Computing can be magical
I fell in love with this magic early in life

As a closeted trans teen, code was my escape

I programmed virtual worlds where I could flee my body and avoid my gender plight

I used the internet to learn about a world I couldn’t see
Captivated, I’ve spent the past 20 years lowering barriers to programming.

I studied why programming is hard, and how to make it easier.

I invented new ways of making, breaking, and fixing software.
All of this followed from two particular notions of justice...

1. Computing should be useful and **usable** to everyone
2. Everyone should be **empowered** to harness computing
Then I started reading...

Over the past five years, I learned how software is being used to oppress, marginalize voices, erode discourse, dissolve safety nets, surveil communities, shrink the middle class, and encode anti-Black racism.

Code, my lifelong professional and personal interest, was both a tool of empowerment and injustice.
My research **amplified** algorithmic bias

My research on **debugging tools** made software faster and cheaper to make, helping developers write biased algorithms faster and more correctly than ever before.

Faster fixes, more disruption (Ko and Myers, 2008).
My research *centralized* and privatized power

My inventions largely served powerful platforms owned by Amazon, Google, Facebook, Microsoft, and Apple, disempowering smaller organizations and the public.

Microsoft replicated our developer tool ideas and restructured teams based on my productivity research increasing its power.

Google built upon our research on ML development, streamlining data debugging.

Apple replicated our learning technologies to attract learners to its walled garden through Swift Playgrounds.

Facebook leveraged our work on help systems to lower usability friction, keeping people on its platform.
My research replaced people with machines

Our research on help systems, and the startup that grew out of it (AnswerDash), created two dozen jobs replaced tens of thousands of customer service agents with information retrieval algorithms, while enriching investors (a little)

We eagerly found ways to replace human effort with machine effort.
Coming out, I faced oppression first-hand

- TSA body scanning leads to body searches when I travel
- ACM/IEEE digital libraries deadname me every day
- Trans-exclusive health IT has led to medical errors
- Twitter has facilitated cyberbullying at the scale of thousands
- News aggregators remind me daily of violence against trans people

Being trans is in a transphobic world is hard. Software makes it harder.
A moral quandary...

I love computing as a medium and want other people to love it too!

I’ve spent my whole career trying to share that love

Computing is harming me and others, and few in CS seem to care or do anything about it.

How can I continue advocating for something that is doing such harm?
I had to revise my notions of justice...

1. Computing should be useful and usable. It must respect everyone.
2. Everyone should be empowered to harness computing to dismantle systems of oppression, rather than reinforce them.
I had to reconsider my research

1. How can we educate the public to ensure that computing respects everyone?

2. How can we educate developers to ensure they make choices that dismantle systems of oppression, rather than reinforce them?
Today, I want to share my lab’s nascent efforts to explore these notions of justice in our research, teaching, and service, and entice you to join us.
I’ll discuss three ways that computing reinforces systems of oppression

For each, I’ll share research by my Ph.D. students that explores how to teach these reinforcing patterns, and empower students to disrupt them.
Warning
This is an unconventional talk. There will be research, but there will also be teaching, service, and politics.
Computing has limits

https://xkcd.com/2237/
Many of us think of computing as endlessly powerful.

The world is using it to simplify retail and transportation, but also poverty, crime, hunger, climate, health, wellness, and more.
But this neophylic myth has ignored real harm.

Judges are delegating sentencing decisions to racially-biased predictions.

States are delegating food stamp eligibility judgements to inhumane algorithms.

Filter bubbles are dividing our discourse.

etc.
What everyone should know about computing

- Code is often wrong (O’Neil, 2016)
- Code embodies its creator’s values and biases (Costanza-Chock, 2020)
- Code can’t solve every problem; it often causes new ones (Toyama, 2015)

The public doesn’t know these facts because we’re not teaching them. In fact, many in CS are saying the opposite, that software is neutral, that it is infinitely powerful.
How can we teach these limits?

Three examples from my lab.
Greg Nelson has invented tutors that teach limits of programming languages. His tutor teaches the mechanics of program execution and reflects on its limits. Students learn that what little intelligence programs have is bestowed by people.

Stefania Druga has shown youth the limits of AI.

When children begin to create with machine learning embodied in robots, they begin to debate the capabilities with peers, and come to see the power of AI skeptically.
Alannah Oleson has invented methods for uncovering designers’ assumptions.

Over a period of weeks, students come to realize the narrow ways in which they understand human diversity and the complexity of designing for it.

—I study HCI and Design Education from a social justice lens.

"I feel slightly less confident in my inclusive design skills [now]... the result of a reality check the [method] gave me."

"... helped us understand the assumptions that we had but didn’t notice while we were creating the design."

"... taught us to think about all kinds of users rather than just a generic one"

"My biggest takeaways from [class] were that I had prejudices... that I didn’t even realize, that I actively needed to change those biases."
These are just a few questions of many, e.g.,

How can we develop views of CS that balance skepticism with optimism?

How well do shifts in beliefs about CS persist long-term?

How do new conceptions of diversity influence algorithm design choices?
Data has limits

https://xkcd.com/1838
Many think of data as being inert.

Computing makes it easy to capture, easy to store, easy to process. Therefore, computing is the powerful thing, data is just input, right?

Data is just bits, right?
It’s biased crime data that has bolstered the carceral state.

It’s cisnormative data on that cause gender nonconforming people be harassed by the TSA.

It’s our desire for data that’s driving increases in carbon output.
What everyone should know about *data*

- Data is a record of the *past*, not the future (Boyd, 2011)
- Data encodes values, assumptions, and goals (Garcia, 2018)
- Uses of data harm people in *unequal* ways (Costanza-Chock, 2020)

The public doesn’t know these limits because we talk about data in *abstract*, *static* terms. But developers, harnessing data for computational ends, can do *concrete*, *dynamic* harm.
How can we teach these limits?

Two examples from my lab.
Yim Register is teaching the limits of data with tutors.

Their tutor teaches prediction concepts with personal data, which their recent study shows is superior at helping people learn a model’s semantics, and frame self-advocacy arguments in terms of those semantics.

I’m interested in creative ways to develop machine learning literacy!
Benji Xie is modeling inequity in learning contexts. His latest project is investigating a new ways for teachers to gain insights about inequities in their classrooms, amplifying minoritized voices while preserving their privacy.

---

I'm interested in equitable human-AI interaction in learning contexts!
These are just a few questions of many.

How can we develop learners’ understanding of data harm?

How should we reframe algorithm design in a way that embraces the limits of data?

How should we reframe data structure education to highlight the role of structure in erasing diversity?
Computing has responsibility

Related to moral relativism, it states that ethics become subjective only when you approach the speed of light. That is, it's ok to be self-serving, steal, and murder as long as you're going really, really fast.

(Note: This is why rap sounds better on the highway at 90 mph)

https://xkcd.com/103/
CS values innovation, disruption, power, and speed

We celebrate students who secure powerful roles, companies that restructure markets, and innovations that reshape society.
But CS often leaves the moral choices about *what* to create to investors, or worse, a myth of technological determinism. But *what* we make is not an individual, inevitable choice, but a moral, collective one. It’s our responsibility to center these individual and collective value tensions and impacts in student learning.
What everyone should know about the responsibility of CS

- Software design choices are collective choices (Vakil, 2018)
- Developers’ values and politics are infused in their choices (Vakil, 2018)
- Developers are responsible for what they make (Friedman, 1992)

Too few people—including CS faculty—understand these responsibilities. It’s our job as teachers to develop awareness of these them, creating a global sense of accountability amongst people who amplify social forces with computing.
How can we teach these responsibilities?

Two examples from my lab.
Neil Ryan is understanding career choices.

Their latest project has found that most undergraduate CS majors rapidly absorb the dominant narratives about careers from their departments and shape career trajectories accordingly.

"I want to change the narrative of what's standard, acceptable practice in computing to better serve humanity!"

"it really normalizes going into big tech just after undergrad. Which I think implicitly normalizes not going down other paths, and not asking questions of these big tech companies. I mean, this is what makes (CS) money, right? When they can have recruiters and big tech companies come recruit successfully, and get, like, tech workers into their capitalist agendas, then (CS) gets more money."

"CS-P1, reflecting on the large gifts given to CS departments by tech philanthropists."
Jayne Everson is studying how to prepare justice-focused K-12 CS teachers. She’s just starting to observe how critical reflection CS and society changes the how they see computing and how they frame computing to youth, across STEM learning contexts.

I’m interested in tools and teaching methods for supporting project-based learning about CS and social justice.”
For my lab, this is just the beginning.
Justice-Focused Secondary CS Education

https://criticalcsed.org

A new NSF grant with my College of Education that will:

- Launch a unique **masters in teaching** that teaches foundations of CS in terms of justice, graduating hundreds in the next 10 years
- Produce a new book on teaching methods for critical computing education
- Investigate shifts in CS teacher identity as they engage sociopolitical issues in their teaching

Can’t make progress without $.
Justice-Focused Undergraduate Teaching

This summer, I revised all of my online books to center issues of justice. Dozens of faculty, including me, use these to teach HCI, design, and software engineering.

So far, students appreciate raising issues at the intersection of race + tech, but are hungry for more.
CS for All Washington

https://csforallwa.org

I lead a coalition of 300+ teachers, researchers, district leaders, industry advocates, not-for-profits.

We’ve passed legislation that 1) requires CS electives in all schools, 2) legally defines CS in justice terms, and 3) secures a $3 million in annual state funding to support teacher professional development.

Critical Computing Education — Amy J. Ko
AccessComputing
https://www.washington.edu/accesscomputing/

With Richard Ladner and Jacob Wobbrock, I co-PI AccessComputing, an NSF-funded Broadening Participation in Computing alliance that creates pathways for students with disabilities to learn CS, and identifies ways to integrate accessible computing into CS curricula.

Students with disabilities are systematically excluded from CS learning throughout K-12 and higher education. Let’s change that.

Critical Computing Education — Amy J. Ko
Research in the Code & Cognition Lab

Many more justice-focused CS literacy projects on motivating and supporting:

- Learning at home
- Learning in school
- Learning in communities
- Learning at work
- Self-advocacy

Students of the Code & Cognition Lab, eating donuts without me.
What can you do?
Augment your research with BPC plans

NSF CISE now requires broadening participation in computing plans.

Use them to seriously address not only issues of equity and inclusion, but also justice.

See bpcnet.org for guidance.

NSF wants us to focus on broadening participation.
Interrogate your teaching

What values are you supporting?

How are your explanations, examples, and assessments reinforcing these values?

How can you be more explicit about your values?

Students are looking to us for moral guidance.
Recognize your power and the responsibility that comes with it.

You have far more than you think, and unless you carefully reflect on the consequences of your choices, you won’t see them.

Being *apolitical* is just another kind of political.
Learn to have *hard conversations* about CS + justice. We can’t let our fear of conflict and politics let code and data further aggravate an already unjust world.
Questions?

The gist:

1. Computing has limits
2. Data has limits
3. CS has responsibility

We need research, teaching, and service that seriously engage these facts; I’m trying to figure out how to do that in CS education.