A Case for Feminism in Programming Language Design

Felienne Hermans
f.f.j.hermans@vu.nl
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands

Ari Schlesinger
ari.schlesinger@uga.edu
University of Georgia
Athens, Georgia, USA

Abstract

Two critical and interrelated questions regarding the design and study of programming languages are: 1) What does it mean to design a programming language? and 2) Why does minimal demographic diversity persist in the programming language community? In this paper, we present feminism as a philosophical lens for analyzing the programming languages field in order to help us understand and answer the motivating questions above. By using a feminist lens, we are able to explore how the dominant intellectual and cultural norms have both shaped and constrained programming languages. A key contribution of this analysis is the explanation of how marginalization in the programming language community limits the intellectual and demographic makeup of the field. We see this paper as an invitation to everyone in the programming languages field to deepen our collective understanding of the forces shaping our field. Our goal is to illustrate opportunities for more inclusive practices that will introduce greater diversity to the design of programming languages *and* the demographic makeup of the programming language community.

CCS Concepts: • **Social and professional topics** → **Gender**; • **Software and its engineering** → **General programming languages**.

Keywords: Programming Languages, Feminism

ACM Reference Format:
Felienne Hermans and Ari Schlesinger. 2024. A Case for Feminism in Programming Language Design. In *Proceedings of the 2024 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! '24), October 23–25, 2024, Pasadena, CA, USA.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3689492.3689809

Onward! '24, October 23–25, 2024, Pasadena, CA, USA

ACM ISBN 979-8-4007-1215-9/24/10
https://doi.org/10.1145/3689492.3689809

1 Introduction

For as long as I[1] can remember, I have been in love with programming. As a pre-teen, I excitedly taught myself programming with BASIC books from the local library, and as a late teenager, I joined my high school's computer club, and my high school capstone project was a rudimentary machine learning algorithm. In university, I chose compiler courses as electives, and my final MSc project included the development of a small programming language for SAT solving. It seemed a logical next step, when given the opportunity, to pursue a PhD in the direction of programming languages (PL). I am, for all intents and purposes, a very normal programming languages person, with one notable exception: I am female. For the longest part of my career, I did not want to think about this fact; it was painful to even see myself as different from my peers: I just wanted to blend in, do the things I liked doing, and not cause a stir.

I tried to control surface aspects of myself in order to fit in: no earrings, no dresses, no nail polish, and certainly no knitting. Even though knitting, like programming, uses formal languages and requires mathematical thinking and extreme levels of care and preciseness and would in theory be a fitting hobby. I thought things would be easiest if I would just be 'one of the boys'.

Over time though, me and the PL community parted ways, and I drifted to Software Engineering and to Computer Science Education, but I always kept a loose connection to the PL community, which was fitting because I ended up living my dream late in my career: I created a programming language. Yet, I never really fit into the PL community. It turned out that the surface aspects of my identity were not the only thing that made me different—I was interested in different things.

Only recently, I began to understand that the way the field is shaped is not welcoming to people who are interested in studying the contexts and cultures of programming—people like me.

Many of my feelings were vague and ill-defined, and it required the liberation of formally being a programming language designer for me to be brave enough to fully embrace and question my discomfort: *How come I, lover of grammars and parsers, creator of a programming language, do not feel at home in the PL community? What are the dynamics that cause this feeling, and who else might feel left out?* I found

answers in a place I would never have thought to look: feminism.

1.1 Why Feminism?

I understand thinking of feminism is confusing for PL people, trust me, it did not come naturally to me either! I thought feminism was just about gender. How can gender, of all things, play a role in programming language design? Yes, programming languages are mostly made by men, but how can that matter?

Feminism, as I understood it, concerned itself with inequality; it encouraged people to start Women in Tech groups and such[2], things I did not think I needed. After all, I had not needed anyone's permission or encouragement to study Computer Science (CS). Feminism was something that others needed, not me—embracing that would be embracing my gender, and I wanted to rid myself of it. And because of my interests and studies, most of my friends were male, so in my social circle and in my workplace, feminism was not looked upon with great admiration or praise, making it even more unlikely that I would think that it could be of any value to me. When I sometimes cited feminist work that I found interesting, I wasn't met with enthusiasm, so I stopped bringing feminism up in those circles. It wasn't until I met more women in science (outside of CS) and in my personal life (I joined a knitting group) that I found spaces to learn about feminism and how it can help us understand the world, including the PL world.

Why Do We Value 'Hard'? This brings us to a central point of this essay, the evidence standards of the PL community explicitly value quantitative work, making qualitative work much less likely to be published—a topic we cover in more depth in Section 4.

As I re-entered the academic field of PL in the early 2020s, now with some understanding of feminism, I started to see dynamics I could not see before. As I came to the field to talk about Hedy, a multi-lingual programming language I created for programming education, it struck me that the type of questions coming from the community centered around the *hardness* of my work. Why is it hard to build a programming language in Arabic? Is that not trivial, since English ones already exist. Creating one in Arabic would be, in theory, trivial? Why would that be worth your time?

Initially, I went along with the questions, explaining why indeed it was very challenging to build a non-English language [121], but I felt weird while doing it. I realized I did not care all that much about it being hard! Why wasn't I asked about things that I did care about, about emails I had received from teachers all around the world about how they used Hedy? Or about the sense of belonging I felt in the Hedy community of 400+ people who volunteered to translate, to run Hedy events? Or about the day our website went down because 600 kids in South-Africa logged in at the same time? Reflecting upon conversations I had in the PL space, it became clear to me that my value systems were not aligned. PL, I saw, values work that is difficult. Hard work, which is understood as mathematical work such as writing proofs, or work that pushed the boundaries of programming in ways that requires difficult tools that not everyone can use, like proof assistants. Doing those things was hard because not everyone can do it, while the work I was doing was doing the exact opposite thing: I was trying to make programming easier. What made my heart warm was not that Hedy was hard to build (even if it was!) but that people were genuinely loving a thing I made.

Feminism helped me to understand that what academic communities value is a social system in which some people's values are more represented than others.

Feminism Can Help Question Values and Priorities of Programming Languages. As I was trying to understand my mismatch with the PL community, I started to see that the current state of the art in PL research and design prioritizes machines over people, there is a focus on programming languages that are easy to read for the machine, rather than easy for people. The current standards of evaluation in the PL community are set from a masculine perspective as well, valuing formalism and formal methods over user studies, quantitative over qualitative work, and the examining of technical aspects over context and people. Feminism in PL is an invitation to consider different ways of designing and studying programming languages. Feminism in PL introduces interesting and exciting challenges that will require the deliberate work of examining different perspectives; seeking out users from a variety of different backgrounds and identities, and carefully investigating our programming languages and systems from diverse perspectives.

Feminism Can To Better Us Understand Science. While I was reflecting on my experiences, I read the paper we based our methodology on called 'Glaciers, gender and science'. This paper helped me to understand the feelings of otherness I was experiencing in the 'hard is good' conversations. I found this paper when it was first published in 2016, and I became curious what feminism could mean for glaciers. At the time I thought it was an interesting way to look at the world, but nothing more. But when I recently reread the paper, when thinking about the value system of PL research, and with new eyes, I could see how this paper can be the start of a conversation for our field.

In this paper, Carey *et al.* write: "*Most research, and hence discourse and discussions ... stems from information ... with manly characteristics and within masculine discourses*" [18]. Reading Carey's paper I was struck by similar undercurrents in the PL community. Valuing hard, technical, work *over* work done for the sake of helping others is a masculine discourse.

The impact of men on PL research goes beyond what Carey *et al.* describe, since programming languages are not discovered out in the wild but are made by and for people. With that I decided I wanted to write a paper deeply engaging with Carey's work applied to our field.

1.2 What Is Feminism?

To explain how feminism can help us improve the design of programming languages, we need to address the question: What is feminism? One way to characterize feminism is as "*a world view that values women and that confronts systemic injustices based on gender*" [20]. According to the Stanford Encyclopedia of Philosophy **feminism is an "*intellectual commitment and a ... movement that seeks an end to gender-based oppression*"**[91]. Oppression and discrimination can be explicit (e.g., the right to vote or have a bank account) or implicit (e.g., preferring things that are deemed *hard* as described above). In an academic context, feminism helps us understand and address discrimination that is harder to see and identify, implicit and systemic discrimination.

Feminism as a word carries a lot of weight; you might want to think of the term as being like an overloaded operator; it can mean different things in different contexts. Feminism as a social movement is associated with the struggle for the right of individual women for more gender equality; from suffragettes to birth control to workplace harassment to closing the pay gap. Academically, feminism focuses on a broad range of topics, from feminist social movements to feminist theory, from the rights of individual women to the wider mechanisms of inequality in society [3, 7, 26, 63, 93]. Over the years, there have been many different ways to describe feminism (and many different types of feminism).

A central principle in feminism, particularly in feminist philosophy and feminist science and technology studies, is that ***knowledge* is shaped by the context in which it is made**, by the people who are creating knowledge, and that knowledge shapes our world (physically and ideologically) [25, 51, 83]. Knowledge has power.

Feminist standpoint theory argues that knowledge is socially constructed and that people from marginalized social standpoints (like women) have lived experiences that provide perspectives in the construction of knowledge that are valuable (despite often being undervalued) [52, 54, 82]. An important point for feminist standpoint theory is that even within a social group like women, there are many different contexts and standpoints—gender, race, class, sexuality, ability, nationality or historical period. All of these impact one's standpoint.[3]

In the context of PL, the programming languages that have been created and the ways we study programming languages reflect the values of the creators and their social context. A programming language, as all technology, "*bears the imprimatur of social context*" [69]. In aggregate, programming language knowledge creation reflects implicit and explicit values of their creators—creators who have historically belonged to a relatively homogeneous community. **That narrow window of acceptable knowledge creation is limiting** programming language research and design.

Feminism Is About More Than Gender. Feminism starts from a place of ending gender-based oppression, but generalizes this to think about systems of values, and hope those shape and are shaped by the discourse of dominant groups, in many places and in PL, men.

In this paper, we are drawing from an inclusive and expansive view of feminism and feminist theory (including intersectionality and third-world feminism [3, 27, 28, 63, 93, 96]). From this lens, **feminism takes into account how discrimination across many different social and cultural identities is interconnected**. To address and reduce discrimination we need to understand how gender, race, class, sexuality, (dis)ability, nationality, etc. intersect. Feminism is invested in mitigating the oppression of all marginalized peoples. As American Civil Rights activist Fannie Lou Hammer famously said, "*Nobody's free until everybody's free*" [49]. Moreover, to truly address discrimination, we need people from dominant and marginalized groups to work together. In an academic and programming languages context, the means that looking to the connections between feminism and programming languages is an opportunity to pursue more diverse and inclusive intellectual and cultural practices.

Feminism Is For Everyone. One common misconception about feminism that we want to address up front is about who feminism is for—about the roles people of all genders (including men) play in feminism. Feminist scholar bell hooks famously asserts that *feminism is for everyone* [64]. [4] Gender-based oppression negatively impacts everyone (men, women, and folks outside the gender binary). We all benefit in a world that is free from systems of oppression. Likewise, in this paper when we talk about the roles of masculinity and men in perpetuating current discriminatory norms, we want to clarify that we are discussing the normative patriarchal and hegemonic representations of masculinity. As discussed by bell hooks in her book *The Will to Change* [65], masculinity does not need to be hegemonic or dominating. Feminism provides insights and opportunities for everyone to end systemic injustices.

1.3 Structure of This Essay

By examining the PL community through a feminist lens (inspired by the framework of Carey *et al.* [18]) this essay will explore how different factors of masculinity currently define PL design and research in three ways: 1) by constraining the type of PL research that is accepted and acceptable, and 2) by creating structural systems of exclusion, and 3) by

limiting the stories we share on the contributions of both women in the PL field and what it means to create a programming language. The essay closes with the fourth aspect of Carey *et al.*: exploring alternative representations that can widen participation and the scope of PL work.

We will work within this **Thesis statement**: Diversity in both the design of PL and the demographics of the community are limited because of the dominant culture that prioritizes theory and formalism over people and social impact. By using a feminist lens, we can take steps to build a more inclusive intellectual and social culture in PL that will benefit the the scholarship we create, the languages we design, and the experiences of all members of the PL community.

2 Setting the Scene: Why Feminism and PL?

2.1 Exclusion in Modern PL Design

Building on ideas introduced in Section 1.2, knowledge is not absolute and neutral; knowing cannot and should not be separated from context, from the 'knower' who is doing the knowing [25, 37, 50, 70, 80, 83]. This leads us to question what type of knowledge and knowers exist in PL, and how that shapes the resulting languages?

This might be a hard to grasp statement for people with a CS background, because, 1+1 = 2 whatever you think about it, right? However, so much knowledge is contextualized without people realizing it, even in PL. Knowledge creation and technology design are impacted by the context, norms, and values of the people and culture that create them.

For example, western PL designers might disregard cultural aspects of non-Western languages, like non-English numerals (a finding I myself encountered when building my language Hedy [43, 44, 56]). Let's consider a definition in a simple grammar: `digit : 0..9`.
I had never realized before I started to work in localized PL design, that this does not include all people's experiences! In Arabic, digits are not 0-9 but ٠..٩.[5] These numerals are not included in most programming languages, creating a situation in which none of the languages in the TIOBE top 10 [6] can be used to calculate ١ + ١ (1+1) in Arabic (or in the dozens of other numerals around the world). Even what may sounds like the most plain building blocks of programming languages are built on the knowledge that we have, the life we have lived, and the norms of the context we live in.

Likewise, PL designers often fail to take visually impaired programmers into account. `system.cout>>` is not a great user interface (UI) if you are consuming code via screen reader, which might read this as: 'system dot cout greater than greater than', or even 'system <pause> c out greater than greater than', since a full stop is not read in regular text. At the moment, accessible programming languages that ensure usability by people with disability, like the language Quorum, are uncommon [115, 118]. Accessibility is not a frequent feature or value embedded into PL design.

Error messages are another PL design feature shaped by peoples' standpoint, by their lived experiences and values. For many programming language designers, error messages are an afterthought. Other aspects of design will likely overpower the careful crafting of error messages, such as an optimized compiler or an elegant syntax. Many language designers, explicitly or implicitly, take the stance that 'users will just have to deal with error messages when they occur'. Error messages in many languages explicitly do not include the user in their phrasing, saying things like 'missing bracket' or 'unknown variable', which leaves the user out of the equation. For a person with the lived experience of being told they don't belong in programming and being afraid of failure in the programming world (a common experience for women in computing), unclear or unhelpful error messages are more likely to have negative impacts. More readable error messages are not just helpful for those with less prior knowledge, clear error messages are a better experience and will help all users—even professional developers. A recent study on error messages in Elm showed that their well-phrased error messages were the most-named positive experience of professionals working with the language [108].

The absence of non-western numerals, of accessibility, of inclusive error messages, and other PL design choices contribute to larger systems of discrimination within PL and computing at large. Feminism can compliment and extend prior efforts around user-centered design by clarifying and centering the needs, experiences, and values of underrepresented and marginalized people. The researchers that currently form our community both miss out on the lived experience of people not currently in the community, and miss out the many tools (many of which are qualitative) needed to elicit and factor in those experiences into their design. Throughout this paper, we will explore many different ways that feminism can help us understand and mitigate discrimination in the design and study of programming languages.

2.2 Exclusion of Women in CS

Programming in general used to be considered women's work (something we explore more in Section 3.4) [2, 35]. However, this perception has changed. More recent data paints a picture of under-representation and exclusion of women in computing. In 2015 only 20% of people receiving a PhD in the US were women [14], and female researchers are estimated to only have authored about 10% of CS papers [89]. Underrepresented problems are magnified when considering the intersection between race and gender [104]. Not all subfields of Computer Science share the same gender makeup either; the average percentage of women CS faculty

members in the U.S. is almost 17%, but the programming languages (PL) community is second lowest with 14% (second only to theory with 13%) [74]. Counting the first 20 Dagstuhl seminars found by searching for 'programming languages' reveals only 10 female organizers out of 78, a percentage of 13%.

A Childhood of Missed Opportunities and Experiences. With such disparities, it is easy to believe that women simply do not like working in CS and PL, and that innate differences are contributing to or creating this disparity. However, it is much more likely that the experiences of women and men growing up with technology are so different, and this causes long-lasting differences in how people of different genders approach computers. In fact, there are many studies that explain how culture and experience greatly impact who chooses to study CS, who stays in CS, and what motivates why someone is interested in CS [6, 12, 13, 41, 124, 128].[7]

Girls are less likely to participate in out-of-school programming clubs than boys [4] and activities that girls and women enjoy are less likely to be labeled technical, such as sewing and embroidering [120, 131]. Therefore, girls as a group tend to have less prior knowledge about programming, and thus they have different perspectives on computing, which can influence their understanding of, and experiences with, programming. Girls are socialized into working with technology differently, Hallström *et al.* show that "*girls and boys learn to approach and handle technology differently, thereby confirming rather than dissolving gender boundaries*" [48]. Turkle further explains that girls feel that they cannot enter the world of computers without endangering their sense of femininity [127]. Hallström also finds that teachers encourage boys' technological play in different ways than they do girls' [48].

These experiential differences also persist in the domain of mathematics [9, 132]. Prior research has found teachers view girls as successful in mathematics thanks to their hard work [67, 111], while they believe boys' success comes from their talent.

Stereotype and Stereotype Threat. Stereotypes around gender and CS/STEM contribute to an environment where women often have a different experience with technology, computing, and programming [17, 39, 59, 86]. For example, reminding women that stereotypes say they are bad at math will often induce worse performance on math evaluations, a principle called *negative stereotype threat*. However, introducing a positive stereotype by telling women that women perform as well as or better than men on a math exam will often induce performance improvements on math evaluations [29, 45, 81, 113]. There is a wealth of literature empirically observing and exploring mitigation strategies for stereotype threat across marginalized identities [77, 97, 101, 102, 129]. This illustrates how culture and society frame and constrain the (academic) career success of people from mar-ginalized groups, because stereotypes and stereotype threat impact performance, achievement, and sense of belonging. The impact of people's beliefs on discriminatory career outcomes is further demonstrated by research showing that STEM fields where people believe that innate talent is essential for success are more likely to have gender imbalances than fields that believe hard work is important for success [79].

A Lifetime of Different Opportunities and Experiences. Simone de Beauvoir famously said that "*One is not born, but becomes a woman*" [8] which seems to fit here too; girls are, in various ways, encouraged out of working with technology and programming.

These small differences build up to larger patterns of how people view technology later in life. Programming self-efficacy, the belief that one can achieve success in programming, is lower for girls and women than for boys and men [98].

Recent work in Denmark has shown that gender-based preferences impact their choices in a CS context: when university students were given the choice between equivalent coursework, female students favor assignments around people and male students prefer those around things [85], and high-school-aged students show similar preferences [21]. Gender stereotypes shape people's interests and behaviors starting at a young age and continue to impact academic and career choices throughout people's lifetimes [87]. Notably, there are also examples where women choose to actively participate with programming and object/technology focused work. For example, the community around the LilyPad, an Arduino variant that can be used to add electronics to clothing (e-textiles) is 65% female [16]. Critically, interest and behavior is malleable, even in light of factors like gender. It has been observed that gender-based *differences in interest disappear* in more demographically balanced and culturally inclusive settings [12, 13].

In summary, women and men are seen differently both by themselves and by others, resulting from different stereotypes around women and men (and technology), and as such, the women and men have different experiences with computing. These factors continue to influence the experiences (and career outcomes) of women inside and outside of the PL community.

2.3 Research Method

In this paper, we analyze PL through a feminist lens. But what is a feminist lens exactly? Let's first unpack the word 'lens': a lens is a filter through which we view the world. Much like in photography, different lenses provide different ways to view the world, bringing some things into focus while obscuring others. We are always looking at the world through a lens, even if we are not aware of the lens by which

we see the world. Bringing awareness to the lens we are using, either by default or intentional choice, allows us to critically assess, analyze, and build our understanding of the subjects that we are examining.

In this paper, we are not creating, but reusing an existing feminist lens, namely the feminist framework that Carey *et al.* previously used when analyzing the academic glaciology community. I will apply this lens both from within and from outside of the field.

Social scientists, particularly qualitative researchers, often makes the distinction between work where the researcher is part of the community (an insider), and work where the researcher is explicitly outside of it (an outsider) [33]. Generally speaking, the distinction between insider/outsider is important because pre-existing relationships a researcher has to a community shapes how a community interacts with the researchers and how researchers interact with and study a community. Insiders can gain better access to a community, allowing them to better understand the experiences, norms, and beliefs of the community being studied; however, the researcher's closeness may leave things certain things unexplained or unexamined because of their familiarity. Outsiders, on the other hand, can examine a community with fresh eyes, allowing them to see things that insiders may not notice as noteworthy due to being a part of a community; however, outsiders can struggle with restricted access to community members or to the genuine experiences and beliefs of community members. Being a female programming language designer, I have both the insider and the outsider perspective at the same time, which allows me to both deeply understand the workings of the community but also allows me to see what we are currently missing.

2.4 Goals Of This essay

Coming back to my insider-outsider perspective, I sometimes wonder what we are even researching. What exactly is a programming language for? What does it mean to design a programming language? And I keep coming back the the question: why are women of all colors[8] so underrepresented in the programming languages community?

Applying a feminist lens, questioning the shape of the field through systems of power, we can better understand how the PL field operates, what type of research is valued, and which people feel welcomed. This essay will not answer, but raise questions about how we can make the field inclusive of more types of research, and more types of people, including all genders.

In the remainder of this essay, we will apply the four-part feminist framework of Carey *et al.* [18] to programming language design, to examine where perspectives are missing, and why.

3 Gendered Science and Knowledge

The first aspect of Carey *et al.*'s framework that we are examining in the context of PL is the way that science aggregates, weights, and evaluates collected data. They write: "*... natural science fields have historically been defined by, and their credibility built upon manly attributes*" [19]. In glaciology these manly attributes are triumphs over hostile lands; in programming, these are triumphs over complex maths. Like glaciology, the PL culture is largely masculine, which you can see in a number of distinct ways.

3.1 What Aspects of PL Do We Study?

Programming languages have a wide range of aspects, from their syntax and semantics to the way they are used in different fields, how they are extended with libraries, how they change over time and how people prefer to use them. Research in dominant PL conferences, however, most notably POPL or PLDI, focuses on language features and to a much lesser extent on, for example, applications, adoption, or the needs of users. Kaijanaho studied over 2000 papers authored between 1973 and 2012 in PL conferences, and only identified 65 empirical studies into aspects of language design [68]. PL research of course is much broader than POPL or PLDI, there are conferences like <PROGRAMMING> and workshops like PPIG, PX, PLATEAU or PAINT. However, only POPL and PLDI are A* conferences according to 2023's CORE ranking[9], so researchers, especially in their early career, will feel the need to publish there, which means following the ways of doing research in those spaces.

A problem with this limited view is the fact that it is not clear where the other topics can be studied. Views of users on programming languages and their accompanying tools for example are sometimes situated in Software Engineering, either by explicit rejection at PL venues, or by implicit consideration of authors. Let's consider, for example, static analysis tools. Construction of these tools is in scope of PL (POPL 2022 has two tracks on Static analysis, POPL 2023 had one such track), but understanding the use of static analysis tools is published in software engineering [66]. If the building of such tools is in scope of PL, why isn't understanding of how they are perceived by users? Other work studying programming languages, like large-scale analyses of programming language features, does sometimes appear in PL venues [73, 76], but also in Information Systems [99], Software Maintenance [90], Software Testing and Analysis [60] or Mining Software Repositories [133].

The spread-out nature of research on programming languages is problematic, since it prevents the PL community from having a more holistic view of programming language use. We are robbing ourselves of a place for conversations on the different perspectives on the ways people use with programming languages.

3.2 How Do We Study PL?

In addition to the choice of research topics, we can also consider the methods that are in scope in PL, a more subtle issue. In many subfields of computer science, a wide range of different methods is accepted. The call for papers of ICSE mentions two types of evaluation criteria for papers: "*The soundness, clarity and depth of a technical or theoretical contribution, and the level of thoroughness and completeness of an evaluation.*". Such formulation leaves room for user studies, which are also published in SE [55, 66].

However, what is accepted in PL are overwhelmingly quantitative methods and formal methods. For example, the SPLASH call for papers seems to invite a broad range of submissions: "*The paper presents sufficient evidence supporting its claims, such as proofs, implemented systems, experimental results, statistical analyses, case studies, and anecdotes*". Experimental results could, in theory, refer to controlled experiments involving users, a case study could report on an implementation of a new feature, and anecdotes might include reports of user experiences. But in practice what we see in PL papers are new features of programming languages: "*Mathematical approaches play an essential role in many of these papers – for example, the authors describe a new language construct, define its semantics and a type system for it, and then show a proof of type soundness*" [117] (see further Section 5.1 for the structural impact of research methods in PL).

There have been PL researchers arguing for the need to study of human factors of programming language features, most notably Stefik and Hanenberg [116, 118, 119]. In itself, the goal of taking users' views into account is entirely aligned with feminist beliefs that perspectives of diverse stakeholders should be taken into account. However, from the broad arguments that Stefik and Hanenberg made to collect and present more evidence, only the quantitative parts were published and discussed. Other methods that could be used for a deeper understanding of the experiences of users like diary studies, observations and the application of theoretical lenses have not been published at all in the mainstream PL community.

Feminism can help us to understand the narrow and numerical methodological practices; Research methods and practices that do not fall within the masculine discourse of hard, are seen as less valuable. These are frequently associated with women/feminity, such as research involving people and qualitative work more generally. That means that our community does not employ these research methods, even though they could shed a different light on how to improve the state of the art for all people that work with programming languages and systems. This view not only limits our view, excludes people of all genders, who are interested in not only building things, but seeing them through, trying their tools with users, and iterating over their tools.

But because that work is so tied to programming languages, it also does not fit in today's PL community, and sadly it also does also not easily fit in other subfields of computer science, unless it is applied to education, where more diverse research methods are in scope.

3.3 PL as Mathematical

One more system of scientific framing in PL is the dominant view that programming is math and that to demonstrate that a new feature of a programming language is valuable, what is needed is a proof.

It is tempting to respond by saying 'of course PL is mathematical, it came from math!', as I have heard many people in my circle of friends exclaim while I was working on this line of reasoning. I too believed for a long time that mathematics is the true origin of PL and CS. Many computer science departments were part of mathematics departments before they became separate departments, although the history of CS and math is long and complicated [125], and not true all CS programs are part of math; MIT CS originated as part of the Electrical Engineering department, and Harvard's came from Applied Mathematics, part of Engineering.

However, programming did not start off as mathematics; it had to be *made* mathematical. Programming started as an activity closely related to electrical engineering, when programs were created by connecting wires. Only later was it reinvented in academia as mathematical, for example, Dijkstra argued in the mid-70s, decades after programming languages were created, that it must be understood as mathematical [31]—he also argued that 'soft sciences' had no place in software engineering, that it required 'hard science' [32]. For programming to be a hard science, other sciences needed to be put at a distance; Dijkstra argues that "*the soft scientists make themselves even more ridiculous ... by claiming that they can contribute to software engineering*". This is interesting in the light of the large contributions made to PL by social scientists, like Noah Chomsky. Mahoney and van Toen reflect on the role of formalism in the exclusion of women in computing; they have argued that there exists a common notion of "*'hard' computing: computing that is technically demanding, mathematical, formal*" and that this notion marginalizes other less-mathematical aspects of computing [84].

Taking a position that centers mathematics is not without its benefits: Edwards argues that "*by association with the miracles of its machinery, computer work is taken to require vast mental powers, a kind of genius with formalism akin to that of the mathematician*" [34]. Clegg adds "*University computing established itself in relation to the dominant mores of the academy: intellectually challenging, tough, abstract.*" [23].

And this undercurrent is still very much alive today. At SPLASH '23 I witnessed a senior academic jokingly say that a paper was rejected because 'it did not have enough Greek

letters for POPL'. Papers at POPL, they meant, need formal mathematical notation of the sort used for presenting proofs, and such formal notation and proofs are considered hard to understand. Similarities to the higher value assigned to data gathered by climbing inhospitable mountains as outlined by Carey *et al.* [18] are easy to draw.

The mathematical nature of programming was further stressed by explicitly disregarding experimentation as a form of gathering knowledge. Turing award winner Hartmanis argues in his 1994 award lecture that "*in computer science, there is no history of critical experiments that decide between the validity of various theories*" and "*the underlying mathematical model of digital computing is not seriously challenged by theory or experiments*". Such statements reinforce the mathematical nature of programming and limit what type of research is in scope. And this is not limited to the methods we can use, but it also impacts the research questions we can ask. Studies examining, for example, whether programmers prefer curly braces over indentation, or whether programmers make more mistakes in an untyped language, which experiments could certainly help us with, are explicitly placed outside of relevant PL work by Hartmanis.

3.4 PL as Masculine

The field of programming languages was not only made mathematical and theoretical, but also masculine. When programming started, all computation depended on the manual labor of (mostly) female 'computers'. As Ensmenger describes, the early days of computing were "*unusually open to females ... [because of the] lack of a fully established scientific or engineering discipline identity left space open for women*" [36]. However, the computing labor these women performed became automated (resulting in the job loss) when electronic computers became commonplace. The addition of male programmers to the field of programming coincided with a shift in how the field saw itself, and programming became aligned with mathematics and with engineering since those fields had prestige. However, with that shift, existing stereotypes and hurdles (such as needing a formal degree) started to hamper women to the point that participation was reduced to the percentages around 10% that we see today [2, 36]

Hence, if we talk about the lack of representation of women in programming today, we should keep in mind that it was once, not even so long ago, seen as women's work. People used to believe that women are good at programming because they are fastidious, "*they worry how all the details fit together while still keeping the big picture in mind*" [22] and "*programming requires patience and persistence and these are traits that many girls (sic.) have*" [47]. In this regard, programming differs fundamentally from glaciology and other subfields in engineering and academia, in that programming started female and *became* male, while other fields were male from the outset and remained as such.

Modern feminist work in the computer science space aptly describes how the fact that CS is male-coded presents issues for women. Kronberger and Horwath find that women experience lower degrees of social acceptance and belonging in the academic environment [72]. Tassabehij describes that software development puts women in "*the ambivalent position of being either female or a coder, but not both*" [123]. Reading that work, I remembered my rejection of knitting, of earrings and of dresses, and I saw my own history through new eyes. This too is the power of feminism, it connects women through time and space and gives them a deeper understanding of themselves. In a field with so little women, it is hard to gain this understanding.

There are many goals for more feminism in PL: one reason is that I believe that allowing more room for different perspectives will make programming languages more usable and more interesting. But another reason is that when our field as a whole knows more about feminism, that will make it easier for women to fit in; I was never exposed to feminist or inclusion research before I deliberately went looking for it, and yet reading about the experiences of all those other women, in different times and places who were also struggling with belonging in a field that they love, made me understand a part of me I that was never able to before. By excluding feminist discourse from PL, we are withholding women (and other non-traditional PL creators) knowledge about themselves, and we are requiring them to do deep work to understand these dynamics.

4 Systems of Scientific Domination

This essay so far has described ways in which scientific practice in PL uses masculine discourses. The next part of Carey's framework explains that this is not a stable situation, but rather an ongoing phenomenon, and invisible powers—definitely not always intentional—are at play to ensure the system remains in its current state. Collins [26] separates the forces of domination into four forms: **Structural** how domination happens through formal structure, such as laws or regulation, and **disciplinary** domination in the enforcement of laws and regulations. There is also **Hegemonic** domination, on how what is included discourse shapes dominated alternatives, and finally **interpersonal** domination concerns how people as individuals reproduce systems of domination.

4.1 Structural

Structural domination in an academic field does not occur as formal laws, but rather as a set of norms—implicit and explicit agreements about the field which the field itself sets. However, these certainly aren't unwritten rules. For example, since 2017, PLDI mentions seven guidelines in their Call of Papers that authors are encouraged to use.[10] These seven

guidelines are very clearly underpinned by a quantitative way of thinking. For example, one of the seven guidelines reads 'appropriate metrics': meaning that we should measure relevant things. Ways of knowing that aren't numerical are hard to fit into this list; there are no guidelines about appropriate observations, or appropriate interviews. Another category is named 'appropriate comparison', detailing how authors should compare 'against an appropriate baseline'. This does not apply to, for example, papers using a theoretical lens to look at systems (such as this paper) or papers that use observations or interviews in which subjects discuss the differences between systems. That is not to say they can never be accepted, but reviewers then cannot use this checklist and are thus required to decide on their own on acceptable evidence.

POPL in their principles[11] describe that reviewers should focus on "*whether the approach is fundamentally sound*" and "*whether the paper contains sufficient information for others to reproduce and build on the results*". This strongly points in the direction of formal proofs and places several types of work as out of scope, including a lot of qualitative work that cannot be reproduced, since its goals aren't to be reproduced but to inform further thought.

4.2 Disciplinary

A critical reader might say, yes, all these guidelines exist, but it is merely an advice on what an author or reviewer should do, these guidelines do not prevent other types of work from being included. But the community also upholds the guidelines, representing the **disciplinary** domination. While indeed, papers can be accepted that deviate from the norm, the lists very clearly delineate the out-group from the in-group, and presenting qualitative work will be a lot harder than quantitative and formal work. One could also argue that these rules can be changed; they come from the community itself. However, the people who reach the seniority to be allowed in decision making are very likely to have done mainstream work, as we argued above.

4.3 Hegemonic

The **hegemonic** domination, or domination of discourse happens in everything that we produce: in papers, journals, conferences and anything that can be considered representative of the PL community. In all those expressions, a certain way of thinking is common, and other perspectives are less likely to be included.

The system stays in play, because people in the field (in any academic field) research not only what they love and what they think is valuable, but also what they know will advance their career. If you want to have a shot at a research career in PL, it is clear that you will need a PLDI or POPL paper every year, and a safe path towards that is to do types or performance work, and not work outside of the mainstream papers, which reviewers are less likely to accept and other researchers are less likely to cite.

4.4 Interpersonal

Finally, there are patterns of **interpersonal** domination. This should not be seen as the dynamic in which people speak over each other, or actively exclude some voices from the discussion. Often it is a lot more subtle but still causes people who fall outside of the current way of doing things to be either excluded or 'absorbed' into regular discourse.

One way in which this happens is by disallowing non-traditional PL researchers deep discussions, for example at conferences. A person researching a traditional PL topic such as dependent types is quite likely to be greeted at a conference with immediate deep questions about their work: 'Have you seen this paper?' or 'Do you know such and such?'. From their perspective, conferences are rife with opportunities to learn.

In contrast, those who are working outside of regular topics, as I have personally experienced both when working on spreadsheets and on non-English programming languages, are only presented with entry-level questions: 'Why is that interesting?' or 'Why is that hard?'–the latter question tying into the notion that hard is valuable and valuable is hard. Answering similar entry-level questions for years at end is not only boring and thus emotionally draining, but it also robs those researchers from learning more. Colleagues mentioning to me that attending a conference is valuable because they always come back with new perspectives has largely felt foreign to me: I was in so many cases the person *providing* the new perspectives and not receiving them.

A similar dynamic happens for those using non-traditional research methods: basic questions about qualitative research rob qualitative researchers of the experience of learning (if I never again get a question about small sample sizes, it'll be too soon) and cause PL conferences to be exhausting rather than fun. When you then also do not look like a traditional PL researcher, for instance if you are female, or non-binary, or if you are a researcher with disabilities, or if you are not white, you face one more distracting dynamic, and that is to be confronted with questions about *yourself*. It is not rare for me to have been asked in PL spaces where the women are, and what I personally thought of the reasons for the lack of women, or to be asked to give a lecture on gender, or have breakfast with female students. While I sometimes enjoy these activities—and while this paper shows that those conversations are necessary and can be surely valuable—participating in such conversations always comes at the opportunity costs of discussing research work. This type of diversity-based service work has shown to be often uncompensated [103], especially for women of color, their race and gender become their identity, whether they choose this or not [94]. None of this is new: the first female police officer in the US embarked on a national tour to promote female participation in policing, in addition to her

police work [88]. Such diversity work is not just an unfortunate consequence of being the first, but is often an ongoing process. As Morrison says "*Know the function, the very serious function of racism, is distraction. It keeps you from doing your work. It keeps you explaining, over and over again, your reason for being... None of that is necessary. There will always be one more thing.*"[12] Ahmed similarly notes that so much feminist and antiracist work is the work of trying to convince others that sexism and racism have not ended; which this essay, and my life in programming over the last two decades, are clear examples of [3].

These three interpersonal dynamics are all connected in intricate ways. Women are more likely to study non-traditional PL contexts, such as end-user programming education or widely used languages seen as less valuable, and are more likely to use non-traditional methods such as qualitative work centered more around people, meaning they will spend large parts of their professional lives educating others in these topics rather than learning.

5 Knowledge Production

Another aspect of Carey *et al.* is knowledge production: who is involved in gathering data and what data is being gathered? Carey *et al.* name examples of exclusionary data gathering around glaciers: more data is collected about hard-to-climb mountains, while data from women in agriculture, whose data is not an adventure, is collected to a lesser extent, distorting our views on glaciers. We apply this aspect of Carey to PL; whose stories do we collect, and what do those stories concern?

5.1 Whose Stories Are Collected?

Women have been involved in the creation of programming languages for decades. To name just a few: Kathleen Booth (Assembly-1949), Grace Hopper (FLOWMATIC-1955, COBOL-1959), Jean E. Sammet (FORMAC-1962), Cynthia Solomon (LOGO-1966), Adele Goldberg (Smalltalk-1972), Barbara Liskov (CLU-1973), Sophie Wilson (BBC BASIC-1981), Christine Paulin-Mohring (Coq-1989), Patricia Hill (Gödel-1992), Audrey Tang (Raku/PERL 6-2000), Crista Lopes (AspectJ-2001) and Heather Miller (Scala-2004). However, if we look at writing about programming languages, one might be inclined to believe that only men have created widely used languages. Someone looking at the proceedings of the ACM SIGPLAN conference on History of programming languages 2020 (HOPL IV) [1] for example: Only 4 female authors contributed, out of 46, who authored 2 out of 20 papers.[13] Or let's consider the 2009 book 'Masterminds of Programming'(MoP)[10], which interviews the creators of 17 different programming languages, all 27 of them male.

We see a familiar dynamic at work: men are labeled experts, while female contributions are erased from history. This phenomenon is common in the history of science and technology [114], also called the Matilda effect [106]. Because of this reframing, we come to believe that the field of programming is shaped by men, and that women simply don't have a place in it. This type of exclusion of women from technology is both a cause and a result of a gendered interpretation of what programming is and what contributions matter, a phenomenon that Cockburn calls the circuit of technology [120].

5.2 What Stories Are Collected?

There is a second question about the stories of programming as collected in books such as HOPL IV or MoP: why do they only collect stories on the *creation* of programming languages?

HOPL aims to collect "*contributions that discuss and analyze the historical development of individual programming languages, programming language families, language features, design themes, and other strong influences on the direction of programming language design, implementation, and usage*" but then presents 18 papers that discuss only the implementation of programming languages, one that discusses a more generic PL idea (macros) and only one that discusses what a programming language is (the only one authored by a woman alone). None of the papers explicitly aim to study the usage of a language. Where design is discussed, it is done from the eyes of a language creator. The authors of MoP explicitly choose to interview PL creators, but then labels them 'Masterminds of Programming', and not 'Masterminds of Programming Languages'.

Highlighting language creators reinforces the story that only building the programming language is where the meaningful work is. This singular focus means that we are missing other research directions around programming, such as the meaning or semiotics of programming by Tanaka-Ishii [122] or the history of programming in a broader sense by Sammet [107]. The missing perspectives are not limited to the perspectives of female authors, but also other broader views. For example, voices critical of technology, such as Lanier [78] are also missing.

The erasure of female contributions thus happens in two ways: directly, by failing to include female PL creators, and indirectly by shaping the perspective of what a mastermind of programming is, a language creator—not a user of language, an extender of language, a thinker about language, or a historian of such languages. We could instead also focus on the design team or the design process, or on the community of people impacted by the language rather than language creator.

5.3 How Do We Develop Languages?

Developing languages, variants and libraries is a large part of the knowledge production of PL. But how do we go about that? A common practice in technology, wider than PL, but

certainly also in PL is called 'dogfooding': meaning to use your own tools (in our case: a programming language) as much as possible, and to 'bootstrap' this, to be able to use it as early as possible. Prior research termed this *I methodology* [5, 100].

Such a methodology suffers from several limitations that are rarely addressed. Firstly there is the unspoken assumption that the original creator can take on the viewpoint of all potential users. This of course has the risk of becoming a self-fulfilling prophecy, surely people who think like the original creator will feel more at home in such a language or system. This again ties into the current community, which is homogeneous in gender, and also in race and abilities. As such, creators stemming from the PL community might make assumptions about what their users know, and what they care about that do not generalize to a wider population. Furthermore, a creator of a system will, by building it, have so much more understanding of a system than any other users (even a user similar in gender, race and ability) that it becomes harder, over time, to identify with novice or casual users, a form of the 'curse of expertise' [61].

5.4 What Counts As a Programming Language?

A final dynamic present in the PL community is gatekeeping about what even is, and isn't a programming language. An interesting example are spreadsheets [58]. While having a number of characteristics of a programming language, like allowing users to perform calculations with variables and conditionals, spreadsheets are excluded from the dominant discourses in PL, as I have experienced extensively when I worked on them early in my career. This can be understood through the lens of masculinity: many people can use spreadsheets. "*The Dutch Bureau of Statistics has reported a rise in people able to use formulas in spreadsheets from 44% in 2006 to 54% in 2013.*" [130], as reported in on [57]. This undercuts the belief that programming languages are hard to learn, and require special training or a mathematical way of thinking, it thus goes against the idea that programming is seen as hard, which is desirable.

Reflecting again on MoP shows how easy the goalposts can be moved; for some of the languages included it can be argued they are not a programming language, such as UML, AWK or SQL. UML is not executable, so could easily be disregarded as a programming language, but is not. HTML, which is executable however, is not seen as a programming language in regular PL discourse [46].

Even Python, at a certain point in time, was seen as a 'scripting language' and not a real programming language. My own experience supports the consistent moving of the 'real programming' goalposts. When I was working on spreadsheets, Excel being not Turing complete was an often-heard argument. My creation of a Turing machine in spreadsheet formulas however did not convince [38], but only led to 'yes but…'. A recent paper review I received (mid 2023) included the sentence "*I am uncomfortable with the definition of Excel as a programming language, despite many considering it as such.*" Observe how the discomfort of this reviewer overpowers both the evidence presented and as well as the opinion of others.

Overlooking spreadsheets as a programming system has an impact in the real world. A recent paper that investigated how to best teach programming to people in the prison system describes a struggle to install the right software to do so [62]. When at their conference presentation they were asked why they did not consider Excel (which in addition to formulas includes a VBA interpreter), which was available, the audience burst out in laughter.[14]

In addition to executable computer languages like spreadsheets being excluded from programming language discourse, so are other formal languages that are coded as female, including but not limited to patterns for knitting, crochet, weaving or sewing, and recipes. Not seen as formal languages they are excluded from study, even while having a history connected to programming; Jacquard looms were the original use case of punch cards [53], and early forms of computer memory were *woven* by hand (e.g., women hand-threaded wire rope "*through and around magnetic ferrite cores*" for NASA's Apollo Guidance Computer) [105].[15]

6 Alternative Representations

In the final aspect of the work of Carey *et al.*, they argue for inclusion of "*greater plurality in knowledge about and representations*". Our paper also aims to explore such diversification. Firstly, the expansions of the diversity of the people in the community through expanding the *methodology* by which programming languages are created, to attract people with different interests and backgrounds into the conversation. Secondly, exploring diversification of programming languages themselves, aimed at including more ways of programming, and attracting people that value those kinds of programming languages.

6.1 Methodological Diversity

This essay is by no means the first essay to address the lack of diversity in research methods in the PL community, other researchers have argued for different methods too, or have simply applied non-traditional methods and got their results published.

SocioPLT. One notable initiative was the SocioPLT research program of Leo Meyerovich *et al.* from Berkeley, who studied what factors lead to programming languages being adopted and found that these are often social and not technical factors [92]. Meyerovich's paper won best paper at OOPSLA 2013 and most influential paper in 2023, so to a certain extent, we can call this work successful; the PL community recognized that sociological work into understanding the human aspects of language adoption is important.

However, if we look at change in the field, the impact seems to be smaller; Meyerovich's paper only has two dozen citations, most of which fall outside of PL. In terms of changing the methods for papers, this paper seems not to have made a dent in getting review guidelines to accept more human-centric work (e.g. by explicitly helping reviewers to understand how to review such work). Ultimately Meyerovich decided not to pursue this line of research further, as he thought in the long term there would not be enough change in the community enabling the mainstream acceptance of such work.[16]

(Controlled) Experiments. A more explicit argument for different research methods came from Tichy [126], who called for more experimentation in computer science, he argued it is needed to get a better understanding of the field we study in addition to proofs and theories. Tichy points out that claims regarding the productivity and quality improvements of using functional or object-oriented programming have not been tested in systematic ways despite existing for 30 years. A related line of papers came from researchers Stefik and Hanenberg, more than a decade later at Onward 2010, as we discussed above. They argued that the programming languages community should focus not only on technical aspects of systems, but should also be studied through people's experiences with the languages, echoing the sentiments made by Tichy. Stefik and Hanenberg's work did not have much impact in the mainstream PL community; while in software engineering empirical work is relatively mainstream[17] in mainstream PL there is hardly more human-centric research done today than there was in 2010.[18]

User-centered Design. Some researchers have also argued for more user-centered design in PL, suggesting a large number of different research methods, including qualitative work, theory building and experimentation [24], and some of these authors have also reflected critically on how to perform user studies [30]. These suggestions are valuable and I think PL should take these to heart, however, it should be noted that [24], which is explicitly about designing *programming languages*, is published in a human-computer interaction venue, and not in PL. There are programming languages created outside of academic PL that explicitly use a user-centered design methodology. For example, Elm designed their error messages to be understandable, with the explicit goal of making them understandable to novices, arguing that people with the least access to mentorship, and those with low confidence are most likely to drop out because of confusing error messages, and encouraging the community to share their struggles with error messages so the language designers could learn from them [108].

6.2 Programming Language Diversity

In addition to people inside and outside of PL arguing for different methods for creating and evaluating languages, some researchers have also built languages serving different needs, or designed languages in more inclusive ways. These new ways of programming open the discussion of what is a programming language, and force us to deeply reflect on that.

Wordplay. A recent example of a programming language that is built to include non-traditional users is Amy J. Ko's Wordplay.[19] In the essay in which Ko describes the creation of Wordplay, she echoes feminist notions: "*In hindsight, I wondered how such a small, privileged group of Western, mostly white people managed to shape so much of computing, and through computing, so much of our modern world.*"[20] The result of Ko's exploration is a programming language in which we can manipulate emoji and letters, where a program is called a 'performance' catering to people with an interest in writing and text. It is a lovely exploration of programming that stretches the limits of the traditional definition of programming languages.

Non-English programming languages. One line of work which has been questioning who has the power to create programming languages—and hence fits perfectly in a feminist PL tradition—is work on non-English programming languages. A well-known example is قلب[21] an Arabic variant of Lisp.[22] Another example is WenYan in classical Chinese.[23] Research arguing for non-English programming languages in a more generic sense, without creating and discussing an implementation of a non-English programming language, has also been appearing, for example by Laiti [75]. Laiti argues that because currently keyboards and programs are lacking support for indigenous languages, we are missing valuable perspectives. It is again interesting to note that this work appears in a Computing Education venue. The case study they present is situated in education, but the point that PL misses these perspectives should also be discussed within the PL community. Something similar happened with I was working on Hedy [56], a programming language for teaching. I was in doubt of where to send the paper describing Hedy, and I chose to send it to a CSed venue, because I was afraid of reception in PL. Interestingly enough, the paper was almost desk rejected because the chair found it too PL centered! Hedy has recently also incorporated non-English programming [121], and the response of the PL community to that work has contributed to the creation of this

essay, because I saw the dynamics of pushback against my earlier work on spreadsheets, and against my interest in doing empirical, qualitative work, were all connected to an underlying belief system.

A final form of work examining the role of natural languages is the programming language Pegasus [71] which allows for 'naturalistic programming', i.e. programming using natural languages and supports both German and English.

Exploring programming in feminine coded formalisms. Activities associated strongly with women are typically not covered in the programming languages context, unless used as a way to make programming languages more relatable for the layperson, for example in [110] where in a recipe is used to demonstrate a formal language like a computer language.

By diversification of PL to include female coded formalisms into the PL discourse, we allow ourselves to learn from formalisms that now fall outside of our view, such as weaving. As Harlizius-Klück [53] summarizes in her excellent paper on the history of weaving (with and without machines), weaving is a form of binary computation: "*To control a weave means to decide whether a warp thread is to be picked up or not.*"

Feminine coded formalisms are not entirely out of scope, a recent Onward! paper has examined machine crochet [109], but it takes the reverse approach: it brings formalism coming from PL to crochet patterns, rather than learning from existing patterns. This view is common, other work on knitting also aims for proof properties of patterns, and points at aspects of patterns as having issues, i.e. "*existing representations for machine-knitted objects are incomplete (do not cover the complete domain of machine-knittable objects) or overly specific (do not account for symmetries and equivalences among knitting instruction sequences).*" It is interesting to see how the researchers take their norms about formalisms (they must be complete and not too specific) and push them on an existing world, rather than learning from the way the existing formalism currently works. This is a common, hegemonic form of reasoning, dismissing and rejecting knowledge made by people from different backgrounds [25, 37, 112]. As Buck *et al.*[15] note in their reflections on Gender and Geoengineering, traditional, masculine research in the natural sciences has a strong tendency to "*classify, measure, map, and, ideally, dominate, and control*". Such tendencies fit the PL world too, controlling an actual machine, without looking at a programming language in its context with users, libraries, mods, and human confusion, where it becomes more dynamic and chaotic and messy.

7 What Came Before; What Comes Next?

We are by no means the first people to discuss the history of women in programming languages, or to report on the effect of the 'mathematicalization' of our field on gender diversity, or even the impact that feminism could have on CS and PL. Diving into this topic, I was surprised how much there was to read: from books and papers detailing the history of women in our field [2, 36] to papers deeply engaging with feminism and computing such as [11, 23]. All of these are recommended for those who want to know more. If you want to gain a more basic understanding of feminism and technology you might want to start with the introduction of Gill and Grint [42] who explain the different forms of feminism and how they relate to technology and programming.

Seeing the amount of researchers who have explored feminism and programming in so many ways, including dozens of women I had never heard of, was as much comforting as it was painful. How is it that I can exist in CS for over two decades and never hear about any of this work; in my education, in conferences, in books and talks?

This paper has summarized research directions and ideas of what research to read if you want to know more, and all referenced papers can get you started in understanding feminism and programming, but the most important thing is where we go from here.

Our hope is that this essay gives space for others who want to broaden PL in any imaginable way: by widening the definitions of what a programming language is, including parts of programming currently populated by women, such as end-user programming, scientific computing and e-textiles, by allowing PL papers to study things that are not just the language itself, such as error messages, IDEs, or ways of thinking that programming languages encourage, and by allowing qualitative work seeking to understand, to challenge, to document and to exist alongside the formal and quantitative work we have seen in the past.

Here's to a future that is more inclusive of people of all interests, genders and races, bound by a love for programming and all that it brings us.

Notes

[1]Wherever the first person is used in this essay, the first author is writing from their personal experience. The second author's contributions were in shaping the paper by developing the argumentation, adding sources, and contextualizing the experiences of the first author.

[2]I now understand this is only one of the forms of feminism, called liberal feminism. For an overview of different forms, see Gill and Grint [42].

[3]Said another way, there is no single, essential standpoint for the category of women.

[4]It is customary to use all lowercase when writing about the scholar bell hooks as this is the capitalization style used by the author herself.

[5]Formally, numerals used in Arabic today are called Hindu–Arabic numerals. 0-9 as commonly used in English are, confusingly, referred to as Arabic numerals since they reached Europe through the Arabic world.

[6]https://www.tiobe.com/tiobe-index/

[7]Moreover, much research about gender differences tends to reinforce discriminatory stereotypes and assumptions, often suffering from confirmation bias and issues with study design, see [39, 40] for more.

[8]The term *women* often implicitly connotes the experiences of white women, excluding the experiences and needs of women of color. We use the term women of all colors to acknowledge this history of exclusion in activism and feminism, and to clarify that we are approaching this work from an inclusive and intersectional lens.

[9]https://portal.core.edu.au/conf-ranks/?search=4612&by=all&sort=arank

[10]https://www.sigplan.org/Resources/EmpiricalEvaluation/

[11]https://www.sigplan.org/Conferences/POPL/Principles/

[12]From a 1975 lecture at Portland State University called 'A Humanist View'

[13]It must be noted that it is somewhat remarkable that there even *is* a small subfield on the history of programming languages at all. SIGPLAN is the only SIG of ACM which has such a conference. There is no History of Operating Systems or History of Computer Architecture conference. Jean Sammet, the first chair of HOPL, devoted considerable effort to create a space where people could share stories of creation of languages, and earlier editions of HOPL considered the human side more.

[14]Personal correspondence with an attendee

[15]Racial and gender stereotypes have played a key role in the way this type of knowledge (and labor) is undervalued. For more information, we encourage you to explore the historic exploitation of Navajo women in early electronics manufacturing [95].

[16]Personal correspondence with the author.

[17]Although not all of empirical SE is experiments, a lot of it is large-scale data analysis

[18]Again, we acknowledge the work done in workshops and smaller conference on human-centric PL work, but we also note that these are not comparable to the career building work published in A* conferences workshops.

[19]https://wordplay.dev/learn

[20]https://medium.com/bits-and-behavior/wordplay-accessible-language-inclusive-interactive-typography-e4b9027eaf10

[21]Pronounced as 'elb' or 'qelb' and meaning heart in Arabic, see https://www.theregister.com/2013/01/25/arabic_programming_language

[22]It took the authors of this paper considerable effort and knowledge of the Arabic alphabet to properly typeset قلب in Latex, a case in point. This is also the reason we use endnotes rather than footnotes; footnotes are not compatible with arabtex.

[23]https://wy-lang.org/

Acknowledgments

Thanks to Leo Meyerovich and Walter Tichy for their input on their work on diverse research methods and its impact. Thanks to Neil C.C. Brown for his words of encouragement and extensive comments on several drafts. And finally, to Andreas Stefik and Stefan Hanenberg: words do not suffice to cover your impact on my thinking. You are the academic friends I needed to feel welcome in PL, and without you, I would not be here today to write all of this.

References

[1] 2020. HOPL IV. *Proc. ACM Program. Lang.* 4, HOPL (2020). Publisher: Association for Computing Machinery.

[2] Janet Abbate. 2012. *Recoding Gender: Women's Changing Participation in Computing*. The MIT Press. https://www.jstor.org/stable/j.ctt5vjp2p

[3] Sara Ahmed. 2017. *Living a Feminist Life*. Duke University Press.

[4] Efthimia Aivaloglou and Felienne Hermans. 2019. How is programming taught in code clubs? Exploring the experiences and gender perceptions of code club teachers. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. 1–10.

[5] Madeleine Akrich. 1995. User Representations: Practices, Methods and Sociology. In *Managing Technology in Society. The Approach of Constructive Technology Assessment*. Pinter, 167. https://shs.hal.science/halshs-00081749

[6] Christine Alvarado and Zachary Dodds. 2010. Women in CS: An Evaluation of Three Promising Practices. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE10)*. ACM. https://doi.org/10.1145/1734263.1734281

[7] Sonya Andermahr, Terry Lovell, and Carol Wolkowitz. 1997. *A Concise Glossary of Feminist Theory* (1st edition ed.). Hodder Education Publishers, London.

[8] Simone De Beauvoir. 1949. *The Second Sex*. Knopf Doubleday Publishing Group. Google-Books-ID: _hywlrNuYvIC.

[9] Joanne Rossi Becker. 1981. Differential treatment of females and males in mathematics classes. *Journal for research in Mathematics Education* 12, 1 (1981), 40–53. Publisher: National Council of Teachers of Mathematics.

[10] Federico Biancuzzi and Shane Warden. 2009. *Masterminds of Programming: Conversations with the Creators of Major Programming Languages*. Theory in Practice (O'Reilly). https://www.amazon.com/Masterminds-Programming-Conversations-Creators-Languages/dp/0596515170

[11] Christina Björkman. 2005. Feminist research and computer science: Starting a dialogue. *Journal of Information, Communication and Ethics in Society* 3 (Nov. 2005), 179–188. https://doi.org/10.1108/14779960580000271

[12] Lenore Blum and Carol Frieze. 2005. The Evolving Culture of Computing: Similarity Is the Difference. *Frontiers: A Journal of Women Studies* 26, 1 (2005), 110–125. https://doi.org/10.1353/fro.2005.0002

[13] Lenore Blum, Carol Frieze, Orit Hazzan, and M. Bernardine Dias. 2007. A Cultural Perspective on Gender Diversity in Computing. *Reconfiguring the firewall. Recruiting women to information technology across cultures and continents* (2007), 109–133.

[14] Kevin S. Bonham and Melanie I. Stefan. 2017. Women are underrepresented in computational biology: An analysis of the scholarly literature in biology, computer science and computational biology. *PLOS Computational Biology* 13, 10 (Oct. 2017), e1005134. https://doi.org/10.1371/journal.pcbi.1005134 Publisher: Public Library of Science.

[15] Holly Jean Buck, Andrea R. Gammon, and Christopher J. Preston. 2014. Gender and Geoengineering. *Hypatia* 29, 3 (2014), 651–669. https://doi.org/10.1111/hypa.12083 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/hypa.12083.

[16] Leah Buechley and Benjamin Mako Hill. 2010. LilyPad in the wild: how hardware's long tail is supporting new engineering and design communities. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems (DIS '10)*. Association for Computing Machinery, New York, NY, USA, 199–207. https://doi.org/10.1145/1858171.1858206

[17] Tracy Camp. 2012. 'Computing, We Have a Problem...'. *ACM Inroads* 3, 4 (Dec. 2012), 34–40. https://doi.org/10.1145/2381083.2381097

[18] Mark Carey, M. Jackson, Alessandro Antonello, and Jaclyn Rushing. 2016. Glaciers, gender, and science: A feminist glaciology framework for global environmental change research. *Progress in Human Geography* 40, 6 (Dec. 2016), 770–793. https://doi.org/10.1177/0309132515623368 Publisher: SAGE Publications Ltd.

[19] Mark Carey, M. Jackson, Alessandro Antonello, and Jaclyn Rushing. 2016. Glaciers, Gender, and Science: A Feminist Glaciology Framework for Global Environmental Change Research. *Progress in Human Geography* 40, 6 (July 2016), 770–793. https://doi.org/10.1177/0309132515623368

[20] P. L. Chinn and C. E. Wheeler. 1985. Feminism and nursing. *Nursing Outlook* 33, 2 (1985), 74–77.

[21] Ingrid Maria Christensen, Melissa Høegh Marcher, Paweł Grabarczyk, Therese Graversen, and Claus Brabrand. 2021. Computing Educational Activities Involving People Rather Than Things Appeal More to Women (Recruitment Perspective). In *Proceedings of the 17th ACM Conference on International Computing Education Research (ICER 2021)*. Association for Computing Machinery, New York, NY, USA, 127–144. https://doi.org/10.1145/3446871.3469758

[22] Wendy Hui Kyong Chun. 2011. *Programmed Visions: Software and Memory*. The MIT Press. https://doi.org/10.7551/mitpress/9780262015424.001.0001

[23] Sue Clegg. 2001. Theorizing the Machine: Gender, Education and Computing. *Gender and Education* 13, 3 (2001), 307–24. ERIC Number: EJ634060.

[24] Michael Coblenz, Gauri Kambhatla, Paulette Koronkevich, Jenna L. Wise, Celeste Barnaby, Joshua Sunshine, Jonathan Aldrich, and Brad A. Myers. 2021. PLIERS: A Process that Integrates User-Centered Methods into Programming Language Design. *ACM Transactions on Computer-Human Interaction* 28, 4 (Aug. 2021), 1–53. https://doi.org/10.1145/3452379

[25] Lorraine Code. 1991. *What Can She Know?: Feminist Theory and the Construction of Knowledge*. Cornell University Press, Ithaca.

[26] Patricia Hill Collins. 2000. *Black Feminist Thought: Knowledge, Consciousness, and the Politics of Empowerment* (second ed.). Routledge, New York.

[27] Patricia Hill Collins. 2019. *Intersectionality As Critical Social Theory*. Duke University Press. 376 pages.

[28] Kimberlé Crenshaw. 1989. Demarginalizing the Intersection of Race and Sex: A Black Feminist Critique of Antidiscrimination Doctrine, Feminist Theory and Antiracist Politics. *University of Chicago Legal Forum* (1989), 139–167.

[29] Kelly Danaher and Christian S. Crandall. 2008. Stereotype Threat in Applied Settings Re-Examined. *Journal of Applied Social Psychology* 38, 6 (May 2008), 1639–1655. https://doi.org/10.1111/j.1559-1816.2008.00362.x

[30] Matthew C. Davis, Emad Aghayi, Thomas D. Latoza, Xiaoyin Wang, Brad A. Myers, and Joshua Sunshine. 2023. What's (Not) Working in Programmer User Studies? *ACM Transactions on Software Engineering and Methodology* 32, 5 (Sept. 2023), 1–32. https://doi.org/10.1145/3587157

[31] Edsger W. Dijkstra. 1973. Programming as a discipline of mathematical nature (EWD 361). https://www.cs.utexas.edu/~EWD/transcriptions/EWD03xx/EWD361.html

[32] Edsger W. Dijkstra. 1975. How do we tell truths that might hurt? (EWD498). https://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD498.html

[33] Sonya Corbin Dwyer and Jennifer L. Buckle. 2009. The Space between: On Being an Insider-Outsider in Qualitative Research. *International Journal of Qualitative Methods* 8, 1 (March 2009), 54–63. https://doi.org/10.1177/160940690900800105

[34] Paul N. Edwards. 1990. The Army and the Microworld: Computers and the Politics of Gender Identity. *Signs* 16, 1 (1990), 102–127. https://www.jstor.org/stable/3174609 Publisher: University of Chicago Press.

[35] Nathan Ensmenger. 2010. *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*. The MIT Press, Cambridge, Mass.

[36] Nathan Ensmenger. 2010. Making Programming Masculine. In *Gender Codes*. John Wiley & Sons, Ltd, 115–141. https://doi.org/10.1002/9780470619926.ch6 Section: 6 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470619926.ch6.

[37] Arturo Escobar, Carlos Frederico Marés de Souza Filho, João Arriscado Nunes, João Paulo Borges Coelho, Laymert Garcia dos Santos, Lino João de Oliveira Neves, Luis Carlos Arenas, Margarita Flórez Alonso, Maria Paula Meneses, Mauricio Pardo, et al. 2020. *Another Knowledge Is Possible: Beyond Northern Epistemologies*. Verso Books.

[38] felienne. 2013. Excel Turing Machine. https://www.felienne.com/archives/2974

[39] Cordelia Fine. 2005. *Delusions of Gender*. Icon Books, New York. Description based upon print version of record.

[40] Cordelia Fine. 2017. *Testosterone Rex*. W. W. Norton & Company. Description based on publisher supplied metadata and other sources..

[41] Allan Fisher and Jane Margolis. 2002. Unlocking the Clubhouse: The Carnegie Mellon Experience. *SIGCSE Bulletin* 34, 2 (2002), 79–83.

[42] Rosalind Gill and Keith Grint. 1995. *The Gender-Technology Relation: Contemporary Theory And Research: An Introduction*. Routledge. https://www.routledge.com/The-Gender-Technology-Relation-Contemporary-Theory-And-Research-An-Introduction/Gill-Grint/p/book/9780748401611

[43] Marleen Gilsing and Felienne Hermans. 2021. Gradual Programming in Hedy: A First User Study. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–9. https://doi.org/10.1109/VL/HCC51201.2021.9576236 ISSN: 1943-6106.

[44] Marleen Gilsing, Jesús Pelay, and Felienne Hermans. 2022. Design, implementation and evaluation of the Hedy programming language. *Journal of Computer Languages* 73, 101158 (Dec. 2022), 1–17. https://doi.org/10.1016/j.cola.2022.101158

[45] Catherine Good, Joshua Aronson, and Michael Inzlicht. 2003. Improving Adolescents' Standardized Test Performance: An Intervention to Reduce the Effects of Stereotype Threat. *Journal of Applied Developmental Psychology* 24, 6 (Dec. 2003), 645–662. https://doi.org/10.1016/j.appdev.2003.09.002

[46] Olivia Guest and Samuel H. Forbes. 2023. Teaching coding inclusively: if this, then what? (Sept. 2023). https://doi.org/10.31235/osf.io/3r2ez Publisher: OSF.

[47] Denise Gürer. 2002. Women in computing history. *ACM SIGCSE Bulletin* 34, 2 (June 2002), 116–120. https://doi.org/10.1145/543812.543843

[48] Jonas Hallström, Helene Elvstrand, and Kristina Hellberg. 2015. Gender and technology in free play in Swedish early childhood education. *International Journal of Technology and Design Education* 25, 2 (May 2015), 137–149. https://doi.org/10.1007/s10798-014-9274-z

[49] Fannie Lou Hamer. 2010. "Nobody's Free Until Everybody's Free,": Speech Delivered at the Founding of the National Women's Political Caucus, Washington, D.C., July 10, 1971. University Press of Mississippi, 0. https://doi.org/10.14325/mississippi/9781604738223.003.0017

[50] Donna Haraway. 1988. Situated Knowledges: The Science Question in Feminism and the Privilege of Partial Perspective. *Feminist Studies* 14, 3 (1988), 575–599. https://doi.org/10.2307/3178066

[51] Sandra Harding. 1986. *The Science Question in Feminism*. Cornell University Press, Ithica. https://doi.org/10.1080/00201748708602126

[52] Sandra G. Harding. 2004. *The Feminist Standpoint Theory Reader: Intellectual and Political Controversies*. Psychology Press. Google-Books-ID: qmSySHvIy5IC.

[53] Ellen Harlizius-Klück. 2017. Weaving as Binary Art and the Algebra of Patterns. *TEXTILE* 15 (April 2017), 176–197. https://doi.org/10.1080/14759756.2017.1298239

[54] Nancy C. M. Hartsock. 1983. The Feminist Standpoint: Developing the Ground for a Specifically Feminist Historical Materialism. In *Discovering Reality: Feminist Perspectives on Epistemology, Metaphysics, Methodology, and Philosophy of Science*, Sandra Harding and Merrill B. Hintikka (Eds.). Springer Netherlands, Dordrecht, 283–310. https://doi.org/10.1007/0-306-48017-4_15

[55] Felienne Hermans. 2013. *Analyzing and visualizing spreadsheets*. PhD Thesis. Delft University of Technology.

[56] Felienne Hermans. 2020. Hedy: A Gradual Language for Programming Education. In *Proceedings of the 2020 ACM Conference on International Computing Education Research (ICER '20)*. Association

for Computing Machinery, New York, NY, USA, 259–270. https://doi.org/10.1145/3372782.3406262 event-place: Virtual Event, New Zealand.

[57] Felienne Hermans and Efthimia Aivaloglou. 2016. Do code smells hamper novice programming? A controlled experiment on Scratch programs. In *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*. IEEE, 1–10.

[58] Felienne Hermans, Bas Jansen, Sohon Roy, Efthimia Aivaloglou, Alaaeddin Swidan, and David Hoepelman. 2016. Spreadsheets are code: An overview of software engineering approaches applied to spreadsheets. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 5. IEEE, 56–65.

[59] Catherine Hill, Christianne Corbett, and Andresse St. Rose. 2010. *Why So Few? : Women in Science, Technology, Engineering, and Mathematics*. American Association of University Women (AAUW), Washington, D.C.

[60] Mark Hills, Paul Klint, and Jurgen Vinju. 2013. An empirical study of PHP feature usage: a static analysis perspective. In *Proceedings of the 2013 International Symposium on Software Testing and Analysis (ISSTA 2013)*. Association for Computing Machinery, New York, NY, USA, 325–335. https://doi.org/10.1145/2483760.2483786

[61] Pamela J. Hinds. 1999. The curse of expertise: The effects of expertise and debiasing methods on prediction of novice performance. *Journal of Experimental Psychology: Applied* 5, 2 (1999), 205–221. https://doi.org/10.1037/1076-898X.5.2.205 Place: US Publisher: American Psychological Association.

[62] Emma Hogan, Ruoxuan Li, Adalbert Gerald Soosai Raj, William G. Griswold, and Leo Porter. 2024. Challenges and Approaches to Teaching CS1 in Prison. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 512–518. https://doi.org/10.1145/3626252.3630802

[63] bell hooks. 1984. *Feminist Theory: From Margin to Center*. South End Press, Cambridge, MA.

[64] bell hooks. 2000. *Feminism Is for Everybody: Passionate Politics*. South End Press. 120 pages.

[65] bell hooks. 2005. *The Will to Change* (1 ed.). Washington Square Press, New York. Hier auch später erschienene, unveränderte Nachdrucke.

[66] Brittany Johnson, Yoonki Song, Emerson Murphy-Hill, and Robert Bowdidge. 2013. Why don't software developers use static analysis tools to find bugs?. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*. IEEE Press, San Francisco, CA, USA, 672–681.

[67] Lee Jussim and Jacquelynne S Eccles. 1992. Teacher expectations: II. Construction and reflection of student achievement. *Journal of personality and social psychology* 63, 6 (1992), 947. Publisher: American Psychological Association.

[68] Antti-Juhani Kaijanaho. 2015. Evidence-based programming language design : a philosophical and methodological exploration. *Jyväskylä studies in computing* 222 (2015). https://jyx.jyu.fi/handle/123456789/47698 Accepted: 2015-11-17T10:33:20Z ISBN: 9789513963880 Publisher: University of Jyväskylä.

[69] Anne Karpf. 1987. Work ; Gender relations in the construction of jobs. In *Gender and expertise*. Free Association Books, London.

[70] Robin Wall Kimmerer. 2013. *Braiding Sweetgrass*. Milkweed Editions.

[71] Roman Knöll and Mira Mezini. 2006. Pegasus: first steps toward a naturalistic programming language. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications (OOPSLA '06)*. Association for Computing Machinery, New York, NY, USA, 542–559. https://doi.org/10.1145/1176617.1176628

[72] Nicole Kronberger and Ilona Horwath. 2013. The ironic costs of performing well: Grades differentially predict male and female dropout from engineering. *Basic and Applied Social Psychology* 35, 6 (2013), 534–546. https://doi.org/10.1080/01973533.2013.840629 Place: United Kingdom Publisher: Taylor & Francis.

[73] Filip Křikava, Heather Miller, and Jan Vitek. 2019. Scala implicits are everywhere: a large-scale study of the use of Scala implicits in the wild. *Proceedings of the ACM on Programming Languages* 3, OOPSLA (Oct. 2019), 1–28. https://doi.org/10.1145/3360589

[74] Nicholas Laberge, K. Hunter Wapman, Allison C. Morgan, Sam Zhang, Daniel B. Larremore, and Aaron Clauset. 2022. Subfield prestige and gender inequality among U.S. computing faculty. *Commun. ACM* 65, 12 (Nov. 2022), 46–55. https://doi.org/10.1145/3535510

[75] Outi Laiti. 2016. The ethnoprogramming model. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research (Koli Calling '16)*. Association for Computing Machinery, New York, NY, USA, 150–154. https://doi.org/10.1145/2999541.2999545

[76] Wing Lam, Stefan Winter, Anjiang Wei, Tao Xie, Darko Marinov, and Jonathan Bell. 2020. A large-scale longitudinal study of flaky tests. *Proceedings of the ACM on Programming Languages* 4, OOPSLA (Nov. 2020), 202:1–202:29. https://doi.org/10.1145/3428270

[77] Ruth A. Lamont, Hannah J. Swift, and Dominic Abrams. 2015. A Review and Meta-Analysis of Age-Based Stereotype Threat: Negative Stereotypes, Not Facts, Do the Damage. *Psychology and Aging* 30, 1 (March 2015), 180–193. https://doi.org/10.1037/a0038586

[78] Jaron Lanier. 2011. *You Are Not a Gadget: A Manifesto* (reprint edition ed.). Vintage, New York.

[79] Sarah-Jane Leslie, Andrei Cimpian, Meredith Meyer, and Edward Freeland. 2015. Expectations of Brilliance Underlie Gender Distributions across Academic Disciplines. *Science* 347, 6219 (Jan. 2015), 262–265. https://doi.org/10.1126/science.1261375

[80] Max Liboiron. 2021. *Pollution Is Colonialism*. Duke University Press. 224 pages.

[81] Songqi Liu, Pei Liu, Mo Wang, and Baoshan Zhang. 2021. Effectiveness of Stereotype Threat Interventions: A Meta-Analytic Review. *Journal of Applied Psychology* 106, 6 (June 2021), 921–949. https://doi.org/10.1037/apl0000770

[82] Helen E. Longino. 1993. Feminist Standpoint Theory and the Problems of Knowledge. *Signs: Journal of Women in Culture and Society* 19, 1 (Oct. 1993), 201–212. https://doi.org/10.1086/494867

[83] Helen E. Longino. 2001. *The Fate of Knowledge*. Princeton University Press.

[84] Karen Mahony and Brett Van Toen. 1990. Mathematical Formalism As a Means of Occupational Closure in Computing — Why 'Hard' Computing Tends to Exclude Women. *Gender and Education* 2, 3 (Jan. 1990), 319–331. https://doi.org/10.1080/0954025900020306

[85] Melissa Høegh Marcher, Ingrid Maria Christensen, Paweł Grabarczyk, Therese Graversen, and Claus Brabrand. 2021. Computing Educational Activities Involving People Rather Than Things Appeal More to Women (CS1 Appeal Perspective). In *Proceedings of the 17th ACM Conference on International Computing Education Research*. ACM, Virtual Event USA, 145–156. https://doi.org/10.1145/3446871.3469761

[86] Jane Margolis and Allan Fisher. 2003. *Unlocking the Clubhouse: Women in Computing*. The MIT Press.

[87] Allison Master, Andrew N. Meltzoff, and Sapna Cheryan. 2021. Gender Stereotypes about Interests Start Early and Cause Gender Disparities in Computer Science and Engineering. *Proceedings of the National Academy of Sciences* 118, 48 (Nov. 2021). https://doi.org/10.1073/pnas.2100030118

[88] Nathan Masters. 2019. Pillars of Fire. https://medium.com/truly-adventurous/pillars-of-fire-d442e8b8e9d

[89] Sandra Mattauch, Katja Lohmann, Frank Hannig, Daniel Lohmann, and Jürgen Teich. 2020. A bibliometric approach for detecting the gender gap in computer science. *Commun. ACM* 63, 5 (April 2020), 74–80. https://doi.org/10.1145/3376901

[90] Davood Mazinanian and Nikolaos Tsantalis. 2016. An Empirical Study on the Use of CSS Preprocessors. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. 168–178. https://doi.org/10.1109/SANER.2016.18

[91] Noëlle McAfee, Ann Garry, Anita Superson, Heidi Grasswick, and Serene Khader. 2024. Feminist Philosophy. In *The Stanford Encyclopedia of Philosophy* (Spring 2024 ed.), Edward N. Zalta and Uri Nodelman (Eds.). Metaphysics Research Lab, Stanford University.

[92] Leo A. Meyerovich and Ariel S. Rabkin. 2013. Empirical analysis of programming language adoption. In *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications (OOPSLA '13)*. Association for Computing Machinery, New York, NY, USA, 1–18. https://doi.org/10.1145/2509136.2509515

[93] Chandra Talpade Mohanty. 2003. *Feminism without Borders: Decolonizing Theory, Practicing Solidarity*. Duke University Press. 300 pages.

[94] Sheila Nair. 2014. Women of Color Faculty and the "Burden" of Diversity. *International Feminist Journal of Politics* 16, 3 (July 2014), 497–500. https://doi.org/10.1080/14616742.2014.929357 Publisher: Routledge _eprint: https://doi.org/10.1080/14616742.2014.929357.

[95] Lisa Nakamura. 2014. Indigenous Circuits: Navajo Women and the Racialization of Early Electronic Manufacture. *American Quarterly* 66, 4 (2014), 919–941. https://doi.org/10.1353/aq.2014.0070

[96] Jennifer C. Nash. 2019. *Black Feminism Reimagined: After Intersectionality*. Duke University Press. 184 pages.

[97] Hannah-Hanh D. Nguyen and Ann Marie Ryan. 2008. Does Stereotype Threat Affect Test Performance of Minorities and Women? A Meta-Analysis of Experimental Evidence. *Journal of Applied Psychology* 93, 6 (2008), 1314–1334. https://doi.org/10.1037/a0012702

[98] Vidushi Ojha, Leah West, and Colleen M. Lewis. 2024. Computing Self-Efficacy in Undergraduate Students: A Multi-Institutional and Intersectional Analysis. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. ACM. https://doi.org/10.1145/3626252.3630811

[99] Aleksandra Orlowska, Christos Chrysoulas, Zakwan Jaroucheh, and Xiaodong Liu. 2021. Programming Languages: A Usage-based Statistical Analysis and Visualization. In *Proceedings of the 4th International Conference on Information Science and Systems (ICISS '21)*. Association for Computing Machinery, New York, NY, USA, 143–148. https://doi.org/10.1145/3459955.3460614

[100] Nelly Oudshoorn, Els Rommes, and Marcelle Stienstra. 2004. Configuring the User as Everybody: Gender and Design Cultures in Information and Communication Technologies. *Science, Technology, & Human Values* 29, 1 (2004), 30–63. https://www.jstor.org/stable/1558005 Publisher: Sage Publications, Inc..

[101] Charlotte R. Pennington, Derek Heim, Andrew R. Levy, and Derek T. Larkin. 2016. Twenty Years of Stereotype Threat Research: A Review of Psychological Mediators. *PLOS ONE* 11, 1 (Jan. 2016), e0146487. https://doi.org/10.1371/journal.pone.0146487

[102] Katherine Picho, Ariel Rodriguez, and Lauren Finnie. 2013. Exploring the Moderating Role of Context on the Mathematics Performance of Females under Stereotype Threat: A Meta-Analysis. *The Journal of Social Psychology* 153, 3 (May 2013), 299–333. https://doi.org/10.1080/00224545.2012.737380

[103] Kamaria B. Porter, Julie R. Posselt, Kimberly Reyes, Kelly E. Slay, and Aurora Kamimura. 2018. Burdens and benefits of diversity work: emotion management in STEM doctoral students. *Studies in Graduate and Postdoctoral Education* 9, 2 (Jan. 2018), 127–143. https://doi.org/10.1108/SGPE-D-17-00041 Publisher: Emerald Publishing Limited.

[104] Yolanda A. Rankin and Jakita O. Thomas. 2020. The Intersectional Experiences of Black Women in Computing. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 199–205. https://doi.org/10.1145/3328778.3366873

[105] Daniela K. Rosner, Samantha Shorey, Brock R. Craft, and Helen Remick. 2018. Making Core Memory: Design Inquiry into Gendered Legacies of Engineering and Craftwork. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174105

[106] Margaret W. Rossiter. 1993. The Matthew Matilda Effect in Science. *Social Studies of Science* 23, 2 (1993), 325–341. http://www.jstor.org/stable/285482 Publisher: Sage Publications, Ltd..

[107] Jean E. Sammet. 1969. *Programming Languages: History and Fundamentals* (first edition ed.). Prentice Hall.

[108] Ari Schlesinger. 2023. *Addressing Computing's Discrimination Problem: A Framework for Anti-Discriminatory Computing*. GeorgiaTech. https://grad.gatech.edu/events/phd-defense-ari-schlesinger

[109] Klara Seitz, Patrick Rein, Jens Lincke, and Robert Hirschfeld. 2022. Digital Crochet: Toward a Visual Language for Pattern Description. In *Proceedings of the 2022 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2022)*. Association for Computing Machinery, New York, NY, USA, 48–62. https://doi.org/10.1145/3563835.3567657

[110] John Shore. 1986. *The Sachertorte Algorithm and Other Antidotes to Computer Anxiety*. Penguin Books. Google-Books-ID: CcjaAAAAMAAJ.

[111] Del Siegle and Sally M Reis. 1998. Gender differences in teacher and student perceptions of gifted students' ability and effort. *Gifted Child Quarterly* 42, 1 (1998), 39–47. Publisher: Sage Publications Sage CA: Thousand Oaks, CA.

[112] Rebecca Solnit. 2014. *Men Explain Things to Me*. Haymarket Books.

[113] Steven J. Spencer, Claude M. Steele, and Diane M. Quinn. 1999. Stereotype Threat and Women's Math Performance. *Journal of Experimental Social Psychology* 35, 1 (Jan. 1999), 4–28. https://doi.org/10.1006/jesp.1998.1373

[114] Autumn Stanley. 1995. *Mothers and Daughters of Invention: Notes for a Revised History of Technology*. Rutgers University Press. 708 pages.

[115] Andreas Stefik, Willliam Allee, Gabriel Contreras, Timothy Kluthe, Alex Hoffman, Brianna Blaser, and Richard Ladner. 2024. Accessible to Whom? Bringing Accessibility to Blocks. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. ACM. https://doi.org/10.1145/3626252.3630770

[116] Andreas Stefik and Stefan Hanenberg. 2014. The Programming Language Wars: Questions and Responsibilities for the Programming Language Community. In *Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software (Onward! 2014)*. Association for Computing Machinery, New York, NY, USA, 283–299. https://doi.org/10.1145/2661136.2661156

[117] Andreas Stefik and Stefan Hanenberg. 2017. Methodological Irregularities in Programming-Language Research. *Computer* 50, 8 (2017), 60–63. https://doi.org/10.1109/MC.2017.3001257 Conference Name: Computer.

[118] Andreas Stefik and Richard Ladner. 2017. The Quorum Programming Language (Abstract Only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 641–641. https://doi.org/10.1145/3017680.3022377 event-place: Seattle, Washington, USA.

[119] Andreas Stefik and Susanna Siebert. 2013. An Empirical Investigation into Programming Language Syntax. *Trans. Comput. Educ.* 13, 4 (Nov. 2013), 19:1–19:40. https://doi.org/10.1145/2534973

[120] Marilyn Strathern and Cynthia Cockburn. 1992. The circuit of technology Gender, identity and power, Roger Silverstone and Eric Hirsch (Eds.). Taylor & Francis, Abingdon, UK, 32–47. https://doi.org/10.4324/9780203401491_chapter_2 Book Title: Consuming Technologies.

[121] Alaaeddin Swidan and Felienne Hermans. 2023. A Framework for the Localization of Programming Languages. In *Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E (SPLASH-E '23)*. ACM. https://doi.org/10.1145/3622780.3623645

[122] Kumiko Tanaka-Ishii. 2010. *Semiotics of Programming* (1st ed.). Cambridge University Press, USA.

[123] Rana Tassabehji, Nancy Harding, Hugh Lee, and Carine Dominguez-Pery. 2021. From female computers to male comput♂rs: Or why there are so few women writing algorithms and developing software. *Human Relations* 74, 8 (Aug. 2021), 1296–1326. https://doi.org/10.1177/0018726720914723 Publisher: SAGE Publications Ltd.

[124] Joy Teague. 2002. Women in Computing: What Brings Them to It, What Keeps Them in It? *ACM SIGCSE Bulletin* 34, 2 (June 2002), 147–158. https://doi.org/10.1145/543812.543849

[125] Matti Tedre. 2014. *The Science of Computing*. CRC Press.

[126] W.F. Tichy. 1998. Should computer scientists experiment more? *Computer* 31, 5 (May 1998), 32–40. https://doi.org/10.1109/2.675631 Conference Name: Computer.

[127] Sherry Turkle. 1988. Computational reticence: why women fear the intimate machine. In *Technology and Women's Voices*. Routledge. Num Pages: 17.

[128] Ruth van Veelen, Belle Derks, and Maaike Dorine Endedijk. 2019. Double Trouble: How Being Outnumbered and Negatively Stereotyped Threatens Career Outcomes of Women in STEM. *Frontiers in Psychology* 10 (2019).

[129] Hannah VanLandingham, Rachael L. Ellison, Aamir Laique, Andrea Cladek, Humza Khan, Christopher Gonzalez, and Megan R. Dunn. 2021. A Scoping Review of Stereotype Threat for BIPOC: Cognitive Effects and Intervention Strategies for the Field of Neuropsychology. *The Clinical Neuropsychologist* 36, 2 (July 2021), 503–522. https://doi.org/10.1080/13854046.2021.1947388

[130] Centraal Bureau voor de Statistiek Nederland. [n. d.]. StatLine - WO voltijd; rendement en uitval, 1995 - 2005. https://opendata.cbs.nl/statline/#/CBS/nl/dataset/71063ned/table?fromstatweb

[131] Judy Wajcman. 1991. *Feminism Confronts Technology*. The Pennsylvania State University Press, University Park, PA.

[132] American Association of University Women. 1991. *Shortchanging girls, shortchanging America: A call to action*. Vol. 4792. American Association of University Women.

[133] Yi Yang, Ana Milanova, and Martin Hirzel. 2022. Complex Python features in the wild. In *Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22)*. Association for Computing Machinery, New York, NY, USA, 282–293. https://doi.org/10.1145/3524842.3528467