159450.3159573.

[28] S. Vakil, "Ethics, Identity, and Political Vision: Toward a Justice-Centered Approach to Equity in Computer Science Education," *Harvard Educational Review*, vol. 88, no. 1, pp. 26–52, 2018. DOI: 10.17763/1943-5045-88.1.26.

[29] ——, "'I've Always Been Scared That Someday I'm Going to Sell Out': Exploring the relationship between Political Identity and Learning in Computer Science Education," *Cognition and Instruction*, vol. 38, no. 2, pp. 87–115, 2020. DOI: 10.1080/07370008.2020.1730374.

[30] A. N. Washington, "When Twice as Good Isn't Enough: The Case for Cultural Competence in Computing," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*, New York, NY, USA: ACM, 2020, pp. 213–219. DOI: 10.1145/3328778.3366792.

[31] L. Winner, "Do Artifacts Have Politics?" *Daedalus*, vol. 109, no. 1, pp. 121–136, 1980. [Online]. Available: https://www.jstor.org/stable/20024652.

[32] R. Y. Wong, V. Khovanskaya, S. E. Fox, N. Merrill, and P. Sengers, "Infrastructural Speculations: Tactics for Designing and Interrogating Lifeworlds," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA: ACM, 2020, pp. 1–15. DOI: 10.1145/3313831.3376515.