CONTENTS

Introduction

WHICH FAIRNESS? THE ONE WE CHOOSE!

Decisions, Decisions, Decisions

This morning, I awoke at 6:45 a.m. to prepare breakfast for my daughter and help her get ready for school. The decision to wake up at that time was one I made yesterday when I set my alarm. Yet, I have plenty more decisions ahead of me today. Thousands, perhaps, depending on how you count them.

Many of these decisions will be minor, like choosing the color of my T-shirt this morning (black, if you are wondering). But some will significantly impact other people. As a parent, many of my choices, such as helping with homework or deciding on summer camps, affect my children, though probably much less than I imagine. Later today, in my role as a professor, I may support or object to the candidacy of a faculty applicant to our Computer Science Department at Stanford. On the way there, as a driver, a moment's inattention could change another person's life, and I pray it never does. While the stakes in academia are famously said to be quite low, people in other professions, such as judges, physicians, police officers, and social workers, much more frequently find the fates of others resting in their hands.

With all these decisions we make, big and small, it is no wonder that, as a society, we are increasingly embracing the help of algorithms. Algorithms, like the spell-checker reviewing these words, spare us tedious tasks. They can also match, and sometimes outperform, humans in

some of the surprisingly complicated challenges we face at work and in daily life.

In health care, for example, diagnostic systems are sometimes as accurate as experienced clinicians. Computational health care may be a partial answer to a growing crisis brought about by shortages of medical professionals, increasing demand due in part to aging populations, and the rapid expansion of medical knowledge. In finance, automated systems monitor streams of transactions no human could follow. In education, learning platforms evaluate students and tailor assignments to each student's pace. Whether or not we notice it, a growing share of decisions that affect us are made with, or by, computational systems.

For better or worse, algorithmic decision-making is everywhere. From the moment we wake up to the moment we fall asleep, and even while we sleep, we are being measured, ranked, and classified. Computational systems help determine what we get, what we pay, and what opportunities we see.

This "day in the life" I've described is increasingly shaped by algorithmic interventions. As their effects play out, the story again starts with an alarm, but the exact time is now decided by an algorithm monitoring sleep patterns and one's calendar. The person wakes up refreshed, with extra time to scroll an algorithmically ordered feed, or to consult a chatbot about summer camps or work decisions, or even to seek comfort regarding a personal setback. They may take a self-driving car through an algorithmically chosen route to an engagement party with a partner matched by a dating app.

Or perhaps this person will encounter algorithms in higher-stakes settings: interacting with a physician deciding on a treatment, a loan officer approving a mortgage, or a judge making a bail decision.

This is all happening today, and it seems likely that no matter when you are reading these words, and no matter how many algorithms affect your life today, algorithms will soon have an even greater impact. Therefore, if I focus too much on specific technologies, this book will age faster than I can write it. Instead, I will focus on scientific ideas that transcend technology.

As algorithms move to center stage, they bring with them a constellation of new societal questions. Some involve privacy and surveillance; others touch on free speech, disinformation, intellectual property, and labor. Fairness is a central piece of this puzzle and the one we will spend most of this book exploring. But before we zoom in on fairness, we should first take a brief look at the wider landscape.

Computation and Society

With the growing prominence of algorithms, it may feel like we are living in "the Matrix," controlled by invisible code.[1] Perhaps. But even if algorithms govern humans, humans still govern algorithms. People design them, use them, and make policy decisions that constrain them.

Because algorithms now touch nearly every aspect of our lives, it is not surprising that many of the major policy questions debated today revolve around computation. Fairness is the central theme of this book, but it sits within a wider set of societal considerations.

Take privacy. Decades ago, a primary fear was government surveillance. Today, surveillance has been privatized. The governments of old would have killed (pun intended) to possess the granular data on us that technology companies collect every day. We casually quote that "data is the new oil" and "if you're not paying for the product, you are the product," but the reality is even more expansive. It is not just the tech giants; an entire industry buys and sells our data behind closed doors. Even the savviest users find it nearly impossible to fully protect their privacy.

Privacy is just one member of this family of challenges. Other questions at the heart of public debate also revolve around computation: the balance between free speech and content moderation, the spread of disinformation and deepfakes, creators' rights over training data, job displacement, and the environmental toll of large-scale computation. Issues of transparency, accountability, and consent only complicate the picture. These concerns, alongside many other examples, underscore

1. A reference to the 1999 science fiction movie of the same name that explores a dystopian vision of an invisible "algorithmic hand" controlling reality.

the growing entanglement of computation with nearly every facet of social, economic, and political life—a complex and evolving puzzle with no easy solutions.

While this book is written by a devotee of (theoretical) computer science, even I understand that none of these problems—including the question of algorithmic fairness, which is our focus—*will* or even *should* be resolved by computer scientists alone. The expertise required to form an appropriate policy response goes well beyond our training. More importantly, these are questions of values, and the answers belong to everyone, not just to experts of *any* field. But I am also convinced that no real progress can be made without the deep understanding of computation that computer scientists can provide. Computer scientists should not only inform the public debate, they should also extend the range of solutions available to society.

The field of algorithmic fairness is a prime example of this collaboration. It challenges us to translate moral values into mathematical and computational frameworks, often revealing just how difficult that translation truly is. With this broader landscape in mind, we can turn to the concrete case of fairness and see how this intricacy unfolds.

Algorithmic Fairness?

If I asked you what comes to mind when I mention fairness, what would you say? Would you reflect on discrimination and injustice in our imperfect world? Or would you be drawn to personal memories? You might remember a moment in childhood when you felt mistreated by a parent or a teacher. Or perhaps a recent moment when the actions of a manager, a colleague, or a spouse made you exclaim, "That's not fair!"

These are all very natural responses, and they are how I myself might have answered if you had asked me this same question sixteen years ago. But now that I have devoted much of that time to studying algorithmic fairness, my response is much more complicated. When I think about fairness, the image that comes to mind is an intricate landscape of many possible ethical choices. And it is not only me. The word "fairness" is still used to respond to a burning insult or to express righteous indignation,

but in recent years it has also become central to discussions of the interface between people and algorithms.

But hold on a minute. Algorithmic fairness? What do algorithms have to do with fairness? Isn't an algorithm simply an inanimate mathematical object? Surely an algorithm cannot discriminate any more than a building or a playground could. It is true that the building and the playground do not *choose* to discriminate, yet their inaccessible design may be discriminatory nonetheless. Similarly, algorithms do not *want* to discriminate because, so far, algorithms do not have wants or wishes. Even so, they can be designed, inadvertently or maliciously, in ways that discriminate. And this risk is not theoretical; in reality, algorithms often exhibit discriminatory behavior, frequently leading major technology companies to face significant embarrassment as they try to explain that behavior.

Readers who have followed technology news over the last decade will not be surprised. In 2016, ProPublica argued that software used across the United States to predict future criminals was biased against Black defendants. Two years later, Reuters reported that Amazon had to scrap an experimental AI recruiting tool because it learned to penalize women. By 2019, Science News was covering how a common health care algorithm disproportionately hurt Black patients. If only these were isolated incidents from the past. But the headlines have not stopped; they have simply moved to new frontiers. In 2024, a class action lawsuit alleged that Workday's AI screening tools discriminate against Black, older, and disabled applicants, while researchers at the University of Washington demonstrated that even the newest AI tools exhibit clear biases when ranking job applicants based on their names.

Yet, despite such failures, the use of algorithms is bound to continue, and I believe it should. These systems are far too valuable to individuals and to societies to abandon altogether. Even though progress will not stop, it should be carefully governed. When humans or institutions make decisions about other humans, we expect, regulate, and legislate that these decisions do not discriminate. When algorithms are making or informing such decisions, we must similarly make sure that these algorithms are fair.

You may be thinking: Fine, we need fair algorithms. *But what does fairness mean?* In trying to articulate an answer, we might reach for words like equality, equity, or impartiality. Those are important starting points, but how can we impose such abstract notions on an algorithm?

Fairness, like beauty, has been studied since ancient times. And like beauty, it still often seems to be in the eye of the beholder. Philosophers, legal experts, and other scholars have suggested important, and sometimes conflicting, perspectives on how to define fairness for millennia.

However, our increasing reliance on algorithms in high-stakes decisions demands that we augment the ways we already talk about fairness—whether in everyday life, philosophy, or law—with language that is more precise and, at times, more mathematical in nature. Throughout the twentieth and twenty-first centuries, economists, statisticians, and computer scientists have developed such a language. Computer scientists, in particular, have brought a perspective shaped by a deep understanding of the nature and limitations of computation. In the next sections, and throughout the book, we will see how this more formal, computational language can help us express and sometimes challenge our intuitive ideas of fairness. We will begin, however, with a more modest step: looking carefully at the everyday meanings of "fair." Then we can start turning them into definitions that algorithms can work with.

Fairness: Asking a Little, Asking a Lot

Working on algorithmic fairness has one unique characteristic that sets it apart from most of my past research: Everyone has an opinion. In the past, very few people shared their thoughts with me on, say, pseudorandom generators for small-width branching programs (and if your reaction is "pseudo-what for what now?" then that was exactly my point). For me, this shift has been wonderfully refreshing, since it lets me share my work more easily with the people in my life. But it can also be a little annoying, especially when I encounter major misconceptions that stem from confusion between how terms like fairness and equality are used

in day-to-day language and their more formal interpretations in scientific literature.

One time, a colleague from a different field invoked the dictionary definition of fairness to support his view of what algorithmic fairness should be. Unceremoniously, I then asked whether we should next base our scientific debates on newspaper comics. With the benefit of hindsight, though, I can see that there is a lot we can learn from those dictionary definitions, and that is where we are going to start.

One way Merriam-Webster defines the term "fair" is "marked by impartiality and honesty: free from self-interest, prejudice, or favoritism." This definition resonates naturally with ideas often associated with procedural fairness, a perspective emphasized in legal scholarship, where attention is directed to the fairness of the decision-making process itself—its impartiality and freedom from favoritism—rather than solely to outcomes. In contrast, fields such as computer science and economics have tended to focus more on substantive fairness, which evaluates whether the outcomes of a process are fair, regardless of how they were produced.

The gap between procedural and substantive fairness is one of several tensions that we will encounter in this book while exploring the landscape of algorithmic fairness. But the real reason I am quoting dictionary definitions is two other ways the word "fair" has been used. The first is "not very good or very bad: of average or acceptable quality." This definition reminds us that fairness is really a rather minimal requirement. Unlike altruism or generosity, all members of society deserve to be free from unfair discrimination, and such expectations are often grounded in law and regulations. One could argue, as the late philosopher John Rawls did, that fairness is about the basic contract of individuals with the institutions of a liberal society.

Before we imagine that fairness is easy, however, let us consider yet another usage of "fair." The word can mean "pleasing to the eye or mind, especially because of freshness, charm, or flawlessness." This beauty-related interpretation has taken on another meaning: "having very little color, coloring, or pigmentation: very light." The implication is hard to miss: The ideal of beauty is linked to lightness of skin. No wonder the

"fairest of them all" is none other than Snow White. We might wonder: How easy could it be to prevent discrimination when our very language has long entwined beauty, skin color, and virtue?

With that sobering observation, let us put the dictionary aside and turn to definitions that are meaningful and actionable in the context of algorithmic fairness.

So, What Is It?

You may be thinking: Enough stalling, just come out with it and specify the definition of fairness. And I would love to oblige. I could save so much time writing this book and even more time studying algorithmic fairness. An early retirement does not sound so bad.

So yes, I would love to spell out a definition that is precise enough and mathematical enough for us to produce algorithms that are fair. A definition that scholars of all disciplines would hail as true. A definition that policymakers would gladly enforce. A definition that would allow activists to put down their pens and their swords. Can't things be simple for once?

Unfortunately, or perhaps fortunately, nothing worth understanding is ever that simple. Not only are we unlikely to get philosophers, legal experts, economists, statisticians, and computer scientists to agree on a single definition, but even within each discipline no definitive answer exists.

Why is it so hard to pin down? Because fairness forces us to navigate a series of profound dilemmas, each with valid competing arguments. First, there is the tension between process and outcome that we already encountered. Many would consider a hiring process fair if every applicant were evaluated using the same test and interview questions. But success in answering those questions—even when they are carefully designed—may be affected by unequal access to preparation. In such cases, the procedure itself may be impartial, while the outcomes it produces are not. Then there is the clash between equality and merit. Should we distribute resources like vaccine doses or school slots evenly among everyone? Or should they go to those who need them most, or

perhaps those who "earned" them? We must also weigh the individual against the group. A policy might look fair when we compare statistical averages across demographic groups, yet it might still treat a specific individual within those groups unjustly. Additionally, how do we account for individual preferences? A "fair" distribution of food is ineffective if we give meat to a vegetarian and nuts to someone with an allergy. On the other hand, some preferences are themselves rooted in bias and may be harmful to others.

We will explore these tensions and many others in depth. But as we do, we will find that there is no universal tiebreaker. The right choice often depends heavily on the context.

Consider the difficult trade-offs we face in the real world. In health care, who should receive a scarce, life-saving organ donation: the patient whose condition is most severe, the one who is likely to gain the most years of life from it, or the one who has waited the longest? In college admissions, should an algorithm prioritize the student with the highest grades, or the one with slightly lower scores who earned them while working night shifts to support their family? Even in advertising, the context shifts the moral ground. Targeting skin-care ads toward a certain gender might be accepted as "relevant marketing," but targeting high-paying job ads in the same way can exclude people from economic opportunity.

These examples are not just hard choices; they show that the fairness criteria we find compelling can change from one domain to another. What feels appropriate in one setting can seem deeply unfair in another. Fairness, and the mathematical definitions we use to express it, depends on the specific decisions we (or our algorithms) need to make and on the values of the society we are part of. Therefore, in each context, we may reasonably arrive at a different answer to the questions about fairness that have concerned scholars for generations.

I have been promising a mathematical language of fairness, but such a language is not meant to suppress public debate by offering simple solutions. Instead, it offers a new perspective on ancient arguments: not settling them, but refining them, and exposing trade-offs that were previously elusive.

The Theory of Computing

So far, our story has centered on fairness itself: a deeply human concept, shaped by values, context, and difficult trade-offs. But if fairness is to go beyond abstract ideals and affect people's lives, it must guide the processes (whether algorithmic or human) that make decisions about them.

It is therefore time to get more familiar with computation, the second major character in our story, and ask what it can tell us about decision-making procedures that are meant to materialize fairness.

The perspective we will take is that algorithms do more than merely implement our definitions of fairness; they shape them. Once we commit to making fairness effective, questions about what can be computed, checked, and enforced begin to define what fairness can reasonably require. To unpack this, we first need to look at the Theory of Computing—the field that studies the fundamental capabilities and inherent limitations of computation. Since this book approaches fairness through a computational lens, this is our natural starting point.

We have been using the term "algorithm" rather casually. A common way to think of an algorithm is as a recipe: a set of precise instructions that transform ingredients (inputs) into a finished dish (output). Take the grade-school algorithm for long addition. This familiar arithmetic algorithm fits the analogy perfectly. It provides a systematic way to combine two numbers (inputs) into their sum (output) using a sequence of basic steps: Align the numbers, add the columns from right to left, carry the tens, and so on. While you may not think of yourself as a computer, you have almost certainly executed this algorithm many times.[2]

The recipe analogy serves us well for now, though it does not capture every form of computation. For example, learning algorithms, the kind that learn a rule from data instead of following a fully handwritten

2. Perhaps you are aware that, before digital computers, the word "computer" was used to refer to humans whose work is to perform computations. If you saw the movie *Hidden Figures*, you can appreciate the importance of human computers through the remarkable story of Katherine Coleman Goble Johnson, the outstanding NASA computer who received the Presidential Medal of Freedom in 2015.

recipe, operate quite differently. However, the analogy helps clarify the scope of the Theory of Computing. While other areas of computer science focus on how algorithms are implemented by digital computers or networks, the Theory of Computing studies the algorithms themselves. The logic of long addition is the same whether it is executed by a supercomputer or by a human with a pencil.[3]

A central notion in the Theory of Computing is computational efficiency: the resources required to execute an algorithm. We will focus mostly on "running time" even though there are other important resources (memory for example). The term can be misleading because we do not measure it in time units. Since a supercomputer is obviously faster than a human, measuring in seconds would tell us more about the hardware than the method. To analyze the algorithm itself, we therefore measure running time in the number of basic steps it requires.

Consider the difference between adding two large numbers and multiplying them. The long-addition algorithm we learned in school is highly efficient: The number of steps grows roughly in proportion to the number of digits. If you double the length of the numbers, the work roughly doubles. Long multiplication, however, is more demanding. The grade-school method involves multiplying every digit of the first number by every digit of the second. If you double the length of the numbers, the work required doesn't just double; it quadruples. For numbers with thousands of digits, multiplication becomes significantly more expensive than addition.

This brings us to algorithmic design, the search for smarter, better algorithms. The grade-school method for long multiplication was known in one form or another for centuries and it was tempting to

3. In fact, some of the impact of the Theory of Computing in areas like biology, physics, economy, and social sciences stems from a more general perspective that finds computations all around. This transformative perspective suggests that many processes composed of simple local steps can be understood as computations, regardless of their domain. Therefore, in this view, computation is the essential activity that drives much of what's around and inside us: from atomic particles and biological cells to our brains, the actions of animal societies, evolutionary processes, financial markets, and social networks. But to capture the computational lens on fairness, we will not need to go that far.

believe that it is essentially the best. But in the 1960s, researchers discovered a faster algorithm that brings the cost of multiplying huge numbers much closer to that of adding them. This highlights a central idea of the field: just because we have a working recipe doesn't mean it's the best one. There may be a clever shortcut waiting to be discovered.

But can we always find a shortcut? This question leads to computational complexity, the study of the inherent difficulty of problems. While algorithmic design looks at specific computational problems and tries to find the best algorithms to solve them, complexity theory tries to argue that for some problems there is an inherent limit on how efficiently they can be solved.

Complexity theory classifies problems based on the resources required to solve them. Some problems, like multiplication or sorting a list, are "efficiently solvable" (they belong to a class known as P). Others appear to be fundamentally intractable. These problems do have an algorithm but executing it would require an unreasonable amount of time. For example, if the running time of an algorithm grows exponentially with the size of the input, then even moderately sized instances become hopeless. Such a calculation could take longer than the remaining lifespan of our sun, even if every computer on earth was dedicated to the task. We will slightly refine our definition of efficiency later, but this high-level intuition is enough for now.

The distinction between efficiently solvable and computationally intractable problems leads to the most famous open question in complexity theory: P vs. NP. Essentially, it asks, if we can quickly check a solution (like verifying a completed sudoku puzzle), can we also quickly find that solution from scratch? Most researchers believe the answer is no ($P \neq NP$). If true, this means there is a vast class of important problems, known as NP-complete, that are inherently intractable. In other words, hard limits on computation are not just theoretical curiosities; they are walls we hit every day.

The reason we have taken this detour into the properties of computation is to ground our ethical discussions in computational realities. This brings us directly back to fairness.

The Computational Lens on Fairness, Take I

We will soon make the argument that a computational perspective can shape how we think about fairness itself. But before making that claim, let us begin with a more modest and concrete connection between the Theory of Computing and fairness.

When we examine how to make fair decisions, we are inherently asking algorithmic questions. We are asking not only *what* constitutes a fair decision, but *how* such a decision can be reached. The procedures by which we reach fair decisions lie in the domain of algorithms.

This process leads to an initial, fairly unsurprising, observation: Caring about fairness means caring about algorithm design. For every task with fairness implications, we need to design algorithms that are fair. And because there is no single definition of fairness, this does not mean designing a single "fair algorithm," but rather a family of algorithms, each aligned with a particular interpretation of fairness and the tradeoffs it entails.

But which algorithmic tasks have fairness implications? Practically any task when it affects people. Consider, for example, a problem that appears in every introductory algorithms course: matching. In its abstract form, matching asks us to connect entities so as to optimize some notion of cost or benefit. Students might meet it as a purely technical challenge, such as assigning software applications to machines in a large cloud-computing system. Each assignment has a cost (it might run slower on one machine than on another). In that context, the goal is simple: maximize speed and minimize cost. The servers have no feelings, and the apps have no rights. We simply want the system to run efficiently.

Yet the very same mathematical structure arises when we match medical interns to hospitals, students to public schools, kidney donors to recipients, or people to potential partners on a dating app. Once humans are involved, questions of fairness quickly surface, and the answers need not look the same across contexts.

In assigning students to public schools, we may prioritize equality and broad access. In matching interns to hospitals, qualifications and training

needs may take center stage. In kidney exchange, considerations of deservedness, such as the level of medical urgency or how long someone has waited, can outweigh other concerns. The role of preferences may also vary. In a dating app, for example, individual preferences cannot be ignored altogether, since a match that no one accepts is a failure. But we may still want to limit the influence of preferences that systematically disadvantage certain groups. Even when the underlying algorithmic problem looks the same, the appropriate definition of fairness depends on the context and on the values we are trying to protect.

As fairness is increasingly recognized as a design objective, we are forced to revisit a vast body of algorithmic work through a new lens. For many tasks, this means identifying where fairness concerns arise, choosing a definition of fairness that fits the context, and then designing algorithms that reflect that choice. At the same time, those algorithms must still satisfy more traditional design goals, most notably computational efficiency.

For now, the takeaway is simple: Once fairness becomes a design goal, much of the algorithmic literature should be revisited. But algorithms as a medium for implementing fairness is only the beginning. Next, we will see a deeper connection: The limits of computation can shape what fairness can reasonably demand in the first place.

The Computational Lens on Fairness, Take II: Ought Implies Can

We have just seen how fairness enters the picture as a design goal, shaping the algorithms that make decisions about people. But as we turn to designing fair algorithms, a deeper question arises: Is the kind of fairness we want even achievable? A useful guide here is the philosophical principle "ought implies can." Roughly speaking, it says that we cannot place a moral obligation on someone to act in a particular way if acting in that way is impossible.

The same idea applies to decision procedures. Asking a decision-maker to take an action that is "fair" under some definition is problematic if it is impossible to identify a fair action, or to reach one in a reasonable way.

The mere existence of a fair solution is not enough; it is also critically important that the solution can be found and used under realistic constraints.

As we discussed earlier, complexity theory teaches us that some problems are inherently intractable. Even the most powerful computers or the most brilliant people would take longer to solve them than the entire history of human civilization. Defining fairness in a way that requires solving an intractable problem is not merely impractical; it is conceptually flawed as an ethical requirement, because it cannot meaningfully guide action.

To see this tension, let's briefly borrow a problem from part II of this book: cake cutting. Imagine dividing a cake among a group of people who all want different parts. One person loves colorful sprinkles, another adores the buttercream flowers, and someone just wants a piece without candle wax. Economists have proposed various fairness concepts for this scenario, one of which is envy-freeness: A division is fair if, after we cut and distribute the cake, no one prefers someone else's piece to their own.

Envy-freeness can be appealing in some contexts, and it is known that an envy-free division is guaranteed to exist. But guaranteeing its existence and finding it are very different matters. For two people, it is easy to find an envy-free division. For three, it is complicated but doable. But as soon as we have four people, the complexity explodes. The best-known general procedures for guaranteeing envy-freeness require a number of steps so vast it defies imagination. Even if such a procedure could be magically carried out, it would likely leave each person with nothing but a pile of subatomic crumbs.

In short, pure envy-freeness is a beautiful ideal that becomes unwieldy as soon as the group grows even modestly. This example already hints at a broader lesson: In algorithmic systems, fairness ideals must contend not only with preferences, but with scale and complexity. Throughout the book, we will see other fairness notions with a similar profile—desirable in principle but demanding prohibitive computation in practice. The computational lens helps us recognize these trade-offs. It also suggests constructive responses: We may approximate the ideal,

relax the requirement slightly, or restrict attention to settings in which the guarantee becomes feasible (as is the case in the example of envy-freeness in cake cutting), preserving the spirit of fairness while accepting the constraints of reality.

But the computational lens goes deeper still. Sometimes, computation is not just a constraint on implementing fairness; it becomes part of the definition of fairness itself. A prime example is the notion of multigroup fairness, a concept that has gained significant influence over the last decade.

Many traditional definitions of fairness focus on averages over broad demographic groups: Does a system treat "women" or "men" similarly, in terms of outcomes, error rates, or opportunities? But averages can hide pockets of discrimination. A system may appear fair to women as a whole while systematically failing for specific subpopulations, such as low-income women over 50, or low-income women over 50 with a history of heart disease.

Multigroup fairness responds to this concern by extending attention beyond a small, fixed set of demographic categories to a much broader collection of groups defined by combinations of attributes, including domain-specific ones such as medical history, financial circumstances, or prior interactions with the system. In this way, fairness is no longer only about a few preselected categories, but about guarding against systematic disadvantage wherever it may arise.

This raises a natural question: How broad can such protection be? Here, computational considerations become decisive. A fairness guarantee can apply only to groups that can be identified and tested in a reliable way. If a group is so complex that no efficient algorithm can distinguish its members from nonmembers, then the system cannot even detect that the group exists, let alone that it is being harmed.

Multigroup fairness therefore aims to protect all groups that can be efficiently identified (that is, groups for which membership can be tested efficiently). Efficiency here is not binary; it comes in degrees. The more computational resources we invest, the more fine-grained the family of groups we can realistically protect. Computation therefore plays a major role in multigroup definitions: It functions like a knob. Specifying how

much fairness we require also means specifying how much computational effort we are prepared to expend.

The underlying truth, illustrated by cake cutting and reinforced by multigroup fairness, is that the computational lens is critical in determining which fairness guarantees are even coherent. This finding represents a profound shift. Computation is no longer just a tool to implement fairness; it is a core ingredient in conceptualizing what fairness is.

Computation beyond Machines

At this point, it may sound as though our discussion of fairness has drifted away from people and toward machines. But the insights drawn from the computational lens are not restricted to digital systems; they apply just as well to human judgment. We focus on procedures, the ways decisions are made, regardless of who or what carries them out. In a sense, the rise of algorithms forced us to adopt a computational perspective on fairness but in doing so we learned just as much about fairness carried out by people.

When a judge weighs evidence, a physician decides on a course of treatment, or an admissions committee evaluates candidates, the process may be informal, discretionary, or guided by experience rather than by an explicit rulebook. But it is still a process. Information is taken in, distinctions are drawn, trade-offs are made, and an outcome is produced. Whether that procedure is written in code, spelled out in policy, or implemented implicitly through networks of neurons and synapses in a human brain, it is subject to constraints imposed by the limits of computation.

This is where the perspective of the Theory of Computing becomes relevant beyond machines. Theory of Computing studies algorithms as abstract procedures, independent of how they are implemented. Its results do not depend on whether an algorithm is executed by silicon, or by biological hardware shaped through training and experience. Limits on what can be computed efficiently, what can be reliably identified, or what trade-offs can be simultaneously satisfied apply to procedures as such.

Seen from this angle, when a fairness requirement runs into a computational impossibility, it is not a quirk of automation. It is a statement about the structure of the decision problem itself. If no procedure can consistently achieve a certain combination of fairness properties under realistic constraints, then neither can a digital system, nor a judge or a physician be expected to do so. The difficulty does not vanish when the algorithm is not explicit but rather replaced by intuition.

This is the sense in which insights from algorithmic fairness apply to fairness in human institutions as well. They do not tell us that people and machines are the same, or that moral judgment can be reduced to computation. Rather, they clarify which fairness ideals are compatible with systematic decision-making at all. Understanding these limits is not about excusing unfair outcomes; it is about distinguishing between failures of execution and demands that no decision procedure—human or otherwise—can satisfy.

Who Chooses Which Fairness?

Up to this point, our story has been carried by two characters: fairness and computation. We have seen that fairness cannot be reduced to a single definition, and that computation is not merely the medium through which fairness is implemented, but part of what makes a fairness requirement meaningful.

But there is a third character in this story, one that has been present all along, mostly in the background: society.

Society is not a single mind with a single set of values. It is a web of institutions, incentives, and traditions. It includes courts and legislatures, regulators and professional norms. It is hospitals, schools, and police departments. It is the private companies whose business models shape what gets built and deployed. And it is academia itself, where philosophers, legal scholars, economists, and computer scientists debate fairness, sometimes using the same word to mean very different things.

Once we acknowledge this third character, "which fairness?" becomes an even sharper question. A definition of fairness is not chosen

in a vacuum. It is chosen within a particular institution, under specific constraints, by people with distinct roles. A hospital does not face the same trade-offs as a court. A school district does not face the same incentives as a technology company. Even within a single institution, different stakeholders may hold conflicting views on what is fair and possess vastly different power to enforce them.

This raises broader questions: How do societies adopt one notion of fairness over another? What mechanisms determine which values become policy and which trade-offs solidify into everyday practice? These questions sit at the heart of long-standing scholarly traditions. Political philosophy, law, sociology, and economics have all developed rich frameworks for thinking about power, legitimacy, and how values are negotiated over time.

The Theory of Computing intersects with these traditions in fascinating ways. Fields such as algorithmic game theory and computational social choice build upon economic foundations to study what can be achieved when many individuals, each with their own incentives, must make collective decisions. They bring a computational perspective to voting, strategic behavior, and institutional design, revealing fundamental trade-offs that arise even before we write a line of code. We will occasionally draw on these fields when they sharpen a dilemma. But the primary focus of this book lies elsewhere.

Our goal is not to adjudicate among competing social mechanisms for choosing fairness, but to clarify the space of possibilities those mechanisms operate within. By understanding what different definitions of fairness demand, what they protect, and what trade-offs they force, we gain a clearer view of the choices that societies face. We will return to this broader context at various points in the book, not as a detour from formal analysis, but as the setting in which that analysis belongs.

The Case for and against Mathematical Definitions

Algorithmic fairness is a messy, noisy, and intellectually invigorating field. When I present my work to a mixed academic audience, the critiques from colleagues of different fields often pull me in opposite directions.

This friction should be cherished. Each discipline brings a unique piece of the puzzle, and our goal should be mutual understanding, not uniformity.

This book aims to share the computational perspective on fairness, which is often obscured by a language barrier of math. But we want to place this perspective within a wider context, including critical theories, which pose what may be the fiercest critique of mathematical approaches.

A major concern raised by critical theories is reductionism. Fairness is subtle, contested, and deeply entangled with history and power. Reducing it to a mathematical definition risks shrinking it into whatever is easiest to measure. Worse, a formal definition can create a false aura of objectivity. An institution might claim its system is fair simply because it satisfies a specific metric, using the math as a shield against deeper criticism. And even when a metric is satisfied, it can legitimize a practice that should have been challenged at the level of power and governance.

These concerns are valid. Formalization can be misused, treating a provisional tool as a moral endpoint. But this vulnerability is not unique to math. Legal standards can devolve into box-checking; philosophical principles can turn into empty slogans ("bring all stakeholders to the table," "ensure inclusivity").

The main cure for reductionism is transparency. We will therefore position our models within their larger context throughout the book, identifying clearly what they address and, just as importantly, what they do not. I will also provide additional discussion and suggestions for further reading in this book's bibliographic notes for those who wish to engage with these critical perspectives more deeply.

Ultimately, the question is not whether a framework can be misused, but what it enables when used well. This is the best case for mathematical definitions: They offer a level of clarity and accountability that would otherwise be impossible, making our disagreements precise.

The Theory of Computing has repeatedly provided such success stories. Consider, for example, the history of cryptography. In its early days, security was a game of cat and mouse: Someone invented a code; someone

else broke it. For most practical schemes, there was no general way to justify security beyond the fact that they hadn't been broken yet. Modern cryptography changed this landscape by introducing rigorous definitions. Security became a precise claim: Secure against what kind of adversary? Under what assumptions? This gave deeper meaning to the protection of systems: A breach was no longer just a failure; it was information about which assumption had been violated. This approach let us transfer insight from one system to another, rather than arguing from scratch each time.

In fairness, the need for precision is even more acute. In security, a breach is often obvious (the message was stolen). In fairness, almost every outcome is arguable. A hiring algorithm can be attacked for unequal outcomes and defended as meritocratic. Without definitions, we lack a stable language to even say what counts as "breaking" fairness. Arguing about definitions is often more productive than arguing about particular systems. It makes claims checkable and enables us to carry lessons from one case to the next.

But definitions do more than just clarify; they create things that previously seemed impossible.

In the Theory of Computing, formal definitions have repeatedly unlocked new capabilities. Privacy is a prime example. Long before differential privacy, researchers tried to pin down privacy in precise terms, but differential privacy offered an especially clean and appealing promise: It enables us to learn useful facts about a population while tightly limiting how much any single individual can affect what is revealed. And the privacy promise is obtained at the level of the procedure rather than by hoping the data has been "anonymized" in some informal sense. In other words, differential privacy does not merely measure privacy; it makes a new kind of guarantee possible.

Another paradoxical-sounding success story involves zero-knowledge proofs: the ability to prove that you know a secret without revealing it. That idea feels impossible until a precise definition makes it thinkable, and then, remarkably, achievable.

We see the same potential starting to materialize in fairness. For example, one distinctly computational innovation is multigroup fairness,

which we discussed earlier. The concern of intersectionality—the idea that overlapping identities can create distinct patterns of disadvantage—was well understood. But the idea of simultaneously protecting a vast, intersecting web of subpopulations, defined by the limits of what an algorithm can efficiently detect, was unlikely to emerge before the computational lens was directed to fairness. It was the language of computation that enabled us to articulate such a rich guarantee and show that meaningful versions of it are achievable. Multigroup fairness is still not an ultimate fairness guarantee, but it introduces a new fairness trade-off that didn't exist before.

When we define fairness, we are not ending the conversation. We are just starting it. Fairness is too subtle to be captured by any single framework. Every perspective—whether legal, philosophical, political, or mathematical—is incomplete, but each is essential. This book offers a computational perspective on algorithmic fairness within this wider conversation.

The Case for and against Algorithms

Up to this point, I have argued that the computational lens applies to decision procedures no matter who executes them—digital systems and human institutions alike. Still, the urgency of today's fairness debate comes from the growing role of machines. Algorithms are now making, or shaping, decisions about bail, loans, hiring, health care, and education at a scale that is hard to ignore. That raises a fundamental critique: Should we really let algorithms make decisions about people in the first place?

The prison abolition movement argues that the current system of incarceration is inherently unfair and calls for a radical overhaul, including the possibility of eliminating prisons entirely. Inspired by this approach, some scholars and activists argue for forms of technology refusal: limiting or rejecting algorithmic decision tools in certain domains. The argument is not just that particular models are flawed, but that automation can entrench the values, incentives, and unjust power structures of the institutions that deploy it. Unsurprisingly, algorithmic tools

in the justice system attract particularly strong opposition. But criticism extends well beyond criminal justice to health care, education and other domains, where targeted systems raise fears about surveillance, exclusion, and discrimination.

It is easy to ridicule "abolish algorithms" as futile in the face of technological progress. I see it differently. Even if one rejects the strongest version of the argument, this critique still matters as a counterweight. It reminds us not to treat algorithmic deployment as the default. We should not assume that algorithms should be used whenever and wherever they exist.

That said, when we consider restricting algorithms, we must also weigh the alternatives. The key question is not only whether an algorithm is flawed (of course it is) but whether it is better or worse than the realistic human practice it would replace.

The debate around autonomous vehicles captures this tension: Should they be deployed only when near perfect, or is it enough that they are safer than human drivers? Just because algorithms are imperfect does not mean they are worse than humans, who are imperfect too.

In practice, the choice is rarely "algorithms or humans." There is a spectrum where human decision-making is supported by algorithmic tools such as scoring systems, structured guidelines, and risk assessments. Finding the right point on this spectrum requires comparing the strengths and weaknesses of each approach. In multidisciplinary meetings, I have often seen this comparison drift in a particular direction. We romanticize human judgment, caricature algorithms, and conflate the limits of current technology with the fundamental potential of computation.

Algorithms often excel at consistency, at absorbing large amounts of information, and at detecting subtle patterns, especially in high-volume settings where humans are forced to make rapid judgments. But it is also true that today's algorithms face substantial limitations. They struggle to grasp context, incorporate ethical reasoning, and handle situations outside their training data. This lack of common sense matters most when evidence is messy and circumstances are unusual, which is frequently where the hardest and most consequential decisions arise.

At the same time, I have yet to see a convincing general argument that humans are inherently superior along any of these dimensions. My expectation, therefore, is not that these limitations mark a fundamental boundary, but that many reflect the current state of our systems and the way we design, train, and deploy them.

Turning our attention back to human decision-making, we find that it, too, has deep structural flaws. Humans are not only biased; we are noisy. As the late Nobel laureate Daniel Kahneman and his colleagues documented, human judgment is shockingly inconsistent. A judge's decision can vary with factors that should be irrelevant, such as their mood, the time of day, or even the weather. In law and medicine, this variability is rarely a charming feature; it is a likely source of unfairness.

In health care, it is tempting to compare algorithms to an idealized physician: someone with the expertise of a top medical researcher and the familiarity, time, and empathy of Doc Baker from *Little House on the Prairie*. But the reality is that many patients see physicians who are struggling to keep up with medical knowledge and have precious few minutes per patient. These physicians, dedicated as they are, can still benefit from algorithmic support. When we compare humans to algorithms, we must compare them in the actual conditions in which they operate.

Other critiques claim that algorithms are inherently different because they rely on statistical correlations. But is that fundamentally different from what humans do? We are all shaped by experience. When a judge assesses testimony, that judgment is influenced by the implicit correlations they have internalized over a lifetime. If anything, algorithms have an advantage here. While we cannot easily audit a judge's upbringing, we have significant control over how an algorithm is trained and far more ability to test it for bias.

Another objection treats consistency as a disadvantage. A defendant facing a human judge feels they have agency. They can object, tell their story, and perhaps change the decision-maker's mind. An algorithm can feel like a fixed, indifferent machine. But this critique assumes algorithmic decisions must be rigid. Even if many current algorithms are limited

in this way, they need not be. Algorithms can be designed to accept new information, process objections, and even incorporate randomness when that serves a fairness goal. A defendant's objection can matter to an algorithm, if the system is designed to listen.

So, am I squarely on "Team Algorithms"? Not at all.

First, when we deploy algorithms, we need to evaluate what they can do now, not what we hope they might do someday.

Second, capability does not imply alignment. Even a powerful algorithm will not automatically do what we want it to do; it will do only what it is designed to do. Unless we build fairness considerations into the objective and the evaluation, optimization can improve overall performance at the expense of justice.

Third, algorithms come with unique dangers of scale. A biased hiring manager might unfairly reject one candidate, but that candidate can find work elsewhere. A biased resume-filtering algorithm, used across an entire industry, can lock that candidate out of the workforce permanently. This kind of algorithmic monoculture, where one model quietly becomes the default gatekeeper across a whole industry, can amplify harm in a way a single decision-maker usually cannot.

Finally, there is a difference that is not only about capability, but about legitimacy and relationship. Many people find comfort in a human in the loop (a person who can review, override, and be accountable for the system's recommendation) because the interaction can carry meaning: discretion, empathy, and the sense of being heard. Being judged by an algorithm can feel alienating. For that reason, even if algorithms outperform humans on particular tasks, societies may be understandably reluctant to let them replace physicians or judges.

People, Algorithms, and People Again

We started this book with an important, if frustrating, realization: There is no single definition of fairness. "Fair" is not one idea but a family of ideas—reasonable on their own, sometimes incompatible,

and always dependent on context. Choosing among them is not a technical detail; it is a value judgment. The real debate is not over whether fairness matters, but over which definition of fairness we choose.

This is where algorithms become oddly clarifying. They don't invent our values, but they refuse to run on ambiguity. The moment we demand that a system be fair, moral intuitions must be made explicit—precise enough to be checked, enforced, and debated. That translation is dangerous when it pretends to be neutral, but powerful when it is honest. A mathematical definition of fairness is not a moral verdict; it is a commitment about what we protect, what we ignore, and what we are willing to trade off. It sharpens disagreement and makes it more productive by exposing exactly where values diverge.

The computational lens adds another discipline, examining not only what we want, but also what can actually be done. Fairness is not only an ideal; it is a requirement placed on decision procedures operating under real constraints—limited time, limited data, limited attention. This is "ought implies can" in computational clothing. Once taken seriously, limits on computation are inseparable from the fairness demands we can coherently make.

Now for the people again. Algorithmic fairness is often framed as a problem about machines, but the deeper lesson is about decision-making itself. Whether carried out by code, policy, or human judgment—explicitly or implicitly—systematic decisions expose trade-offs. Studying fairness for algorithms forces those trade-offs into daylight, revealing which fairness ideals can coexist, which conflict, and which failures are failures of will rather than of possibility.

The arc of this introduction is therefore a loop. We begin with people, because fairness is a human demand. We pass through algorithms, because computation forces precision and exposes trade-offs. And we return to people again, because the real question is not whether machines can be fair, but whether we are willing to make our values explicit, defend them, and revise them when they collide with reality. This book is my effort to help you do exactly that.

Embrace the Math (and the Philosophy, Too)

If you've reached this far, you're likely starting to see why a computational perspective can clarify debates about fairness. At this point, readers might fall into one of two camps: those eager for the mathematics (enough with the chitchat), ready to dive into this exciting research area, and those with a touch of math-phobia. Readers in this latter group may acknowledge the importance of this work but expect that it will ultimately be beyond their grasp. While this book can't undo the math anxiety many of us carry, I hope it can offer a small corrective experience. Fairness impacts us all, and we should be empowered to take part in this conversation.

Mathematician Cathy O'Neil, author of the best-selling *Weapons of Math Destruction*, warned about the use of complex algorithms by corporations and institutions to disempower stakeholders, leveraging mathematical complexity to shield decisions from scrutiny. In truth, using complexity to create confusion is not unique to mathematics; fields like law and medicine also rely on specialized language that can exclude those without years of expert training. This sense of exclusion is powerful. And yet math anxiety is widespread, and modern learning models are undeniably complex. In that sense, I hope this book serves as a step toward "mathematical disarmament," not by eliminating the math but by demystifying it and giving readers tools to "argue with the equations."

To make this goal attainable, I will do a few things. First, we will not complicate matters where it isn't needed. Often, all we will need are a few variables to make the discussion more concrete—so x might denote the data associated with an individual, sparing us from repeating that phrase again and again. We will also rely on a bit of grade-school algebra and some basic probability intuition. Second, whenever the math creeps in, I will slow down to provide intuition and explanation. Finally, the book's narrative is designed so that you can skim, or even skip, the mathematics without losing the main thread. That said, I'll encourage you, whenever possible, to step just a little outside your comfort zone and engage with it.

To the math-savvy reader, let me say that you, too, may be encouraged to step outside your comfort zone. The book will repeatedly place formal ideas within broader philosophical and social contexts. You could skim or skip those parts, but a deeper understanding of algorithmic fairness requires the philosophy just as much as it requires the equations.

How to Read This Book

Throughout the rest of this book, we will explore a wide range of computational tasks with fairness at stake. In each context, we will define what fairness means, examine the kinds of discrimination we are trying to prevent, and highlight forms of unfairness that may persist even when a particular definition of fairness is satisfied. Finally, we will ask a question that is central to this book: Do efficient algorithmic procedures exist that can realize these fairness ideals, and what trade-offs do they inevitably entail?

This book was written with a diverse audience in mind, and I expect it to be read in many ways. You might read it cover to cover, insisting on understanding each idea in depth. Or you might skim, holding on to the overall storyline while letting the technical details blur. Alternatively, you can dip in, circle back, and take different paths depending on what brought you here.

What follows is some guidance for five common ways of reading this book:

- *As a big-picture reader*: If you are reading to sense what the computational lens can say about such a deeply human concept, focus on the motivating discussions and examples. Engage with the mathematical definitions only to the extent that they make the big ideas concrete. The technical details deepen the analysis, but the main narrative is designed to remain intact even if you skim them.
- *As a practitioner*: If you are a policymaker or a system designer, pay particular attention to the assumptions at the core of each

definition and to the values those definitions attempt to capture. Many of the most consequential fairness choices appear not as explicit moral declarations, but as technical decisions about objectives, constraints, and data.

- *As an academic*: If you are interested in an introduction to this subarea of the Theory of Computing, follow the definitions and results. While this is not a research monograph, the formal material is presented with enough care to support genuine reasoning. I have included several opportunities, in clearly marked sections, to go deeper into the mathematics. At the same time, I encourage you to engage with the broader context, as this field is ultimately motivated by real-world questions.
- *As an educator*: If you are reading as an educator, from high school through university, know that this book can be used as a textbook, even if that was not its primary intent. It is modular by design and can support a wide range of classroom settings: interdisciplinary undergraduate courses on technology and society, computer science courses on algorithmic fairness, or discussion-based seminars that bring together students from different backgrounds. My own algorithmic fairness course at Stanford overlaps substantially with the material here, though the classroom version goes a bit more deeply into formal definitions, theorems, and proofs.
- *As a student*: And if you are reading as a student, whether in a course or on your own, you should know that this book was written with you very much in mind. There is no single "correct" way to read this book. Go deep where your interest pulls you, and don't hesitate to skim or circle back.

Finally, I encourage you to approach each definition we discuss with appropriate skepticism. In which contexts would you choose it as the right notion of fairness? What does it protect, and what does it leave exposed?

Moving to the structure of the book, it is organized into two main parts. The topics covered in these two parts are not often considered

under the same umbrella, but I see both as essential for a more complete understanding of algorithmic fairness.

Part I: Training the Beast: Fairness in Machine Learning: We begin where the fire is hottest. The first part of the book tackles the messy, headline-grabbing world of machine learning and automated decision-making. Here we deal with algorithms that learn from data to make predictions and classifications: who gets a loan, who is flagged for medical risk, who is selected for a job. In these chapters, we will see how fairness concerns interact tightly with data quality, historical bias, and the objectives we encode in our models. We will explore why intuitive goals of fairness become so complicated—and at times even unattainable—when algorithms attempt to generalize from the past to predict the future. The discussion in this part is organized around several major axes of research, most notably the tension between fairness to individuals and fairness to groups, and between equality of outcomes and faithfulness to the patterns in the data.

Part II: Balancing the Scales: Fairness in Resource Allocation: The second part steps back into the cleaner world of classic algorithms, where there is no historical data involved, just inputs and outputs. Our central focus here is how resources should be allocated among individuals. Such problems are ubiquitous in everyday life and arise at every level of social organization, from families to states and even global systems, often with very high stakes. In this part, we will examine settings such as chore division, school assignment, hospital placement, carpooling arrangements, and even the organization of blockchains. While part I is heavily influenced by statistical analysis, this part draws many of its tools from economics. But just as in the first part, computational considerations will take center stage.

A note on citations. For the sake of readability, I have chosen not to interrupt the main text with detailed citations. References and suggestions for further reading are listed at the end of the book. There I point to key results discussed here, along with recommended material for anyone who wishes to explore algorithmic fairness in greater depth.

While my choice of topics is broad, the topics themselves are not the main point. More important are the opportunities they provide to

(continued...)

INDEX

Note: Page numbers followed by *f* and *t* indicate figures and tables.