UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI STUDI UMANISTICI

PHD THESIS

Bias and Miscomputation

A Philosophical and Formal Framework for Machine Learning Unfairness

CANDIDATE

Chiara Manganini

TUTOR

Prof. Giuseppe Primiero

COORDINATOR

Prof. Andrea Sereni

Academic Year 2024/2025

Università degli Studi di Milano
Corso di Dottorato HUME · XXXVIII Ciclo
Dipartimento di Filosofia Piero Martinetti

Tesi di Dottorato di Ricerca

Bias and Miscomputation

A Philosophical and Formal Framework for Machine Learning Unfairness

CANDIDATA

Dott.ssa Chiara Manganini

ORCID: 0009-0007-9360-8363

Matricola: R13840

TUTOR

Prof. Giuseppe Primiero

COORDINATORE

Prof. Andrea Sereni

Tesi di Dottorato del settore disciplinare M-FIL/02 redatta con il contributo finanziario dell'Unione Europea Next Generation EU e Kube Partners Italy CUP G43C22002260004.

A.A. 2024/2025

Abstract

As Machine Learning (ML) systems are increasingly being used in critical domains, the need for a coherent framework to account for the discriminatory effects of their outcomes becomes urgent. In computer science, existing approaches tend to emphasise the role of ML design, presenting algorithmic fairness as a matter of making better design choices, especially at the level of the data used to train the model. This dissertation challenges the adequacy of such a design-centric perspective on the problem of unfair ML predictions. It does so by reconnecting it with broader and longer-standing issues within the philosophy of computational artefacts that have largely been overlooked in the current debate in the ethics of artificial intelligence. The primary contribution of this thesis is to reframe the analysis of algorithmic discrimination around the notions of use, maintenance, and repair of ML systems. Specifically, I argue that the correctness criteria of an ML system should be reformulated in terms of the contextual convergence of the diverse normative requirements of the agents who use it. Compared to other accounts of ML normativity, this reconceptualisation avoids succumbing to scepticism about implementation ascriptions, while returning a more dynamic and realistic understanding of how normative requirements circulate, feed back, conflict, and adapt across complex ML systems. Crucially, I claim that this shift has the advantage of allowing for a richer understanding of algorithmic fairness, viewing it as a plurality of data repair practices, rather than a static value embodied by certain ML designs. Two main formal contributions follow from this analysis: the introduction of validity criteria for ML predictions and the development of a novel logical framework to reason about the impact of errors in the input data on the fairness of algorithmic outcomes.

Sommario

I modelli di *Machine Learning* (ML) vengono sempre più utilizzati in contesti critici. Di conseguenza, diventa urgente delineare un quadro teorico per studiare gli effetti discriminatori delle loro predizioni. In informatica, gli approcci esistenti tendono a enfatizzare il ruolo della fase di progettazione iniziale di questi modelli, presentando l'equità algoritmica come una questione di migliori scelte di design, soprattutto a livello dei dati di allenamento. Questa tesi problematizza una tale prospettiva fortemente incentrata sul ruolo delle scelte di design per mitigare le conseguenze discriminatorie delle decisioni algoritmiche. Lo fa ricollegando quest'ultimo problema a questioni più ampie e di più antica data nell'ambito della filosofia della tecnologia, largamente trascurate nell'attuale dibattito sull'etica dell'intelligenza artificiale. Il contributo principale di questa tesi è quello di riformulare l'analisi della discriminazione algoritmica attorno ai concetti di uso, manutenzione e riparazione dei sistemi di ML. Nello specifico, sostengo che i criteri di correttezza di un sistema di ML dovrebbero essere riformulati in termini di convergenza contestuale dei diversi requisiti normativi degli agenti che lo utilizzano. Rispetto ad altre interpretazioni, infatti, questa riconcettualizzazione evita di soccombere allo scetticismo riguardo alle attribuzioni di implementazione, restituendo al contempo una comprensione più dinamica e realistica di come i requisiti normativi circolano, creano dei *feedback*, entrano in conflitto e si adattano all'interno dei complessi sistemi di ML. Inoltre, sostengo che questo cambio di prospettiva abbia il vantaggio di consentire una più ricca comprensione dell'equità algoritmica, considerandola come una pluralità di pratiche riparative, piuttosto che un valore staticamente incarnato da determinate scelte di design. Da questa analisi, seguono due principali contributi formali: la definizione di criteri di validità per le predizioni dei sistemi di ML e lo sviluppo di una logica che sia in grado di formalizzare l'impatto degli errori dei dati di input sull'equità delle decisioni algoritmiche.

Acknowledgments

First and foremost, I am grateful to my supervisor, Giuseppe Primiero, for believing in this project and for providing guidance throughout, especially in the early stages. I thank Kube Partners for generously contributing to funding my research, and the reviewers of this thesis, Sabina Leonelli and Vaishak Belle, for taking the time to carefully read my work and offering their insightful comments. To Vaishak, I am additionally thankful for giving me the opportunity to carry out part of my research in his lab at the University of Edinburgh, which has been a unique professional and personal experience. Finally, I owe my gratitude to Andrea Guardo, who has been not only my professor and Master's thesis supervisor but also my first academic mentor, encouraging me to navigate the academic world and pursue an academic career. Moreover, the first chapter of this thesis would never have come to light without the countless hours of discussion with him.

My PhD has been packed with experiences my younger self would probably never have dared to dream of: presenting my own work at prestigious universities, solo travelling outside of Europe, and even founding a startup! On this wonderfully unpredictable journey, a group of colleagues — mostly, brilliant philosophers and computer scientists — have soon become the first people I genuinely turned to, sharing achievements, ideas, free time, and a good dose of worries, too. To all of them, I want to say a heartfelt thank you. To those in MIRAI and in the Logic, Information, and Computation Group: Francesco, Alessandro, and especially Greta, with whom I spent the longest and most rewarding hours at the whiteboard. To those in my PhD programme, who made me a sort of honorary member of the 37th cycle: Yazan, Marina, Sara, Lucia, and Rodrigo. To those at the Politecnico di Milano, Camilla and Giacomo, who involved me in the Meta group in literally every way possible. To those I met in Edinburgh at the School of Informatics – Xenia, Jessica, Sophie, Katya, Alessio,

Alberto, Marcell, Inigo, Victoria, and Wenhao – with whom I organised, besides many hikes to the Scottish Highlands, a rather unlikely yet legendary "International Conference of Visiting Students". Finally, to my favourite conference buddy, Fahim, with whom I regularly planned to meet at computer science conferences across Europe to catch up.

A special mention goes to my dearest friends of a lifetime, Francesca, Raffaello, and Federico, who constantly supported me through the ups and downs of this journey. Competing together in the Italian Logic Games in 2016 remains one of my fondest memories of high school and of our friendship; devoting a large part of my PhD to logics for fair AI is, therefore, a bit of your merit and your fault.

Finally, Patrizia and Serena – my beloved family – and Walter: thank you for always being there no matter what; this thesis is for you.

Contents

Introduction **1**

1 A Sceptical Paradox for Computational Artefacts **7**
1.1 Introduction . . . 7
1.2 Related Work . . . 9
1.3 Two Theories of Computational Artefact Correctness . . . 14
1.4 The Rule-Following Paradox and Its Sceptical Solution . . . 17
1.5 Replicating the Paradox . . . 19
1.6 Machine Functions and Machine Functionalism . . . 21
1.7 Intentionalism Without Semantic Intentions: A Pragmatic View . 24
1.8 Conclusion . . . 27

2 How to Do Things with Bits. A Pragmatic Framework for Data-Driven Miscomputations **28**
2.1 Introduction . . . 28
2.2 Background: the LoA Taxonomy of Errors . . . 29
2.3 An LoA Taxonomy of ML Errors . . . 31
2.3.1 Errors of Design . . . 33
2.3.2 Errors of Prediction . . . 34
2.3.3 Errors of Justification . . . 35
2.4 Pragmatic Miscomputations . . . 38
2.5 A Pragmatic View of ML Miscomputations . . . 41
2.6 Defining Formal Validity Criteria for Machine Learning Models . . 45
2.6.1 The Story So Far: On the Validity Criteria of Deterministic Computer Simulations . . . 47
2.6.2 ML as Opaque and Nondeterministic Computer Simulation 50
2.7 Conclusion . . . 53

3 Algorithmic Fairness as a Repair Practice **55**
3.1 Introduction . . . 55

3.2 The Design Paradigm and its Critics . . . 56
3.3 On Maintaining and Repairing AI . . . 60
3.4 Algorithmic Unfairness as a Design Error . . . 63
3.5 Three Problems for the Design-Centric View of Algorithmic Fairness 67
3.5.1 Just Fixing Errors Can Miss Their Value . . . 67
3.5.2 The Atemporal Illusion of Fairness-by-Design . . . 69
3.5.3 The Fairness-Accuracy Trade-off as a Design Myth . . . 70
3.6 Conclusion . . . 72

4 Data Speak but Sometimes Lie. A Formal Approach to Data Bias and Algorithmic Fairness 75
4.1 Introduction . . . 75
4.2 A Medical Example . . . 78
4.3 Preliminaries . . . 80
4.4 Modelling ML Decisions as Ulam Games . . . 82
4.4.1 Modelling ML Features . . . 82
4.4.2 Modelling ML Decisions . . . 84
4.4.3 Modelling Data Bias . . . 85
4.5 Model-based ML Testing Within Ulam . . . 87
4.5.1 The (UMO) Rule of Constrained Monotonicity . . . 89
4.5.2 A Generalised (UMO) Rule for Weighted Evidence . . . 91
4.6 Related Work . . . 95
4.6.1 Ulam Game for Data-Driven Reasoning . . . 95
4.6.2 Logical Formalisations of Bias . . . 96
4.6.3 Fairness and ML Correctability . . . 96
4.7 A Proof-Theoretic Interpretation of Biased Predictions . . . 98
4.7.1 Preliminaries . . . 98
4.7.2 Inference Rules . . . 100
4.7.3 A Completeness Result for $\mid\!\sim_{RJ}$ . . . 106
4.8 Conclusion and Future Work . . . 113

Conclusion 115

Bibliography 120

Introduction

Rethinking Algorithmic Fairness Between Old and New Issues in the Philosophy of Technology

Machine Learning (ML) systems are increasingly being used in high-stakes domains such as healthcare, criminal justice, and finance, due to their superior predictive abilities compared to conventional rule-based ones. As this technology begins to shape the way large-scale critical decisions are made in our society, the need for structured approaches to address the discriminatory effect of ML outcomes becomes more urgent than ever.

In the last decade, tackling the phenomenon of *algorithmic unfairness* has been the goal of a growing number of 'fair ML' approaches within the computer science community. Broadly speaking, this family of methods has ended up returning what might be called a *design-centric view* of algorithmic fairness. According to this perspective, unfair predictions are ultimately caused by poor design choices – especially with regard to the quality of the data used – tragically gone undetected during the construction of the ML model. Overall, this has reinforced the idea of ML unfairness as an unintended, yet largely inevitable side effect of ML systems.

The design-centric view of algorithmic fairness influences how fair ML researchers and practitioners materially devise mitigation strategies. Therefore, establishing whether this understanding is adequate in the first place becomes of primary importance. Despite this, the fundamental concepts upon which such

a view rests – intended functionality, side effect, and error – have rarely been acknowledged as deserving proper philosophical scrutiny in their own right by fair ML scholars.

This dissertation seeks to build a bridge between long-standing philosophical debates on the nature of technical artefacts and the current discussion on algorithmic fairness: What exactly constitutes the intended functionality of an ML system? What distinguishes its proper functionality from a mere side effect? What counts as an ML error? Philosophers of technology have long debated similar questions, well before the surge of the fair ML research programme. Originating in the 1980s in relation to technical artefacts, by the 2010s these issues were adapted to the particular context of computational ones, understood as a subset of the former category. Once we look more closely at the relatively new problem of algorithmic fairness, these core questions resurface with renewed significance and urgency. The main methodological contribution of this work is to show that revisiting these debates clarifies how algorithmic unfairness should be conceived and, consequently, tackled.

The central aim of this dissertation is to clarify how the problem of unfair predictions should be philosophically understood, both in terms of its 'diagnosis' and 'therapy'. In line with the wide-ranging methodological approach just motivated, this interrogative is approached from quite afar, by preliminarily investigating a number of core and more general aspects of the nature of ML artefacts, regarding their ontology, normativity, and epistemology. More specifically, the theoretical contributions of this thesis are guided by four main research questions:

Research Question 1 *Do existing accounts adequately address the normativity of ML systems?*

Research Question 2 *Under what conditions does an ML system malfunction?*

Research Question 3 *In what ways can we know whether an ML system malfunctions?*

Research Question 4 *In what ways is ML normativity relevant to our understanding of algorithmic fairness?*

To address Research Question 1, Chapter 1 offers a preliminary examination of existing accounts of computational artefact functions. These positions are critically assessed from what is arguably the most uncompromising perspective possible, namely, a form of radical scepticism inspired by Kripke's reflections on semantic normativity. Kripke's well-known 'rule-following paradox' is here replicated for the computational domain, leading to the conclusion that existing theories of computational correctness cannot determine which function a given artefact implements. To avoid the untenable result of implementation indeterminacy, the paradox compels us to adopt a less intentionalist account of computational functions – one in which correctness is not grounded in the intentions of the designer of the artefact, but on the pragmatic convergence of uses established by the plurality of agents interacting with it. A recent proposal along these lines is provided by the User Levels theory, which I introduce and discuss.

Chapter 2 builds on this conclusion by extending the User Levels account to the specific context of ML systems, in order to address Research Question 2. Compared to existing intentionalist accounts of computational errors, the User Levels theory is argued to be better suited to capture ML miscomputations. The normative rigidity of the former theories, in fact, appears limiting when it comes to complex ML systems whose criteria of correctness are markedly dynamic, i.e., shift between multiple agents, feed back from the output, and involve non-deterministic outcomes. Based on the User Levels account, a novel taxonomy of ML errors is hence introduced and discussed.

After addressing the ontological problem of ML normativity, the discussion turns to Research Question 3, which concerns the corresponding epistemology. Drawing on the philosophy of science and the formal verification literatures, the final part of Chapter 2 lays the groundwork for a formal definition of the epistemological criteria of validity that are most appropriate for these data-driven computational artefacts. The work revisits three traditional formal definitions of simulation relations, adapting them to the ML case. For ML systems, these relations must be substantially revised because the objects to which they apply are characterised by non-determinism and opacity. To account for these, probabilistic and weak versions of the simulation relations are proposed in order to capture different levels of isomorphism between an ML model and its target system. On this basis, three corresponding validity criteria are introduced and discussed.

The second half of the dissertation, constituted by Chapters 3 and 4, is devoted to Research Question 4, concerning algorithmic fairness. Specifically, in Chapter 3, the broader pragmatic picture of ML normativity returned by the previous two chapters is eventually connected to the problem of unfair ML predictions as follows. While prevailing fair ML conceptualisations tend to overemphasise the role of design choices in mitigating unfairness, the User Levels theory invites us to shift our focus from the moment of artefact design to the set of human activities that continuously challenge, preserve, and renegotiate the normativity of the artefact *after* its creation. From this shifted perspective, algorithmic fairness describes a plurality of repair practices, rather than as a static value embodied in specific ML design choices. Specifically, I emphasise the theoretical importance that data repair and data maintenance acquire within this novel epistemological paradigm.

Finally, Chapter 4 directly follows up on this claim, providing a concrete example of what a practice of data repair might look like in the context of fair ML. Specifically, I focus on a logic to reason about how, under specific assumptions, correct information can be retrieved from gappy, inconsistent, and corrupted observations. Crucially, in line with my previous claim that algorithmic fairness should aim at ML repair rather than ML redesign, the proposal I put forward is not aimed at *correcting* biased data nor at *retraining* the ML model to align with a predetermined fair distribution. In contrast, it aims to reason about the way in which a prediction made on biased ML features can be *repaired* by modelling the noise contained in these features. This logical machinery is hence exploited to formalise how ML predictions are impacted by data bias, i.e., the uneven distribution of data noise between sensitive groups, and how this should be accounted for by a fairness-aware ML system.

Taken together, the four chapters build a conceptual bridge between the relatively new and urgent debate on algorithmic fairness and a longer philosophical tradition of issues on the normativity of technical artefacts. By reconsidering ML fairness beyond simply a (however technically complex) design task, but as a deeper issue concerning how we use, repair, and maintain computational systems, the dissertation aims to expand the philosophical foundations of the field and to open new directions for both conceptual analysis and technical practice.

As the name 'algorithmic fairness' suggests, the object of this work is

inherently multidisciplinary, encompassing computer science and logic, as well as ethics and moral philosophy. In light of this, the dissertation employs a combination of methodologies. Conceptual analysis, which remains the preferred and primary tool of inquiry, is progressively complemented by a variety of formal methods. The result is a coherent progression: the first two chapters refer to two prominent tools of formal verification, namely temporal logic (Section 1.2) and isomorphic relations (Section 2.6); the third chapter engages with existing mathematical formalisations of algorithmic fairness and causal models (Section 3.4); and the fourth culminates in a novel logical framework for reasoning about data bias. Throughout the discussion, the potential high stakes of unfair ML predictions are highlighted by references to real-world, societally-critical decision scenarios taken from biometrics (such as the face recognition system discussed in Section 2.1), access to education (such as ML predictions of college success discussed in Section 3.4), and healthcare (such as ML evaluations of intensive care risk during the COVID-19 pandemic in Section 4.2).

The dissertation reworks a selection of materials that have been developed over the past three years. In particular, Chapter 1 corresponds to an accepted article [104]; Chapter 2 corresponds to a published article [24], with the exception of Section 2.6, which instead draws from an accepted contributed chapter [107]. Chapter 3 constitutes a stand-alone work that is currently being reviewed in a philosophy journal. Chapter 4 corresponds to a published paper [105], with the exception of Section 4.7, which draws on a previously published contributed chapter [106].

What follows is a complete list of published contributions by the author of this dissertation within her PhD project.

1. Heilmann,[1] X., Manganini,[1] C., Cerrato, M., Kestel, L., and Belle, V. "A Neurosymbolic Approach to Counterfactual Fairness". *Neurosymbolic Artificial Intelligence. Forthcoming.*

2. Manganini, C. "A Sceptical Paradox for Computational Artefacts". *Philosophical Inquiries. Forthcoming.*[2]

[1]These authors contributed equally to this work.

[2]Best Paper in Metaphysics, Young Scholar Prize at SIFA (Società Italiana Filosofia Analitica) 2025.

3. Manganini, C., Corsi, E. A., and Primiero, G. "Data Speak but Sometimes Lie. (2026). A Game-Theoretic Approach to Data Bias and Algorithmic Fairness". *International Journal of Approximate Reasoning*, 109608.

4. Manganini, C. and Primiero, G. (2026). "Defining Formal Validity Criteria for Machine Learning Models". In: *Durán, J.M., Pozzi, G. (eds) Philosophy of Science for Machine Learning.* Synthese Library, vol 527. Springer, Cham.

5. Buda, A. G., Manganini, C., and Primiero, G. (2025). "A Philosophical Framework for Data-Driven Miscomputations". *Philosophies* 10.4.

6. Heilmann,[3] X., Manganini,[3] C., Cerrato, M., and Belle, V. "A Neurosymbolic Approach to Counterfactual Fairness". (2025). *Proceedings of The 19th International Conference on Neurosymbolic Learning and Reasoning, in Proceedings of Machine Learning Research.*

7. Manganini, C. and Primiero, G. "Reasoning With and About Bias". (2024). In: *Hosni, H., Landes, J. (eds) Perspectives on Logics for Data-driven Reasoning. Logic, Argumentation & Reasoning*, vol 35. Springer, Cham.

8. Manganini, C. and Primiero, G. "Reasoning With Bias". (2023). In: Proceedings of the 1st Workshop on Fairness and Bias in ML co-located with 26th European Conference on Artificial Intelligence (ECML 2023), Kraków, Poland, October 1st, 2023. In: *Calegari, R., Tubella, A. A., González-Castañé, G., Dignum, V., and Milano, M. (eds), vol. 3523. CEUR Workshop Proceedings.*

[3]These authors contributed equally to this work.

Chapter 1

A Sceptical Paradox for Computational Artefacts

1.1 Introduction

Coding is an inherently linguistic endeavour based on the use of programming languages, written idioms largely derived from mathematics and characterised by a peculiarity: making computers "do things".

In *Wittgenstein on Rules and Private Language* [91], or *WRPL* for short, Kripke puts forward a very bizarre paradox about (any) language. The paradox goes roughly like this: since there is ultimately no such thing as "meaning something by a certain sign", no word can have meaning either. How does Kripke's conclusion affect the linguistic activity of coding? And are these consequences relevant to our understanding of what computers are and how, by coding them, we make them "do things"?

The present work seeks to shed some clarity on these philosophical worries by juxtaposing two distinct questions: one in the metaphysics of language, or metasemantics ("What fixes the *meaning* of a word?") and one in the metaphysics of computer science ("What fixes the *function* of a computational artefact?") to show that they run, in a very relevant sense, in parallel. To respond to both, in fact, the respective received philosophical views presuppose the existence of a specific mental state of "meaning something by a certain word", which goes under the name of 'semantic intention'.

As for the former question, a well-known appeal to semantic intentions is given right at the beginning of *WRPL* by the sceptic's interlocutor: I mean addition and not, say, quaddition[1] by the sign "+" because, until now, I have had the

[1]Where the bizarre quaddition function, denoted by the "quus" operator $\oplus$ (Kripke 1981:

semantic intention to mean the former, and not the latter, by that sign. On this basis, it is metalinguistically correct for me to give "125" rather than, say, "5" as an answer to "68+57" [91, p. 8]. This response soon becomes the target of the objections of the sceptic, who argues that there is no such thing as "meaning something by a certain sign", to the conclusion that there is ultimately no way to determine which answer I should give.

Likewise, in the philosophy of computer science, both Primiero [128] and Turner [154] address the latter question – "What determines the function of a computational artefact?" – by endorsing an intentionalist view of computational functions. Namely, the function of a computational artefact is fixed by the content of its functional specification, a mathematical object that represents the intentions of its designer. One might correctly notice that these intentions spell out what the designer wants the artefact to *do* (e.g., managing bank transactions, controlling traffic lights, providing real-time communication) and therefore are not *prima facie* the target of Kripke's paradox, which is about semantic intentions, i.e., what an agent wants a sign to *mean*, thus getting us back to the previous question. In the present work, however, I show that both leading accounts of what it is for a programmer to "want a computational artefact to do something" ultimately entail that the same programmer must entertain semantic intentions directed at the semantic rules of the programming language itself, causing the problem of meaning to resurface at a meta-level. For this reason, the rule-following paradox can be replicated for the computational domain, this time to the conclusion that it is indeterminate whether a computational artefact implements this or that function.

Being such a claim unacceptable in much the same way as the non-existence of meaning is, it forces us to explore a sceptical solution. With regard to this, I take into examination a recent proposal by Buda and Primiero [25] as a possible candidate. Simply put, according to this account, a computational artefact can be said to correctly implement a function when there is a convergence of a particular type among its different users, much recalling Kripke's idea that the agreement on the use of a word among members of a linguistic community – rather than a mental state – is what ultimately grounds meaning ascriptions.

The work is structured as follows. After introducing the distinction between intentionalist and evolutionary accounts of technical artefact functions at large,

9), is defined as:

$$x \oplus y = \begin{cases} x + y \text{ if } x, y < 57, \\ 5 \text{ otherwise} \end{cases}$$

I focus on the notion of functional specification as a formalisation of the intentions of the computational artefact's designer (Section 1.2). Section 1.3 offers a detailed exposition of Primiero's and Turner's accounts, uncovering that they both entail the existence of semantic intentions. I then summarise the rule-following paradox alongside Kripke's famous sceptical solution (Section 1.4). Hence, I show how the paradox can be replicated for the existing intentionalist formulations, to the conclusion that it is not possible to determine which function is being implemented by a computational artefact (Section 1.5). On a more exegetical note, I discuss Kripke's response to functionalism (Section 1.6), to finally discuss a possible sceptical solution to the paradox (Section 1.7) and draw some concluding remarks (Section 1.8).

1.2 Related Work

The question of the significance of Kripke's paradox for our understanding of computing machines is not entirely new. Since the publication of *WRPL*, it has gained attention in relation to the debate on the nature of computation *qua* physical process in material systems [26, 27, 138, 139, 140, 144, 145, 147]. However, most of the times these works avoid any clear-cut connection with the other – largely independent – strand of the debate within the philosophy of computer science, which instead deals with the question of what sort of technical artefacts computers are (their ontology), and what grounds the notion of correct functioning for them (their normativity). Some other times, these connections are made explicit, but fatally misconceived. For instance, Buechner [27] maintains that *WRPL* ends up undermining what he calls "the standard view of what computers are". Upon such a view, there is an objective fact about the function a computer is implementing – a fact that can be directly read off from its outputs:

> if [...] the physical computer is operating normally, then when the number '2' is input to that computer, and it physically computes the number '4,' then the physical computer is physically computing the square function. If the physical computer physically computes the number '5,' when the input is the number '2,' then it has suffered a breakdown of some kind in computing the square function. [27, p. 496]

The problem is that the view described by Buechner is not quite "standard",

at least within the philosophy of computer science, where both of the most prominent ontological accounts [128, 154] clearly state that no inherent physical property of the machine, including the output it returns, can determine what function it is implementing. This is ultimately a matter of what constitutes the normal conditions of functioning and what are its desirable outputs. Both conditions are stipulated by the agent who is behind the computational artefact's design.[2] Therefore, while Buechner's claim that *WRPL* undermines our current understanding of computers may be correct – and I will argue it is – his argument needs to be formulated differently.

Notably, among the philosophers of computer science, Turner has consistently integrated the rule-following considerations into a theory of computational artefacts [153, 154, 156]. In particular, at the very end of his most comprehensive work on the topic, *Computational Artifacts. Towards a Philosophy of Computer Science* [154], he discusses Kripke's paradox, and the objections moved against it by McDowell [108] and Miller [115]. The work closes with a quite general remark on the philosophical import of the paradox: "This is a complex and ongoing debate. Our objectives here are modest: we have tried to put enough bones on matters to indicate how it affects the issue of program correctness." [154, p. 239] Turner, clearly, worries that the paradox undermines his own theoretical proposal – and I will argue he is right.

Philosophers of technology more in general have long engaged with the question of the nature of computational artefacts. Central to this debate is the notion of function, as computational artefacts are human-made objects that serve specific purposes – like managing bank transactions, controlling traffic lights, and providing real-time communication. As such, they belong to the broader category of technical artefacts, together with objects like hammers, cars, mirrors, and tables. For this reason, I will begin by introducing the debate on technical artefact functions at large.

Simplifying a lot, the design and the use of technical artefacts co-occur in human agents with a specific mental state that involves desires, beliefs, and action plans, that we normally call "intention". For example, I build a box with the intention of carrying heavy objects or use a fork with the intention of eating. Henceforth, I will refer to such a mental state by "teleological intention" to mark the distinction with that of semantic intention, introduced earlier in relation to Kripke's sceptical paradox.

[2]Buechner briefly mentions this position, arguing that this, too, is vulnerable to Kripke's argument against functionalism [27, p. 506], but without elaborating much. The present work might be regarded as expanding on this very claim.

In the current debate, there is a lack of consensus on how we should understand the relationship between these teleological intentions and the functions of technical artefacts. Broadly speaking, two main views have been discussed in the literature. According to the prevailing, intentionalist view of functions [75, 110] teleological intentions are *necessary* for ascribing functions to technical artefacts. In the strongest formulation, teleological intentions are *necessary and sufficient* for functional ascription: in virtue of its being designed for ϕ-*ing*, the function of artefact kind K is to ϕ. In other words, "all we have to do is talk to the microwave engineers to know what the microwave oven's function is" [53, p. 6]. More cautious intentionalist positions see teleological intentions as just *necessary* – yet not sufficient – for functional ascription. In this way, the intentionalist can rule out undesirable "phantom functions" – that is, functions associated with a kind K that is actually incapable of performing them [74]. We might want to require that other support conditions hold as well. For example, that at least certain K-tokens are capable of ϕ-*ing,* or that the agent has justifiable reasons to intend to use an artefact of kind K for ϕ-*ing* [75].[3]

By contrast to intentionalism, on the evolutionary view [53, 78, 79, 169, 124, 125], intentions are *not necessary nor sufficient* for ascribing functions to an artefact. By analogy with natural selection, artefacts are like phenotypes that have survived a history of selective pressures in virtue of some kind of utility of theirs. This very utility corresponds to their function. Put differently, the function of artefact kind K (e.g., chairs) is the effect by which past K-tokens (e.g., past chairs) have been continuously replicated up to the present moment (e.g., to provide seating). For the evolutionary theory, teleological intentions and artefact functions are two different concepts, co-extensive at best. Artefacts can in fact replicate for all sorts of unintended reasons. For instance, we might observe that regardless of the intentions high-heeled shoes were designed for, their true function is to oppress their wearers (by limiting their movements, etc.) as such an effect has arguably been the ultimate driver of their long history of selection [146].

Despite their theoretical differences, evolutionary and intentionalist accounts often agree in practice on the functions of many artefacts. This is because teleological intentions can be seen as strong selection pressures–for example, a designer may discard all K-tokens that fail to serve the intended purpose.

[3] I will return to these support conditions in Section 1.5, in the context of a form of scepticism about computational artefact functions that I contrast with my own.

Moreover, both views also share the same notion of function, intended as 'what the artefact is supposed to do', which may differ from what it *actually does*, given its current physical properties. To mark this difference, henceforth I will refer to the notion of function (the one that intentionalism and evolutionism share) by "teleological function", and to the latter by "causal-role function", borrowing from [45]. Accordingly, the discussion that follows will focus exclusively on teleological functions. Specifically, in the remainder of the section, I shall deal with the question of what determines the teleological function of computational artefacts.

What does it mean that a computational artefact is made for managing bank transactions, controlling traffic lights, or providing real-time communication? On the surface, these functions seem just as those of candles (to illuminate) or sunglasses (to protect the eyes from intense light). However, unlike candles or sunglasses, computational artefacts perform their functions through physical computations. The debate over the nature of physical computation is complex and spans a range of philosophical views of how we should define such a notion, whether semantically [137], mechanistically [123], syntactically [60], or causally [33]. For our purposes, it will suffice to say that it is broadly accepted that the physical computations involved in (digital) computers are defined and controlled through symbols.

To say that computational artefacts perform their teleological functions through physical computations and the latter are defined and controlled symbolically is to say that the teleological functions of computational artefacts are defined and controlled symbolically. Computer programs are typically taken to be the formalisation *par excellence* of such functions. However, this quite intuitive idea has to undergo some refinement, based on the following observations. First, two computer programs can in principle fulfil the same teleological function while not being identical at the symbolic level. For example, I can code two programs for computing the greatest common divisor of a tuple of integers using different programming languages, for example Python and Java. These programs, respectively called P and J, fulfil the same teleological function, despite having a different syntax. Furthermore, P and J might also use a different sequence of instructions to calculate the greatest common divisor. For instance, the sequence of instructions realised by P could be computationally more efficient than those of J. Still, both programs would be said to fulfil the *same* teleological function.

Based on these two observations, it can hence be concluded that the

formalisation of a computational artefact function is more abstract than both its program and the sequence of instructions it realises. As Turner says, this formalisation must distil "the what from the how" [154, p. 33], meaning that it should abstract away what the artefact should do from how it should do it. This is precisely what we call "functional specification" in computer science.

Concretely, a functional specification is a set of well-written formulas, expressed in a suitable mathematical language. From case to case, specifications can be expressed using various mathematical languages, like linear logic, temporal logic, set theory, and many more. Regardless of the language used to express them, specifications play a pivotal role in the software engineering practice, when it comes to verifying the correctness of computational artefacts.

For instance, let us think of a program that controls a safety-critical system, like the traffic light system placed at the intersection of two roads, North and East. Suppose that the designer of the system holds the following teleological intention: making vehicles navigate the intersection as safely as possible.[4] Simplifying, this translates into the following temporal logic condition:

$$S : \Box \quad (North = Red \vee East = Red)$$

S imposes that it must always be the case (symbolised by the box operator) that at least one traffic light displays red.[5] Verifying the traffic light system corresponds to finding out whether S is satisfied by any possible physical state the system might step into. Concretely, this is done either empirically or formally. On the one hand, software testing allows one to gather empirical information about the lower bound of S's satisfaction – i.e., from a battery of N successful tests, we can conclude that S is satisfied in *at least* N cases. On the other hand, formal verification techniques give us stronger guarantees on S's satisfaction, as every possible state of the system is automatically inspected, until a counterexample is found (if any). This gives us information about the upper bound of this satisfaction – we can conclude that S is satisfied in *at most* N states of the systems.

To close, it is important to stress that both software testing and formal verification are possible because, operationally, program specifications establish the criteria of correctness for computational systems. From a philosophical

[4]Notice that, in this example, the teleological intention of the designer does not involve pedestrians. This will become relevant in Section 1.7.

[5]One might notice that S leaves completely indeterminate what actual control sequence should make the traffic lights transition from one state to the other (from green to yellow, from yellow to red, from red to green).

perspective as well, the normativity of specifications has been central to the debate on the ontology of computational artefacts, as exemplified by the accounts of [128] and [154], which I discuss in the next section.

1.3 Two Theories of Computational Artefact Correctness

Drawing from the method of abstraction introduced by Floridi [58], Primiero [128] has proposed a view of computational artefacts as objects that exist at many different Levels of Abstraction (LoAs). The theory individuates six LoAs, ordered from the most to the least abstract: the designer's intention level, the functional specification level, the algorithm level, the program level, the machine code level, and the hardware level [7]. Importantly for our discussion, the uppermost LoA is described as "express[ing] the objective of constructing systems able to perform some desired computational tasks" [7, Section 2.1], which exactly corresponds to the notion of teleological intention as we defined it in the previous chapter.

Primiero defines implementation as a relation between a given LoA and a more abstract one. Specifically, saying that "A implements B" is equivalent to saying that "A is an instantiation of B at a lower level of abstraction". The relation is surjective, i.e., can map different elements of a LoA to the same element belonging to a level above (i.e., more abstract). This makes sense of the observations of the previous section, regarding programs P and J: although they implement different algorithms to calculate the greatest common divisor (say, GCD and GCD', respectively), these implement the same functional specification, say $\mathcal{GCD}$. Accordingly, in the theory every level acts as a control device on the correctness of the one immediately below.[6] For instance, both GCD and GCD' are correct algorithms as long as they constitute valid procedures to yield an output that satisfies the mathematical properties spelled out by $\mathcal{GCD}$. In their turn, P and J are correct programs as long as they comply with the sequences of instructions specified, respectively, by GCD and GCD', and so on and so forth for the levels below.[7] For an illustration of the implementation and abstraction relations, see Figure 1.1.

Both implementation and abstraction are transitive relations. By

[6] In this way, the concept of correctness is broken down into level-specific criteria [128, pp. 213–214], allowing for a layered taxonomy of information [127] and computational malfunctions [61, 126]. The latter will be discussed in Section 2.3.

[7] Angius and Primiero [5, 6] have formally developed the notion of *computational copy* to reason about classes of programs that realise the same functional specification, like P and J.

transitivity, every level below the intention LoA can be viewed as implementing it and is ultimately judged correct or incorrect with respect to it. If this is so, Primiero's account must address one fundamental question: how can the intention LoA establish the conditions of correctness for the level immediately below, the specification LoA? In other words, how can we validate that a given specification correctly represents the teleological intention of its designer, given that the former is a linguistic object and the latter is not? An act of interpretation seems to be called for:

> The notion of specification requires therefore necessarily an act of interpretation [...]. An analysis of the notion of correctness and validity in physical computation systems preserving both a syntactic qualification of its states and a semantic interpretation of its intended specification seems therefore required. [128, p. 189]

This becomes clearer if we come back to the traffic light example introduced before. In S, the "Red" predicate, as well as the "North" and "East" constants, the disjunction connective, the box operator, and all non-logical symbols (like parentheses and equals signs) are defined in the logical language of S. But, sure enough, the definitions in such a language are made of symbols, too – either logical or natural language ones – whose meaning is ultimately fixed by the author of S. For example, by "Red", the author of the specification could equally mean the colour red, the colour green, or any other property. It is based on the semantic intention held by S's author, that this specification poses logical constraints on red traffic lights, rather than on green ones, or on any other set of traffic lights. The same applies to any other symbol in S. It is therefore clear that such a necessary "act of interpretation", as Primiero calls it, corresponds to semantic intention.

Summarising, the LoA theory ultimately requires the semantic content of functional specifications to be fixed by certain corresponding semantic intentions held by the designer. This is basically all we need to conclude that Primiero's account requires semantic intention to exist in the first place, as a substantive mental state, based on which the correctness of the computational artefact can be ultimately verified.

On Turner's view, instead, the normativity of functional specifications stems from the actual practice of programming. Recalling Austin's locutionary/illocutionary distinction [8],[8] Turner starts with noticing that

[8]For Austin [8], there are three components of an utterance: locutionary, corresponding to the propositional content of an utterance; illocutionary, corresponding to its force (e.g., ordering, asking, declaring, etc.); perlocutionary, corresponding to its effects in the world.

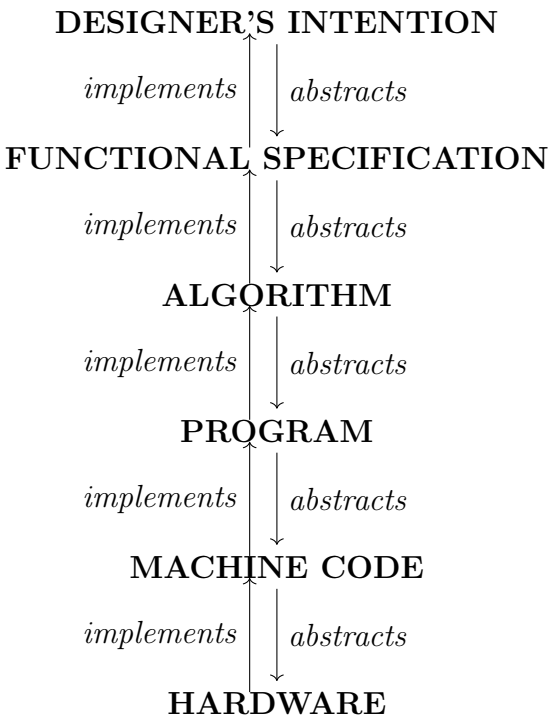

Figure 1.1: The schema illustrates the basic elements of the hierarchy of abstraction, as conceptualised by the LoA ontology.

functional specifications and functional descriptions of a computational artefact's behaviour can have the same semantic content. However, only specifications have the illocutionary force of prescriptions [155, p. 359].[9] The difference between functional specifications and functional descriptions qualifies as a temporal one as well: "if it is given ahead of the construction it is a specification whereas if it is given after construction it is a functional description" [153, p. 142]. Namely, two phases are distinguished. At an earlier stage, the designer establishes the semantic content of the program's specification based on the clients' indications. *Only once this is fixed,* an "intentional shift" occurs [156, p. 24]: such a semantic content stops being just a description – an empirical statement about the artefact's behaviour – and truly becomes the normative standard of the artefact's behaviour (i.e., what the artefact *should do*). Of course, this same shift can iterate more than once during the artefact's lifecycle. For instance, when debugging a program, one can find out that the original specification did not consider a number of possibilities, leading to execution errors and unhandled exceptions. By correcting the code, the programmer is substantially revoking the illocutionary force from the

[9] It is noteworthy that the distinction made by Turner seems to track the one between proper functions and causal-role functions illustrated in Section 1.2.

original specification, to give it to an improved specification, while the old one becomes a mere description of the computational artefact's (unwanted) behaviour.

Turner's theory, in general, offers a realistic account of how functional specifications acquire their normativity in software engineering. However, the entire framework hinges on a crucial assumption: that the locutionary content of program specifications can be semantically determined in the first place. As Turner remarks, there is no way to get around this:

> Unfortunately, as an agent, I can have all sorts of images in my head. How do any such images connect with the structure of an artifact? How do they deliver extensional agreement? While the latter may not be sufficient to provide an adequate account of correctness, it is certainly necessary. [154, p. 232]

Once again, semantic intention seems to be the only way to gauge the locutionary content of program specifications to their written expressions, so that we have to conclude that Turner's account necessarily requires semantic intentions to exist, in order to determine what is the function of a computational artefact and consequently establish its correctness. The remainder of this work is devoted to showing that, due to their reliance on semantic intentions, both Primiero's and Turner's accounts fall prey to a sceptical argument analogous to Kripke's rule-following paradox.

1.4 The Rule-Following Paradox and Its Sceptical Solution

According to Kripke's exegesis in *WRPL*, at the heart of the *Philosophical Investigations* [167], Wittgenstein is formulating and then solving a paradox about the meaning of words. As I reconstruct it, the paradox is an argument in two steps: a premise, constituted by a claim in the metaphysics of mind, and a conclusion concerning the metaphysics of language.

The premise of the paradox is the claim that no fact can constitute our intending something by a given sign – regardless of how we try to characterise it, whether as a mental grasp of an idea, a mental image, a disposition, or a cognitive function. These characterisations correspond, respectively, to what Kripke [91, p. 54] calls the Platonist Response, the Representationalist Response [91, p. 17], the Dispositionalist Response [91, pp. 26–37], and the

Functionalist Response [91, p. 37, note 24]. All these responses count as "straight" responses to this paradox, as they are aimed at proving the paradox wrong on its very own sceptical premise. In this work, I will not revisit Kripke's arguments against each of these views, except for the last, which will be the focus of Section 1.6.

For now, it is enough to note that many commentators have argued that the scepticism raised by Kripke is ultimately metaphysical rather than merely epistemological: it challenges the very existence of a fact that could constitute semantic intention [93, 163].[10] As such, the paradox has deep consequences for metasemantics. In fact, semantic intention is standardly assumed to be what fixes a rule to a word. By denying the existence of semantic intention – the paradox premise – Kripke is substantially rejecting this standard view of meaning: words and rules can exist, but no word can in fact be governed by a rule, as nothing can determine whether a certain word is governed by this or that rule [167, § 28]. To put it differently, as soon as the question about word meaning hinges on the ontological status of semantic intentions, Kripke's sceptical claim inevitably leads to the paradoxical conclusion that there is "no such thing as meaning anything by any word" [91, p. 55]. The conclusion is paradoxical not only in the sense that it is unacceptable and completely counters our proto-theoretical intuitions [167, § 201], but also because it is pragmatically self-defeating – how can it be, in fact, that we come to express in a language an argument about its very impossibility?

In order to escape this paradoxical conclusion, it seems, one must let go of the received view of meaning, which justifies the inference from the premise to the conclusion of the argument, i.e., from the sceptical claim in the philosophy of mind to the sceptical claim in metasemantics. This constitutes the fundamental move at the basis of Kripke's so-called "sceptical solution" [91, p. 22] in contrast to the straight ones mentioned above. The core claim of Kripke's sceptical response goes like this. Semantic intention attributions are useful assertions we make to attribute a linguistic ability, not a mental state, to other speakers. In a community of speakers, Jones' being judged to agree with the responses of the other members in enough cases constitutes the condition to assert – not to *determine* – that they mean addition by the sign "+" [91, p. 92]. Based on this, our everyday talk about semantic intentions and meaning is still legitimate, despite the non-existence of the corresponding facts.

The ongoings of the long debate on Kripke's paradox are difficult to summarise

[10] Although the sceptical claim is fundamentally metaphysical, it has epistemological implications – for example, that one cannot have direct access to one's own semantic intentions [69].

briefly. Here, I will limit myself to mention one: while the sceptical solution is typically understood as endorsing a non-factualist view of meaning ascriptions– that is, the idea that sentences like "Jones means addition by the sign '+'" do not express any factual claims about the world – other scholars [28, 93, 109, 166] have defended a reading in which Kripke allows for the existence of minimal semantic facts, albeit non-mental ones.

1.5 Replicating the Paradox

According to Primiero and Turner, a computational physical system is said to implement a function on the grounds of the teleological intentions of its designer, as they are expressed formally in a linguistic object corresponding to its functional specification. Crucially, both accounts presuppose that semantic intentions are capable of attaching the right semantics to the symbols in the specification. However, if we accept Kripke's claim that there is no such thing as a mental state of semantic intention, this leaves ultimately indeterminate what function is implemented by a computational artefact. By analogy with Kripke's rule-following scepticism, I take this line of reasoning to constitute a sceptical paradox of implementation that jeopardises both of these two intentionalist accounts of computational artefact functions.

The two paradoxes may be seen as two stand-alone independent problems running in parallel, both engendering from parallel questions around the relationship between an abstract and a concrete realm, one in the metaphysics of language (L) and the other in the metaphysics of computer science (C):

(L) What fixes the meaning of a word?

(C) What fixes the function of a computational artefact?

Standard answers to both questions appeal in fact to semantic intentional facts. The symmetry between the two questions holds as soon as we accept the common premise that no fact of the matter corresponds to intending something by a certain sign. This is all we need to formulate the two parallel sceptical claims that share the same antecedent:

(L) Since semantic intention does not exist as a mental state, nothing fixes the meaning of a word.

(C) Since semantic intention does not exist as a mental state, nothing fixes the function of a computational artefact.

The original sceptical claim was that there is no mental fact capable of determining which rule a person is intending by a certain sign. Similarly for the computation domain, no fact can determine which function a machine is implementing, according to any of the two theories of implementation outlined in Section 1.3.

Analogously to the Kripke's original paradox, the scepticism entailed by the paradox of implementation is fundamentally metaphysical rather than merely epistemological. The conclusion, in fact, is not that we can never be certain of the correctness of the machine output – in other words, that there exists a sheer eventuality that the machine may be computing a different function from the intended one *without us knowing that*. Rather, the paradox tells us that regardless of the quantity of evidence we can acquire about a given computational artefact (the output it returns, its internal states, etc.) the truth-value of utterances such as "Computational artefact C implements function F" is always indeterminate, ultimately.

This scepticism being at core metaphysical marks the main difference from another attack to the intentionalist position [79]. In a recent article, Hurshman argues that such a view is unable to ascribe functions to a particular class of computational artefacts, namely machine learning systems.[11] To substantiate his claim, he takes the theory by Houkes and Vermaas [75] as the paradigm for intentionalist views of technical artefact functions[12] and shows that machine learning systems lack the necessary requirements to be ascribed proper functions according to this account. Specifically, the epistemic opacity typical of many machine learning models prevents their designers from holding the right type of beliefs about how they perform their function.[13] Being these support conditions for functional ascription unattended, machine learning systems cannot be ascribed functions from an intentionalist perspective. Therefore, what Hurshman suggests

[11] A machine learning algorithm, rather than being the result of standard manual coding, it is itself the output of a (manually coded) computational artefact that we can call "training engine", whose function is to find an optimal output, given certain constraints. Next chapter will be entirely devoted to key ontological, epistemological, and normative aspects related to machine learning systems and their malfunctions. My claim will be that machine learning systems are, for the LoA theory, computational artefacts that lack a proper functional specification level.

[12] Although oukes and Vermaas programmatically aim at mixing intentionalist and evolutionary elements in their theory, which is in fact baptised with the acronym ICE (Intention, Causal-Role, Evolution) [75].

[13] Although it seems that a problem arises here in relation to those machine learning systems that we cannot legitimately describe as "opaque". Being opaque and being a machine learning algorithm are not co-extensive predicates. In other words, the property of "being a machine learning algorithm" fails to pick all and only the cases he takes to be problematic for the intentionalist, as it is simultaneously too restrictive and too relaxed as a criterion (see [78] for an exploration of the evolutionary view of non-ML computational artefacts).

is a form of epistemological scepticism about the function of machine learning systems, given that we do not have the right of kind of knowledge about their inner workings:

> In both cases, algorithms' past effects do not indicate their performance in new, even subtly different contexts. The inductive justification that we have for using opaque algorithms, which is implicit in ascriptions of evolutionary functions, is fragile with respect to moves across contexts. Knowledge of how an algorithm works could allow one to foresee whether a new context will produce malfunctions or misfunctions. To the extent that an algorithm is opaque, however, it is difficult to predict its reliability in new contexts. [79, p. 91]

By exclusion, it is concluded that machine learning systems can only be ascribed functions in evolutionary terms: similarly to natural evolution, the machine learning function emerges from the initial randomisation of parameters, as the model is trained, validated, and deployed. Therefore, what Hurshman suggests is a form of scepticism about (certain) technical functions which, in contrast to mine, is at core epistemological.

1.6 Machine Functions and Machine Functionalism

The paradox premise, as reconstructed in Section 1.5, concerns the metaphysics of mind. Namely, it claims that no mental fact whatsoever univocally determines what we mean by a certain sign. If we accept this claim – and note that I have *not* rehearsed the reasons for accepting it in the present work – it follows that the meaning of words cannot be determined. The very premise of the paradox of implementation, however, must sound surprisingly false to those who are convinced that human rule-following can be no different in essence from that of machines, like the functionalist. The reason being that the human brain is just a special kind of machine, in particular a complex type of computing machine which cognitively computes a set of functions.[14]

[14] By "machine functionalism" [129], I refer to a philosophical position that combines together the computationalist claim that core mental processes can be described in terms of a Turing-style computational model and the functionalist claim that all we need to do to individuate a mental state is to specify the role it plays within the functional organisation of the mind, as the abstract schema of a Turing Machine does.

Kripke elaborates on this very topic in *WRPL* while discussing dispositionalism [91, p. 30].[15] He takes the functionalist response to the paradox to go as follows. Let us suppose that I am endowed with the technical skills necessary to build from scratch a computing machine that implements the function I intend by the sign "+", i.e., addition. If I succeed, the machine would "simply grind out the right answer, in any particular case, to any particular addition problem. The answer that the machine would give is, then, the answer that I intended" [91, p. 33]. Put differently, since I set up the machine to reproduce my own rule-following, *the fact* that it is implementing addition is evidence of the content of my own semantic intention. Namely, the machine's implementing addition would be evidence of my own intending addition; contrarily, the machine's implementing quaddition would be evidence of my own intending this other function.

It is crucial to note that this implementation fact must itself be independent of any mental state of mine – like my teleological intentions – or we risk falling into an infinite regress. In fact, if functionalism is true, my mental states correspond to cognitive functions in my brain, which is a machine in its turn. Therefore, it must be possible to decide whether the computing machine I have built implements addition or quaddition *just by looking* at its behaviour.

Of course, for Kripke this possibility is not given: no physical fact of the matter determines which function a machine is implementing. There are at least two inherent reasons for this gap [91, p. 34]. The first is related to its limitedness: in its entire lifespan, a concrete machine only receives a finite number of inputs and returns a finite number of outputs, hence covering just a limited portion of the function it is taken to implement. Secondly, as any other concrete object, the machine is subject to the inevitable phenomenon of physical decay, so that the mere possibility of a malfunction makes even the (limited) outputs actually obtained from the machine not conclusive of any fact of implementation whatsoever.[16] A case in point that perfectly illustrates Kripke's argument is the phenomenon of planned obsolescence of computational artefacts like smartphones and televisions. These artefacts are manufactured to

[15] Indeed, Kripke sees dispositionalism and functionalism as one: the former, in fact "can be viewed as if it interpreted us as machines, whose output mechanically yields the correct result" [91, p. 33]. The core dispositionalist claim is that possessing a particular semantic intention is identical to having a corresponding disposition, in the same way that the fragility of a crystal glass is identical to its disposition to break. In other words, I mean addition by "+" because up until now I have been disposed to give "125" as the answer to "68 + 57", even though such a disposition never manifested in the past.

[16] As the reader will notice, these reasons basically repropose two fundamental points made by the sceptic earlier in *WRPL*, regarding the famous addition/quaddition example.

have a suboptimal lifespan by design, leading to the premature failure of certain functionalities in order to push consumers to replace these devices with new ones. According to Kripke's reconstruction, we should regard the diminished capacity of this hardware as a proper functioning, as long as intended by its designer.

For these two reasons, for Kripke, there is ultimately no way of labelling the output of a machine as correct or incorrect, without establishing which outputs it is supposed to return beforehand. And, clearly, the designer is the one who decides that:

> How is it determined when a malfunction occurs? By reference to the program of the machine, as intended by its designer, not simply by reference to the machine itself. Depending on the intent of the designer, any particular phenomenon may or may not count as a machine 'malfunction'. A programmer with suitable intentions might even have intended to make use of the fact that wires melt or gears slip, so that a machine that is 'malfunctioning' for me is behaving perfectly for him. Whether a machine ever malfunctions and, if so, when, is not a property of the machine itself as a physical object but [...] as stipulated by its designer. [91, pp. 34-35]

To take stock, Kripke is claiming that the designer's teleological intentions are necessary for grounding the normativity of a computational artefact. In the context of *WRPL*, this is instrumental to the refutation of the functionalist response to the paradox.

In the context of the present discussion, instead, what should be stressed is that such a claim takes the form of a defence of the intentionalist view of artefact functions. In Section 1.5, in fact, I called intentionalist any account that treats the designer's teleological intention to be necessary for an artefact to have a function. This raises a pressing question: how can Kripke's endorsement of the core *intentionalist* position be reconciled with the fact, apparently at odds, that the very intentionalist view falls prey to the sceptical paradox of implementation? This will be the object of our final discussion, about the sceptical solution to it.

1.7 Intentionalism Without Semantic Intentions: A Pragmatic View

Kripke uses the core intentionalist claim (i.e., that teleological intentions are necessary for ascribing functions to artefacts) to undermine the functionalist response to the paradox of rule-following. However, if the paradox of rule-following is successful, the very same intentionalist accounts by Primiero [128] and Turner [154] end up leaving implementation ascriptions indeterminate.

In Section 1.4, the sceptical solution to the rule-following paradox was qualified as one that, while accepting its premise, succeeds in saving our everyday talk of semantic intentions and meanings, nevertheless. Analogously, a sceptical solution to the paradox of implementation should be able to make sense of our everyday practice of ascribing functions to computational artefacts, but without entailing the existence of semantic intention. As an additional requirement such a sceptical solution should endorse the core intentionalist claim – i.e., that teleological intentions are necessary to implementation ascriptions – to secure Kripke's objection to machine functionalism discussed in the previous section. To this regard, I am going to show that a recent proposal by Buda and Primiero [25] succeeds to meet these *desiderata*.

The authors move from the observation that the LoA theory account is highly theoretical and ultimately inadequate to account for the complexity of contemporary information technologies. In particular, it is observed that the LoA theory fails to accommodate certain important phenomena in the current coding practice. These include the emergence of deviant uses of computational artefacts, the re-evaluation of bugs as features, and the widespread use of software development methodologies in which there is no clear normative dominance of the functional specification over the implementation.[17] In order to capture these pragmatic aspects of coding, it is argued, the LoA theory should be revised in two ways. First of all, the scope of the theory must be much broadened to include a plurality of agents other than the designer, which include software developers, policymakers, hardware manufacturers, and end-users alike – all called "users" (see Figure 1.2). Secondly, the assumption of a rigid normative direction always flowing from the functional specification to the more

[17] This recalls earlier doubts on the normativity of specifications raised by Cantwell Smith [30, p. 22]: "As a matter of technical practice, specifications tend to be extraordinarily complex formal descriptions, just as subject to bugs and design errors and so forth as programs." In other words, on Cantwell Smith's view, there is no reason to believe that the specification is any more correct than the program. What happens in practice is that the two are adjusted to converge with each other.

concrete implementations of the artefact is to be replaced with one in which "the influence of the specification on the implementation is not any stronger than the reverse relation, in a context of continuous communication and collaboration between the several actors involved in the process" [25, p. 145].

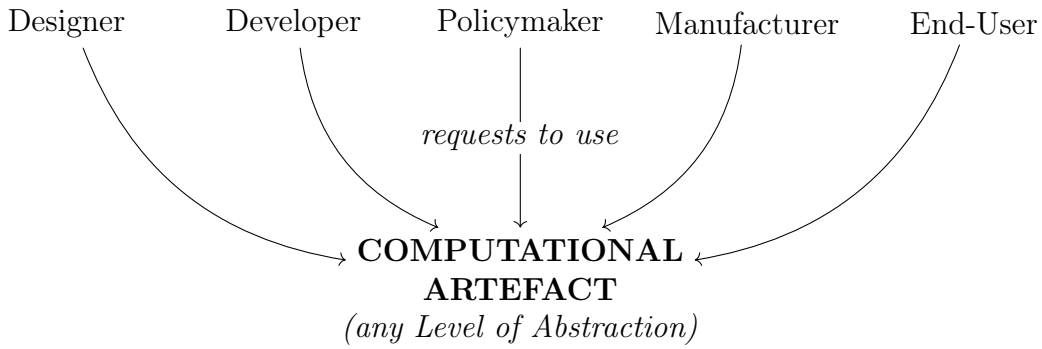

Figure 1.2: The schema illustrates the most common User Levels of a computational artefact, as conceptualised by the pragmatic view of computational functions. With their various requests and modes of use, users exert a pressure on the content of the functional specification, as originally formalised by a special user, the system designer. As a response, the functional specification evolves to explicitly accommodate certain uses and explicitly disallow certain others. Formally, it is a collector of maximally consistent uses of the computational artefact.

To illustrate the full potential of this "pragmatic view" of computational artefact functions, let us go back to the traffic light example. Consider a scenario where a minor hardware malfunction causes one of the two traffic lights to remain red for a few seconds longer than originally established by its specification, before finally switching to green. Pedestrians might start to take advantage of the lax time when the two traffic lights are both red to cross the intersection in a relatively safer way. Arguably, from their perspective, the system is functioning properly when it manifests the glitch, as this makes it possible for them to fulfil their teleological intention of crossing the street in relative safety. However, for the LoA theory, the delayed light switch remains a glitch, as this behaviour ultimately departs from the original teleological intentions of the designer (the pedestrian crossing itself counts as a *deviant* use of the traffic light system, as the latter was designed to allow *vehicles*, not pedestrians, to navigate the intersection safely). On the other hand, the pedestrians' requirement that both lights display red at some point may not even be expressed linguistically at all; nevertheless, this expectation plays a crucial role in shaping their normative judgments about the system.

The pragmatic theory accounts for this scenario differently, by holding that

the functions of computational artefacts are not determined by the content of their functional specifications. Rather, the functional specification just formalises, a posteriori, a set of compatible uses of the artefact, as they emerge through a gradual and dynamic process. This process often begins with the emergence of novel, idiosyncratic uses that gradually evolve into more widespread and innovative uses of a computational artefact, and even to the expert redesign of its functional requirements (Buda and Primiero 2024). These three phases of use provide a more effective framework for understanding the traffic light example: some pedestrians try to cross the street when, by chance, both lights display red (idiosyncratic use). Over time, the very same pedestrians begin to rely on the traffic lights displaying simultaneously red by repetition and habit, and the practice quickly starts to spread among pedestrians thanks to imitation, communication, and mutual coordination (innovative use), to the point that even the system designer may be prompted to redesign its specification S – rather than fix the system – to explicitly accommodate this innovative use into it (expert redesign). The same process may iterate again and again throughout the entire lifespan of the computational system. Before the last phase of expert redesign is reached, designers and pedestrians do have conflicting normative requirements towards the same computational artefact. Specifically, while the delay in the light switch counts as an unforeseen behaviour for the former, it ultimately constitutes the desired output for the latter. This is because the two groups ultimately ascribe different functions to the same system – something the Level of Abstraction (LoA) fails to capture, as it restricts the analysis only to the designer's intentions. To conclude, on Buda and Primiero's view, the system is functioning correctly as S does not explicitly disallow for the possibility that both traffic lights display red simultaneously – indeed, both lights displaying red trivially satisfies the condition expressed by S that at least one of them is red. A line of reasoning like the one just expounded lies behind the pragmatic understanding of implementation and computational correctness:

> [Pragmatic Implementation is] the correspondence [...] between modes and requests of use of the artefact. A physical artefact [...] is pragmatically correct if and only if there is convergence of all the modes and requests of use of the artefact expressed at the different ULs [User Levels] [...], converging in the specification. [25, p. 163]

1.8 Conclusion

Although, in principle, it might be possible to come up with an evolutionary reconstruction of the traffic light example, this could be an unwarranted step, since the only intentions we need our account to do without are the semantic ones, not the teleological ones. In contrast, the pragmatic theory fully qualifies as an intentionalist view of computational functions, as users' teleological intentions towards the artefact are still regarded as necessary for it to be ascribed a function.[18] Yet, in contrast with the two previously discussed intentionalist accounts by Primiero [128] and Turner [154], such ascriptions are not mediated semantically by semantic intentions. Therefore, while the two former accounts count as straight responses to the paradox of implementation, the pragmatic theory qualifies as a sceptical solution to it, as it is substantially compatible with the non-existence of semantic intentions. Crucially, upon the pragmatic view, the correctness of a computational artefact should be reformulated in terms of a contextual convergence of the different agents' use requests, much recalling Kripke's idea that the agreement on the use of a word among members of a linguistic community is what ultimately grounds meaning ascriptions.

In the next chapter, the User Levels theory is extended to the specific case of Machine Learning (ML) systems. Preliminarily, I ask whether the traditional taxonomy of errors given by the LoA ontology can be revisited to account for the miscomputations specific of ML systems. Although the answer is positive, I point out that the rigid top-down normativity entailed by the LoA theory is ultimately inadequate to account for the dynamic correctness criteria of complex ML systems. In contrast to this, the User Levels theory proves to be able to better accommodate how the normative requirements of different users circulate, feed back, compete, and possibly conflict among the agents interacting with the ML artefact. Upon this alternative taxonomy, two types of pragmatic ML miscomputations are distinguished. Finally, I explore the question of what epistemological criteria are most adequate for evaluating the validity of ML outcomes.

[18] Contrarily to Turner's view [154], according to which agents are supposed to cooperatively converge to an agreed-upon set of functional specifications, for Buda and Primiero [25], such intentions can be divergent and, to some extent, even conflicting with each other, as users are seen as the bearers of their own normative requirements towards the artefact.

Chapter 2

How to Do Things with Bits. A Pragmatic Framework for Data-Driven Miscomputations

2.1 Introduction

We have all wasted endless minutes waiting in vain for a web page to load; by now, we are so accustomed to this and other typical computational malfunctions that most of the time, almost automatically, we try to reload the blocked page or restart an application that does not respond to our commands. More rarely, we have to give in to their annoying presence and shut down the device, in the hope that a reboot will solve the problem and not end with a monochrome (black, blue, white, or gray) screen of death.

As trivial as computational errors may seem in our everyday life, their existence is daunting for safety-critical systems (such as aircrafts, medical devices, etc.) and for disciplines trying to address them, from logic to theoretical computer science and software engineering. Within the philosophy of computer science, the notion of miscomputation, a central aspect of the theory of the Levels of Abstraction (LoA) as formulated by [128], has led to the definition of a detailed taxonomy of miscomputation [61, 59].

Although we show that the LoA ontology can be revisited to capture ML errors, in this work we also point out that the normative rigidity of the resulting taxonomy seems limited in accounting for these complex computational artefacts, whose criteria of correctness are markedly dynamic. For example, let us consider the following two scenarios. In the first case, an ML face recognition system – properly trained on an adequate set of images – ends up labeling a male with

pronounced feminine facial features as female. In the second case, the same recognition system correctly identifies the same male as such. Arguably, the intended function of the algorithm is to classify individuals as men or women based on their facial traits. If this is so, we might concede that in the former case (where misgendering occurs) the system is implementing the correct functionality, despite causing *incorrect predictions* for certain data points, especially outliers. Conversely, the latter case (where the misgendering does not occur) seems to be precisely the opposite: the outlier data point is predicted correctly, although this seems hardly *justified* in terms of the training process the model was presumably exposed to.

In cases like this, compared to the LoA account, the User Levels account [25] proves to better accommodate how the normative instances of different users could circulate, feed back, compete, and possibly conflict among the agents such as designers, end-users, and developers interacting with the artefact. Upon this alternative taxonomy, two types of pragmatic ML errors are distinguished: instruction miscomputations and implementation miscomputations.

The work is organised as follows. Based on the considerations of Section 2.2, we complement the traditional LoA taxonomy of errors with two new types of miscomputations specific to the ML context: errors of prediction and errors of justification (Section 2.3). After that, the LoA framework is contrasted with its pragmatic alternative, the UL theory (Section 2.4), which is accordingly extended to the ML case in Section 2.5. The work concludes with Section 2.6, devoted to a preliminary discussion of the formal validity criteria for ML systems.

2.2 Background: the LoA Taxonomy of Errors

In the past decade, philosophical interest in computational errors has reignited after a brief period of attention on formal verification's validity between the 1980s and 1990s [47, 57, 30]. The basis of this fresh interest in miscomputations in the philosophical arena has been the definition of the ontology of levels of abstraction (LoAs), which has been the subject of extensive exposition in Section 1.3. In particular, in [61], five different LoAs are considered:[1]

1. *Functional Specification Level* (FSL). This level specifies the *intended*

[1]This hierarchy differs slightly from the one in Section 1.3, due to a minor difference. Contrarily to the one given previously, the present hierarchy considers both the high-level programming code and its low-level machine translation as part of the same LoA.

function of the system, i.e., what the system is supposed to do according to its designers. This function is mathematically specified in terms of an input-output (I/O) mapping that reflects the teleological intentions of the system's designers.

2. *Design Specification Level* (DSL): an abstract description of the system, formulated by means of dedicated *specification languages* (e.g., ACSL, CASL, Alloy etc.), identifying and explaining the specific functions and operations required to fulfil the intended function.

3. *Algorithm Design Level* (ADL): a translation of the specifications formulated at the DSL into executable procedures (algorithms) expressed in high-level languages such as pseudo-code, i.e. sequences of rules and operations that the system must execute in order to accomplish its specification and thus fulfil its intended function.

4. *Algorithm Implementation Level* (AIL): a translation of the procedures specified at the ADL into actual executable programs in some high-level programming language (such as Python or Java), then automatically translated (through compiling and linking) into low-level instructions in Assembly, a programming language with a one-to-one correspondence with the machine code controlling the behaviour of the physical components.

5. *Algorithm Execution Level* (AEL): a hardware translation of the instructions provided by the machine code into physical actions performed by electric circuits and gates.

Based on this stratified ontology, a detailed taxonomy of miscomputations was systematised by Fresco and Primiero [61]. First, the authors distinguish between three types of errors: *conceptual* errors reflect breaching of validity conditions; *material* errors reflect breaching of correctness conditions; *performable* errors reflect breaching of physical conditions. The resulting taxonomy of computational errors includes mistakes, failures, slips, and operational malfunctions, defined as follows.

1. *Mistakes* describe those conceptual errors that occur at the FSL, DSL, ADL, or AIL due to the miscomprehension or misinterpretation between the agents responsible for different levels (for example, the designer's intentions might be misinterpreted or translated inconsistently by the software developer).

2. *Failures* are material errors that occur at the DSL, ADL, and AIL, e.g., when the requirements are wrongly formulated in the specification or when the latter is poorly implemented in an algorithm.

3. *Slips* are syntactic or semantic errors induced at the level of algorithm implementation in software, hence occurring only at the AIL.

4. *Operational malfunctions* are runtime errors due to hardware failures.

Finally, Floridi et al. [59] categorise any of the error types just listed as "dysfunctions" or "misfunctions". Any of these error occurrences can in fact manifest either through the disappearance of the intended functionality – if this is so, we talk about *dysfunction* – or through the occasional emergence of unintended and undesirable side effects along with the intended functionality – in this case, we talk about *misfunction*. The authors conclude that, counterintuitively, software tokens cannot neither dysfunction nor misfunction. Hence, software results in a very peculiar kind of artefact type whose tokens cannot malfunction. This underlines and reinforces the normative role of the specification with respect to other LoAs.

2.3 An LoA Taxonomy of ML Errors

The ongoing paradigm shift from the methodology of manual coding to that of curing – which refers to the set of increasingly sophisticated techniques for preprocessing data in the context of data-driven technologies [4] – is indicative of the pivotal role played by training data in determining the behavior of an ML model. In fact, during the training phase the functionality of the model emerges from the training data, as if training itself could be seen as an automatic version of the traditional techniques of requirement engineering [161, 12]. Therefore, we argue, the ML model should not be considered in isolation but always associated with the training process through which its functionality was found. In other words, what we should consider is not an ML *model* but an ML *system*, composed of three distinct, though functionally integrated, components [54]:

1. The training sample: The set of inputs used to train the ML model;

2. The training engine: The computational process that allows the ML model to learn from data during the training process;

3. The learned model: The resulting ML model obtained by running the training engine on a specific training sample.

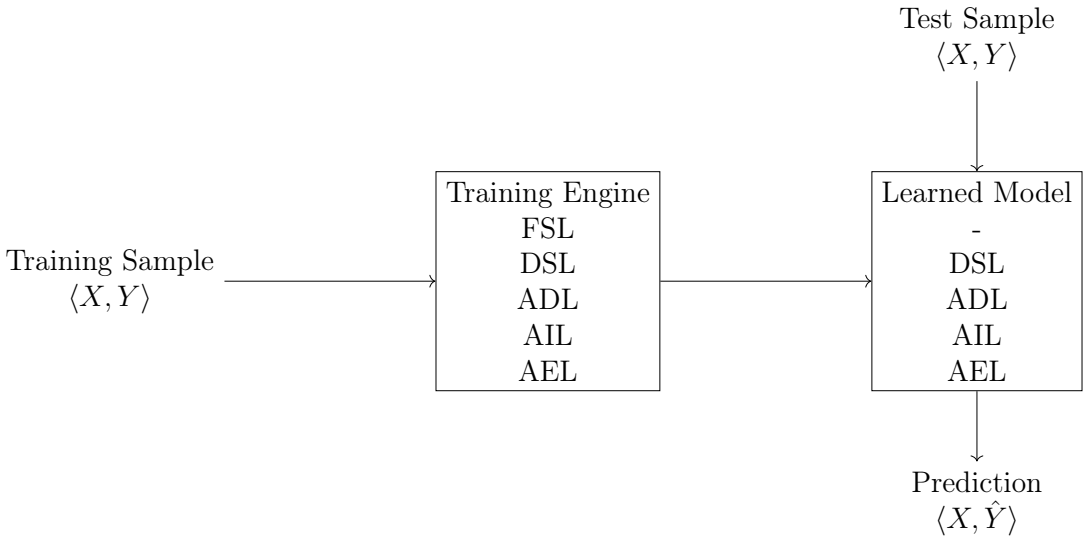

Figure 2.1: The training sample, the test sample, and the prediction are data artifacts. The training engine and the learned model are, instead, computational artifacts. The training sample, training engine, and learned model together constitute an ML system.

From the LoA perspective, data curing seems to suggest that ML models inherently lack a proper Functional Specification Level (FSL), since their specification is the output of an algorithmic process, i.e., the execution of the training engine on the training sample (see Figure 2.1). Moreover, it is important to note that the three components are artifacts themselves: the training sample is a data artifact; the training engine is a traditional (non-ML) computational artifact; and the third is a computational artifact without FSL. Consequently, the traditional error taxonomy is revised in two distinct ways. In fact, any type of errors occurring in the training sample and the training engine can be re-categorized as an error of design for the ML system (as discussed in Section 2.3.1). However, once training is over, the ML system shows two additional types of errors that were not accounted for in the traditional LoA taxonomy of miscomputations: errors of prediction (see Section 2.3.2) and errors of justification (see Section 2.3.3). Table 2.1 summarizes the novel taxonomy that we are going to discuss in the remainder of the section.

2.3.1 Errors of Design

In the context of traditional computational artifacts, an error of design refers to the incorrect coupling between the function intended by the designer and the specification actually implemented by the system. In [61], it is further distinguished between mistakes and failures, respectively corresponding to inconsistent and incomplete requirements expressed at the Design Specification Level of the LoA hierarchy. However, when it comes to ML systems, we point out that *any* malfunction that occurs at the level of the training engine or at the level of the training sample affects the functionality of the resulting learned model and can therefore be seen as an error of design for the ML system in its entirety. For both the training engine and the training sample, we hence distinguish between mistakes and failures.

Starting from the training engine, one important consideration is that this artifact specifies all of the hyperparameters, i.e., the technical details of the training process: the learning rate, the batch size, the loss function, the number of epochs, etc. Hence, hyperparameter choices may be considered erroneous to the extent that they are either inconsistent (mistakes) or incomplete (failures). Here, a mistake may occur when, for instance, the optimization method is not consistent with the loss function chosen: pairing a binary cross-entropy loss function with ReLU activation in the output layer violates the assumptions of the loss function and leads to undefined or misleading gradients. In contrast, a failure may occur when a critical hyperparameter is omitted. As an example, failing to specify a batch size in a deep learning pipeline can result in the use of an undesired default or incompatible value, resulting in immediate runtime errors or in suboptimal training.

Coming to the training sample, data errors can be classified along several data quality dimensions [14, 81]. Similarly to the distinction just made for hyperparameters, we can conceptually distinguish between inconsistent and

ML Component	**Description**	**Miscomputation**
Training Sample	data artifact	design error
Training Engine	traditional computational artifact	design error
Learned Model	computational artifact with no FSL	prediction error justification error

Table 2.1: The three basic components of an ML system, alongside their associated malfunctions: design, prediction, and justification errors.

incomplete data, respectively, corresponding to data mistakes and data failures. In this regard, recalling the discussion on data curing by [4], we notice that the data preparation process (data cleaning, integration, resampling, etc.) constitutes a potential source of errors of both kinds. Furthermore, the training sample itself is frequently the result of an automated process known as "feature learning". For instance, within the context of natural language processing, feature learning involves the training of an additional ML model to embed lexical items as vectors such that lexically similar items are situated in close proximity within the vectorial space, while those that are dissimilar are positioned significantly further apart. In this scenario, an error within a training sample derived through feature learning would simultaneously count as a design error, in relation to the training of the ML system, and a prediction error, in relation to the embedding methodology.

2.3.2 Errors of Prediction

Given an input, a prediction error occurs whenever there is a mismatch between the predicted label and its true value (called the ground truth). The first obvious observation is that prediction errors are inevitable for ML systems. As the "No Free Lunch Theorem" [168] demonstrates, all supervised classification algorithms have the same test performance (accuracy, precision, and recall calculated on new input) when averaged over all possible datasets. This means that a model may perform well on certain data but will inevitably perform poorly on others.

One first reason for this mismatch is given by the fact that unlike in traditional settings, in ML contexts, the model is learned from examples rather than manually coded. Hence, what is learned is an approximation of the target function, inferred on the available examples within the constraints imposed by the hyperparameters (see the previous subsection). Put differently, the ML system does not learn the target function $F(X)$ but a suitable approximation $\hat{F}(X)$ thereof:

$$\hat{F}(X) \approx F(X)$$

A second and more fundamental source of prediction error in ML systems arises from the inherent difference between the predicted and true labels, known as the irreducible minimum error, or Bayes error [85], an independent random

noise term representing the influence of all kinds of unobservable factors on Y.

$$Y = \hat{F}(X) + \epsilon$$

Therefore, ϵ is the lowest possible classification error in principle achievable by a classifier (the Bayes classifiers, precisely). Note that ϵ cannot be reduced even if the true probability distribution of the data is fully known, i.e., that $\hat{F}(X) = F(x)$.

Just to exemplify the significance of predictive errors, we can briefly touch on the notion of adversarial examples. These refer to unusual input data that cause deep neural networks to output verdicts that look to humans like very bizarre prediction errors. For instance, small imperceptible perturbations added to images cause a well-performing image classification model to misclassify a panda as a gibbon with high confidence [66]. Adversarial examples represent a paradigmatic type of "strange" —to use the terminology of [131]— prediction errors in ML systems, whose occurrence and content are substantially impossible to foresee. Adversarial examples are not only carefully crafted inputs that are intentionally designed to fool ML models into making incorrect predictions. Rather, they naturally occur in datasets, like image datasets where textures, patterns, colors, and light conditions can fool the ML system in the most bizarre ways. According to Alvarado [2], such unpredictability should count as a serious obstacle to securing a sufficient level of reliability for ML systems in science.

2.3.3 Errors of Justification

The training process might induce correlations in the model that, while not necessarily detrimental to the performance, are undesirable for some other reasons, such as epistemological ones. The ML system might, for instance, return the right prediction for a data point but for the wrong reasons. We hence take these widespread kinds of ML malfunctions to be errors at the level of justification.

An illustrative example of a justification error was presented in [134], where a classification algorithm was trained to classify images of wolves and dogs. Contingently, in the training dataset, wolves were often depicted against a snowy background. Using a novel explainability technique called LIME (Local Interpretable Model-Agnostic Explanations), the authors showed that the

resulting ML system had learned to classify new input pictures based on the color on the background, deeply questioning the trustworthiness of the system. Even if the classifier contingently performed well, in fact, one would still want to claim that an error had occurred because of what justified such predictions.

The problem of misjustified ML predictions has particular import in the context of science, where we want well-defined epistemic guarantees to hold. In molecular design, for instance, AI systems are capable of achieving results once thought impossible, particularly in terms of discovery speed and overall efficiency. Given the abundance of data accumulated from previous reactions, ML models can now identify completely new chemical entities as well as suitable catalysts for specific chemical tasks. Unlike the wolf–dog classifier example mentioned earlier, in this case the prediction seems justified by the significant amount of raw data available from prior reaction pathways. Based on this, it has recently been argued that this could even allow "the creative part of this process [of formulating a chemical hypothesis] to be progressively aided by a procedural logic" [56, p. 13], opening up to the possibility to establish a long-awaited logic of discovery.

However, to concede this point, we should first examine what we mean by creativity scientific, whether there are different types thereof, and how these relate to scientific discoveries on the one hand, and computational systems on the other. Boden [19], for example, offers a simple and exhaustive taxonomy that distinguishes between three types of creativity in a broad sense (encompassing the artistic sphere) which are defined "combinational", "exploratory", and "transformational".

> The first type involves novel (improbable) combinations of familiar ideas. [The second] involves the generation of novel ideas by the exploration of structured conceptual spaces. This often results in structures ("ideas") that are not only novel, but unexpected. [The third] involves the transformation of some (one or more) dimension of the space, so that new structures can be generated which could not have arisen before. The more fundamental the dimension concerned, and the more powerful the transformation, the more surprising the new ideas will be. [19, p. 348]

How these categories relate to the process of artistic creation is easily deducible. Consider the case of painting: combinational creativity is involved when we move within the same artistic style, such as the Flemish style or the

Michelangelo school; exploratory creativity occurs when a new and unexpected figurative painting style is created, as in the case of the Impressionist school or Surrealism. The abandonment of the figurative form typical of abstract painting can instead be considered a form of transformational creativity.

With respect to the creation of new scientific knowledge, this taxonomy can be exemplified as follows: the above mentioned molecular design example can be placed at the first level of the taxonomy; research on mRNA vaccines, which began in the late 1980s, provides an excellent example of exploratory creativity; regarding the last level of creativity, we need to go back in time to the epochal discoveries that led to substantial paradigm shifts, such as the concept of quantum introduced by Max Planck, Albert Einstein's theories of relativity, or Niels Bohr's atomic model.

This taxonomy has already been used to evaluate the artistic capabilities of computational systems, both classical and data-driven: Generative AI systems are primarily combinatorial systems, so the first level of creativity is assured. Exploratory and transformational creativity "shades into one another" [19, p. 348], and ML systems have been shown to be able to express not only the former but also the latter, thanks to the introduction and manipulation of random values (e.g., temperature) capable of creating new and unexpected results.

Even though chance can create new and pleasant artistic forms, it would be dangerous to rely on randomness when dealing with scientific discovery. Using randomness to create new scientific knowledge obliterates the possibility of providing a justification based on the most probable statistical distribution: the more randomness we introduce to scale the hierarchical levels of creativity, the less likely we are to find a rationale for the results obtained. In other words, ML systems could provide scientifically reliable results only in the combinational setting, thanks to their brute force.

Even in these cases, the risk of falling into the temptations of pseudoscience is very high. Statistical systems, by their very nature, lend themselves to favoring homogeneous results and consequently to being vulnerable, on the one hand, to the risk of discrimination against outliers or minorities, and on the other, to the risk of dystopian Lombrosian research tendencies that excite eugenicists (for a widely criticized example, see [170]). In these and other cases (such as credit scoring, recruitment, access to public services, etc.) the laws of statistics can serve as scientific justification for unfair results and unjust policies.

To conclude, the risk arising from the statistical nature of AI systems must be seriously taken into account, and particular attention must be paid to the even

greater risks of using this type of technology to generate scientific theories that draw on forms of exploratory and transformational creativity.

2.4 Pragmatic Miscomputations

Recently, de Haas and Houkes [71] have questioned the philosophical assumptions underlying the LoA error taxonomy, particularly criticising that the LoA account presupposes the complete control and the self-containedness of software. In light of current coding practices (cloud computing, open source, collaborative code, etc.), the authors have argued that it is no longer possible to consider these assumptions as generally valid. A series of practical examples are used to show that many agential activities are involved in the development and execution of software and that, under the aforementioned presuppositions, the LoA-based theory wrongly attributes too many responsibilities to software engineers.

While de Haas and Houkes' criticisms of the assumptions of complete control and self-containedness are agreeable, their alternative account does not seem to succeed fully, as it ends displacing such responsibility on the shoulders of yet other software engineers. As an example, their interpretation of the EVE Online bug ends up blaming Windows software engineers for not denying problematic file deletion requests [71, p. 11]. This possibly suggests that giving up the assumptions of complete control and self-containedness is just *not enough.*

Moving from roughly the same philosophical worries, Buda and Primiero [25] come to challenge a more fundamental assumption of the LoA theory, namely that of a rigid top-down normative structure in which the correctness criteria of the computational artifact flow from the specification to the hardware. In this pragmatic view, which has already been introduced in Section 1.7, the functional specification is reinterpreted just as the maximally consistent set of the semantic and normative instances encompassing human agents such as designers, end users, coders, policy makers, owners, etc. From this very plurality, the failure of self-containedness and complete control noted by de Haas and Houkes. As already mentioned in the previous chapter, pragmatic correctness is defined as the convergence of all modes and requests of use of the computational artifact, expressed at different user levels [25].

To substantiate the key difference between the LoA and the UL theories, the following two examples can be useful:

Example 1 (Excel Art) *In 2000, Tatsuo Horiuchi retired and decided to devote himself to painting. He did not want to spend money on brushes, paints, and canvases, and he did not want to spend a lot of time cleaning up. So, he opted for digital painting, but he did not want to spend money on expensive graphics programs either. Then, he decided to use the programs pre-installed on his Windows computer, but he found Paint to be unintuitive and not user-friendly. Instead, he found the perfect instruments to create his digital art by means of Excel's line tool (to draw shapes) and bucket tool (to fill such shapes with color and gradients), raising Excel art, which until then had just been pixel art created using cells, to a new level.*

Example 2 (Malicious Software) *During the same year, Onel de Guzman was a talented but poor 23-year-old student from the Pandacan district of Manila, besides being an activist devoted to the fight for the right to Internet access. Since he could not afford the cost of an Internet connection, he decided to create a password-stealing Macro virus written and compiled in Visual Basic, disguised as a common text file named LOVE-LETTER-FOR-YOU.TXT, and to attach it to an email with the subject "ILOVEYOU" containing the following text: "Kindly check the attached LOVELETTER coming from me.". The virus was able to replicate itself, replacing Windows library files and other random files in the infected system; to resend itself to all of the addresses in Outlook's address book; and (obviously) to steal passwords.*

For the LoA theory, both examples equally qualify as cases of misfunction. In fact, in light of the distinction made in Section 2.2 between dysfunction and misfunction, the two cases involve computations that produce the intended effects alongside with other ones, totally unforeseen by the systems' designers.

However, this reconstruction finishes to obscure an obvious difference between these two examples, which concerns the different effects produced in the world in the two cases: while Onel de Guzman was charged by Philippines state prosecutors for the (in)famous Love Bug that affected 10% of internet-connected computers, Tatsuo Horiuchi is now a well-known 80-year-old digital artist whose technique was promoted by Excel with the release of an improved set of drawing tools. At a more abstract level, the Excel example tells us about a virtuous use of a highly limited tool (spreadsheet line and bucket tools), while the malicious software example is the story of a malicious use of a very powerful tool (macros).

The UL theory conceptually captures the difference between these two examples by focusing on the sequence of effects triggered by the unconventional

uses of Excel and macros, respectively. Initially, these uses were neither included nor explicitly excluded by the original designer's specification or any other UL. As idiosyncratic uses similar to the initial ones became more widespread, both cases evolved to the stage of innovative use: on the one hand, Excel art has never been so popular; on the other hand, more than 25 variants of ILOVEYOU were created and sent across the Internet. Then, they both evolved to the expert redesigning stage, when expert digital artists started to develop new techniques for creating art using spreadsheets and when new improved variations of Love Bug were developed by black hat developers and social engineers. Finally, the product design phase is reached when the idiosyncratic use is integrated into the specification as correct or is rejected as not correct. In our examples, this corresponds to the moment when the original designers release a new version of the computational artifact. The crucial difference between the two being that, in the first one, this version includes new and improved Excel drawing tools to promote the creation of pixel art; in the second one, this includes a new mechanism for preventing the execution by default of any macro downloaded from the Internet and hence discouraging their use. The specification thus temporarily regains its normative role over the system, until the next idiosyncratic use, in a continuous iteration of the same process (see Figure 2.2).

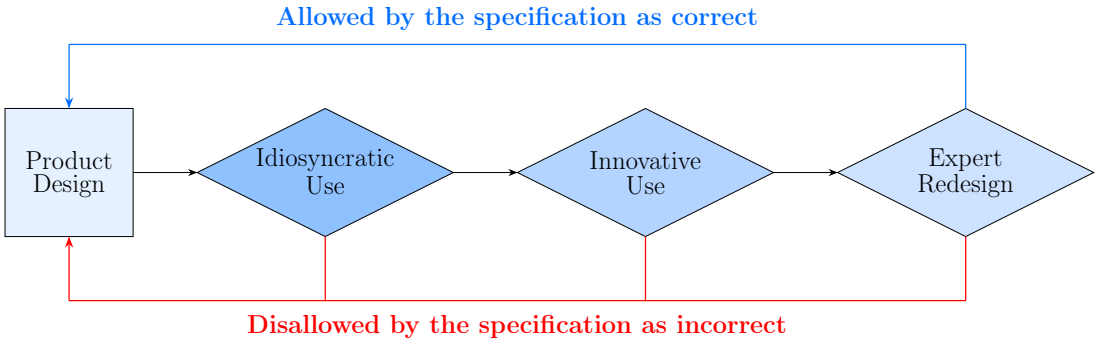

Figure 2.2: The iterative spectrum of correctness for UL theory spans from the correct standard use of the original product design to idiosyncratic use, to innovative use, to expert redesign.The idiosyncratic use in Example 1 is permitted by the specification, whereas that in Example 2 is prohibited.

To capture the difference between the two examples, a new pragmatic criterion for computational correctness is called for, one according to which the ILOVEYOU case (but not the Excel art one) can be described as involving a miscomputation:

Definition 1 (Pragmatic Miscomputation) *A physical artifact displays a pragmatic miscomputation when, as a result of an idiosyncratic use, it manifests a functionality that is explicitly disallowed by the specification.*

Quite expectedly, the pragmatic view returns a picture of computational error in which fewer phenomena count as proper computational errors. This narrowing effect is indeed typical of pragmatic perspectives more in general (in philosophy of language, philosophy of science, etc.) and is the result of centering the focus on the way the emergence of random, overtly unconventional and even malicious uses of a computational artifact can have an active and productive role, rather than passive and inherently destructive one, in shaping its normative requirements. In the next section, we finally apply the same framework to the specific case of ML systems.

2.5 A Pragmatic View of ML Miscomputations

Within the UL framework, the three components of an ML system illustrated in Section 2.3 are taken to correspond to three different User Levels: the training sample UL, the training engine UL, and the learned model UL. For completeness, we might think that these three user levels always come with two more levels: the designer level and the final user level, that is, the level at which the model outcome comes to fruition for some human agent.

For the sake of simplicity, let us start by applying this structure to the case of the simplest neural structure, the multilayer perceptron, to exemplify the general behavior of an ML system (see Figure 2.3):

1. *Initial Randomization.* At the beginning of the training phase, when the weights and biases are randomly distributed throughout the Deep Neural Network (DNN), any input of the training sample UL will result in an idiosyncratic output in the model output UL. At this stage, it is very unlikely that the system performs with the intended accuracy.

2. *Training.* During training (e.g., by means of backpropagation in supervised deep learning), idiosyncratic outcomes start to converge towards similar results, and innovative outcomes will start to emerge in the model output UL. The performance of the system accordingly increases by the adjustment of weights and biases dictated by the cost function minimization.

3. *Reinforcement Learning.* Through reinforcement learning techniques, the model is iteratively shaped by end-user feedback, with rewards and penalties guiding it toward outputs that reflect specific situated preferences and goals that are difficult to encode directly into the training data.

4. *Final ML Model.* The process stops when the phase of product design is reached, i.e., when the system satisfactorily approximates the performance intended by the designer UL. At this point, the entire process is ready to start anew, for example, when the DNN is retrained.

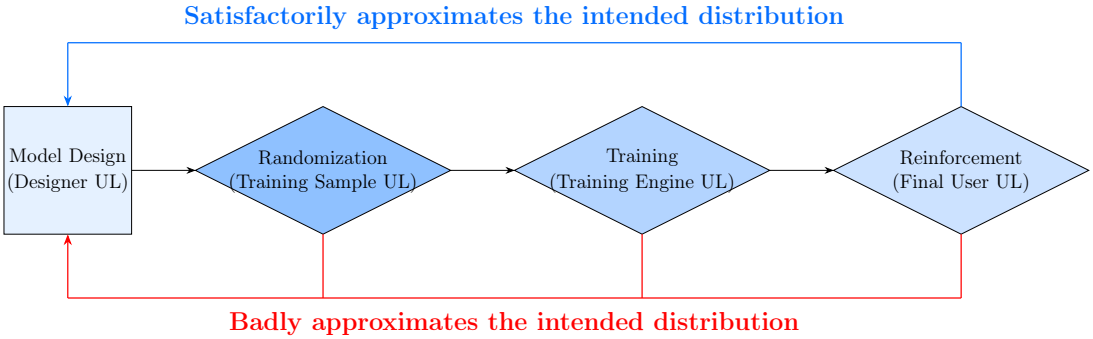

Figure 2.3: The randomization at the beginning of the training qualifies as an idiosyncratic use; during training certain innovative uses emerge and get further refined when stepping into a phase of expert redesign, for example thanks instantiated by a reinforcement learning mechanism.

As the previous section has illustrated, each idiosyncratic use – and its subsequent evolutions at the stage of innovative use and expert redesign – can either be accepted or rejected by the designer UL. In the former case, the ML engineer will be encouraging the emergence of certain statistical patterns or behaviors, in the latter, the ML will be changed at the level of its structural constraints (hyperparameters, optimization function, architecture, etc.) to discourage them. Crucially, however, the ML designer cannot explicitly disallow nor allow certain behaviors by directly acting upon its functional specification. This is because the trained ML model does not properly have one (in LoA terms, we defined it as a computational artifact with no FSL).

Recall the fictitious face recognition scenario described in Section 2.1, where a man with feminine traits is misclassified as a woman.[2] While for the LoA perspective, there is no doubt that such misgendering constitutes a

[2]This biased representation might, for example, end up having a negative impact on the life of transgender men. For a discussion of the specific data quality issues arising in classifying transgender individuals using binary gender labels, see [130].

miscomputation *per se* (specifically, an error of prediction), from the pragmatic perspective, this is ultimately a matter of choice, to be established based on the intentions of the users involved, such as designers and end-users. For example, we can think of a situation where such a misgendering is caused by the fact that the training sample (containing, presumably, only faces of cisgender individuals) offers an inadequate representation of the true distribution (containing faces of transgender individuals as well). For example, in a scenario where the face recognition system might be used to let workers in their workplaces, a user who wants the ML to treat individuals as fairly as possible will deem such bias an error.[3]

Figure 2.4: Images generated with Microsoft Bing Image Creator using as prompt "A person holding a spoon" [(a), (b), (c), (d)], "A person holding a fork" [(e), (f), (g), (h)], "A person holding a knife" [(i), (j), (k), (l)]

An analogous tension between the goals of the different users is illustrated in Figure 2.4 in relation to the context of Generative AI (GenAI). When given the

[3]Bias present in the data can be replicated and even amplified across all user levels: at each level the distance between the considered distribution and the reference distribution can increase. Elsewhere, we have described this phenomenon as a Bias Amplification Chain [23]. For a similar holistic view of bias that occurs throughout the entire ML life cycle and not only at the level of output predictions, see [151].

(a) (b) (c) (d)

Figure 2.5: Images generated with Microsoft Bing Image Creator using as prompt "A person holding a knife in the kitchen"

text prompt "A person holding a spoon" or "A person holding a fork", Microsoft Bing Image Creator outputs us four photorealistic and unbiased images, in which gender and races are fairly distributed, and even left-handed people are accurately represented with respect to their distribution in the real world. In contrast, if we use "a person holding a knife" as prompt, to prevent violent or racially biased images, automatic bias mitigation operations lead to results in which photo realism disappears, together with gender equality, and accuracy: not only are there only cartoon-like male characters in the last row of Figure 2.4, but also no one of them is holding a knife, and in one case the character is not even a person, but a happy and hungry cannibal tomato.[4]

Figure 2.5 shows the effects of adding the contextual information "in the kitchen" to the sentence "A person holding a knife": we regain photo-realism, together with accuracy and better racial and gender fairness. This may be due to the fact that such contextual information makes automatic mitigation operation useless, hence it frees the model from mitigation operations. The group of tokens included in "in the kitchen" is semantically associated with other tokens e.g. cooking, which allow to avoid associations with undesirable semantic subspaces e.g. involving violence and racial bias. Also in this case, accuracy and fairness, and hence the direction of normativity, are dictated by both the intentions of the designer (manifested in the bias mitigation mechanism embedded in the model) and those of the end user writing the text prompt (as expressed by the prompt text).

The broad Definition 1 mentioned earlier can be further detailed by distinguishing between the two following cases:

[4]This is only a selection of the tests conducted using Microsoft Bing Creator based on OpenAI's DALL-E 3 models PR13 and PR16, from August 2024 to May 2025 (15 tests from August 2024 to September 2024, 15 tests from November 2024 to January 2025, and 34 tests from march 2025 to May 2025. These tests were carried out both in isolation and sequentially, and always showed comparable results.

Definition 2 (Implementation Miscomputation) *An implementation miscomputation occurs when any of the lower user levels of the system (training sample UL, training engine UL, learned model UL, and model output UL) badly approximates*[5] *the designer's intentions, as represented by an ideal unbiased distribution (when evaluating fairness) and/or by real world ground truths (when evaluating accuracy).*

Definition 3 (Instruction Miscomputation) *An instruction miscomputation arises when the output of the system is in contrast with the instruction provided by end users and/or their informational states.*

Upon this distinction, the first two rows of Figure 2.4 are to be considered as pragmatically correct computations, whether accuracy is our goal or we aim for the fairness of the system, since they accurately implement the designer's intentions and follow the final user's instructions, while preserving a fair balance of genders and races. Figure 2.5 can be seen as a case of implementation miscomputation with respect to gender fairness, given the non-uniform distribution of male and female outcomes. The last row of Figure 2.4 represents a more severe case of pragmatic miscomputation, involving both implementation *and* instruction: it is an unfair implementation miscomputation because of the non-uniform gender distribution; it is an inaccurate implementation miscomputation in virtue of the cartoon-like style of the outputted pictures; and it is an instruction miscomputation because all characters are holding a wooden spoon, instead of a knife as requested by the end user's text prompt.

A final question to address stems from Definition 2, concerning the adequate epistemological criteria to establish whether a given outcome distribution significantly departs from a reference one in the specific context of ML. This is the object of the next section, where we lay the groundwork for a formal approach to these validity criteria.

2.6 Defining Formal Validity Criteria for Machine Learning Models

In a recent article, Angius and Plebe [4] have suggested that the most appropriate notion of correctness for ML systems might be that of 'experimental

[5] In the original text, we used the expression "badly implements" [24]. For a clearer formulation, here I am using the expression "badly approximates" instead.

validity', where a simulation is experimentally valid whenever there is an isomorphism between a "usable" computational model and a mathematical model that is "fit for purpose" [128].[6] In the present section, we want to build upon Angius and Plebe's intuition that ML systems can be seen as (scientific) simulations of a special kind, and consequently that the work on the validity of scientific simulations can be of help in reasoning about the validity criteria of ML outputs.

Formal isomorphism relations have long proved useful instruments in many epistemological endeavours, especially in the scientific context, where especially the surge of computational modeling techniques and simulation software have called for the definition of specific methodological principles to establish the validity of a scientific result obtained through algorithmic processes. The problem of validity as a property of a scientific result obtained by computational methods could be formulated as follows:

Problem 1 (Computational Model Validity) *What are the conditions for accepting as valid the results of a computational model, and how do we express them formally?*

To this end, we observe that most ML systems exhibit two epistemologically troubling features: opacity, stemming from their structural complexity, and non-determinism, arising from the way learning methods are instantiated to detect statistical patterns in data. These characteristics – that have already been extensively examined in the context of traditional simulations [77, 49, 50, 1] – lead us to restate the question of ML validity in the following terms:

Problem 2 (Opaque and Nondeterministic Computational Model Validity) *What are the conditions for accepting the results of an opaque and non-deterministic computational model, and how do we express them formally?*

While there is an abundance of technical and philosophical analyses of ML opacity [70, 20, 54] and nondeterminism [150, 38, 44], a comprehensive theoretical examination of Problem 2 remains underdeveloped at present; with this contribution, we aim to contribute laying the theoretical grounds for its systematization.

[6] A computational is usable "when it allows to accomplish the desired experimental analysis and to construct a fit for purpose mathematical model"; a mathematical model is said to be fit for purpose "when it is able to represent some of the requested variables and inferential properties of the represented system" [4, p. 47].

2.6.1 The Story So Far: On the Validity Criteria of Deterministic Computer Simulations

The conceptual study on what kind of validity conditions can be established for results obtained by implemented computational systems (such as simulation software) is a complex issue. The origin of this debate is notoriously traced back to Humphreys:

> Inasmuch as the simulation has abstracted from the material content of the system being simulated, has employed various simplifications in the model, and uses only the mathematical form, it obviously and trivially differs from the 'real thing', but in this respect, there is no difference between simulations and any other kind of mathematical model, and it is primarily when computer simulations are used in place of empirical experiments that this element of falsity is important. But if the underlying mathematical model can be realistically construed (i.e. it is not a mere heuristic device) and is well-confirmed, then the simulation will be as 'realistic' as any theoretical representation is. [77, pp. 501-502]

From the assumed epistemological identity of the validity of computer simulations and their use in place of empirical experiments, the question arises of how to formulate their validity conditions. In particular, it becomes pressing to establish whether a formal treatment of such criteria is possible. Here the debate has diverged, with different positions being formulated in the literature. On the one hand, it has been argued that such a question cannot be answered in absolute terms, and that the very conditions of knowledge of the target system determine the appropriate methodology to formulate what validity means:

> For the data to hit the target, the experiment, simulation, or "simulating experiment" must mirror the target in a number of relevant aspects. Whether the mirroring should be purely formal or material in character, depends on the kind of question we are asking, and on the amount and quality of background knowledge we have accumulated regarding the target system itself. [68, p. 15]

On the other hand, detractors of formal methods – including Humphreys himself – have argued against the usefulness of formalization, seen as imposing a further departure from the observed system, to be added to the abstractions and idealizations already implied by equation-based models:

> [...] no comprehensive model at the level of the total system exists. Computationally irreducible processes thus occur in systems in which the most efficient procedure for calculating the future states of the system is to let the system itself evolve. [F]ormalization takes one away from what is important about the form of the equations, and hence is at best unnecessary; at worst it distorts the important formal features. [...] logic is not well suited to deal with every goal of science, or even particularly well with any one of them. [77, pp. 149-153]

Nonetheless, provided that computational models have proven time and again their extraordinary capacity for accurately predicting the behavior of real systems, their reliability inevitably requires formal criteria to assess their correctness:

> If computer simulations are reliable processes that render correct results, then we need a way to characterize what a 'correct' result would be. [...] Verifications and validations are two fundamental pillars for asserting the reliability of computer simulations [...] [R]esearchers are on good grounds for claiming that their computer simulations are reliable processes, and thus that they are justified in believing that the results of the simulation are correct. [...] [R]eliability of computer simulations support the claim that researchers are able to claim for explanation, prediction, and [...] for understanding of the results. [...] Although neither software nor hardware can be fully verified or validated, researchers are still developing methods that reduce the occurrence of errors and increases the credibility of the simulation. [49, pp. 91-96]

As stressed also in [76] with the notion "construction assumptions" for formal templates, the role of formal methods is crucial to establish what kind of idealizations and approximations are appropriate in knowledge construction. Determining such assumptions for computational methods is essential to distinguish which validity criteria hold. It is, in fact, necessary if we want to know under which conditions the results of such models can be deemed valid.

One attempt to ground such formal validity conditions is the notion of isomorphism, well-known in the philosophical debate on what enables a scientific model to *represent* the target system [13, 21]. The isomorphic view anchors scientific representation in the existence of a formally definable one-to-one, bijective correspondence between the set of states representing the dynamic evolution of the target and the one representing the dynamic evolution

of the computation. Notoriously, it has been argued that such a conception is somewhat too strong, as it requires perfect matching between the target's and the model's structures. Moreover, it imposes properties like reflexivity and symmetry that seem undesirable for qualifying the representation relation (see, for example, the directionality requirement by Frigg and Nguyen [63]). To respond to these shortcomings, weaker variants of target-model correspondence have been proposed, like partial isomorphism and homomorphism.

Primiero [128] has applied these relations to the issue of validity of computer simulations, to formalize the correspondence between the deterministic model of a target system and the deterministic computational model of a simulation software that should predict or describe it.[7] Such an application defines the underlying formal mappings between:

1. an observed target system, that is the phenomenon that we want to study
2. an abstract mathematical model that describes the dynamic transformations of its relevant variables,
3. a computational model, that is, an algorithm that translates the mathematical model into discrete state transitions that computers can process,
4. a computational system that physically implements such an algorithm.

Leaving aside the last element of the simulation, corresponding to the physical implementation itself, three main types of formal isomorphic relations have been discussed in the literature to variously qualify the correspondence between the target system and the computational model:

Definition 4 (Bisimulation) *The computational model displays all and only the behaviors of the target system.*

Definition 5 (Simulation) *The computational model displays only behaviors of the target system.*

Definition 6 (Approximate Simulation) *The computational model displays some, but not only, behaviors of the target system.*

[7] This formal strategy relies on an idealization process which assumes the ability to construct models for both the target and the computational systems. Especially for the former, such an idealization is not trivial and it may have to take into account certain limitations, such as partial observability (see Section 2.6.2).

Bisimulation is the strongest isomorphic relation, as it holds when the two systems exhibit identical stepwise behavior; Simulation holds between a computational model and the target system whenever the former tracks all stepwise behaviors of the latter, but not *vice versa*; Approximate Simulation allows for incompleteness and divergence of the computational model with respect to the target, as some computational behaviors have no counterpart in it.

More formally, the behaviors of both the target system and the computational model are defined by their labeled state transition systems, that we can denote by $\vec{TS}$ and $\vec{CM}$, respectively. The three relations of Bisimulation, Simulation, and Approximate Simulation correspond, respectively, to the equality ($\vec{CM} = \vec{TS}$), the inclusion ($\vec{CM} \subset \vec{TS}$), and the non-null intersection between the two ($\vec{CM} \cap \vec{TS} \neq 0$). These three relations have also been used to define corresponding notions of exact, inexact, and approximate copies of computational artifacts by Angius and Primiero [5, 6].

2.6.2 ML as Opaque and Nondeterministic Computer Simulation

Against this conceptual background, ML simulations constitute a departure from traditional ones with regard to the actual process through which the computational model is constructed. In non-ML scenarios, the mathematical model – corresponding, for instance, to a set of differential equations – is typically *top-down* translated into the computational model. Alvarado has correctly emphasized that this manual process of translation "includes many idiosyncratic engineering practices that are far from the sound theoretical principles in virtue of which the initial [mathematical] model was constructed" [3, p. 1190]. Still, it remains true that the validity guarantees we have on the resulting computational model largely depend on how adequate said engineering practices happen to be for a given mathematical model, both of which are to a significant extent *epistemically transparent*.

This process seems to have entirely opposite directionalities and modalities in the ML case, as the mathematical model is obtained from the computational one (rather than the other way around) through a procedure that is largely automatic (rather than manual). Concretely, through the training process we obtain the learned ML model, concretely returned as a matrix that specified the neural network parameters. The learned ML model is a computational model

that implements the predicted function $\hat{F}$, where the latter expresses a joint probability distribution of the ML features and can be substantially regarded as the mathematical model of the simulation. As a result of this bottom-up process, ML systems inherently exhibit two fundamental characteristics: nondeterminism and opacity.

For the former, in fact, the result of the learning process is a computational model that implements the learnt function $\hat{F}$, which constitutes the mathematical model of the simulation based on ML. This describes a probability distribution associating each input with its outcome probabilities. Therefore, the input-output relation is non-deterministic to the extent that it is stochastic, i.e., can be reasoned upon using probabilities. A well-studied and natural way to represent the uncertain behaviors of ML systems is provided by probabilistic extensions of state transition systems such as Probabilistic Process Algebra and Markov Models (such as discrete-time Markov chains, Markov reward models, and Markov decision processes) [87, 102].[8] An equivalent approach is that of inferential systems like the probabilistic typed natural deduction calculus discussed in [46, 64].[9] Either way, the common strategy underlying these formalizations is the following:

1. Assign probability distributions to training and test data as frequencies in the population to represent possible data for the trained model and possible test inputs;

2. Assign, to each feature, the probability to hold for the current data point;

3. Compute, by an appropriate probabilistic consequence (resp. inference) relation, the probability that any set of assignments of features leads to an assignment of the target feature (resp. an equivalence class).

As for opacity, the mathematical model expressed probabilistically by $\hat{F}$ is not immediately accessible to us. A non-trivial work of abstraction is further required to extract it from the trained ML model implementing it. It should be noticed, additionally, that even when the mathematical model is successfully reconstructed from the outputs returned by the resulting ML model, this may not be a guarantee of epistemic transparency, due to its inherent complexity. In fact, $\hat{F}$ potentially contains a huge number of variables $X_1, \ldots, X_n$ of $\hat{F}$, which

[8] For instance, in [152] probabilistic model checking is exploited to verify the accuracy and the trustworthiness of *post-hoc* explanations of ML models.

[9] Based on this, a bias detection tool called BRIO has recently been implemented to assess ML decisions in societally-critical scenarios, like in the financial sector [39, 40, 23].

make its decomposition into simpler marginal distributions $P\left(X_i|X_{j\in\{1\ldots n\}\setminus i}\right)$ both practically and theoretically challenging [54]. In the literature, the problem of model surveyability has been conceptualized in terms of a difference between *strong* and *weak* isomorphic relations to distinguish, respectively, between those cases where the states of the systems are completely observable and those where the systems' states are only partially so [9].

Therefore, in light of these two characteristics just discussed of ML systems, nondeterminism and opacity, we turn to define the following weak and probabilistic versions of the three isomorphic relations introduced above, this time capable of expressing partially observable correspondences between the probabilistic transitions of the computational ML model and those of its target:

Definition 7 (Weak Bisimulation) *The nondeterministic computational model only exhibits behaviors whose probabilities match those of the target system.*

Definition 8 (Weak Simulation) *The nondeterministic computational model only exhibits behaviors with at least the same probabilities as those of the target system.*

Definition 9 (Weak Approximate Simulation) *The nondeterministic computational model exhibits some, but not only, behaviors whose probabilities match those of the target system.*

The relation of Weak Bisimulation expresses in formal terms a complete match between the computational model and its target system: what the model predicts is exactly as in the target system. This relation can only be instantiated by a perfect classifier. The relation of Weak Simulation admits the case of *underfitting*: the ML system does not predict beyond the scope of the target system, while it may assign to certain predictions a higher probability than the corresponding value in the target system. Conversely, the Weak Approximate Simulation relation captures the notion of *overfitting*: the model may make predictions that are in the scope of the target system, but it may also predict features of variables which the target system does not admit. In this sense, the computational model may create predictions which do not reflect the complexity of the model.

Based on these relations, different validity can be induced, from an extremely strict one, defined on the relation of Weak Bisimulation, to the most relaxed one, defined on the relation of Weak Approximate Simulation. However, the formal

articulation of these principles is left for another occasion, as our attempt here has just been that of laying the ground for a formal investigation on adequate validity criteria for ML systems.

2.7 Conclusion

We have shown how the LoA taxonomy of classical miscomputations can be quite naturally adjusted and extended to accommodate the errors of ML systems. Based on its revision, three types of ML miscomputations can be identified: errors of design, errors of prediction, and errors of justification.

However, the LoA taxonomy offers just a static and local (i.e., relative to a specific data point) snapshot of the ML system, as it is unable to represent the inherently dynamic and evolutionary nature of data-driven computation, where continuous evolution (through training) and user interaction (through reinforcement) are key elements. Under these respects, the UL theory fares better, as it correctly captures the way normativity flows back and forth among a plurality of agents variously involved in the design, the training, and the use of the ML system. While in our analysis we have intentionally focused on just the most fundamental user levels (the designer UL, the training sample UL, the training engine UL, the learned model UL, and the final user level), in more complex data-driven scenarios, this framework needs to be expanded to include, for example, the pre-trained model UL, the data labeler UL, the prompt UL, the data scientist UL, etc. This seems particularly needed in the case of Generative AI, where several ML systems are combined to produce even more complex results (e.g., in a text-to-image model, a textual prompt is first mapped into a semantic space and then converted into an image by means of reverse diffusion).

To complement this picture, we have advanced a preliminary proposal for a formal inquiry on the epistemological validity criteria for ML systems. Namely, starting from the traditional definitions of isomorphic relations familiar to the philosophy of science and the formal verification literature, we have developed weak probabilistic versions of the traditional formal simulation relations that can express partial correspondences between the probabilistic state transitions of an ML model and its target system.

As we have emphasized, the central contribution of UL theory lies in situating the ML designer's intentions as only one element among many that shape the articulated and dynamic normativity of an ML artefact. Building on

this reasoning, the next section turns to the issue of algorithmic fairness from the same pragmatic perspective. From this standpoint, we propose moving beyond what might be described as the prevailing design-centric view of fair ML, and instead advance an alternative framework, that we claim to be epistemologically richer: one that understands the fairness of an ML system as a plurality of practices of repair and maintenance that continuously unfold after its initial design.

Chapter 3

Algorithmic Fairness as a Repair Practice

3.1 Introduction

Philosophers have long investigated the many ways technical artefacts *stop doing* what they are supposed to do, for reasons like transient malfunction, poor design, and ordinary wear and tear. Far less often, they have been interested in how artefacts *keep on doing* what they are supposed to do, mainly thanks to timely maintenance, skilled repair, and a good dose of elbow grease.

While seemingly casual, this omission is deeply rooted in a way of looking at technology "through the eyes of the designer". It has already been pointed out that this *design paradigm* [172] has long dominated the philosophical debate on technical artefacts, indirectly contributing to leaving unexplored some important ontological, normative, and ethical aspects concerning the way we maintain and repair them. According to Steven Jackson [83], shifting from the designer's to the fixer's perspective calls for a radically new understanding of technical failure. Rather than an unexpected interruption, malfunction should be seen as manifesting the inherent fragility of the technological world we inhabit. Not idealised 'standard conditions', but friction, degradation, and falling apart of components are the real circumstances in which computational artefacts normally operate. Therefore, his suggestion is that we adopt a "broken-world thinking" to reason about them:

> [Broken world thinking] asks what happens when we take erosion, breakdown, and decay [...] as our starting points in thinking through the nature, use, and effects of information technology and new media. [83, p. 221]

The present work is "an exercise in broken-world thinking" [83, p. 221] within a discourse where design talk is nearly ubiquitous: that concerning the discriminatory effects of Machine Learning (ML) decisions. Current conceptualisations of algorithmic unfairness are in fact strongly focused on the role of ML design in both identifying causes and proposing remedies, and can therefore be described as design-centric. Within this view, algorithmic discrimination is understood as the harmful consequence of failing to design the right ML model, ultimately traced back to the poor quality of training data. Accordingly, ML fairness is framed as achievable through *better design choices*, such as selecting higher-quality training data or enforcing stricter benchmarks.

While, undoubtedly, these are all serious and legitimate strategies within the context of responsible Artificial Intelligence (AI), I argue that the activities of ML *redesign* and ML *repair* are not equivalent when it comes to their underlying epistemologies: the design paradigm in the former case, broken-world thinking in the latter. To substantiate this distinction, I highlight three aspects of ML errors that the design paradigm struggles to address but that broken-world thinking accommodates more naturally. From this, I suggest that broken-world thinking offers a more adequate framework than the design paradigm and, therefore, should be preferred in framing our understanding of algorithmic fairness.

The structure of the work is as follows: after outlining the main critiques of the design paradigm in the context of information technology (Section 3.2), in Section 3.3, I explore the question of what counts as maintaining and repairing an ML system and who are the actors involved in these activities. Section 3.4 recalls the main accounts of algorithmic discrimination, highlighting that they all entail an understanding of the phenomenon in terms of a failure of design, substantially in line with the design paradigm. Subsequently, in Section 3.5, I explore three aspects of ML errors that challenge the design paradigm: one epistemological, one temporal, and one practical. Finally, Section 3.6 makes explicit the main outcome of the work, namely that algorithmic fairness should be better framed as a repair practice, rather than as a static value embodied by specific ML designs.

3.2 The Design Paradigm and its Critics

Mark Young describes the *design paradigm* as the view that "understands function as stemming primarily from the way a technology was designed" [172]. This paradigm permeates a dominant narrative of engineering that identifies the

properness of such discipline with the moment of *designing technical artefacts*, in particular stressing certain markedly mentalistic aspects of design such as planning, creativity, expert knowledge and intentions of designers.

In light of the discussion in Chapter 1, it is clear that such a view has great affinities with the intentionalist position that dominates the philosophical debate on technical functions at large [75, 110], as well as on computational functions specifically [128, 154]. The intentionalist view of artefact functions can be regarded as the philosophical articulation of the design paradigm, as it posits that mental facts in the head of the designer are at least necessary, if not even sufficient, for establishing what an artefact is for (for a detailed discussion of the position, see Section 1.2).

More or less explicitly in contrast to this, an increasing number of scholars from various disciplines have proposed shifting the discourse on computational functions from the moment of artefact creation to the continuous process of human practices that shape the behaviour of the artefact *after* it has been designed.

It is worth noting that seminal work on the maintenance of computational artefacts dates back to the 1970s, during the early phases of software engineering literature, when authors and practitioners began to recognise maintenance as a vast and largely understudied set of technical practices that extend well beyond the simplistic notion of bug fixing. A clear sign of this emerging awareness can be found in Canning's metaphor of maintenance as an 'iceberg' [29] of invisible practices, as well as in Lehman's laws of software evolution:[1]

1. Continuing Change — The system must be continually adapted or it becomes progressively less satisfactory.

2. Increasing Complexity — As the system evolves, its complexity increases unless work is done to maintain or reduce it.

3. Self-Regulation — The system evolution processes are self-regulating with the distribution of product and process measures close to normal.

4. Conservation of Organizational Stability — The average effective global activity rate in an evolving system is invariant over the product's lifetime.

5. Conservation of Familiarity — The pace of change is limited by the need for stakeholders to retain system knowledge.

[1] Descriptions adapted from [128, p. 185].

6. Continuing Growth — The functional content of a system must be continually increased to maintain the same level of user satisfaction over time.

7. Declining Quality — The quality of a system will decline unless rigorously maintained and adapted to operational environment changes.

8. Feedback System — Evolution processes are multi-level, multi-loop, multi-agent systems and must be treated as fundamentally feedback-driven to achieve significant improvement over any reasonable base.

Based on empirical studies of IBM's internal processes of software development, Lehman [95] argued that computational artefacts do not remain static but evolve continuously in response to user needs and environmental pressures, following certain universal patterns comparable to laws in the natural and social sciences. His eight laws highlight the necessity of ongoing maintenance for large, real-world software systems, in order to counter the progressive erosion of their functionality. The laws fundamentally emphasise that maintenance is not an auxiliary activity but the very condition of software survival, showing that the meaning and functionality of computational artefacts are emergent properties of long-term maintenance and socio-technical co-evolution [65]. In particular, laws (2, 6, 7) explicitly recognise a direct nexus between maintenance activities and certain core dimensions of software artefacts, such as complexity (2), size (6), and quality (7).

Within the philosophical literature, two accounts that distance themselves from the design paradigm were already introduced in Chapter 1: the evolutionary view of technical functions and the User Levels account. The first has been described in Section 1.2 as rejecting the core claim of intentionalism that intentional facts are necessary for establishing functions. By analogy with natural selection, the function of an artefact of kind K is identified with the effects produced by past instances of K that both managed to persist and facilitated the proliferation of new tokens of the same kind. This perspective has recently been applied to computational artefacts specifically in [78]. The User Level theory has been extensively discussed in Chapters 1 and 2. As already pointed out, this ontology offers a more flexible intentionalist account of computational functions, according to which the normativity of a computational artefact is not determined solely by the content of the designer's intentions. Instead, it is negotiated among a wider range of agents—including low-level software implementers, policymakers, and end-users—each contributing their

own semantic and normative requirements. In Section 2.4, this position was applied specifically to the ML context.

Likewise, scholars in Science and Technology Studies (STS) have noted that the prevailing emphasis on design and innovation in computer engineering should be reassessed in light of the *actual* daily practices of software developers—such as bug fixing, versioning, backup, updating, and code documentation—most of which concern the maintenance rather than the design of software systems [120]. Building on this observation, Russell and Vinsel argue that information technology professionals are, first and foremost, maintainers rather than innovators, and that their professional ethics should be reframed accordingly [135]. In a similar vein, Jackson analyses the massive maintenance work underpinning data infrastructures, drawing attention to novel aspects such as the continuity between their material and virtual dimensions, as well as the political implications of data surveillance, data loss, and data errors. Finally, Steinert emphasises that repair activities are far from being merely conservative acts of preservation or restoration; instead, they involve a skilled, creative, and intentional production of values, comparable in significance to the design process itself [148].

It is also important to stress that Jackson's notion of broken-world thinking draws on a broader philosophical tradition. Pragmatist theorists have long highlighted the generative role of cognitive gaps and failures in domains such as pedagogy (e.g., Vygotsky's notion of scaffolding [162]) and communication (e.g., violations of Gricean maxims [67]). Most notably, the feminist notion of *care ethics* interprets never-ending tasks—such as housework, childcare, and emotional labour—as arising from the social expectation that women continually recognise and take responsibility for the inherent fragility of people, objects, and relationships. Seen in this light, women's often invisible labour is fundamentally oriented toward *keeping things going*, and can therefore be understood as a form of (unpaid) repair and maintenance labour. Feminist scholars have further argued that such care practices are epistemologically generative: marginalised individuals make sense of the world through their lived experiences of care and oppression, while remaining acquainted with dominant perspectives. This condition, described as "a bifurcation of consciousness" [72, p. 27], provides them with a richer understanding of social dynamics than privileged individuals, who are often blind to certain forms of oppression [111].

Jackson's notion of broken-world thinking builds on these insights, extending the idea of repair beyond domestic and social spheres to computational and

technological artefacts. Just as care work maintains the continuity and functioning of everyday life, broken-world thinking emphasises the ongoing practices required to maintain, correct, and sustain the normative and functional integrity of artefacts within complex digital infrastructures. His suggestion is that broken-world thinking endows fixers with a special understanding on the artefact that cannot be ultimately attained by designers and users:

> The question is this: can repair sites and repair actors claim special insight or knowledge, by virtue of their positioning vis-à-vis the worlds of technology they engage? Can breakdown, maintenance, and repair confer special epistemic advantage in our thinking about technology? Can the fixer know and see different things [...] than the better-known figures of 'designer' or 'user'? [83, p. 229]

Taking seriously this idea that repair work is an epistemological endeavour in its own right, by means of which a genuinely new type of knowledge on the artefact can be attained, the aim of the work is to explore *which distinctive insights* are gained once we replace the mainstream design paradigm with the broken-world perspective in understanding ML errors. However, to do this, it should first be clarified what "maintenance" and "repair" mean in the specific context of ML.

3.3 On Maintaining and Repairing AI

As already discussed in Section 2.3, Angius and Plebe observe an ongoing paradigm shift from the practice of manual *coding* in traditional software development to that of *data curing* in the ML context.

Drawing on a real-world example from autonomous driving Deep Learning (DL) technologies, the authors observe that it is increasingly common for the "effort in encoding different DL models [...] [to] not correspond in significant performance improvements" [4, p. 47]; conversely, they notice that "[m]ajor efforts are being spent in curing differently the dataset instead" [4, p. 48]. Here, 'curing' data refers to a set of sophisticated techniques—such as *labelling*, *cleaning*, *massaging*, *migration*, and *augmentation*—aimed at preprocessing the data used to train ML systems. The authors interpret this shift as evidence of the pivotal role that training data play in shaping the behaviour of an ML model. In light of this, my analysis will programmatically focus on data-related

maintenance and repair activities, setting aside those directed elsewhere, for instance, to the material support of ML systems (such as hardware, infrastructures, networks, and storage) or to the many manually coded components of the ML system (such as the training engines, data preprocessing pipelines, APIs, etc.).

Interestingly enough, the terminology used to refer to data curing activities sometimes evokes the semantic domain of more traditional *care* practices, suggesting a conceptual parallel between these data processing tasks and the broken-world perspective. A pivotal question about AI maintenance and repair becomes: Who is responsible for the maintenance of this enormous amount of data? Who are the *actors* of these 'curing/caring' practices, apart from the always-mentioned ML engineers in their shiny offices? In what follows, I identify three groups of data maintainers that, while largely operating outside public recognition, are essential to the good and continuous functioning of the data curing process. I call them the *unknown*, the *unrecognised*, and the *unaware* data maintainers, respectively.

The first category of *unknown* maintainers includes a diverse typology of data workers, generally outsourced from low-income countries through crowdsourcing platforms such as Amazon Mechanical Turk or Scale AI. These workers perform tasks such as labelling, ranking, and classifying the data that fuels supervised learning and reinforcement learning from human feedback. In the case of Large Language Models (LLMs), annotators are tasked with ranking model completions to train reward models or with labelling toxic and non-toxic content to filter harmful outputs. Data workers constitute a structurally invisible and low-skilled workforce that often lacks knowledge of the broader objectives of the systems they help construct [98]. Nonetheless, ML models inevitably end up reflecting their cultural predispositions and amplifies them on an unprecedented scale. For example, [43] have noted that in one widely used image dataset, "a photograph of a woman smiling in a bikini is labelled a 'slattern, slut, slovenly woman, trollop', while a young man drinking beer is categorized as an 'alcoholic, alky, dipsomaniac, boozer, lush, soaker, souse"'.

The second category is that of *unrecognised* data maintainers, primarily domain experts who engage with ML systems as part of their specialized work (e.g., to screen patients, interpret complex datasets, or support decision-making). Although they function as typically that of ML end-users, these experts also play an indispensable role in the construction, evaluation, and validation of ML models—yet their contributions are rarely formally

acknowledged. For example, in medical imaging applications, radiologists may be tasked with validating the outputs of ML classifiers. Their involvement is often largely extractive: they assess the quality of training data and model predictions, while being replaced by cheaper alternatives whenever possible [48]. In practice, this instrumentalisation of expertise means that their deep domain knowledge is seldom integrated into the core design process, creating a persistent risk of "frustration with technologies that do not fit their needs, knowledge, and workflows" [142, p. 2]. As a result, the insights and contextual understanding these professionals could provide to improve model robustness, reduce errors, or mitigate bias often remain untapped, highlighting a critical blind spot in conventional ML development practices.

Finally, *unaware* maintainers encompass the immense category of digital service users, whose everyday interactions—such as likes, preferences, flags, and feedback—continuously shape ML model behaviour. Active forms of data maintenance include downvoting irrelevant Google Search results, flagging misinformation on Facebook, or reporting traffic slowdowns in Google Maps. Beyond these explicit actions, virtually any publicly available online content can be scraped and incorporated into LLM training sets, meaning that almost any kind of online user activity also functions as a form of maintenance as long as it leaves some kind of data signal. When the continuous production of human-generated content ceases or stalls, it leaves behind an outdated set or signals, which social scientists have termed *digital remains* [99]; for example, Facebook profiles of deceased individuals are projected to outnumber those of the living by the end of the century [173]. Instead, as genuine human-generated content becomes increasingly contaminated by model-generated outputs—for example, when LLMs are trained on large volumes of automatically generated forum posts, social media replies that have been produced by other LLMs —the learning process risks degeneration, potentially resulting in catastrophic and unpredictable failures. In such cases, the model may gradually lose its capacity to acquire new knowledge and even forget previously learned information—a phenomenon referred to as *model collapse* [141]. This dynamic highlights how the ongoing, often invisible contributions of unaware maintainers are crucial to the long-term integrity and adaptability of ML systems.

If this concludes the examination of the main actors involved in the activities that *preventively* keep the ML system going (i.e., ML maintenance), other strategies are put in place *posthumously*, i.e., after the ML system has manifested a failure of functionality (i.e., ML repair). As Lin and Jackson

observe, the conception of ML errors practitioners have is first and foremost *negative*:

> the focus of ML practitioners [...] follows a 'limit and eliminate' approach, in which the value of errors is entirely negative; error functions first and foremost as an obstacle to be overcome, in pursuit of ML models that correspond more accurately and comprehensively to the conditions of a complex world. [98, p. 22]

The 'limit and eliminate' approach clearly reflects the design paradigm's ideal of seamless artefact functionality, framing ML errors as interruptions, deviations, or exceptions from intended behaviour. Yet this view proves inadequate once the fundamental nature of ML systems is taken into account. While more sophisticated designs can certainly reduce error rates, the prospect of a completely error-free system is unattainable, owing to the stochastic and probabilistic character of ML itself. As discussed in Section 2.3, the notion of Bayes error establishes a hard lower bound on predictive accuracy: a residue of misclassifications will persist regardless of design choices. To this irreducible margin of error we must add for example the phenomenon of adversarial examples, introduced in Section 2.3.2, i.e., inputs that, while often indistinguishable from ordinary data to humans, systematically fool deep learning models in the most bizarre ways. Crucially, adversarial examples are not only the result of deliberate attacks but also occur naturally within real-world datasets, making adversarial attacks largely unavoidable.

Together, concepts like Bayes error and adversarial inputs reveal the inadequacy of the 'limit and eliminate' strategy as a theoretical framework for understanding ML errors and that the belief that careful design alone can deliver seamless functionality is therefore inadequate, too. What is required instead is a perspective that acknowledges the inevitability of error and foregrounds the continuous labour of maintenance as a response to it. This point becomes even clearer when we consider the problem of algorithmic fairness, a paradigmatic domain in which the 'limit and eliminate' approach fails clearly.

3.4 Algorithmic Unfairness as a Design Error

Distributive fairness is defined as the realisation of a similar distribution of resources or burdens between individuals with different *sensitive attributes* such

as age, race, or gender. This understanding of fairness as a "similarity of distributions" has been transposed to the case of ML-driven decision scenarios, with the only difference that in such cases the optimal allocation of resources and burdens is learnt from observations of the past. Several proposals have been offered about what an algorithmically fair distribution of outcomes should look like and how ML models should technically achieve them [31].

Over the past decade, numerous frameworks have been proposed to reason about the unfairness of Machine Learning (ML) systems. Despite their many differences, some of these accounts have stressed that ML unfairness can be caused by a biased representation of certain sensitive groups within the data. Unfair predictions are often described as the consequence of "*mismeasured* proxies [of the sensitive attribute]", or emerging when "attributes [...] *misrepresent* the true behaviour of the users" [112, pp. 4, 7],[2] where concepts like *mis*-measurement and *mis*-representation clearly pertain to the vocabulary of data quality, in particular concerning the dimension of data correctness. Framing algorithmic unfairness in terms of data quality issues is also reflected in a growing interest in data-centric AI approaches [86]. Furthermore, according to this reading, the type of 'misrepresentation' that produces algorithmic discrimination is a noise with the two following characteristics. First, it is *unevenly distributed* between sensitive groups (e.g., men and women), as only errors that are non-uniform across sensitive groups can impact them differentially [84]. Secondly, it is *systematic*, in the sense that it consistently overestimates or underestimates the true values of an ML feature. This means that, unlike random noise, it cannot be averaged out by simply adding more records. I will refer to such a noise by "data bias" henceforth.[3]

To illustrate how data bias can lead to unfair ML outcomes, let us consider the following example:

Example 3 (Biased Exam Score) *At University X, black candidates consistently score lower on admission tests than their white peers. These disparities do not simply reflect differences in merit. Closer examination reveals multiple contributing factors: some exam questions presuppose cultural knowledge more common among white students; certain examiners exhibit bias against black applicants; and schools in predominantly black neighbourhoods often provide weaker preparation.*

As a result, the exam acts as a biased data-generating process, producing

[2] Emphasis added.
[3] Friedler et al. call it "group skew" instead [62].

uneven levels of noise in the observations of the two sensitive groups.[4] Let us now consider an ML system trained to predict students' academic success based on these exam scores. The university admits students predicted to succeed and rejects the others. Even if a black and a white student are equally diligent and intelligent, the biased scoring system can cause their ML predictions to differ substantially. Moreover, because the scores of black students are noisier, the model struggles to separate truly successful from unsuccessful black candidates, whereas the task is comparatively easier for white students. In other words, the two groups are associated with different levels of *separability* [51].

One way to address algorithmic unfairness is through *demographic parity*. This approach assumes that any observed difference in outcomes between groups entirely arises from the data bias. Under this simplification, academic success is treated as equally distributed across black and white students— corresponding to the "We Are All Equal" (WAE) assumption [62]. Fairness is then enforced in ML predictions by equalising the probability of a positive outcome ($\hat{Y} = 1$) for each group:

Definition 10 (Demographic Parity)

$$Pr(\hat{Y} = 1 \mid white) = Pr(\hat{Y} = 1 \mid black)$$

Demographic parity is conceptually simple and easy to implement, but it has an important limitation: it does not consider the causal pathways that generate the outcomes. In other words, it treats all differences between groups as undesirable, even if some differences are legitimately linked to factors unrelated to bias. In contrast, *counterfactual fairness* takes the complete opposite direction: that of describing, in full detail, how the sensitive attribute causally influences the ML prediction [94]. This requires reconstructing the causal model underlying the prediction.[5] In our university example, this requires quantifying

[4] This does not imply that the exam perfectly measures the abilities of white students either—for example, both groups may include cheaters at similar rates.

[5] A causal model is a pair $(\mathcal{S}, \mathcal{F})$ defined as follows. Signature $\mathcal{S}$ is a tuple $(\mathcal{U}, \mathcal{V}, \mathcal{R})$ where $\mathcal{U}$ is the set of *exogenous* variables, i.e., variables that causally depend on factors that are outside the scope of the model such as noise and background conditions; $\mathcal{V}$ is the set of *endogenous* variables, whose values are causally determined by other variables of the model; $\mathcal{R}$ is a function that associates each variable $X \in \mathcal{U} \cup \mathcal{V}$ with the non-empty set of its possible values. Finally, $\mathcal{F}$ is the set of structural equations that determine the value of Y as a function of those of the other endogenous and exogenous variables, i.e., $\mathcal{F}_X : \mathcal{R}(\mathcal{U} \cup \mathcal{V} - \{X\}) \rightarrow \mathcal{R}(X)$. Denoting by $\hat{Y}$ the ML prediction and by S the binary sensitive attribute with possible values $\mathcal{R}(S) = \{s, s'\}$, the computation of the counterfactual outcome for an individual with sensitive attribute s corresponds to the intervention $\hat{Y}_{S \leftarrow s'}$ [121]. This denotes the value of the predicted outcome $\hat{Y}$ as determined by a minimally modified version of $\mathcal{M}$ in which a new structural equation of

how a student's race affects their exam score and, subsequently, the ML prediction. Establishing the fairness of the ML model corresponds to ascertaining that black students would have obtained the same ML outcomes, had they been white, which formally corresponds to carrying out a causal intervention $\hat{Y}_{Race \leftarrow White}$ [121]:

Definition 11 (Counterfactual Fairness)

$$Pr(\hat{Y} = 1 \mid black) = Pr(\hat{Y}_{Race \leftarrow white} = 1 \mid black)$$

Hence, counterfactual unfairness is quantified as the difference between the actual ML outcomes obtained by the black candidates and counterfactual ones they *would* have obtained, in the counterfactual world where they were white.

To summarise, the key distinction between these approaches is that demographic parity treats algorithmic fairness as equality of outcomes at the group level, without accounting for the underlying causal mechanisms, whereas counterfactual fairness defines fairness at the level of individual predictions, ensuring that each factual prediction would remain unchanged under a counterfactual modification of the sensitive attribute. However, despite this difference, both notions ultimately share a common commitment to the design paradigm. They in fact understand fair ML prediction as ultimately caused by the poor data quality of the input ML data.[6]

According to the LoA taxonomy of ML errors introduced in Section 2.3, incorrect or biased training data are classified as a type of design error. In the ML context, in fact, design choices include selecting and curing datasets [4], choosing architectures, and specifying hyperparameters. In other words, selecting inadequate data for an ML model is analogous to manually implementing a correct program on a misinterpreted specification in traditional software. In both cases, harmful outcomes arise from an unintentionally misguided design process.[7]

In this regard, I contend that framing algorithmic unfairness solely in terms of design choices is epistemologically restrictive and should be treated with

S overrides its value to s'.

[6] It is crucial to remark that the theoretical significance of the notion of data bias to the problem of algorithmic discrimination is in no way disputed here. What is considered problematic is only its categorisation as a design error.

[7] It could be objected that some unfair outcomes stem from the intended functionality of the system—for instance, if a model is explicitly designed to discriminate [78] or assumes that its data perfectly represents individuals. Even in such cases, from a fair ML perspective, intentionally designing discrimination is morally unacceptable. Accordingly, the design-centric view frames discrimination primarily as the byproduct of biased data or flawed design choices.

caution. From this perspective, achieving fairness in ML is reduced to redesigning models through *better design choices*, such as selecting more representative datasets or imposing stricter benchmarks. While these measures are unquestionably valuable and constitute legitimate objectives within responsible AI, a design-centred approach neglects the broader socio-technical processes through which fairness emerges in practice. In particular, it fails to recognise the ongoing work of *repair* and maintenance—interventions that extend beyond the initial design phase and are more naturally captured within the framework of broken-world thinking. To illustrate the advantages of adopting a broken-world perspective, in the next section I highlight three dimensions of ML errors that the design paradigm struggles to address but that are more naturally accommodated within this framework: one epistemological, one temporal, and one practical.

3.5 Three Problems for the Design-Centric View of Algorithmic Fairness

3.5.1 Just Fixing Errors Can Miss Their Value

It is important to emphasise that broken-world thinking does not deny the epistemological status of malfunctions in computer science: testing, debugging, and fault-tolerant architectures are well-established fields, and software engineering textbooks consistently highlight the role of errors in anticipating potential risks and guiding subsequent redesigns. What broken-world thinking *does* challenge, instead, is the assumption that failure represents a clearly identifiable disruption within an otherwise ideal continuum of proper, seamless, and intentional use—an assumption that treats errors solely as anomalies to be eliminated wherever possible. What broken-world thinking *does* challenge, instead, is the assumption that failure represents a clearly identifiable disruption within an otherwise ideal continuum of proper, seamless, and intentional use—an assumption that treats errors solely as anomalies to be eliminated wherever possible. Previous work has already highlighted the limitations of this overly rigid conception of miscomputations in relation to both manually-coded computational artefacts [25] and to data-driven ones [24]. The debate surrounding the ontological status of adversarial inputs – a notion already introduced in Section 3.3 – further illustrates the complexities of this

ML normativity.

According to the prevailing view, adversarial inputs are mere errors, resulting from random glitches in the input image or statistical quirks that fool deep neural networks without corresponding to any genuine patterns. For example, altering just a few pixels in an image of a panda can cause a well-performing classifier to misclassify it as a gibbon with high confidence [66]. Critics of this standard interpretation have, however, pointed out that it is overly simplistic and cannot account for the evidence that adversarial inputs can transfer between ML models that have been trained on totally different data sets. This instead suggests that deep neural networks track certain non-robust features that are largely invisible to the human eye and that nonetheless reflect an "inherent geometry of the data" [82]. From this perspective, adversarial inputs are not mere errors, but convey genuine and potentially valuable predictive information, highlighting systematic vulnerabilities rather than random anomalies.

Interestingly, Buckner offers a third, orthogonal account of adversarial inputs, which differs markedly from both the prevailing and the critics' perspectives. In this view, adversarial inputs are neither mere errors nor purely informative signals; instead, they are analogous to humans' perceptual deceptions, such as optical illusions or acoustic distortions [22]. While adversarial inputs can carry useful predictive information, their opaque origins make them unreliable until interpreted within an appropriate theoretical framework. Crucially, such a framework would allow adversarial inputs to be harnessed rather than simply eliminated, akin to how humans exploit sensory delusions:

> the Doppler effect can cause naive observers to conclude that a train's horn changes frequency as it passes. Once [...] understood [it can be] even used as a reliable source of evidence, as Doppler shifts allow us to estimate the relative speed of an approaching signal source in weather forecasting. [22, p. 732]

This tripartite discussion of adversarial inputs highlights three fundamentally different stances on ML errors. The first, prevailing view treats adversarial inputs as random glitches or noise to be eliminated; the second, a more critical perspective, recognises them as meaningful but non-robust features of the data that provide genuine, though often imperceptible, predictive information. Buckner's orthogonal account, by contrast, positions adversarial inputs as systematic deceptions—unpredictable and opaque, yet potentially

informative—emphasising their role as opportunities for understanding and intervention rather than merely anomalies to be fixed.

In this sense, the first two positions reflect a designer-centric perspective: behaviours are judged either aligned with or deviating from intended functionality, to be tolerated or eliminated. Buckner's view, however, aligns more closely with a "fixer" or broken-world perspective: adversarial inputs reveal the limits of intended functionality, and failures are not simply problems to be removed but windows into the deeper, emergent properties of the system. This approach encourages learning from the system's failures and leveraging them to gain genuinely new knowledge about the model's inner workings, a type of knowledge that is richer and more revealing than the one required to merely assess whether such behaviours should be eliminated or lived with.

3.5.2 The Atemporal Illusion of Fairness-by-Design

Young has observed that the design paradigm largely neglects the temporal dimension of artefacts, since their function is considered fixed once and for all by the teleological intentions of the designer, remaining the same throughout their lifetime [171]. He has highlighted that, by contrast, the maintenance perspective treats technical functions as inherently temporal processes: they emerge and persist through continuous modification. Maintenance practices reveal that technologies are embedded within temporal processes, shaped not only by material and social constraints but also by ongoing human skills, values, and interventions.

It is therefore unsurprising that in the predominantly design-centric literature on algorithmic fairness, important temporal aspects have been largely overlooked. Fairness is often modelled as a static property, measurable at a single point in time, while ignoring long-term phenomena such as retraining, user adaptations in response to model outputs, data drift, evolving labels [130], and even potential negative consequences of fairness mitigation strategies over time [73].

The challenge of integrating temporality into a design-focused view is clearly illustrated by debates surrounding algorithmic recourse. Recourse refers to the set of actions an individual may be advised to undertake in order to reverse an unfavourable ML decision. For example, if a loan application is rejected, the system might recommend strategic steps to achieve a positive outcome, such as reducing existing debt or increasing savings. These actions unfold over very

different temporal horizons: reducing debt may take several years, whereas opening a savings account could be completed within weeks. For this reason, the temporal dimension has been remarked as critical when devising optimal recourse strategies [36, 17].

A deeper challenge, however, emerges when considering the evolution of the ML system itself. Between the time t at which the application is denied and the time $t + 1$ at which the recommended actions are completed, the model may be updated or retrained. Such modifications can render the original recourse recommendations ineffective, as there is no guarantee that a newly trained model will be backward-compatible with guidance issued by its previous versions. This temporal fragility exposes a fundamental limitation of design-centric approaches to algorithmic fairness: fairness cannot be guaranteed solely through static, one-off design interventions.

Ensuring genuinely fair outcomes therefore requires attentiveness to the dynamic interplay between models, their environments, and the individuals interacting with them over time. Broken world thinking provides a compelling alternative lens, emphasising that technical functions are not fixed, but emerge and persist through continuous processes of maintenance, adaptation, and intervention. From this perspective, fairness is not a static property to be realised once and for all, but an ongoing accomplishment, dependent upon sustained engagement with the evolving socio-technical system.

3.5.3 The Fairness-Accuracy Trade-off as a Design Myth

Corbett-Davies et al. observe that the debate on algorithmic unfairness is largely dominated by the notion that achieving fairness inevitably entails a reduction in accuracy, and vice versa [41]. Within this perspective, accuracy itself is taken to reflect ethically desirable outcomes much like fairness; for instance, ensuring that automated decision systems reliably detect fraudsters or tax evaders, or accurately predict the onset of diseases in patients, is considered morally valuable.

This perspective, which could be described as the 'tragic view of fairness', has significantly influenced the fair ML literature, favouring a conception of algorithmic fairness as an optimisation task between two values that stand in an inherent trade-off, i.e., accuracy and fairness [113]. For instance, in the automated assessment of welfare claims, a fair ML system should aim to

identify fraudulent cases with high accuracy while avoiding disproportionate flagging individuals belonging to particular sensitive groups. In other words, fairness is treated as a constraint on the optimisation of accuracy, requiring that the system minimise group-level disparities without compromising its predictive performance.

The adequacy of a fairness–accuracy trade-off was first problematised in Section 2.5, where examples from biometrics and Generative AI illustrated conflicts between the normative requirements of designers and end-users. In the literature, challenges to this trade-off have often been grounded in considerations of data quality, as discussed in Section 3.4 (see, e.g., [165, 18, 51, 37, 62]). These criticisms can be framed as a two-fold dilemma regarding the presence or absence of bias in the training data. Namely, if the training data do not contain data bias (according to the definition of data bias given in Section 3.4), they provide an accurate reflection of the true abilities and attributes of individuals. In this case, the notion of a fairness–accuracy trade-off becomes irrelevant: the most accurate ML model achievable is simultaneously the fairest, as it is the maximally faithful to reality.[8] Conversely, if the training data are biased (according to the same definition), the test data—being drawn from the same distribution—are also biased. As a result, the measured test accuracy is itself a distorted indicator of model performance. Maximising such a biased metric would therefore be meaningless, and once again the purported fairness–accuracy trade-off dissolves.

As Wick et al. put it:

> There is a pernicious modeling-evaluating dualism bedeviling fair machine learning in which phenomena such as label bias are appropriately acknowledged as a source of unfairness when designing fair models, only to be tacitly abandoned when evaluating them. [...] the labels against which we evaluate are often biased (unfairly against a protected attribute) in the very same way as the unfair classifier trained on them. [...] because the labels are biased, we must immediately assume that the corresponding accuracy measurements are also biased. [165, p. 1]

In either case, the fairness-accuracy trade-off dissolves, revealing that framing the problem in terms of a simple balance is fundamentally misguided. As the

[8] This has been referred to as a "conservative" notion of fairness, which treats the current *status quo* as the benchmark for fairness [132].

quote highlights, this trade-off largely emerges from an overly narrow focus on the training data, characteristic of the design paradigm: label bias is acknowledged when constructing the ML model but often ignored when evaluating it on test data. In other words, the evaluation phase treats biased test labels as if they were clean data, producing the illusion of a trade-off between accuracy and fairness.

This "modelling–evaluation dualism" exemplifies the limitations of a design-centric approach, which tends to confine attention to input data while overlooking the broader, systemic pervasiveness of data quality issues beyond the scope of the training sample. Data are always somewhat 'broken', in ways that are not casual but largely socially determined. By emphasising the continuous, relational, and systemic nature of data imperfections, broken-world thinking provides a more realistic and comprehensive framework for understanding and addressing issues of algorithmic fairness.

3.6 Conclusion

In the present work, I have critically examined the prevalent view of algorithmic unfairness as a problem of ML design. As I have argued, such a view risks producing an impoverished understanding of the challenges faced by the fair ML research programme, reducing it to the task of redesigning increasingly sophisticated ML models whenever harmful behaviours are observed. However, minimising algorithmic fairness demands a far more substantial understanding of the challenge to be faced. It requires a continuous renegotiation of the epistemological status of malfunctions (Section 3.5.1), a higher degree of time consistency in ML decisions (Section 3.5.2), and a realistic grip on the real pervasiveness of data quality issues (Section 3.5.3). These desiderata, as I have shown, are extremely difficult to accommodate within a purely design-centric framework.

It is, however, encouraging to note that recent philosophical work on artificial intelligence has begun to engage with the notions of repair and maintenance. This engagement is evidenced by what we might describe as two *broken-world metaphors*. A first metaphor is Crawford and Paglen's notion of "AI excavation" [43], echoed in the language of data set ruins [103] and of AI as a found object [79]. In this perspective, ML training data sets are not neutral repositories of information but rather conglomerates of material traces, akin to the artefacts unearthed and studied by the archaeologist. What comes into

focus is the culture that produced them: the largely invisible workforce of data labellers, annotators, moderators, raters, transcribers, curators, and others, who sustain the vast infrastructures of ML. Their judgements, biases, and evaluative frameworks are sedimented into the data and replicated by the ML behaviour. Much as the archaeologist interrogates artefacts, the metaphor encourages to ask: What do ML data sets reveal about the people and societies that created them? What values and purposes ultimately animate their production? How can we recover, by repair and maintenance, this lost functionality?

A second powerful metaphor is Leonelli's notion of "data lineages" [96]. Here, data are seen as specimens whose traits can only be reconstructed through careful genealogical work. Leonelli likens this to the practice of the evolutionary biologist, who patiently disentangles intricate layers of evidence to understand how organisms adapt within changing environments through time. Data, similarly, do not exist as static, timeless, and abstract entities; they carry the indelible imprint of being continuously curated, mobilised, repurposed, and reinterpreted. The lineage metaphor thus invites us to reconceptualise data not as disembodied abstractions but as material bodies that acquire meaning and function precisely through their capacity to adapt to shifting conditions of use.

Not only does broken-world thinking help us better addressing the above ontological, temporal, and practical blind spots of the design-centric view of algorithmic fairness. Importantly, the adoption of this alternative perspective also counters the risk of a design-centric "epistemic monoculture" of AI, a standpoint described by Messeri and Crockett [114] as one that suppresses alternative epistemologies, singularly amplifying particular viewpoints, methodologies, research questions, and values within scientific inquiry. For these reasons, I conclude that this epistemology may be more effective in establishing a theoretical foundation for our philosophical analyses of algorithmic discrimination. Rather than simply seeing the latter as a value embodied by this or that ML design, algorithmic fairness might be better seen as a plurality of reparative practices that aim to recognise and take responsibility for the inherent "brokenness" of the data with which ML systems are continuously fed.

With these motivations in the background, the next chapter concretely develops the kind of repair practice I here argued for, in the context of ensuring fairer ML outcomes. Crucially, the proposal developed in the next chapter is not aimed at *correcting* biased data nor at *retraining* the ML model to align with a predetermined fair distribution. In contrast, it aims to reason about the way in which a prediction made on biased ML features can be *repaired* by

modelling the noise contained in these features. This logical machinery is hence exploited to formalise how ML predictions are impacted by data bias and how this should be accounted for by a fairness-aware ML system.

Chapter 4

Data Speak but Sometimes Lie. A Formal Approach to Data Bias and Algorithmic Fairness

4.1 Introduction

Currently, there is still a lack of conceptualisation on the relationship between incorrect data and unfair predictions. In this context, we argue that there is value in stepping back and reframing the problem of unfair predictions from a data quality perspective. Taking seriously Friedler et al.'s suggestion that it might be "possible to take a more information-theoretic perspective on the nature of transformations between [construct and observed] spaces" [62, p. 5], we explore an information-theoretic and logic-based approach to data bias.

Specifically, we exploit the setting of a two-player game, called Ulam game [157], in which a Responder selects a secret element from a finite search space, and a Questioner tries to discover it by asking a series of Yes-No questions to the Responder:

> Someone thinks of a number between one and one million (which is just less than 220). Another person is allowed to ask up to twenty questions, to each of which the first person is supposed to answer only yes or no. Obviously the number can be guessed by asking first: Is the number in the first half million? then again reduce the reservoir of numbers in the next question by one-half, and so on. Finally the number is obtained in less than log2(1,000,000). Now suppose one were allowed to lie once or twice, then how many questions would one need to get the right answer? [157, p. 281]

In the game, it is common knowledge that the Responder can lie *at most* $0 \leq m < \infty$ times. Whenever $m = 0$, i.e. whenever the answer of the Responder are always truthful, the Ulam game is a sound and complete semantics for classical logic, while whenever $m > 0$, the game is a sound and complete semantics for the $m + 2$-valued Łukasiewicz logic [118, 117].

In previous work [11], the Ulam game was used to model data-driven reasoning in science. Scientists, like the Questioner, aim at discovering which scientific hypothesis H^* holds (the secret) in a search space $\mathcal{H}$ that can be understood as a set of mutually exclusive and exhaustive hypotheses. They pursue this objective by performing experiments, the outcomes of which can be interpreted as the answers that Nature gives to Yes-No questions. Thus, Nature plays the role of the Responder, and the untruthful answers Nature gives can be seen as errors inevitably slipping into experimental data. One fundamental aspect of data-driven reasoning captured by the game is that the process of hypothesis rejection comes in degrees. A hypothesis is said to be *temporarily rejected* whenever a single (possibly untruthful) question-and-answer entails its negation, while it is said to be *rejected* if it has been temporarily rejected at least $m + 1$ times.

In the present work, we adapt the Ulam game to another data-driven scenario, namely that of ML predictions. A prediction can be understood in fact as the outcome of the game, where Questioner corresponds to the trained ML model, and Responder to the input data. The Questioner wants to know whether the data point is H or $\neg H$ by asking Yes-No questions that correspond to binary features; at each new sequence of question-and-answer, Questioner updates the hypotheses' degrees of rejection accordingly. In this framework, data errors can be accommodated quite naturally in terms of untruthful responses from the Responder, and data bias can be modelled using a variant of the classic version of the game where the Responder can choose to deliver responses through different channels, each characterised by a different number of lies. Each channel represents a possible data-generating process, characterised by its specific noise level. In an unbiased situation, Responder's answers can be noisy, but this noise is uniformly distributed across sensitive groups. In Ulam terms, every data point "is playing the same game". In contrast, in a situation of data bias, the malicious Responder chooses to use a noisier (or less noisy) channel to deliver the information about the individuals that belong to a specific sensitive group.

Our model helps to highlight how problematic the idea of fairness as "similar individuals should receive similar outcomes" [52] is. In fact, in the presence of

data bias, similar individuals belonging to different sensitive groups might produce very different observations depending on the channel used by Responder and, conversely, very similar observations might correspond to factually dissimilar individuals belonging to different sensitive groups. Therefore, similarity between inputs might not be meaningful in the presence of data bias. To illustrate this, we introduce a real-world medical scenario in which a racially biased data-generating process produces significantly different observations for factually very similar patients, who end up receiving significantly different treatments. Instead of the definition of fairness that rests upon input similarity, we hence turn to consider a more procedural principle for fairness according to which all individuals should be evaluated along the same criteria. We refer to this fairness conception as the "Principle of Consistency" and note that, in an ML-enabled scenario, such evaluation criteria correspond to the information encoded in the input features of a data point. Crucially, failing to associate each data point with the correct channel (its actual noise level) results in a violation of such a principle: although an ML model uses the same set of observations for all individuals, there can still be significant disparities in the robustness associated with them. Using a variation of the rational non-monotonic consequence relation inspired by the Ulam game, we manage to effectively quantify such a disparity in a novel way, assuming no causal knowledge on the data-generating process, by only looking at the input data of an ML model and its associated decisions. To the best of our knowledge, this is the first game-theoretic formalisation of ML unfairness.

The work is organised as follows. After introducing a working example from the medical domain (Section 4.2), we give some preliminaries about the Ulam game (Section 4.3) and show how this game-theoretic formalisation provides a suitable model for ML decision-making (Section 4.4). In Section 4.5, we first define a rule of constrained monotonicity for the Ulam game, called (UMO), and its generalised version for weighted evidence (gUMO). After connecting our proposal with already existing related work (Section 4.6), in Section 4.7 we introduce a proof-theoretic system that captures key aspects of the Ulam game and subsequently establish a completeness result for the Ulam consequence relation $\mid\!\sim_{RJ}$. Finally, we conclude by discussing the general applicability of the proposed framework for reasoning about data bias and ML fairness (Section 4.8).

4.2 A Medical Example

To illustrate our methodology, we start with an example taken from an independent report on the equity of medical devices recently released by the English National Health Service (NHS) [164]. In this work, a fictitious story is used to exemplify how a biased measurement impacts patient's outcome in an ML-aided medical scenario. In the story, a pulse oximeter – an optical device that sends light waves of various frequencies through the patient's skin to make measurements of underlying physiology – systematically overestimates the true oxygen saturation level in individuals with higher levels of melanin in the skin. The narrative is realistic, since such a racial bias as been extensively evidenced in the scientific literature [143, 55].

Set during the COVID-19 pandemic, the story is about two patients of different skin colours who are screened by an ML risk-assessment system to decide which one needs intensive care the most. It is implicitly assumed that, given the emergency situation, there are limited resources and *only one of them* can be admitted to the intensive care unit at the moment of initial evaluation. Crucially, the measured level of blood oxygen in the dark-skinned patient (not recognised as biased by the doctors) causes her to obtain delayed care.

> Steven, a 58-year-old **White** British man, has been experiencing worsening **breathlessness, fever and chest pain**. He arrives at his office, sweaty and breathless, and with a new cough. His manager calls 999. Paramedics attend and find him so breathless that they rush him to the nearest emergency department. A PCR test for COVID-19 is positive, and his chest x-ray shows severe **pneumonitis**. A pulse pulse oximeter shows his **blood oxygen levels are low**. The emergency team assesses him using a standard clinical risk prediction score, which shows a **very high score**. The hospital's protocol for scores this high means that he is referred to the intensive care unit (ICU) to receive **respiratory support** [...]. He is seriously ill for a few days, but makes a good recovery. He goes home after 10 days.
>
> Yvonne, a 60-year-old **Black** British woman of Caribbean heritage, has **similar symptoms** and is barely able to manage her job as a cleaner because of her **breathlessness**. As an agency worker, she has no sick pay, so she delays calling 999 until she is home from work. Paramedics find her so breathless that they take her straight to the nearest emergency department. A PCR test for **COVID-19 flags positive**, and her chest x-ray shows **severe pneumonitis**. A pulse pulse oximeter shows her

> **blood oxygen levels are just within normal** range. The emergency team assesses her using a clinical risk prediction score, which comes back **high, but not sufficiently high to trigger a referral to intensive care**. **Further tests** show her **oxygen levels are worse** than the pulse pulse oximeter suggests. She is finally **moved to intensive care** [...]. Her oxygen levels are now critically low, so she is put on a **ventilator** to save her life. (emphasis of the authors, [164, p. 39])

It is important to notice that, from a medical standpoint, a low level of oxygen in the blood is associated with a high risk of requiring intensive care.[1] Therefore, data *do speak in favour* of Steven over Yvonne in this story; the question is: are they telling the truth? Based on the two patients' data, a clinician who is unaware of the bias produced by the pulse oximeter reading would presumably deem it *fair* to choose Steven over Yvonne, as suggested by the risk score returned by the ML model. The clinician would probably appeal to two main procedural aspects of the decision process:

Principle 1 (Justifiability of Evaluation) *The features used to evaluate individuals have been selected based on domain knowledge and their use is justified by their having causal influence on the individuals' outcome.*

Principle 2 (Consistency of Evaluation) *A consistent method of evaluation has been used for both individuals, as the same information has been collected about them.*

According to the clinician who assesses the ML outcome, choosing Steven over Yvonne is fair in virtue of the justifiable and consistent procedure used. In what follows, we want to examine whether this procedure is actually so. While we will assume the Principle of Justifiability 1 to hold – at least defeasibly, based on the clinician's current medical knowledge– we will focus on the Principle 2, to the conclusion that it is ultimately *not satisfied*. It will be shown in fact, that not the same *information*, but merely the same *observations* have been collected about the two patients. Our proposed information-theoretic approach shows that the two patients' data are associated with different levels of information. To quantify it, we start by introducing the key logical aspects of the Ulam game.

[1] Here, we assume that the algorithm is not aware that even a normal oxygen level is associated with high risk for a specific group of patients (dark-skinned ones). This might happen for a number of reasons. The system might have been trained on unbiased measurement methods of blood oxygen (like through blood tests, for instance), or the training set did not contain enough records of dark-skinned patients, or again, the ML model was specifically designed to incorporate medical-specific knowledge about the importance of blood oxygen level to evaluate the risk of serious respiratory issues in patients. Note also that we assume that the ML model is not aware of the race of patients, as this is not an input feature.

	Questions	**Steven's Answers**	**Yvonne's Answers**
0	70+ years old?	No	No
1	Breathlessness?	Yes	Yes
2	Fever?	Yes	Yes
3	Chest pain?	Yes	Yes
4	Cough?	Yes	Yes
5	COVID-19 test?	Yes	Yes
6	Severe pneumonitis?	Yes	Yes
7	Low blood oxygen?	Yes	No
	ML Decision	Move to IC	Not move to IC
8	Follow-up Test 1?	-	Yes
9	Follow-up Test 2?	-	Yes
10	Follow-up Test 3?	-	Yes

Table 4.1: An intuitive formalisation of the medical observations collected for assessing the risk of the two patients of the example illustrated in Sect. 4.2. In the table, we have assumed that both patients' age (58 and 60, respectively) is below a certain fixed risk threshold (arbitrarily set to 70) for the sake of the example.

4.3 Preliminaries

Starting from the idea that data can reject hypotheses, in [11] *degree of rejection* functions have been introduced. A generic degree of rejection is defined over the multiset[2] of data $\mathcal{M}_{\mathcal{D}}$ times the set of hypotheses and has values in the positive real numbers or ∞. The use of multiset is necessary to accommodate the fact that seeing the same data once is not the same as seeing it multiple times. Suppose that $\mathcal{H} = \{H_1, \ldots, H_n\}$ and $\mathcal{D} = \{d_1, \ldots, d_l\}$ are two sets of propositional variables standing for *hypotheses* and *data*, respectively. Then, $\mathrm{Fm}_{\mathcal{H}}$ and $\mathrm{Fm}_{\mathcal{D}}$ are the sets of propositional formulas generated by closing $\mathcal{H}$ and $\mathcal{D}$, respectively, under the usual connectives $\wedge, \vee, \neg$. Note that the definition of degree of rejection does not tell us how to compute it, but only which properties a function should satisfy to be a degree of rejection.

[2] A multiset is a generalisation of a set that allows multiple occurrences of the same element. Unlike sets, where each element appears only once, multisets keep track of the number of times each element appears (its multiplicity).

Definition 12 *We say that* $r\colon \mathcal{M}_{\mathcal{D}} \times \mathcal{H} \to \mathbb{R}_+ \cup \{\infty\}$ *is a* degree of rejection *if, for any* $\gamma, \delta \in \mathrm{Fm}_{\mathcal{D}}$, $\Delta \in \mathcal{M}_{\mathcal{D}}$ *and* $H \in \mathcal{H}$, *the following hold:*

1. *If* $T, \gamma \models \delta$, *then* $r_\gamma(H) \geq r_\delta(H)$.
2. *If* $T, H \models \delta$ *then* $r_\delta(H) = 0$.
3. *If* $T, \delta \models \neg H$, *then* $r_\delta(H) > 0$.
4. *If* $r_\Delta(H) \neq \infty$, *then* $r_\Delta(H) = \sum_{\delta \in \Delta} r_\delta(H)$.

The classical formula T, defined as

$$T := \bigvee_{H \in \mathcal{H}} H \ \wedge \bigwedge_{H, H' \in \mathcal{H},\ H \neq H'} H \to \neg H',$$

formalizes the assumption that the set of hypotheses $\mathcal{H}$ is exhaustive and mutually exclusive.

Once the degree of rejection has been defined, we can identify the set $\hat{\mathcal{H}}_\Delta$ of hypotheses in $\mathcal{H}$ which are *least rejected* by the data in Δ:

$$\hat{\mathcal{H}}_\Delta = \arg\min_{H \in \mathcal{H}^f_\Delta} r_\Delta(H). \tag{4.1}$$

where $\mathcal{H}^f_\Delta$ is the set of hypotheses with a finite degree of rejection. We have then recalled all the necessary elements for defining the *RJ-consequence relation* (where RJ stands for *rejection*).

Definition 13 (RJ-consequence) *Let* Δ *be a multiset of formulas in* $\mathrm{Fm}_{\mathcal{D}}$ *and* $\varphi \in \mathrm{Fm}_{\mathcal{H}}$. *We say that* φ *is an* RJ-consequence of Δ, *written* $\Delta \mathrel{|\!\sim_{RJ}} \varphi$, *if* $T, H \models \varphi$, *for each* $H \in \hat{\mathcal{H}}_\Delta$.

Based on Definition 13, in [11] a family of non-monotonic consequence relations is introduced. One of them, in particular, finds its motivation in the play process of the Ulam game. This consequence relation is defined by instantiating the degree of rejection that was defined in Definition 12 as follows. Suppose that H^* stands for the secret and m for the maximum number of lies Nature can tell. Then, the Ulam rejection degree r^u is defined as:

$$r^u_\delta(H) = \begin{cases} \infty & \text{if } T \models \neg\delta, \\ 1 & \text{if } T, H \models \neg\delta, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

$$r^u_\Delta(H_i) = \begin{cases} \infty & \text{if } H_i \in \mathcal{H} \text{ and } \sum_{\delta\in\Delta} r^u_\delta(H^*) > m \\ m+1 & \text{if } H_i \neq H^* \text{ and } \sum_{\delta\in\Delta} r^u_\delta(H_i) > m \\ \sum_{\delta\in\Delta} r^u_\delta(H_i) & \text{otherwise.} \end{cases} \tag{4.3}$$

Thus, the rejection degree has value ∞ only if we go against the game's rules (the secret H^* has been rejected more than m times) or the sum of the rejection degrees of the single $\delta \in \Delta$ is strictly greater than m.

4.4 Modelling ML Decisions as Ulam Games

4.4.1 Modelling ML Features

Our language is composed of two (finite) sets of propositional variables $\mathcal{D} = \{d_1, \ldots, d_l\}$ and $\mathcal{H} = \{H_1, \ldots, H_n\}$, respectively corresponding to the feature space (or data) and to the target label space (or hypotheses).

In the oximeter example, for instance, the set of hypotheses is $\mathcal{H} = \{IC, \neg IC\}$, which corresponds to the possible prediction of the ML model. Moreover, we denote with $Q_i(Yv)$ the i-th question of Table 4.1 whenever it is referred to Yvonne. Similarly, $Q_i(St)$ denotes the i-th question whenever it is referred to Steven. Then, $Q_i(Yv) =$ "Yes" indicates that the answer to question i on Yvonne is positive. To enhance readability for positive answers to question i we will write $Q_i(a)$, and $\neg Q_i(a)$ for negative answers, where a denotes a generic patient. Thus, $\mathcal{D} = \{Q_i(a) \mid i = 0, \ldots, 10\}$. Accordingly, since the input features used by the ML model to assess the patients' respiratory risk correspond to questions 0-7 in Table 4.1, they can be rephrased as follows:

$$\begin{aligned} \Delta_{Yv} \;=\; & \{\neg Q_0(Yv), Q_1(Yv), Q_2(Yv), Q_3(Yv), Q_4(Yv), Q_5(Yv), \\ & Q_6(Yv), \neg Q_7(Yv)\} \end{aligned}$$

$$\begin{aligned} \Delta_{St} \;=\; & \{\neg Q_0(St), Q_1(St), Q_2(St), Q_3(St), Q_4(St), Q_5(St), \\ & Q_6(St), Q_7(St)\} \end{aligned}$$

Since the level of admissible noise in the data Δ_{Yv} and Δ_{St} is not known, we initially assume it to be greater than the number of questions. Otherwise, Δ_{Yv} and Δ_{St} would yield very similar (or even the same) rejection degrees for both

hypotheses. If r is the degree of rejection function defined by the functions (4.2) and (4.3), then:

$$r_{\Delta_{Yv}}(IC) = 2, \text{ and } r_{\Delta_{Yv}}(\neg IC) = 6,$$

$$r_{\Delta_{St}}(IC) = 1, \text{ and } r_{\Delta_{St}}(\neg IC) = 7.$$

According to a deterministic function that maps the data onto hypotheses, it is possible to rewrite the elements in Δ_{Yv} and Δ_{St} in the language of $\mathcal{H}$. In this example, we assume that for all $i = 0, \ldots, 7$

$$Q_i(a) \equiv IC$$

while

$$\neg Q_i(a) \equiv \neg IC.$$

Thus, from every positive answer to questions 0-7 of Table 4.1 it classically follows that the generic patient a should go to intensive care (IC). The contrary follows from negative answers. Consequently:

$$\Delta_{Yv} \equiv IC^6 \wedge \neg IC^2$$

$$\Delta_{St} \equiv IC^7 \wedge \neg IC$$

where IC^k stands for $\bigwedge_{i=1,\ldots,k} IC_i$ and $\neg IC^k$ stands for $\bigwedge_{i=1,\ldots,k} \neg IC_i$. The pedics of IC_i and $\neg IC_i$ have the sole role of counting the occurrences of IC and $\neg IC$.

In ML terms, these equivalences convey the information one can obtain through feature importance techniques such as SHAP [101]. In fact, $Q_i(a) \equiv IC$ expresses that observing that a satisfies predicate Q_i (e.g., has fever) "contributes" to hypothesis IC rather than $\neg IC$. However, it is worth noting that the analogy needs to be refined, as here every observation is assumed to have the same importance, which is unrealistic for the large majority of ML scenarios. This assumption will be dropped in Section 4.5.2. In addition, if $m = 0$ and the search space consists of only two hypotheses, then we would need only one question-and-answer to determine whether a patient should be admitted to the IC. In practice, each of the questions $Q_0, \ldots, Q_7$ in isolation is not sufficient for the practitioner to make a decision.

Ulam game	**ML interpretation**
Questioner	ML system
Questions	Feature space $\mathcal{X}$
Responder	Nature
Answers	Data Point $\vec{x}$
Different channels	Data Bias
Lies	Incorrect data
Secret	True label y

Table 4.2: Correspondence between the Ulam Game and ML predictions.

4.4.2 Modelling ML Decisions

By ML decision-making, we mean a decision process that involves the use of an ML algorithm to define a function $\hat{f} : \mathcal{X} \to [0, 1]$ which associates individuals with a probabilistic prediction (such as a risk score). This prediction is used to prioritise individuals, given a decision function d and a fixed threshold τ:

$$d(\vec{x}) = \begin{cases} 1 & \hat{f}(\vec{x}) \geq \tau \\ 0 & \text{otherwise.} \end{cases}$$

We model this ML-enabled decision-making as an Ulam game in the following way. First, each ML prediction corresponds to a play of the game where Questioner corresponds to the trained ML model, and Responder to the data point $\vec{x} = \{x_1, \ldots, x_n\}$ it is fed with. Questioner wants to find the data point's true label y between mutually exclusive hypotheses (in what follows these will be assumed to be just two, H and $\neg H$, to model standard risk-assessment scenarios). Questioner asks a number of yes-no questions about $\vec{x}$, each corresponding to a binarised feature x_i. After every answer, Questioner updates the degree of rejection of H and $\neg H$ accordingly. Table 4.2 summarises the mapping between the fundamental concepts of Ulam game and ML predictions.

In our simplified example with only two patients, a decision needs to be made on which patient, between Yvonne and Steven, should be admitted to the IC unit. It is important to note that both patients show symptoms of a respiratory condition. In fact, IC is the least rejected hypothesis for both, i.e., $\hat{\mathcal{H}}_{\Delta_{Yv}} = \hat{\mathcal{H}}_{\Delta_{St}} = \{IC\}$. Therefore, $\Delta_{Yv} \mid\!\sim_{RJ} IC$ and $\Delta_{St} \mid\!\sim_{RJ} IC$. Therefore, by

Definition 13, $\Delta_{Yv} \mathrel{|\!\sim_{RJ}} IC$ and $\Delta_{St} \mathrel{|\!\sim_{RJ}} IC$. Nonetheless, we also know that $d(Steven) = 1$ and $d(Yvonne) = 0$, where 1 corresponds to "Move to IC" and 0 to its negation. In fact, we assumed in Section 4.2 that *only one* patient can be moved to the IC unit, due to limited resources. Denoting by a a generic patient in $\{Yvonne, Steven\}$ and by $-a$ the other one, d is defined as follows:

$$d(a) = \begin{cases} 1 & r_{\Delta_a}(IC) < r_{\Delta_{-a}}(IC) \\ 0 & r_{\Delta_a}(IC) > r_{\Delta_{-a}}(IC) \\ \texttt{undec} & \text{otherwise} \end{cases} \tag{4.4}$$

Had Steven's and Yvonne's rejection degrees been the same, the decision function would have been undecidable ($d(a) = \texttt{undec}$). In fact, the available data Δ_a would have been insufficient to justifiably move either patient to intensive care.

4.4.3 Modelling Data Bias

If we assume the oximeter measurements to be accurate (i.e., error $\varepsilon = 0$), from the data we have, Yvonne's clinical picture differs from Steven's. In reality, we know this is not the case, as the two patients have in fact similar conditions. In fact, the device has captured Steven's (factually low) blood oxygen level in a sufficiently accurate way, while it has overestimated Yvonne's (factually low, as well). More specifically, her blood oxygen level is masked as normal (meaning, above a given threshold t assumed to be the same for both patients). We do not know how much this bias ε amounts to, but we know that had Yvonne's blood oxygen level been lower than $t-\varepsilon$, she would have been flagged as $Q_7(Yv)$. Since this has not been the case, Yvonne's true saturation level is between $(t - \varepsilon)$ and t.

To model data bias, we modify the classic version of the Ulam game by introducing the notion of channels into the game. Similarly to previous applications of the Ulam game to error-tolerant communication [122], we interpret each channel as corresponding to a different number of lies available to Responder/Nature in the classic version of the Ulam game [35]. We can think of each channel as corresponding to a different noise level in the communication *medium* used by Responder/Nature to answer Questioner. It might be that the answer Questioner receives through a channel is totally accurate (no lies), it

might be that Responder/Nature can use up to m lies, or it might even be that Responder/Nature's lies are related to the outcome of an aleatory experiment – in this latter case, we are talking of a channel with probabilistic lies [116, 42].

Crucially, every data point is associated with its specific channel or set of channels (ideally, one for each feature and independent of each other). In the oximeter example, we know that the device underestimates the level of oxygen in dark-skinned patients, while it works sufficiently well for light-skinned ones. In medical terms, a pulse oximeter is put on Yvonne's finger, and then the data appearing on the screen is transcribed as "low" or "normal" (depending on the value of threshold t) on her medical file and given as input to the ML algorithm. In Ulam terms, this corresponds to asking the Responder Question 7. The number of lies actually used by Responder within Steven's game is $m_{St} = 0$, while for Yvonne $m_{Yv} > 0$ (note that we are able to reconstruct this only because we know that $\neg Q_7(Y)$ should have been $Q_7(Y)$). For instance, if $m_{Yv} \leq 2$ – i.e., at most 2 of the observations in Δ_{Yv} are incorrect – then the compatible correct data would be:

1. $\Delta'_{Yv} \equiv IC^6 \wedge \neg IC^2$ (no lie has been said);
2. $\Delta''_{Yv} \equiv IC^8$ (2 lies on $\neg IC$);
3. $\Delta'''_{Yv} \equiv IC^4 \wedge \neg IC^4$ (2 lies on IC);
4. $\Delta''''_{Yv} \equiv IC^5 \wedge \neg IC^3$ (1 lie on IC, 1 lie on $\neg IC$).
5. $\Delta'''''_{Yv} \equiv IC^7 \wedge \neg IC$ (1 lie on $\neg IC$);
6. $\Delta''''''_{Yv} \equiv IC^5 \wedge \neg IC^3$ (1 lie on IC);

Thus, if we apply the decision function (4.4) to cases (1-6), we would obtain:

1. $d(\Delta'_{Yv}) = 0$;
2. $d(\Delta''_{Yv}) = 1$;
3. $d(\Delta'''_{Yv}) = 0$;
4. $d(\Delta''''_{Yv}) = 0$.

When $m_{Yv} \leq 2$, it can no longer be inferred that Steven should be chosen over Yvonne for intensive care. In fact, supposing that Responder can use up to 2 lies, it is no longer true that $d(\Delta_{Yv}) = 0$ for all correct evidence $\Delta^*{}_{Yv}$ compatible

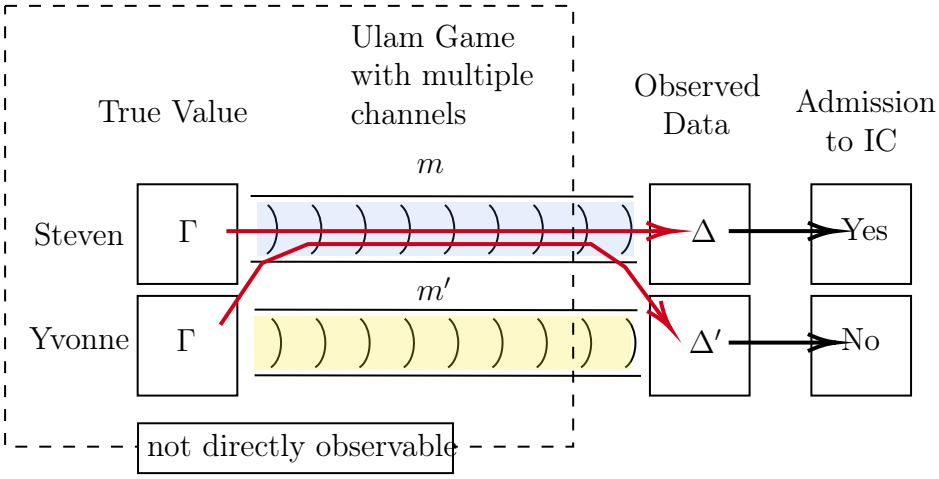

Figure 4.1: Initial and Unfair Position. Observations Δ and Δ' have been produced through two channels characterised by different levels of noise, respectively m and m', with $m' > m$. Despite this, as the red arrows indicate, the unaware Questioner misassociates observations Δ' with the wrong channel, assuming them to be more truthful than what they actually are.

with Δ_{Yv}.[3] Specifically, the observations at hand would be compatible with the situation in which Yvonne is actually *more in need of intensive care* than Steven – this happens when both lies are on $\neg IC$, corresponding to case (2.). In other words, it might be the case that her oxygen level is below the threshold t and $\neg Q_7(Y)$ as result of a lie, just as much it could be the case that her oxygen level is factually above the threshold t and $\neg Q_7(Y)$ is accurate. In a situation like this, data bias corresponds to the presence of more than one channel connecting Question 7 to the patients. Given these channels, unfairness corresponds to the situation in which a data point is misassociated with the wrong channels, while fairness corresponds to the situation in which this misassociation does not occur. Figures 4.1 and 4.2 illustrate how the medical example could produce these unfair and fair decisions, respectively.

Let us now see how a proof-theoretic investigation of the consequence relation capturing the Ulam rejection degree can hint at a way to test a generic ML algorithm for this misassociation.

4.5 Model-based ML Testing Within Ulam

In the classic version of the Ulam game, we can detect a lie whenever we receive contradictory information from Nature. For example, if we ask the same

[3] Here, $* \in \{', '', ''', '''', ''''', '''''' \}$.

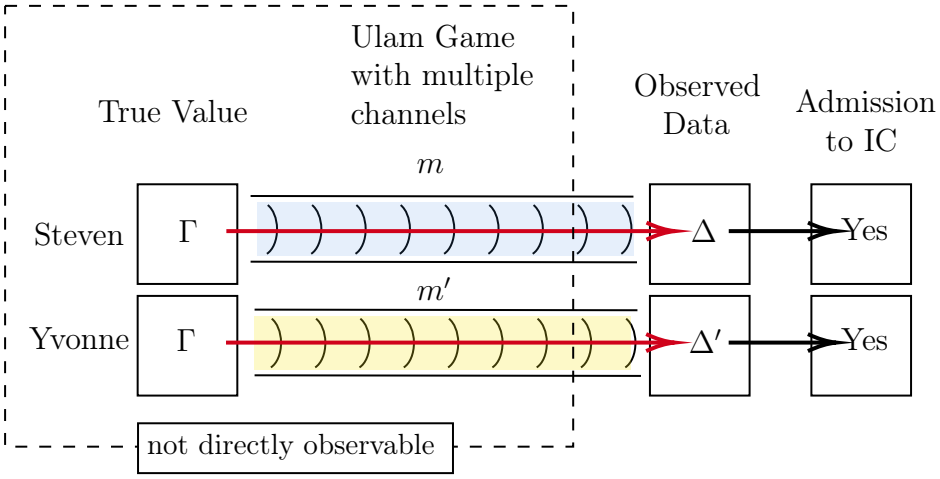

Figure 4.2: Final and Fair Position. Observations Δ and Δ' have been produced through two channels characterised by different levels of noise, respectively m and m', with $m' > m$. The red arrows indicate that Questioner is correctly associating Δ and Δ' to their true noise level.

question twice and obtain different answers, then one of the two must be false, and we can infer that Nature has lied. Suppose l lies have been spotted, then the number of lies m is *at least* l ($m \geq l$).

In the version of the game we consider, however, we assume that m is fixed but it is not common knowledge, i.e., only Nature knows it. This makes it difficult to recognise incorrect data: any data we collect may or may not be so, as we cannot know when the lies are over. Nonetheless, we know that *there must be* a point in the game at which Responder runs out of lies, since m is finite.

In a game setting like this, although we cannot recover m from the data obtained by questioning Nature, it is nevertheless possible to define an order over the hypotheses based on how many times they have been rejected. Moreover, since the number of lies available to Nature will eventually end, we know that from a certain point onwards (that we do not know), Nature's answers will all be correct. This means that the more questions we ask, the more robust – i.e., invariant to new and possibly erroneous information – our rejection function becomes. Let us suppose, for instance, that we could gather all possible information about Yvonne and use it as input of an ML model accordingly trained. Similarly to the oximeter reading, some other observations would be noisy. Yet, the usual assumption of independence prevents them to be *all noisy in the same direction and with the same skew*. While we would not know m and, therefore, we could not be able to establish at which point Responder has run out of lies, we would still notice that, as observations are added, one hypothesis starts being inferred more and more consistently from the data.

The broader our set of observations, the more robust our rejection order on hypotheses becomes and only an increasingly exceptional set of observations could change it. Hence, measuring the distance in rejection degrees between the least rejected hypothesis and the second leats rejected one can indicate whether the observations are in fact converging on a hypothesis. Even though assuming that errors in data might occur at maximum only a fixed and finite number of times is an oversimplification on the usual distribution of errors in data, in the following, we show how this robustness is effectively captured by the rule of Ulam constrained MOnotonicity (UMO) first (Subsection 4.5.1), and then by its Generalised version (gUMO) for weighted evidence (Subsection 4.5.2). In the conclusions, we show how utilizing variants of the Ulam game, such as Ulam games with probabilistic half-lies, could better model the distributions of errors in biased data.

4.5.1 The (UMO) Rule of Constrained Monotonicity

The uRJ-consequence relation obtained by instantiating r in Definition 13 with r^u satisfies the following rule of constrained monotonicity.

$$\frac{\Delta \mathrel{|\!\sim} \psi \quad \{\Delta \setminus \Delta \mathrel{|\!\sim} \psi\}_{\delta \in \Delta}}{\Delta, \varphi \mathrel{|\!\sim} \psi} \text{ (UMO)}$$

Let us now see a possible way to read (UMO). Suppose that whatever there is on the left-hand side of the consequence relation $\mathrel{|\!\sim}$ stands for the data we have, while on the right-hand side of $\mathrel{|\!\sim}$, we find what follows from the data Δ. Therefore, suppose that ψ follows from the data in Δ, in symbols, $\Delta \mathrel{|\!\sim} \psi$. Since Δ formalizes the data we are considering, it is defined as a multiset of formulas, and each $\delta \in \Delta$ can be seen as an observation. Suppose now to remove one single observation from Δ and verify whether from $\Delta \setminus \delta$ we can still infer ψ. If $\{\Delta \setminus \{\delta\} \mathrel{|\!\sim} \psi\}_{\delta \in \Delta}$, then (UMO) tells us that we can add any other observation φ to Δ, but the data in Δ are strong enough to still infer ψ.

This rule of constrained monotonicity can be generalised concerning the cardinality of the set of observations that we remove from Δ. In (UMO), we remove *one* observation at the time from Δ, and then we add only *one* observation. Let d be the cardinality of Δ, then for every set of indexes $I \subseteq [d]$, we can identify those of cardinality 2 and generalise (UMO) as follows.

$$\frac{\Delta \mathrel{|\!\sim} \psi \quad \{\Delta \setminus \{\bigcup_{i\in I} \delta_i\} \mathrel{|\!\sim} \psi\}_{|I|=2, I\subseteq[d]}}{\Delta, \bigcup_{\substack{j\in J \\ |J|\leq 2}} \varphi_j \mathrel{|\!\sim} \psi} \text{ (UMO)}^2$$

Note that the indexes in the conclusion of the rule only have the role of counting the maximum number of new evidence that we can add to the premise Δ and still inferring ψ. Generalising for every $n \leq d$, we can define the following rule.

$$\frac{\Delta \mathrel{|\!\sim} \psi \quad \{\Delta \setminus \{\bigcup_{i\in I} \delta_i\} \mathrel{|\!\sim} \psi\}_{|I|=n, I\subseteq[d]}}{\Delta, \bigcup_{\substack{j\in J \\ |J|\leq n}} \varphi_j \mathrel{|\!\sim} \psi} \text{ (UMO)}^n$$

The rule (UMO) is then recovered imposing $n = 1$, and whenever $n = 0$, the two premises and the conclusion are identical. For $n = d$, $(\text{UMO})^n$ holds only if ψ is a tautology. The evidence $\{\varphi_1, \ldots, \varphi_n\}$ added to Δ could be interpreted as an addition of (potentially disrupted) data that Δ is able to tolerate. The rule $(\text{UMO})^n$ can be used to test a generic ML algorithm in two ways:

1. Suppose that n is the maximum n for which $(\text{UMO})^n$ is valid; then the ML prediction's **robustness** should be proportional to it. As n grows, the more robust the ML prediction.

2. If $(\text{UMO})^n$ is valid but $(\text{UMO})^{n+1}$ is not, then we can identify what additional information must be added to a generic input data Δ to have $(\text{UMO})^{n+1}$ holding and determine how **sensitive** the ML prediction is to data variation.

In Table 4.3, we exemplify some results on the number of checks one needs to perform for some increasing levels of the $(\text{UMO})^i$ rule. In Figure 4.3, an illustration of the UMO is given.

$(\textbf{UMO})^i$	**Number of checks for a generic cardinality d of Δ**	$d = 10$	$d = 50$	$d = 100$
$i = 1$	d	10	50	100
$i = 2$	$\binom{d}{2} = \frac{d(d-1)}{2}$	45	1125	4950
$i = 3$	$\binom{d}{3} = \frac{d(d-1)(d-2)}{6}$	120	19600	161700

Table 4.3: Let us denote with d the cardinality of our data Δ. We compute the number of checks needed to verify the validity of $(\text{UMO})^i$ with $i \in \{1, 2, 3\}$ for a generic d and for some specific d.

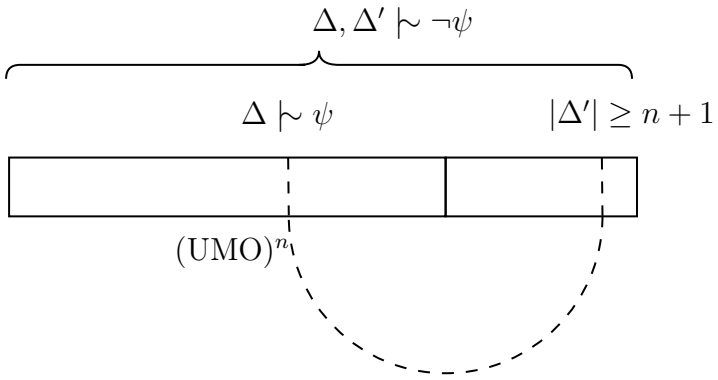

Figure 4.3: In the example, Δ infers ψ satisfying $(\text{UMO})^n$. If we add Δ the set of observations Δ', with $|\Delta'| \geq n+1$ and $\Delta' \mid\sim_{RJ} \neg\psi$, then the initial conclusion gets overtuned.

4.5.2 A Generalised (UMO) Rule for Weighted Evidence

In the setting analysed in [11], data are formalised through multisets to reflect the fact that seeing the same data once is not the same as seeing it multiple times. However, for computing the rejection degree, the relevance of each observation for the prediction is not taken into account. One simple way to generalise the rejection degree is to multiply every $r_\delta(H)$ by a weight w_δ whose value is in [0, 1], and that indicates how relevant δ is to the prediction.

Thus, the *weighted rejection degree* can be defined as follows.

Definition 14 *Let r be a* rejection degree function *and $w\colon \mathcal{D} \times \mathcal{H} \to [0,1]$ a weighted function that reflects the importance of each observation δ relative to a hypothesis H. Then, the* weighed rejection degree function $r^w\colon \mathcal{M}_\mathcal{D} \times \mathcal{H} \to \mathbb{R}_+ \cup \{\infty\}$ *is defined as*

$$r^w_\Delta = \sum_{\delta \in \Delta} r_\delta(H) w_\delta(H)$$

If we consider a weighted rejection degree function instead of a rejection degree function, then Definition 13 can be easily extended. The (UMO) rule formalizes a scenario in which the rejection level of the least rejected hypotheses is sufficiently low compared to the rejection level of all other hypotheses. This ensures that adding a single extra piece of information does not alter the order of rejection of the hypotheses and, ultimately, the set of least rejected hypotheses and what follows from them.

In the context of the Ulam game, since one extra piece of information can change the level of rejection of at most one, the second premise of the rule is equivalent to requiring that the difference between the rejection degree of the

least rejected hypotheses and the rejection degree of the rest of them is at least 2. If this is the case, then we can add any piece of information to a generic Δ, and the level of rejection of any non-least rejected hypotheses will necessarily be greater than the rejection degree of those minimally rejected by Δ. Thus, we have found a condition that guarantees that $\hat{\mathcal{H}}_{\Delta,\varphi} \subseteq \hat{\mathcal{H}}_{\Delta}$. However, if we consider a weighted Ulam-based rejection degree function, it might be that $\hat{\mathcal{H}}_{\Delta,\varphi} \not\subseteq \hat{\mathcal{H}}_{\Delta}$. Thus, the second premise of (UMO) needs to be refined. For any single evidence δ and $H \in \mathcal{H}$, $r^u_\delta(H) \in \{0,1\}$. Thus, if we consider the weighted extension of r^u, denoted by r^{uw}, we have that $r^{uw}_\delta(H) = r^u_\delta(H) w_\delta(H) \in [0,1]$. In general, for any value of w_δ, we have that $r^u_\Delta(H) \geq r^{uw}_\Delta(H)$ and while $r^u_\Delta(H) \in \mathbb{N} \cup \{\infty\}$, $r^{uw}_\Delta(H) \in \mathbb{R}_+ \cup \{\infty\}$.

Suppose that for every $H \in \hat{\mathcal{H}}_\Delta$ $r^{uw}_\Delta(H) > 0$. Then it might be the case that there exists $H' \in \mathcal{H} \setminus \hat{\mathcal{H}}_\Delta$ s.t. $r^{uw}_\varphi(H') = 0$ and

$$r^{uw}_{\Delta,\varphi}(H') \leq r^{uw}_{\Delta,\varphi}(H) \quad \text{i.e.}$$

$$r^{uw}_{\Delta,\varphi}(H') = r^{uw}_\Delta(H') + r^{uw}_\varphi(H') = r^{uw}_\Delta(H') \leq r^{uw}_\Delta(H) + r^{uw}_\varphi(H).$$

If $r^{uw}_\Delta(H') \leq r^{uw}_\Delta(H) + r^{uw}_\varphi(H)$, then H' might belong to $\hat{\mathcal{H}}_{\Delta,\varphi}$ and $\hat{\mathcal{H}}_{\Delta,\varphi} \not\subseteq \hat{\mathcal{H}}_\Delta$. Thus, to make sure that $\hat{\mathcal{H}}_{\Delta,\varphi} \subseteq \hat{\mathcal{H}}_\Delta$, we need to require that the difference of the weighted rejection degrees between any hypotheses not minimally rejected by Δ and the weighted rejection degree between any hypotheses minimally rejected by Δ is strictly greater than the weighted rejection degree of the new information φ on any of the minimally rejected hypotheses, i.e. if for every $H \in \hat{\mathcal{H}}_\Delta$ and $H' \in \mathcal{H} \setminus \hat{\mathcal{H}}_\Delta$ $r^{uw}_\varphi(H) < r^{uw}_\Delta(H') - r^{uw}_\Delta(H)$, then $\hat{\mathcal{H}}_{\Delta,\varphi} \subseteq \hat{\mathcal{H}}_\Delta$. The rule (UMO) can then be generalised to (gUMO).

$$\frac{\Delta \mathrel{|\!\sim} \psi \quad \{r_\varphi(H) < r_\Delta(H') - r_\Delta(H)\}_{H \in \hat{\mathcal{H}}_\Delta, H' \in \mathcal{H} \setminus \hat{\mathcal{H}}_\Delta}}{\Delta, \varphi \mathrel{|\!\sim} \psi} \text{ (gUMO)}$$

Proposition 1 (*gUMO*) *is valid for the RJ consequence relation obtained by considering the weighted Ulam-based rejection degree function.*

Proof 1 *Suppose that* $\Delta \mathrel{|\!\sim}_{RJ} \psi$. *Thus, for any* $H \in \hat{H}_\Delta$ *and* $H' \in \mathcal{H} \setminus \hat{H}_\Delta$,

$$r^{uw}_\Delta(H) < r^{uw}_\Delta(H').$$

If for $\varphi \in \mathcal{D}$,

$$r^{uw}_\varphi(H) < r^{uw}_\Delta(H') - r^{uw}_\Delta(H),$$

	Questions	**Weight**
0.	70+?	0.85
1.	Breathlessness?	0.67
2.	Fever?	0.62
3.	Chest pain?	0.73
4.	Cough?	0.81
5.	COVID-19 test?	0.78
6.	Severe pneumonitis?	0.59
7.	Low blood oxygen?	0.94
8.	Follow-up Test 1?	0.87
9.	Follow-up Test 2?	0.79
10.	Follow-up Test 3?	0.30

Table 4.4: Weights of Questions 0-10 of Table 4.1; each weight is a value in interval $[0, 1]$ that reflects the importance of the correspondent question for hypothesis IC.

then

$$r^{uw}_{\Delta,\varphi}(H) = r^{uw}_{\Delta}(H) + r^{uw}_{\varphi}(H) < r^{uw}_{\Delta}(H') < r^{uw}_{\Delta,\varphi}(H'),$$

which implies that the rejection order is preserved after adding φ. Therefore, the least rejected hypotheses remain the same, ensuring that $\Delta, \varphi \mathrel{|\!\sim_{RJ}} \psi$.

Note that in (gUMO), in contrast with (UMO), the second premise imposes a constraint on which φ can be added to Δ. Based on the distance between the least and the second least rejected hypothesis by data Δ, we can hence define a measure $(\text{gUMO})^j$ of the level of monotonicity that a given set of data can tolerate:

Definition 15 ($(\text{gUMO})^j$) *If $j = \min\{r_\Delta(H') - r_\Delta(H) \mid H \in \hat{\mathcal{H}}_\Delta, H' \in \mathcal{H} \setminus \hat{\mathcal{H}}_\Delta\}$, then we say that Δ satisfies $(gUMO)^j$.*

Conceptually, $(\text{gUMO})^j$ quantifies the robustness of inference $\Delta \mathrel{|\!\sim_{RJ}} \psi$ to the addition of new (and possibly corrupted) data. Translated in Ulam game terms, the rule tells us the level of tolerance of the data at hand to possible new lies from Responder. Returning to the pulse oximeter example once again, let us assume that the clinician knows, from previous knowledge, how relevant the questions in Table 4.1 are for the ML prediction. Therefore, assuming that these

	Steven					Yvonne				
	$r(\neg IC)$	$r(IC)$	gUMO	$\hat{\mathcal{H}}$	d	$r(\neg IC)$	$r(IC)$	gUMO	$\hat{\mathcal{H}}$	d
Δ_0	0	0.85	0.85	$\neg IC$	-	0	0.85	0.85	$\neg IC$	-
Δ_1	0.67	0.85	0.18	$\neg IC$	-	0.67	0.85	0.18	$\neg IC$	-
Δ_2	1.29	0.85	0.44	IC	-	1.29	0.85	0.44	IC	-
Δ_3	2.02	0.85	1.17	IC	-	2.02	0.85	1.17	IC	-
Δ_4	2.83	0.85	1.98	IC	-	2.83	0.85	1.98	IC	-
Δ_5	3.61	0.85	2.76	IC	-	3.61	0.85	2.76	IC	-
Δ_6	4.2	0.85	3.35	IC	-	4.2	0.85	3.35	IC	-
Δ_7	5.14	0.85	4.29	IC	1	4.2	1.79	2.41	IC	0
Δ_8	-	-	-	-	1	5.07	1.79	3.28	IC	0
Δ_9	-	-	-	-	1	5.86	1.79	4.08	IC	0
Δ_{10}	-	-	-	-	0	6.41	1.79	4.62	IC	1

Table 4.5: Degrees of rejection, gUMO level, minimally rejected hypothesis, and final decision computed on the sequence of questions-and-answers Δ_i for Steven and Yvonne, where $\Delta_i = \{Q_0, \ldots, Q_i\}$. Finally, for the sake of the example, we assume that a prediction is considered acceptable when it is associated with a gUMOlevel greater than 4.20; this is why the collection of new observations about Yvonne stops with Δ_{10}.

are transcribed in Table 4.4, it is possible to reconstruct that Steven's data satisfy a higher level of constrained monotonicity with respect to Yvonne. Specifically, Table 4.5 shows that, arrived at Question 7, Steven's data satisfy $(\text{gUMO})^{4.29}$ and Yvonne satisfies $(\text{gUMO})^{2.41}$. Crucially, the difference $1.88 = 4.29 - 2.41$ quantifies how robust Steven's prediction is, in comparison to Yvonne's. Such a difference effectively quantifies the impact of the biased oximeter reading on Yvonne's prediction. To obtain a robustness level similar to Steven's, she needs to undergo three more tests for IC (questions 8-10), which may require time, resources, and effort from her part, possibly worsening her medical condition in the meantime. This hints at the more general consideration that in order to reach parity of gUMO levels between sensitive groups in the presence of data bias, more information must be collected for the affected sensitive group. This strategy has already been discussed in relation to some applications of active learning for fair predictions [51, 119, 10], with which our research shows several connections.

Furthermore, decision (4.4) was originally defined as a function of $r(IC)$. However, we now suggest a possible improvement by redefining it as a function

of gUMO, instead:

$$d(a) = \begin{cases} 1 & \text{gUMO}_a > \text{gUMO}_{-a} \\ 0 & \text{gUMO}_a < \text{gUMO}_{-a} \\ \texttt{undec} & \text{otherwise} \end{cases} \tag{4.5}$$

This better captures the intuition that the decision of admitting Steven over Yvonne to IC could be overturned as we gather more information about her condition, stopping only when her prediction reaches a sufficient level of robustness – to be established beforehand, based on the sample. We argue that, only then, the Principle of Consistency (2) is eventually satisfied, making the corresponding decisions fair.

4.6 Related Work

4.6.1 Ulam Game for Data-Driven Reasoning

The Ulam game, also known as the "Rényi-Ulam game" has been independently introduced by Alfréd Rényi in 1961 [133] and by Stanisław Ulam [157, p. 281].

To our knowledge, Cicalese [34] is the first to use the Ulam game to model a computational scenario, namely supervised *learning*. Questioner's goal is to determine the secret Boolean function f^* based on a sequence of given input-output pairs and knowing that up to m of these are incorrect. At every new question, Questioner is given a binary input x and has to guess the value of $f^*(x)$. Responder either confirms or refutes it, having the possibility of lying at most m times. Questioner's goal is to identify the target function f^* with the minimum number of wrong predictions. As the game goes on, Questioner proceeds by progressively pruning away those functions that are not compatible with the available noisy observations (input-output pairs) anymore, within a given noise tolerance threshold m.

Similarly to our work, in [88] the Ulam game is used to model a typical data quality problem faced by crowdsourcing platforms, where humans are employed to annotate images (e.g., labelling a dog's picture based on its breed). Since human labellers can fail their tasks for many reasons such as fatigue, ignorance, and difficulty of the task at hand, the challenge here is to implement an error-tolerant

annotation process. To do so, the authors break down the original task into many smaller subtasks (i.e., yes-no questions about the dog's hair colour, length, etc.). Given that each subtask is subject to an independent error, they exploit the Ulam game to find an optimal strategy to distribute these subtasks within a group of human workers in order to obtain reliable labels. A computationally efficient heuristic sampling strategy is proposed and evaluated.

4.6.2 Logical Formalisations of Bias

From a logical point of view, the task of finding adequate fairness formalisations can be seen as subsumed under the broader task of applying verification methods to machine learning systems [136]. In this spirit, scholars have recently attempted to formalise fairness constraints usable as specifications for formal verification.

Kawamoto [90] formalises the properties of supervised machine learning models through a statistical epistemic logic. Distributional Kripke models are used to define statistical measures on the prediction distribution of the model. Fairness is measured by means of a modal operator of 'conditional indistinguishability'. Based on this, existing measures of fairness are translated in terms of degrees of indistinguishability between the predictions returned to different sensitive groups.

Belle [15] formalises fairness in a dynamic epistemic setting suitable to logically capture bias in ML predictions, in the presence of actions, beliefs, and plans. [80] focus on assessing the presence of data bias in the training sample used for logic-based algorithms (such as decision trees and decision sets).

Finally, Liu and Lorini [100] introduce a modal language to formalise binary classifiers and their properties, and for which they provide axiomatics and complexity results. With this, not only they formalise a variety of relevant notions of explanation (abductive, contrastive, counterfactual), but also propose a logical formalisation of the principle of Fairness Through Unawareness [160].

4.6.3 Fairness and ML Correctability

Our proposal shows interesting connections with previous work on the correctability of ML predictions. Venkatasubramania and Alfano [159] distinguish between two types of ML correction scenarios: algorithmic recourse

and algorithmic appeal. The former pertains to the situation in which an individual who has received an *unfavourable* ML prediction undertakes strategic actions to obtain a favourable decision from the same ML decision process in the future. Here, recourse corresponds to the effort associated with such a sequence of actions.

The notion of recourse has gained great attention in the ML fairness debate, especially in relation to the development of techniques to ensure its parity between sensitive groups [89, 158, 92].

The notion of recourse has gained great attention in the ML fairness debate, especially in relation to the development of techniques to ensure its parity between sensitive groups [89, 158, 92]. However, one general objection that can be moved against algorithmic recourse is that puts the emphasis on the *wrong* aspect of an ML prediction, specifically its potential to be unfavourable, while the fundamental question of whether such a prediction is justified in the first place remains largely unexamined by the algorithmic recourse perspective. By putting much of the stress on the sequence of actions that the single individual can put in place to reverse an adverse prediction, the responsibility of the outcome itself is somehow deflected from the AI system to the specific end-user who has to undertake a number of actions to reverse it [149]. But what about an outcome, like the one received by Yvonne in Section 4.2, that is ultimately unjustified? In light of the preliminary discussion on data bias given in Section 4.1, we know that systematic errors affecting the data-generating mechanism can produce a differentially noisy representation of sensitive groups. Therefore, we argue that the conception of ML correctability returned by the notion of algorithmic recourse is ultimately inadequate.

Algorithmic appeal characterises situations in which an ML decision is not simply *unfavourable*, but also *unjustified* because of the presence of inaccuracies in the input data used to produce it [159]. In this case, one clearly wants to challenge the ML decision because it was based on *incorrect* information. Unfortunately, the concept has been largely overlooked so far in the literature. [159] themselves seem to have a somewhat simplistic idea of it ("[...] someone may be recorded as having defaulted on a loan when in fact they repaid it on time and in full, or they be recorded as having no credit history when in fact they have taken out and repaid several loans" [159, p. 288]). However, in the above, we extensively discussed the problem of data bias in all its urgency.

However, as conceptually compelling as it might be, the practical applicability of algorithmic appeal seems limited as long as we do not have direct access to

the true values to rectify the incorrect pieces of information. Cerrato *et al.* [32] have introduced the more practically useful concept of "contextual recourse", to denote the situation in which the individual shows that their prediction should be different in virtue of *additional* pieces of information that were not used for the initial prediction. As the authors suggest, "some level of logical reasoning is [...] required in contextual recourse" [32, p. 3]. We hold that the logical machinery presented above exactly addresses this point. Through the Ulam game formalisation, in fact, we have quantified this very correctability as a measure of how robust an inference is to possible errors contained in its premises. As we have shown, this inference robustness is effectively captured by the gUMO level calculated using the gUMO rule. To be deemed fair, an ML model predicting in the presence of data bias should attain similar gUMO levels between sensitive groups, as this grants the prediction process to be consistent, as required by Principle 2.

4.7 A Proof-Theoretic Interpretation of Biased Predictions

In what follows, we introduce the basic elements to define a proof-theoretic approach to a logic for reasoning from biased premises. Our language models a supervised ML classification with a single, binary protected attribute and (possibly many) binary non-protected attributes.

4.7.1 Preliminaries

In the proof theory, we interpret the non-monotonicity as a probabilistic derivability relation implementing the failure of the structural rule of Weakening. We start with introducing the following sets of predicates (also called attributes or features), upon which we will then define the Training and the Test Samples:

Definition 16 (ML Predicates) *Two types of predicates are defined:*

- $\mathbb{P} = \{P_1, \ldots, P_n\}$ *as the set of predicates corresponding to the ML features.*
- $\mathbb{T} = \{T\}$ *as the singleton containing the binary target feature.*

Definition 17 (Training Sample) *A training sample* $\mathcal{S}^{train} = \{\mathbb{D}^{train}, \mathbb{P}^{train}\}$ *is such that:*

- $\mathbb{D}^{train} = \{b_1, b_2, \ldots, b_m\}$ *is a finite set of elements of a domain denoting the data points of the training sample;*
- $\mathbb{P}^{train} = \mathbb{P} \cup \mathbb{T}$ *as defined in Definition 16*

Definition 18 (Test Sample) *A test sample* $\mathcal{S}^{test} = \{\mathbb{D}^{test}, \mathbb{P}^{test}, ww\}$ *is such that:*

- $\mathbb{D}^{test} = \{a_1, a_2, \ldots, a_m\}$ *is a finite set of elements of a domain denoting the datapoints of the test sample;*
- $\mathbb{P}^{test} = \mathbb{P}$
- $ww : \mathbb{P}^{test} \mapsto [0, 1]$ *is a total weighting function that assigns to each predicate a measure of how it is relevant for the prediction. This relevance is acquired by extra knowledge or is obtained through feature importance techniques.*

Moreover, to ensure that the weights are comparable and sum to one, we additionally define a normalised and signed weight function as follows.

Definition 19 (Normalised Weights) *Given a predicate* $P \in \mathbb{P}^{\text{test}}$*, we define function* $w_{norm} : \mathbb{P}^{test} \mapsto [0, 1]$ *as:*

$$w_{\text{norm}}(P) = \frac{ww(P)}{\sum_{Q \in \mathbb{P}^{\text{test}}} ww(Q)}$$

Definition 20 (Signed Weights) *We define function* $\tilde{w} : \mathbb{P}^{test} \times \mathbb{D}^{test} \mapsto [-1, 1]$ *as follows:*

$$\tilde{w}(P_i, a) := \begin{cases} + w_{\text{norm}}(P_i) & \text{if } P_i(a), \\ - w_{\text{norm}}(P_i) & \text{otherwise.} \end{cases}$$

This can be clarified as follows: for any datapoint, we check the current list of predicates (i.e., ML features); knowing that a predicate holds for a datapoint will weight in the classification at hand for the value of the predicate; knowing that the predicate does not hold, we take the inverse, so that satisfied predicates contribute positively and unsatisfied ones negatively. $P(a)$ reads as "It is the case that individual a has property P" or "Individual a is P". Conversely, $\neg P(a)$ is read as "It is not the case that individual a has property P" or "Individual a is not P". Note that an important assumption here is that the predicates satisfied by a 'speak in favour' of prediction $T(a)$, conversely, predicates that

are not satisfied by a 'speak against' prediction $T(a)$.

Finally, for convenience, we rewrite $\tilde{w}(P_i, a)$ with the following shortcut when we are referring to a generic a:

$$w_i := \tilde{w}(P_i, a) \tag{4.6}$$

The language is then defined by predicative expressions closed under negation, conjunction, disjunction and implication.

Definition 21 (Language)

$$\begin{aligned}
&\phi := P(a)^w \mid \neg\phi \\
&\Phi := \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \\
&\psi := T(a) \ \textit{such that} \ T \in \mathbb{T} \ \textit{and} \ a \in \mathbb{D}^{test} \\
&\xi := r \in \mathbb{R}
\end{aligned}$$

Some clarifications are important on the definition of $\phi := P(a)^w$ where $w = \tilde{w}(P, a)$ as per (4.6) and $P \in \mathbb{P}^{test}$. Note that we do not assume a generalised version of the Excluded Middle, as it is sensible to consider only properties that can actually be derived (or their negation can) as a target feature from the current list of available predicates for the datapoint under evaluation. Moreover, the weighted inference relation we introduce requires a weaker than classical definition of negation.

Metavariable ψ denotes the result of the classification $T(a)$ for a predicate T that ranges over the class $\mathbb{T}$ of target predicates. Finally, with ξ, we refer to a real value associated to the degree to which the target predicate can be inferred from the current feature set.

4.7.2 Inference Rules

Two simple rules are used to inductively construct a feature list, thereby defining a method that selects one feature at a time, coupled with its weight (See Figure 4.4).

The induction starts from a list containing a single predicate (*base case*). At any step, an atomic predicative formula of weight w is added (*selection*), where the order in which predicates are added may depend from the availability of incoming information, or could be established by design. The overall weight of a

$$\frac{}{\{Q(a) \mid Q \in \mathbb{P}^{test}\} :: features}\ base\ case$$

$$\frac{\Gamma^{w} :: features \qquad P(a)^{w'} \notin \Gamma \mid P \in \mathbb{P}^{test} \qquad |w + w'| \leq 1}{\{\Gamma, P(a)\}^{W=w+w'} :: features}\ selection$$

Figure 4.4: Construction Rules for the Inductive Selection of Features

feature list is denoted by $W = \sum(w_1, \ldots, w_n)$, as it is the algebraic sum of the weights of the predicates in it. Each selection allows to update the feature list, where a new weight w is added to the previous sum of weights W, always checking the stopping condition that the summation on the overall weight $|W| \leq 1$, which is the total amount of available information about a datapoint. This additive process returns the overall weight of the selected features.

Let us now clarify how, at each step of the induction, the current list of selected features corresponds to the set of premises used by the inferential system to output a probabilistic prediction.

Definition 22 (Classification) *A judgement:*

$$\Gamma^{test,W} \mid\!\sim^{trained} \psi_{\xi}$$

denotes a classification where:

- $\Gamma^{test,W} := \{P_1(a), \ldots, P_n(a)\}$ *is a list of atoms* $P_i \in \mathbb{P}^{test}$ *for* $i = 1, \ldots, n$ *over an* $a \in \mathbb{D}^{test}$ *as per Definition 18 built according to the rules of Figure 4.4, and with overall weight* W*.*
- $\mid\!\sim^{trained}$ *is a non-monotonic inference relation which refers to a (known or unknown) training sample as per Definition 17, and whose semantics is defined by inference rules presented below in Figure 4.5;*
- ψ *is a predicative atomic formula* $T(a)$*, where* $T \in \mathbb{T}$ *is the target feature, and* $a \in \mathbb{D}^{test}$*. Therefore, it is the actual prediction returned by the model for input* a*.*
- $|\xi| \in [0, 1]$ *denotes probability that* $\mid\!\sim^{trained}$ *assigns to* ψ *being validly inferred from* $\Gamma^{test,W}$*, as determined by the inference rules. Prediction* ψ_{ξ} *is returned as a binary classification if we fix a threshold value* τ*, such that the final output is* $T(a)$ *if* $\frac{\xi+1}{2} \geq \tau$ *and is* $\neg T(a)$ *otherwise. This, in fact, allows for mapping the values in the [-1,1] interval onto probabilities.*

$$\frac{}{\Gamma^W, \phi \ \mid\sim \psi_\xi}\, classification$$

$$\frac{\Gamma^W \ \mid\sim \psi_\xi}{\Gamma^W \ \mid\sim \neg\psi_{(-\xi)}}\, failure \qquad \frac{\Gamma^W \ \mid\sim \neg\psi_\xi}{\Gamma^W \ \mid\sim \psi \to \bot_\xi}\, safety$$

$$\frac{\Delta^W \ \mid\sim \psi_{1\xi} \qquad \Gamma^{W'} \ \mid\sim \psi_{2\chi}}{\{\Delta, \Gamma\}^{W+W'} \ \mid\sim (\psi_1 \wedge \psi_2)_{\xi\cdot\chi}}\, \mathrm{I}\wedge$$

$$\frac{\Gamma^W \ \mid\sim (\psi_1 \wedge \psi_2)_\xi \qquad \Gamma^W \ \mid\sim \psi_{1\chi}}{\Gamma^W \ \mid\sim \psi_{2(\xi/\chi)}}\, \mathrm{E}\wedge_R \qquad \frac{\Gamma^W \ \mid\sim (\psi_1 \wedge \psi_2)_\xi \qquad \Gamma^W \ \mid\sim \psi_{2\chi}}{\Gamma^W \ \mid\sim \psi_{1(\xi/\chi)}}\, \mathrm{E}\wedge_L$$

$$\frac{\Gamma^W \ \mid\sim \psi_{1\xi} \qquad \Gamma^W \ \mid\sim \psi_{2\chi}}{\Gamma^W \ \mid\sim (\psi_1 \vee \psi_2)_{(\xi+\chi)}}\, \mathrm{I}\vee$$

$$\frac{\Gamma^W \ \mid\sim (\psi_1 \vee \psi_2)_\xi \qquad \Gamma^W \ \mid\sim \psi_{1\chi}}{\Gamma^W \ \mid\sim \psi_{2(\xi-\chi)}}\, \mathrm{E}\vee_R \qquad \frac{\Gamma^W \ \mid\sim (\psi_1 \vee \psi_2)_\xi \qquad \Gamma^W \ \mid\sim \psi_{2\chi}}{\Gamma^W \ \mid\sim \psi_{1(\xi-\chi)}}\, \mathrm{E}\vee_L$$

$$\frac{\Gamma^W, \phi \ \mid\sim \psi_\xi}{\Gamma^W \ \mid\sim (\phi \to \psi)_\xi}\, \mathrm{I}\to \qquad \frac{\Delta^W \ \mid\sim \phi \to \psi_\xi \qquad \Gamma^{W'} \ \mid\sim \phi_\chi}{\{\Delta, \Gamma\}^{W+W'} \ \mid\sim \psi_{(\chi\cdot\xi)}}\, \mathrm{E}\to$$

Figure 4.5: Rules for Classification

Note, importantly, that in what follows we will assume function ξ to be identical to function W:

$$\xi = id(W) \tag{4.7}$$

The rules in Figure 4.5 define the derivability relation for our classifier. The $(classification)$ rule says that the target feature is derivable with degree ξ from a feature list of weight W. As already said, if we use the identity map between W and ξ, the two values will coincide, discounting any accuracy measure.

The $(failure)$ rule establishes that, if a list of features derives a property for a datapoint with degree ξ, then the negation of that property for that individual will be derivable from the same list of features with degree $-\xi$.

Whenever Γ, Δ are independent, the rules for $\wedge$ and $\vee$ introduce and eliminate the two connectives by making standard arithmetic operations on the associated probabilities. The $I \to$ rule implements the Deduction Theorem: inferring the property ψ with score ξ, from a list of features containing ϕ with weight w, should make it possible to infer ψ with score ξ from Γ alone, once ϕ is assumed

with the associated weight. Implication elimination, as usual, implements Modus Ponens: from the validity of an implication with score ξ and the assumption of the antecedent with score χ, one can infer the consequent with score $\xi \cdot \chi$. Note that, interestingly, only in this rule ϕ appears on the right-hand side of the inference, as inferred from the set of premises $\Delta^{W'}$. In other words, ϕ is predicted on the basis of a (possibly different) set of features $\Delta^{W'}$, and is carried over in the prediction of ψ.

To evaluate bias within the ML classification model means to quantify the amount of information that would be needed for the system to correct a previously erroneous classification. To do so, we construct an external classifier in which the ground truth is inferred:

Definition 23 (Ground Classification) *A judgement:*

$$\Gamma^{test,1} \vdash^{ground} \psi_1$$

denotes a classification where:

- $\Gamma^{test,1}$ *denotes that premises* Γ^{test}*, as per Definition 22, embed the entire available information about an input, which amounts to 1 as said in Definition 18;*
- $\vdash^{ground}$ *is a monotonic inference relation that infers the ground truth;*
- ψ_1*, like in Definition 22, is the prediction, this time inferred with full certainty.*

Of course the one just defined is quite a demanding notion of ground truth, as it requires both the maximum quantity of information possible, and full certainty of the conclusion. It is nonetheless possible to liberalise this definition along both these dimensions. In fact, what we are interested to model through this judgment is not necessarily a perfect classifier, but rather an inferential system that, unlike system $\mid\!\sim^{test}$, confidently outputs the correct prediction with the minimum quantity of information possible. Hence:

Definition 24 (Weak Ground Classification)

$$\Gamma^{test,W} \vdash^{ground} \psi_\xi$$

The condition of initial incorrectness of the classifier's prediction under which we are reasoning, therefore, can be expressed by means of the following pair of judgements:

$$\frac{\Gamma^{test,W} \mid\!\sim^{trained} \psi_{\xi} \qquad \Delta^{test,W'} \vdash^{ground} \neg\psi_{\xi'}}{\Gamma^{W}, \Delta^{W'} \mid\!\sim^{trained} \neg\psi_{(\xi'-\xi)}}\ correction$$

Figure 4.6: Structural Rule for Correction

Definition 25 (Initial Condition)

$$\Gamma^{test,W} \mid\!\sim^{trained} \psi_{\xi} \qquad \Delta^{test,W'} \vdash^{ground} \neg\psi_{\xi'}$$

The intended correction can be expressed as failure of the structural rule of Weakening for the system. In other words, when the above condition of initial discrepancy between a given classification of input a and its ground truth obtains, correction is possible if there exists an amount of information W about a with the following property. Given a list of features Γ^W that let us infer ψ with probability ξ, there is an extension of Γ such that $\neg\psi$ becomes valid with an updated degree $\xi' - \xi$, see Figure 4.6.

Such an extension might correspond to the evaluation of the entire set of features, thus reducing to Δ^{test}. Note that we have defined ξ as $id(W)$, i.e., the score on the judgement derived corresponds to the weight W of the information at hand. Therefore, the absolute difference in scores assigned to the judgement in the conclusion corresponds to the additional information in terms of weight required to reach the correct conclusion starting from the wrong classification. In the following we will keep this assumption. Hence, when the *correction* rule is applied with a sufficient degree for ξ' (i.e. because of a feature in Δ with a sufficiently high weight), or a sufficient number of times, the degree of confidence $\xi' - \xi$ will increase above the threshold τ, and eventually the conclusion ψ will no longer be inferrable. This means that some premises on a given derivation tree of the form $\Gamma^W \vdash \psi_{\xi}$ were biased, in the sense that the weights of currently available data was either insufficient for a correct classification, or skewed in the sense that the available features weighted strongly in favour of a given prediction. In these cases, one has to add *selection* rules until all sufficient information is available for a correct classification, and thus eventually (*correction*) rule is satisfied.

On this basis, we now want to offer a generalised version of the metric underlying the (*correction*) rule, for evaluating (i) how much additional information is needed to correct a prediction, and in particular (ii) how much such measure is imbalanced across different groups of the test sample.

Therefore, given the judgement:

$$\{Q_1(a), \ldots, Q_m(a)\}^M \mid\!\sim^{trained} \psi_\xi$$

and fixed an aggregation function f over the weights of the predicates $Q_1, \ldots, Q_m$, M denotes the amount of information currently available and used for a wrong prediction ψ with a score ξ. Similarly, in judgement:

$$\{Q_p(a), \ldots, Q_n(a)\}^N \vdash^{ground} \neg\psi_{\xi'}$$

N expresses the amount of information sufficient to compute by the same function f over the weights of the predicates $Q_p, \ldots, Q_n$ the correct classification $\neg\psi$ with score 1 respecting the ground truth (or with $\xi' \geq \xi$). It is reasonable to assume that the score ξ' assigned by a classifier trained on the ground truth may converge to 1. Note that it is possible that $\{Q_1(a), \ldots, Q_m(a)\} \cap \{Q_p(a), \ldots, Q_n(a)\} \neq \emptyset$. In the following, we abbreviate with $S := (\Gamma^W, \mid\!\sim^{trained})$ the pair of weighted features set and trained model. Similarly, we abbreviate with $G := (\Delta^{W'}, \vdash^{ground})$ the pair of weighted features set Δ and classifier function which returns a classification that matches the ground truth.

Maintaining the assumption that the scores $\xi = id(W) = M$ and $\xi' = id(W') = N$, the difference $(N - M)$ expresses the amount of additional information that would be sufficient to correct the initial incorrect prediction, by applying the (*correction*) rule. This value can be further normalised by the accuracy of the system. Put differently, $(N - M)$ expresses the amount of additional information that would be sufficient for the Weakening rule to fail. This difference, therefore, is a measure of non-monotonicity.

Definition 26 (Correction Distance) *Given $S := (\Gamma^W, \mid\!\sim^{trained})$ that infers a formula $\psi := T(a)$ such that $T \in \mathbb{T}, a \in \mathbb{D}^{test}$, the correction distance with respect to $G := (\Delta^{W'}, \vdash^{ground})$ that infers formula $\neg\psi$ is defined as:*[4]

[4]In the original paper [106], the correction distance was defined as the inverse of the difference $N - M$, weighted on the accuracy of the ML system (calculated by previous analyses), i.e., $C(S, \psi) = 1 - (N - M) * accuracy$. In the present chapter, we have simplified the original definition to make a clearer correspondence between the Correction Distance and gUMO. Other relevant departures with respect to the original formalism are:

- the redefinition of the weight function (Definition 18) and the addition of the related Definitions 19 and 20,
- the redefinition of the (*base case*) and ($failure$) rules,
- the removal of the distinction between protected and non-protected predicates in Definitions 16 and 18.

$$C(S, \psi) = N - M$$

Given our definitions of N and M – as the amounts of information sufficient to give, respectively, a correct and an incorrect prediction – the measure C formally expresses an evaluation of how far the system is from correcting a wrong classification.

In the light of the above, intuitively, a system is said to be maximally distant from the correction of a wrong prediction $T(a)$ obtained based on amount of information M, if there is no additional amount of information M' such that the sum of $M + M'$ results in an amount of information sufficient for predicting correctly $\neg T(a)$. To formalise this, we use the correction distance to evaluate the amount of information required for the rule (*correction*) from Figure 4.6 to hold.

Definition 27 (Maximally-Distant-From-Correction System) *Given* $\Gamma^W \mid\!\sim^{trained} \psi_\xi$ *and* $f(W) = M$*, then* $C(S, \psi) = 1$ *if and only if* $\nexists M'$ *such that* $f(W') = M + M' = N$ *and* $\Delta^{test,W'} \vdash^{ground} \neg\psi_{\xi'}$.

A maximally distant from correction system is therefore one such that no amount of correct information whatsoever can make it correct. Intuitively, when a system will behave as to be maximally distant from correctness with respect to any given property or (group of) individual(s), it will be deemed just to be an inaccurate classifier. On the other hand, a classifier which cannot be corrected provided novel data when acting on a given property or (group of) individual(s), *but not with respect to another*, can intuitively be said to be strongly biased against the current data.

All things being equal, a system is correctable if there exists some additional information M' such that the sum of $M + M'$ results in an amount of information sufficient for predicting correctly:

Definition 28 (Correctable System) *Given* $\Gamma^W \mid\!\sim^{trained} \psi_\xi$ *and* $f(W) = M$*, then* $C(S, \psi) < 1$ *if and only if* $\exists M'$ *such that* $f(W') = M + M' = N$ *and* $\Delta^{test,W'} \vdash^{ground} \neg\psi_{\xi'}$.

4.7.3 A Completeness Result for $\mid\!\sim_{RJ}$

Let us recall the most relevant aspects discussed so far in the present section. We have used the following judgment to denote the non-monotonic

inference relation $\mid\!\sim$ that corresponds to a prediction of a trained ML model that takes input data $\Delta = \{\delta_1, \ldots, \delta_m\}$ associated with the sum of weights $W = \sum\{w_1, \ldots, w_n\}$ to predict binary target label ψ with probability ξ:

$$\Delta^W \mid\!\sim \psi_\xi \tag{4.8}$$

The rules in Figure 4.5 define derivability relations for our classifier. Moreover, a structural rule called (*correction*) is defined to update a prediction's probability as new information (that possibly allows for deriving contradictory conclusions) is added to the original set of observations. Namely, if we combine observations Δ with weights W inferring conclusion ψ with another set of observations Γ with weights W' and inferring conclusion $\neg\psi$, the following holds:

$$\frac{\Delta^W \mid\!\sim \psi_\xi \qquad \Gamma^{W'} \mid\!\sim \neg\psi_{\xi'}}{\{\Delta, \Gamma\}^{W+W'} \mid\!\sim \neg\psi_{(\xi'-\xi)}}\, correction \tag{4.9}$$

In the calculus, the maximum probability $\xi = 1$ corresponds to the situation in which the inference is based on an amount of information sufficient to infer a conclusion that is *correct*, that is, to infer the ground truth with certainty. In terms of the semantics of the Ulam game, this inference corresponds to the situation in which Responder has used all m available lies. In fact, from this moment onwards, conclusions can be classically entailed on the basis of any new (certainly truthful) answer. In other words, in the game-theoretic semantics, function ξ is the difference between the sum of weighted rejection degrees of those observations that 'speak' against label ψ and the sum of weighted rejection degrees of those observations that 'speak' in favour of it. We capture this intuition more rigorously with the three following definitions.

Definition 29 (gUMO-Norm) *Whenever $\Delta \mid\!\sim_{RJ} \psi$ satisfies $(gUMO)^j$, we say that $\Delta \mid\!\sim_{RJ} \psi$ satisfies $(gUMO\text{-}Norm)^{\tilde{j}}$, where:*

$$\tilde{j} := \frac{j}{\sum_{H \in \mathcal{H}^f_\Delta} r^{uw}_\Delta(H)}$$

and $\mathcal{H}^f_\Delta$ is the set of hypotheses with a finite degree of rejection.

The level of gUMO-Norm tends to 1 as the difference in rejection degrees between the least rejected and the second least rejected hypothesis increases and the number of finitely rejected hypotheses decreases. It is exactly 1 when

$\sum_{H \in \mathcal{H}^f_\Delta} r^{uw}_\Delta(H) = j$, i.e., two finitely rejected hypotheses are given, one of which is never rejected. Conversely, the level of gUMO-Norm tends to 0 whenever the difference in rejection degrees between the minimally rejected and the second least rejected hypothesis gets smaller and smaller.

Definition 30 *Whenever* $\Delta \mid\sim_{RJ} \psi$ *satisfies* $(gUMO\text{-}Norm)^{\tilde{j}}$, *we write* $\Delta \mid\sim_{RJ} \psi_{\tilde{j}}$.

Definition 31 *Let* a *be a generic datapoint in* $\mathbb{D}^{test}$. *We define* γ_Δ *as the sum of the signed weights of the predicates satisfied by* a, *and* $\overline{\gamma}_\Delta$ *as the sum of the magnitudes of the negative contributions of predicates not satisfied by* a. *Formally:*

$$\gamma_\Delta := \sum_{P_i \in \Delta} \tilde{w}_i \tag{4.10}$$

$$\overline{\gamma}_\Delta := \sum_{\neg P_i \in \Delta} (|\tilde{w}_i|) \tag{4.11}$$

Note that, by construction, both γ_Δ and $\overline{\gamma}_\Delta$ are nonnegative as they respectively represent the magnitudes of the predicates speaking 'in favour' and 'against' conclusion ψ.

Proposition 2 *Whenever* $\Delta^W \mid\sim \psi_\xi, \xi = \gamma_\Delta - \overline{\gamma}_\Delta$.

Proof 2 *By* (4.7)*:*

$$\xi = id(W)$$

where by the (selection) rule of Figure 4.4:

$$W := \sum_{i \in \{1,\ldots,n\}} \tilde{w}_i$$

Let us define the set $\mathcal{I} = \{1, \ldots, n\}$ *of indexes of predicates in* $\mathbb{P}^{test} = \{P_1, \ldots, P_n\}$. *Therefore, we can rewrite* ξ *as:*

$$\xi = \sum_{i \in \mathcal{I}} \tilde{w}_i$$

We can partition $\mathcal{I}$ *into two sets: one containing the indexes of predicates satisfied by* a, $\mathcal{S} = \{i | P_i(a) \in \Delta\}$ *and the other set containing the indexes of predicates not satisfied by* a, $\mathcal{U} = \{i | \neg P_i(a) \in \Delta\}$. *Then the total signed weight can be written as*

$$\xi = \sum_{i \in \mathcal{S}} \tilde{w}_i + \sum_{i \in \mathcal{U}} \tilde{w}_i$$

For $i \in \mathcal{U}$, we have $\tilde{w}_i < 0$, so $|\tilde{w}_i| = -\tilde{w}_i$. Therefore,

$$\sum_{i \in \mathcal{U}} \tilde{w}_i = \sum_{i \in \mathcal{U}} (-|\tilde{w}_i|) = -\sum_{i \in \mathcal{U}} (|\tilde{w}_i|)$$

Since $\mathcal{S} = \{i | P_i(a) \in \Delta\}$, by construction

$$\sum_{i \in \mathcal{S}} \tilde{w}_i = \sum_{P_i \in \Delta} \tilde{w}_i = \gamma_\Delta$$

Similarly, since $\mathcal{U} = \{i | \neg P_i(a) \in \Delta\}$, by construction

$$\sum_{i \in \mathcal{U}} |\tilde{w}_i| = -\sum_{\neg P_i \in \Delta} (|\tilde{w}_i|) = -\overline{\gamma}_\Delta$$

By Definition 31, they correspond to γ_Δ and $\overline{\gamma}_\Delta$, respectively.
Therefore, substituting this into the expression for

$$\xi = \sum_{i \in \mathcal{S}} \tilde{w}_i + \sum_{i \in \mathcal{U}} \tilde{w}_i$$

we obtain that

$$\xi = \gamma_\Delta + (-\overline{\gamma}_\Delta) = \gamma_\Delta - \overline{\gamma}_\Delta$$

which proves the desired equality.

Proposition 3 *If $\mathcal{H} = \{\psi, \neg\psi\}$ and $\Delta \mid\!\sim_{RJ} \psi$ satisfies $(gUMO)^j$, we say that $\Delta \mid\!\sim_{RJ} \psi$ satisfies $(gUMO\text{-}Norm)^{\tilde{j}}$ where:*

$$\tilde{j} = \frac{r^{uw}_\Delta(\neg\psi) - r^{uw}_\Delta(\psi)}{r^{uw}_\Delta(\neg\psi) + r^{uw}_\Delta(\psi)}$$

In the formula, r^{uw} is the Ulam weighted rejection degree as we have defined it earlier in Definition 14.

Proof 3 *Since by Definition 30, we know that:*

$$\tilde{j} := \frac{j}{\sum_{H \in \mathcal{H}^f_\Delta} r^{uw}_\Delta(H)}$$

we proceed by proving equivalence for the numerator and the denominator of $\tilde{j}$ separately.

(Numerator Equivalence). *We want to prove that:*

$$j = r_{\Delta}^{uw}(\neg\psi) - r_{\Delta}^{uw}(\psi)$$

If $\Delta \mid\!\sim_{RJ} \psi$, *then* $\hat{\mathcal{H}}_{\Delta} = \{\psi\}$ *by (4.1) and* $\mathcal{H} \setminus \hat{\mathcal{H}}_{\Delta} = \{\neg\psi\}$. *Since* $\Delta \mid\!\sim_{RJ} \psi$ *satisfies* $(gUMO)^j$, *by Definition 15, it follows that* $j = \min\{r_{\Delta}(H') - r_{\Delta}(H) \mid H \in \hat{\mathcal{H}}_{\Delta}, H' \in \mathcal{H} \setminus \hat{\mathcal{H}}_{\Delta}\} = r_{\Delta}^{uw}(\neg\psi) - r_{\Delta}^{uw}(\psi)$.

(Denominator Equivalence). *We want to prove that:*

$$\sum_{H \in \mathcal{H}_{\Delta}^{f}} = r_{\Delta}^{uw}(\neg\psi) + r_{\Delta}^{uw}(\psi)$$

Assuming that neither hypothesis has been definitely rejected, $\mathcal{H}_{\Delta}^{f} = \{\neg\psi, \psi\}$ *and therefore* $\sum_{H \in \mathcal{H}_{\Delta}^{f}} r_{\Delta}^{uw}(H) = r_{\Delta}^{uw}(\neg\psi) + r_{\Delta}^{uw}(\psi)$.

For convenience, we introduce the following notation:

$$\xi_{\Delta-Bin} := \frac{r_{\Delta}^{uw}(\neg\psi) - r_{\Delta}^{uw}(\psi)}{r_{\Delta}^{uw}(\neg\psi) + r_{\Delta}^{uw}(\psi)} \tag{4.12}$$

We introduce the following correspondence.

Proposition 4 *The difference between* γ_{Δ} *and* $\overline{\gamma}_{\Delta}$ *is equal to* $\xi_{\Delta-Bin}$*:*

$$\gamma_{\Delta} - \overline{\gamma}_{\Delta} = \xi_{\Delta-Bin}$$

Proof 4 (Proof of Proposition 4) *Given* γ_{Δ} *and* $\overline{\gamma}_{\Delta}$, *by Proposition 2, we know that the following inference holds in* $\mid\!\sim$*:*

$$\Delta^{W} \mid\!\sim \psi_{(\gamma_{\Delta} - \overline{\gamma}_{\Delta})}$$

Given the structural rule (correction) for the $\mid\!\sim$ *derivability relation (4.9),*

$$\frac{\Theta^{W''} \mid\!\sim \neg\psi_{\overline{\gamma}_{\Delta}} \qquad \Gamma^{W'} \mid\!\sim \psi_{\gamma_{\Delta}}}{\Theta^{W''}, \Gamma^{W'} \mid\!\sim \psi_{(\gamma_{\Delta} - \overline{\gamma}_{\Delta})}}\, COR$$

it is always possible to construct two sets of observations $\Theta^{W''}$ *and* $\Gamma^{W'}$ *such that* $\Theta^{W''}$ *only contains predicates not satisfied by datapoint* a *and* $\Gamma^{W'}$ *only contains predicates satisfied by the same data instance. Note that in this case* $\Theta \cap \Gamma = \emptyset$

because if a predicate belongs to the set of predicates satisfied by a, it cannot also belong to the set of predicates not satisfied by a. Let us denote their union by Δ^W*, where* $\Delta = \Gamma \cup \Theta$ *and* $W = W' + W''$.
The overall weights W' *and* W'' *are the algebraic sums of the normalised weights associated with predicates in* Γ *and* Θ*, respectively. By Definition 22:*

$$W' = \gamma_\Delta$$

and similarly:

$$W'' = \overline{\gamma}_\Delta$$

By Definition 20, their absolute values are determined as follows:

$$|W'| = \sum_{P_i \in \Delta} \frac{ww(P_i)}{\sum_{Q \in \Delta} ww(Q)} = \frac{\sum_{P_i \in \Delta} ww(P_i)}{\sum_{Q \in \Delta} ww(Q)}$$

and

$$|W''| = \sum_{\neg P_k \in \Delta} \frac{ww(P_k)}{\sum_{Q \in \Delta} ww(Q)} = \frac{\sum_{\neg P_k \in \Delta} ww(P_k)}{\sum_{Q \in \Delta} ww(Q)}$$

where

$$\sum_{Q \in \Delta} ww(Q) = \sum_{P_i \in \Delta} ww(P_i) + \sum_{\neg P_k \in \Delta} ww(P_k)$$

Since this is exactly how we defined the Ulam weighted rejection degree in Definition 14, we can now introduce the following two equivalences:

$$\sum_{P_i \in \Delta} ww(P_i) = r^{uw}_\Delta(\neg\psi)$$

and

$$\sum_{\neg P_k \in \Delta} ww(P_k) = r^{uw}_\Delta(\psi)$$

Therefore:

$$W' - W'' = \frac{r^{uw}_\Delta(\neg\psi)}{r^{uw}_\Delta(\neg\psi) + r^{uw}_\Delta(\psi)} - \frac{r^{uw}_\Delta(\psi)}{r^{uw}_\Delta(\neg\psi) + r^{uw}_\Delta(\psi)}$$

Finally:

$$\frac{r^{uw}_\Delta(\neg\psi)}{r^{uw}_\Delta(\neg\psi) + r^{uw}_\Delta(\psi)} - \frac{r^{uw}_\Delta(\psi)}{r^{uw}_\Delta(\neg\psi) + r^{uw}_\Delta(\psi)} = \frac{r^{uw}_\Delta(\neg\psi) - r^{uw}_\Delta(\psi)}{r^{uw}_\Delta(\neg\psi) + r^{uw}_\Delta(\psi)}$$

which is equal to $\xi_{Bin-\Delta}$ *by* (4.12).

We can now prove the correspondence between the proof-theoretic derivability relation $\mid\sim$ and the semantic RJ-consequence relation $\mid\sim_{RJ}$ in the binary case $\{\psi, \neg\psi\}$.

Theorem 1 $\Delta \mid\sim \psi_\xi$ *with* $\xi > 0$ *if and only if* $\Delta \mid\sim_{RJ} \psi_\xi$.

Proof 5 (Proof of Theorem 1) *We show both directions of the equivalence.*

(⇒) From $\mid\sim$ ***to*** $\mid\sim_{RJ}$*:*

Assume $\Delta \mid\sim \psi_\xi$.
By Proposition 2, we have:

$$\xi = \gamma_\Delta - \overline{\gamma}_\Delta$$

By Proposition 4, we know that in the binary case:

$$\gamma_\Delta - \overline{\gamma}_\Delta = \xi_{\Delta-Bin}$$

By (4.12)*, we now that:*

$$\xi_{\Delta-Bin} = \frac{r^{uw}_\Delta(\neg\psi) - r^{uw}_\Delta(\psi)}{r^{uw}_\Delta(\neg\psi) + r^{uw}_\Delta(\psi)}$$

Since $\xi > 0$*, then* $r^{uw}_\Delta(\neg\psi) > r^{uw}_\Delta(\psi)$.
Since by assumption $\mathcal{H} = \{\psi, \neg\psi\}$*, then* $\hat{\mathcal{H}} = \{\psi\}$ *and* $\mathcal{H} \setminus \hat{\mathcal{H}}_\Delta = \{\neg\psi\}$ *so:*

$$r^{uw}_\Delta(\neg\psi) - r^{uw}_\Delta(\psi) = \min\{r_\Delta(H') - r_\Delta(H) \mid H \in \hat{\mathcal{H}}_\Delta, H' \in \mathcal{H} \setminus \hat{\mathcal{H}}_\Delta\}$$

Moreover, by Definition 13, it follows that $\Delta \mid\sim_{RJ} \psi$.
By Definition 15, $r^{uw}_\Delta(\neg\psi) - r^{uw}_\Delta(\psi)$ *is the (gUMO) level satisfied by* $\Delta \mid\sim_{RJ} \psi$.

We also know that:

$$r^{uw}_\Delta(\neg\psi) + r^{uw}_\Delta(\psi) = \sum_{H \in \mathcal{H}^f_\Delta} r^{uw}_\Delta(H)$$

By Definition 29, $\Delta \mid\sim_{RJ} \psi$ *therefore satisfies* $(gUMO\text{-}Norm)^\xi$ *which, by Definition 30, can be rewritten as:*

$$\Delta \mid\sim_{RJ} \psi_\xi$$

(⇐) From $\mid\sim_{RJ}$ ***to*** $\mid\sim$*:*

Assume $\Delta \mid\sim_{RJ} \psi_{\xi}$.
By Definition 30, this means that $\Delta^{W} \mid\sim_{RJ} \psi$ *satisfies* $(gUMO\text{-}Norm)^{\xi}$.
By Definition of gUMO-Norm 29, we know that:

$$\xi = \frac{j}{\sum_{H \in \mathcal{H}^{f}_{\Delta}} r^{uw}_{\Delta}(H)}$$

By Proposition 3, we know that in the binary case:

$$\frac{j}{\sum_{H \in \mathcal{H}^{f}_{\Delta}} r^{uw}_{\Delta}(H)} = \frac{r^{uw}_{\Delta}(\neg\psi) - r^{uw}_{\Delta}(\psi)}{r^{uw}_{\Delta}(\neg\psi) + r^{uw}_{\Delta}(\psi)}$$

By (4.12):

$$\frac{r^{uw}_{\Delta}(\neg\psi) - r^{uw}_{\Delta}(\psi)}{r^{uw}_{\Delta}(\neg\psi) + r^{uw}_{\Delta}(\psi)} = \xi_{\Delta-Bin}$$

By Proposition 4:

$$\xi_{\Delta-Bin} = \gamma_{\Delta} - \overline{\gamma}_{\Delta}$$

By Proposition 2, we can formulate the inference:

$$\Delta \mid\sim \psi_{(\gamma_{\Delta} - \overline{\gamma}_{\Delta})}$$

with $\gamma_{\Delta} - \overline{\gamma}_{\Delta} > 0$ *because* $\gamma_{\Delta} - \overline{\gamma}_{\Delta} = \xi_{\Delta-Bin} = \xi$ *and* ξ *is positive by assumption.*

In light of the same reason, we rewrite it as

$$\Delta \mid\sim \psi_{\xi}$$

4.8 Conclusion and Future Work

In the present investigation, a variation of the rational non-monotonic consequence relation inspired by the two-player Rényi-Ulam game was used to reason about the impact of data bias on ML outcomes. In particular, we focused on defining a novel way to test whether such outcomes are fair, assuming no causal knowledge on the data-generating process, but only knowing the model's input data and its associated decisions.

Our proposal can be expanded further by exploring more realistic and sophisticated models of measurement errors. One way to do so is to explore another variant of the Ulam game with channels, namely the Ulam game with channels and probabilistic half-lies. In this game setting, whether Responder can lie depends not only on the outcome of an aleatory experiment *but also on the true answer itself*. Half-lies help us to model those scenarios in which the probability of false negatives is significantly higher than that of false positives, or *vice versa*.

In the case above of the biased oximeter reading, for example, we would have that Responder tosses a coin biased towards Head with p=0.8 whenever the patient's *true* oxygen level is below the threshold. If Head is returned, then Responder lies and gives back a high oxygen level. In contrast, whenever the oxygen level is above the threshold, no coin is tossed and Responder is forced to return the truthful answer. In other words, while Questioner can trust observations indicating low blood oxygen levels, the same confidence cannot be applied to observations of normal blood oxygen levels. In this version of the game, we never reach complete certainty about the secret but, being the outcome of the aleatory experiments independent, Nature's answers will nevertheless asymptotically converge towards the true hypothesis.

Although the above discussion has mainly focused on the case of measurement error in the medical domain, the general framework is sufficiently flexible to apply to a broader range of cases within fair ML research. Each of these cases may require domain-specific error models and considerations. For instance, the disadvantaged sensitive group is not necessarily the one associated with noisier data; individuals may in fact even be advantaged by noisier channels. Consider, for example, a fraud-risk assessment context in which individuals are evaluated on the basis of features such as past frauds or criminal records. These data may be more accurate for certain sensitive groups – those more frequently scrutinised for fraud or more heavily policed – than for others. Again, probabilistic half-lies can be employed to capture asymmetric informativeness: the presence of a criminal record is a reasonably faithful indicator of past crime, whereas its absence does not necessarily indicate that no crime has occurred. Crucially, the degree of this asymmetry may differ systematically between sensitive groups. In this scenario, individuals associated with the noisier channel end up being *advantaged*, as spotting fraudsters among them becomes a harder task.

Conclusion

By combining philosophical analysis with formal methods, this work explores the foreground role played by the notion of computational error – and in particular ML error – in navigating the concept of algorithmic unfairness.

Specifically, the goal of Chapter 1, *A Sceptical Paradox for Computational Artefacts*, is to motivate from the ground up the choice in favour of a pragmatic view of computational normativity. By analogy to Kripke's claim that no fact of the matter can determine the meaning of a word, the sceptical paradox of implementation is an argument for the conclusion that no fact of the matter can determine the function of a computational artefact. The paradox targets the prevailing view within the philosophy of computer science, according to which the function of a computational artefact is to be identified with the content of its functional specification, a mathematical object that formalises the intentions of the artefact's designer. In existing formulations, such a view requires the existence of certain semantic intentions in the head of the designer. However, if we accept Kripke's claim that there is no such thing as a mental state of semantic intention, this ultimately leaves indeterminate what function a computational artefact is implementing. The only account that meets the desiderata of a sceptical solution to the paradox is the recently proposed theory of User Levels, according to which a computational artefact can be said to correctly implement a function when there is a convergence of a particular type among its different users, much recalling Kripke's idea that the agreement on the use of a word among members of a linguistic community – rather than a mental state – is what ultimately grounds meaning ascriptions.

Chapter 2, titled *How to Do Things with Bits: A Pragmatic Framework for Data-Driven Miscomputations*, is aimed at applying the User Levels theory to the specific case of ML systems. Preliminarily, ML errors are analysed from the

standpoint of the intentionalist account of the Levels of Abstraction (LoA). The traditional LoA taxonomy of miscomputations is revisited to capture the three fundamental errors that could occur in an ML system: errors of design, errors of prediction, and errors of justification. However, it is pointed out that the rigid normativity of the LoA theory seems inadequate to account for the dynamic correctness criteria of complex computational systems, like ML systems. In contrast to this, the User Levels theory proves to better accommodate how the normative instances of different users could circulate, feed back, compete, and possibly conflict among the agents interacting with the artefact, such as designers, end-users, and developers. Upon this alternative taxonomy, two types of pragmatic ML errors are distinguished: instruction miscomputations and implementation miscomputations. After that, the chapter sets the groundwork for a formal approach to the validity criteria for ML systems. In the context of deterministic scientific simulations, formal validity requirements have typically been defined to reason about the relationship between the mathematical model underlying the target system and the computational model used to simulate it. With ML simulations entering the picture, these formal requirements need to be reviewed for two main reasons. First, their output is inherently non-deterministic. Secondly, they are opaque, i.e., their mathematical model is abstracted away from the target system through the mediation of a computational model that remains largely opaque to us due to its high complexity. Accounting for these two characteristics of ML systems – non-determinism and opacity – requires the definition of probabilistic and weaker versions of the traditional relations of Simulation, Bisimulation, and Approximate Simulation. Based on these, three corresponding validity principles for ML systems are proposed. These criteria can account for a nuanced range of cases, from the strongest to the weakest, depending on how much a ML model can be assumed to correctly represent a given target system.

Chapter 3, *Algorithmic Fairness as a Repair Practice*, critically examines the widespread assumption that the unfair predictions returned by an ML system are ultimately due to a flaw in its design. In prevailing computer science approaches, algorithmic unfairness is primarily understood as the harmful consequence of a failure of ML design, most often attributed to the poor quality of the data used for training the model. From this perspective, algorithmic fairness is mainly regarded as a set of improved design choices, such as curating better datasets or adopting stricter benchmarks. More broadly, I point out that this view has

its roots in a *design paradigm*, dominant in the philosophical literature, which conceives of technical malfunctions as disruptions to an otherwise ideal continuity of correct and intentional uses that were predetermined once and for all at the moment of design. Building on the notion of 'broken world thinking', I propose an alternative epistemology of technical errors where malfunction is not seen as a disruptive event but rather as the ordinary condition of operation of the computational artefact. In this sense, the ML functionality emerges through a plurality of constant maintenance and repair practices throughout the lifecycle of the system. To substantiate the epistemological difference between the concepts of redesign and repair, I identify three limitations of the design paradigm related to ML errors: one ontological, one temporal, and one practical. In contrast, broken world thinking naturally accommodates these aspects, suggesting that it can provide a theoretically richer standpoint for our understanding of algorithmic discrimination. Accordingly, rather than conceiving fairness as a static value embedded in certain ML designs, we may better understand it as a plurality of reparative practices that address the persistent brokenness of the data ML systems are continuously fed with.

Finally, the goal of Chapter 4, *Data Speak but Sometimes Lie: A Formal Approach to Data Bias and Algorithmic Fairness*, is to concretely follow up on the epistemology of algorithmic fairness motivated in the previous chapter. To this end, the chapter develops a formal framework for reasoning about how data bias, defined as the differential level of noise of the data of different sensitive groups, impacts the fairness of ML outputs. In the semantics of this logic, a variation of the rational non-monotonic consequence relation is modelled after a two-player game, called the Ulam game, where the presence of differentially noisy channels of communication between the players captures the notion of data bias. Within this game-theoretic setting, one can model how correct information can be *retrieved* from gappy, inconsistent, and erroneous observations. In the corresponding calculus, this translates into the definition of a structural rule of correction that models how the addition of new information can be used to correct an initial ML prediction made on biased data. This can be exploited to achieve algorithmic fairness, as we formulate it in terms of a novel 'Principle of Consistency'.

This dissertation advances the debate on algorithmic fairness in two distinct ways. First, it uncovers some strong yet underacknowledged connections

between the debate on fair ML on the one hand and core philosophical open issues about the normativity of computational artefacts on the other. This contributes to broadening and deepening the conceptual pinpoints of such a research programme, without renouncing to the specificity of its methodologies and formal tools. Secondly, it does so by introducing a novel logical framework to reason about the impact of biased data on the fairness of ML outcomes.

The proposed analysis is subject to certain natural limitations. Most importantly, the exposition focuses programmatically on the context of ML predictions returned to the user and produced in a supervised setting. As such, the analysis does not take into account particular and more complex cases of Generative Artificial Intelligence (GenAI), although an image generation example is discussed in Section 2.5. For instance, certain emerging uses of GenAI seem particularly interesting to explore further, like agentic AI, especially when used to autonomously coding and implementing new computational artefacts in turn, and digital twins, for instance those used in the medical domain to simulate patient conditions. These GenAI cases might raise distinct theoretical concerns as well as ethical issues that call for an extension of the present framework.

To conclude, this exposition opens the way for a number of future research directions. A first natural extension of the work could be to further expand on the epistemological aspects of data errors in data-intensive contexts. This requires investigating how the very notion of error is shaped by the epistemic, social, and political conditions under which AI systems are developed and deployed. In particular, Leonelli's notion of "data shadows" [97] offers a valuable starting point to analyse how inquiry methods influence what counts as error and how typically negative features such as gaps, absences, omissions and inconsistencies in data acquire epistemic significance, as they can be informative by highlighting the structural conditions of data production, curation, and circulation of knowledge.

A second line of future research concerns the implementation of the logical formalism of Chapter 4 into a computable metric that practitioners can use to assess the fairness of their systems. More broadly, this direction aligns with the growing interest in 'Neurosymbolic AI' methods, which allow one to reason symbolically about the neural network's behaviour by establishing a correspondence between its low-level information processing and high-level

logical reasoning. As Neurosymbolic AI combines the expressive rigour of logical formalism with the flexibility of deep learning, it is argued to bring together 'the best of both worlds' and, as such, to constitute a promising instrument towards the development of a more interpretable, trustworthy and responsible generation of ML systems [16].

In this regard, it seems feasible to develop a neurosymbolic pipeline in which the neural network is injected with the logical constraints defined in the formal framework given in Chapter 4. The development of such a neurosymbolic tool would contribute to further bridging the gap between philosophical inquiry and computer science approaches to algorithmic fairness by the means of logical methods, in direct continuity with the spirit of the present dissertation.

Bibliography

[1] Alvarado, R. "Computer Simulations in Science". In: *Synthese Library* (2023), pp. 19–28. DOI: 10.1007/978-3-031-38647-3_2. URL: https://doi.org/10.1007/978-3-031-38647-3_2.

[2] Alvarado, R. "Challenges for Computational Reliabilism: Epistemic Warrants, Endogeneity and Error-based Opacity in Machine Learning". In: *Philosophy of science for machine learning: Core issues and new perspectives*. Ed. by Durán, J. M. and Pozzi, G. Springer, forthcoming.

[3] Alvarado, R. "Computer simulations as scientific instruments". In: *Foundations of Science* 27.3 (2022), pp. 1183–1205.

[4] Angius, N. and Plebe, A. "From Coding to Curing. Functions, Implementations, and Correctness in Deep Learning". In: *Philosophy and Technology* 36.3 (2023), pp. 1–27. DOI: 10.1007/s13347-023-00642-7.

[5] Angius, N. and Primiero, G. "The logic of identity and copy for computational artefacts". In: *Journal of Logic and Computation* 28.6 (Mar. 2018), pp. 1293–1322. ISSN: 0955-792X. DOI: 10.1093/logcom/exy012. eprint: https://academic.oup.com/logcom/article-pdf/28/6/1293/25677660/exy012.pdf. URL: https://doi.org/10.1093/logcom/exy012.

[6] Angius, N. and Primiero, G. "Copying safety and liveness properties of computational artefacts". In: *Journal of Logic and Computation* 33.5 (2023), pp. 1089–1117.

[7] Angius, N., Primiero, G., and Turner, R. "The Philosophy of Computer Science". In: *The Stanford Encyclopedia of Philosophy (Spring 2021 Edition)*. Ed. by Zalta, E. N. Metaphysics Research Lab, Stanford University, 2021.

[8] Austin, J. L. *How to Do Things with Words*. Clarendon Press, 1962.

[9] Baier, C., Hermanns, H., Katoen, J., and Wolf, V. "Bisimulation and Simulation Relations for Markov Chains". In: *Electronic Notes in Theoretical Computer Science* 162 (2006), pp. 73–78. DOI: 10.1016/j.entcs.2005.12.078. URL: https://doi.org/10.1016/j.entcs.2005.12.078.

[10] Bakker, M. A., Valdés, H. R., Tu, D. P., Gummadi, K. P., Varshney, K. R., Weller, A., and Pentland, A. S. "Fair Enough: Improving Fairness in Budget-Constrained Decision Making Using Confidence Thresholds". In: *SafeAI@AAAI*. 2020. URL: https://api.semanticscholar.org/CorpusID:211840427.

[11] Baldi, P., Corsi, E. A., and Hosni, H. "A logical framework for data-driven reasoning". In: *Logic Journal of the IGPL* (2024). URL: https://doi.org/10.1093/jigpal/jzae113.

[12] Banks, A. and Ashmore, R. "Requirements Assurance in Machine Learning." In: *SafeAI@ AAAI*. 2019.

[13] Bas, C. V. F. *The Scientific Image*. New York: Oxford University Press, 1980.

[14] Batini, C. and Scannapieco, M. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer, 2006. ISBN: 978-3-540-33172-8.

[15] Belle, V. "Toward A Logical Theory Of Fairness and Bias". In: *Theory and Practice of Logic Programming* 23.4 (2023), 865–883. ISSN: 1475-3081. DOI: 10.1017/s1471068423000157. URL: http://dx.doi.org/10.1017/S1471068423000157.

[16] Belle, V., Chockler, H., Vallor, S., Varshney, K. R., Vennekens, J., and Beckers, S. "Trustworthiness and Responsibility in AI - Causality, Learning, and Verification (Dagstuhl Seminar 24121)". In: *Dagstuhl Reports* 14.3 (2024). Ed. by Belle, V., Chockler, H., Vallor, S., Varshney, K. R., Vennekens, J., and Beckers, S., pp. 75–91. ISSN: 2192-5283. DOI: 10.4230/DagRep.14.3.75. URL: https://drops.dagstuhl.de/entities/document/10.4230/DagRep.14.3.75.

[17] Beretta, I. and Cinquini, M. "The importance of time in causal algorithmic recourse". In: *World Conference on Explainable Artificial Intelligence*. Springer. 2023, pp. 283–298.

[18] Blum, A. and Stangl, K. *Recovering from Biased Data: Can Fairness Constraints Improve Accuracy?* 2024. arXiv: 1912.01094 [cs.LG]. URL: https://arxiv.org/abs/1912.01094.

[19] Boden, M. A. "Creativity and artificial intelligence". In: *Artificial Intelligence* 103.1 (1998). Artificial Intelligence 40 years later, pp. 347–356. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(98)00055-1. URL: https://www.sciencedirect.com/science/article/pii/S0004370298000551.

[20] Boge, F. J. "Two Dimensions of Opacity and the Deep Learning Predicament". In: *Minds Mach.* 32.1 (Mar. 2022), 43–75. ISSN: 0924-6495. DOI: 10.1007/s11023-021-09569-4. URL: https://doi.org/10.1007/s11023-021-09569-4.

[21] Brading, K. and Landry, E. "Scientific Structuralism: Presentation and Representation". In: *Philosophy of Science* 73.5 (2006), 571–581. DOI: 10.1086/518327.

[22] Buckner, C. "Understanding adversarial examples requires a theory of artefacts for deep learning". In: *Nature Machine Intelligence* 2.12 (2020), pp. 731–736. DOI: 10.1038/s42256-020-00266-y.

[23] Buda, A. G., Coraglia, G., Genco, F. A., Manganini, C., and Primiero, G. *Bias amplification chains in ML-based systems with an application to credit scoring.* 2025. URL: https://ceur-ws.org/Vol-3881/paper9.pdf.

[24] Buda, A. G., Manganini, C., and Primiero, G. "A Philosophical Framework for Data-Driven Miscomputations". In: *Philosophies* 10.4 (2025). ISSN: 2409-9287. DOI: 10.3390/philosophies10040088. URL: https://www.mdpi.com/2409-9287/10/4/88.

[25] Buda, A. G. and Primiero, G. "A Pragmatic Theory of Computational Artefacts". In: *Minds and Machines* 34.1 (2024), pp. 139–170. DOI: 10.1007/s11023-023-09650-0.

[26] Buechner, J. "Not Even Computing Machines Can Follow Rules". In: *Saul Kripke*. Ed. by Berger, A. London: Palgrave Macmillan, 2011, pp. 343–368.

[27] Buechner, J. "Does Kripke's Argument Against Functionalism Undermine the Standard View of What Computers Are?" In: *Minds and Machines* 28.3 (2018), pp. 491–513.

[28] Byrne, A. "On Misinterpreting Kripke's Wittgenstein". In: *Philosophy and Phenomenological Research* 56.2 (1996), pp. 339–343.

[29] Canning, R. "That maintenance 'iceberg'". In: *EDP Analyzer* 10.10 (1972).

[30] Cantwell Smith, B. "The Limits of Correctness". In: *Computerization and Controversy*. Ed. by Kling, R. Academic Press, San Diego: ACM SIGCAS Computers and Society, 1996, pp. 18–26.

[31] Caton, S. and Haas, C. "Fairness in Machine Learning: A Survey". In: *ACM Comput. Surv.* 56.7 (2024). ISSN: 0360-0300. DOI: 10.1145/3616865. URL: https://doi.org/10.1145/3616865.

[32] Cerrato, M., Vallenas Coronel, A., and Köppel, M. "The Case for Correctability in Fair Machine Learning". In: *Proceedings of the European Workshop on Algorithmic Fairness (EWAF'23)*. Winterthur, Switzerland, 2023, pp. 1–4.

[33] Chalmers, D. J. "Does a rock implement every finite-state automaton?" In: *Synthese* 108.3 (1996), pp. 309–333.

[34] Cicalese, F. *Fault-Tolerant Search Algorithms: Reliable Computation with Unreliable Information*. Springer Publishing Company, Incorporated, 2013. ISBN: 3642173268.

[35] Cicalese, F. and Mundici, D. "Recent developments of feedback coding and its relations with many-valued logic". In: *Proof, Computation and Agency: Logic at the Crossroads* (2011), pp. 115–131.

[36] Cinquini, M., Beretta, I., Ruggieri, S., and Valera, I. "A Practical Approach to Causal Inference over Time". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 14. 2025, pp. 14832–14839.

[37] Cooper, A. F., Abrams, E., and NA, N. "Emergent Unfairness in Algorithmic Fairness-Accuracy Trade-Off Research". In: *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. AIES '21. Virtual Event, USA: Association for Computing Machinery, 2021, 46–54. ISBN: 9781450384735. DOI: 10.1145/3461702.3462519. URL: https://doi.org/10.1145/3461702.3462519.

[38] Cooper, A. F., Frankle, J., and De Sa, C. "Non-Determinism and the Lawlessness of Machine Learning Code". In: *Proceedings of the 2022 Symposium on Computer Science and Law*. CSLAW '22. Washington DC, USA: Association for Computing Machinery, 2022, 1–8. ISBN: 9781450392341. DOI: 10.1145/3511265.3550446. URL: https://doi.org/10.1145/3511265.3550446.

[39] Coraglia, G., D'Asaro, F. A., Genco, F. A., Giannuzzi, D., Posillipo, D., Primiero, G., and Quaggio, C. "BRIOxAlkemy: A Bias detecting tool". In: *Proceedings of the 2nd Workshop on Bias, Ethical AI, Explainability and the role of Logic and Logic Programming co-located with the 22nd International Conference of the Italian Association for Artificial Intelligence (AI*IA 2023)*. Ed. by Boella, G., D'Asaro, F. A., Dyoub, A., Gorrieri, L., Lisi, F. A., and Chiara Manganini, G. P. Vol. 3615. CEUR Workshop Proceedings. CEUR-WS.org, 2023, pp. 44–60. URL: https://ceur-ws.org/Vol-3615/paper4.pdf.

[40] Coraglia, G., Genco, F. A., Piantadosi, P., Bagli, E., Giuffrida, P., Posillipo, D., and Primiero, G. *Evaluating AI fairness in credit scoring with the BRIO tool*. 2024. arXiv: 2406.03292.

[41] Corbett-Davies, S., Pierson, E., Feller, A., Goel, S., and Huq, A. "Algorithmic Decision Making and the Cost of Fairness". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '17. Halifax, NS, Canada: Association for Computing Machinery, 2017, 797–806. ISBN: 9781450348874. DOI: 10.1145/3097983.3098095. URL: https://doi.org/10.1145/3097983.3098095.

[42] Corsi, E. A. and Montagna, F. "The Rényi–Ulam games and many-valued logics". In: *Fuzzy Sets and Systems* 301 (2016), pp. 37–50.

[43] Crawford, K. and Paglen, T. *Excavating AI: The Politics of Training Sets for Machine Learning*. 2019. URL: https://excavating.ai.

[44] Creel, K. and Hellman, D. "The Algorithmic Leviathan: Arbitrariness, Fairness, and Opportunity in Algorithmic Decision-Making Systems". In: *Canadian Journal of Philosophy* 52.1 (2022), pp. 26–43. DOI: 10.1017/can.2022.3.

[45] Cummins, R. "Functional analysis". In: *Journal of Philosophy* 72 (1975), pp. 741–764.

[46] D'Asaro, F. A., Genco, F. A., and Primiero, G. "Checking trustworthiness of probabilistic computations in a typed natural deduction system". In: *Journal of Logic and Computation* (2025). exaf003. DOI: 10.1093/logcom/exaf003. URL: https://doi.org/10.1093/logcom/exaf003.

[47] De Millo, R., Lipton, R., and Perlis, A. "Social Processes and Proofs of Theorems and programs". In: *Communications of the ACM* 22.5 (1979), pp. 271–281.

[48] Dumitrache, A., Aroyo, L., and Welty, C. "Crowdsourcing Ground Truth for Medical Relation Extraction". In: *ACM Trans. Interact. Intell. Syst.* 8.2 (July 2018). ISSN: 2160-6455. DOI: 10.1145/3152889. URL: https://doi.org/10.1145/3152889.

[49] Durán, J. M. "Computer Simulations in Science and Engineering - Concepts, Practices, Perspectives". In: *Springer* (2018).

[50] Durán, J. M. and Formanek, N. "Grounds for Trust: Essential Epistemic Opacity and Computational Reliabilism". In: *Minds and Machines* 28.4 (2018), pp. 645–666.

[51] Dutta, S., Wei, D., Yueksel, H., Chen, P.-Y., Liu, S., and Varshney, K. "Is There a Trade-Off Between Fairness and Accuracy? A Perspective Using Mismatched Hypothesis Testing". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by III, H. D. and Singh, A. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2803–2813. URL: https://proceedings.mlr.press/v119/dutta20a.html.

[52] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. "Fairness through Awareness". In: *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*. ITCS '12. Cambridge, Massachusetts: Association for Computing Machinery, 2012, 214–226. ISBN: 9781450311151. DOI: 10.1145/2090236.2090255. URL: https://doi.org/10.1145/2090236.2090255.

[53] Eaton, A. W. "Artefacts and their functions". In: *Oxford Handbook of History and Material Culture*. Ed. by Gaskell, I. and Carter, S. A. Oxford: Oxford University Press, 2020, pp. 35–53.

[54] Facchini, A. and Termine, A. "Towards a Taxonomy for the Opacity of AI Systems". In: *Philosophy and Theory of Artificial Intelligence 2021*. Springer, 2022, pp. 73–89.

[55] Fawzy, A., Wu, T. D., Wang, K., Robinson, M. L., Farha, J., Bradke, A., Golden, S. H., Xu, Y., and Garibaldi, B. T. "Racial and ethnic discrepancy in pulse oximetry and delayed identification of treatment eligibility among patients with COVID-19". In: *JAMA internal medicine* 182.7 (2022), pp. 730–738.

[56] Ferraz-Caetano, J. "The Artificial Intelligence Explanatory Trade-Off on the Logic of Discovery in Chemistry". In: *Philosophies* 8.2 (2023). ISSN: 2409-9287. DOI: 10.3390/philosophies8020017. URL: https://www.mdpi.com/2409-9287/8/2/17.

[57] Fetzer, J. "Program Verification: The Very Idea". In: *Communications of the ACM* 31.9 (1988), pp. 1048–1063.

[58] Floridi, L. "The method of levels of abstraction". In: *Minds and Machines* 18.3 (2008), pp. 303–329.

[59] Floridi, L., Fresco, N., and Primiero, G. "On Malfunctioning Software". In: *Synthese* 192.4 (2015), pp. 1199–1220. DOI: 10.1007/s11229-014-0610-3.

[60] Fodor, J. A. *The Language of Thought*. Cambridge, MA: Harvard University Press, 1975.

[61] Fresco, N. and Primiero, G. "Miscomputation". In: *Philosophy and Technology* 26 (2013), pp. 253–272.

[62] Friedler, S. A., Scheidegger, C., and Venkatasubramanian, S. *On the (im)possibility of fairness*. 2016. arXiv: 1609.07236 [cs.CY]. URL: https://arxiv.org/abs/1609.07236.

[63] Frigg, R. and Nguyen, J. "Models and Representation". In: *Springer Handbook of Model-Based Science*. Ed. by Magnani, L. and Bertolotti, T. 2017, pp. 49–102.

[64] Genco, F. A. and Primiero, G. "A Typed Lambda-Calculus for Establishing Trust in Probabilistic Programs". 2023. URL: https://arxiv.org/abs/2302.00958.

[65] Godfrey, M. W. and German, D. M. "The past, present, and future of software evolution". In: *2008 Frontiers of Software Maintenance*. 2008, pp. 129–138. DOI: 10.1109/FOSM.2008.4659256.

[66] Goodfellow, I. J., Shlens, J., and Szegedy, C. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML]. URL: https://arxiv.org/abs/1412.6572.

[67] Grice, H. P. "Logic and conversation". In: *Speech acts*. Brill, 1975, pp. 41–58.

[68] Guala, F. "Models, Simulations, and Experiments". In: *Model-Based Reasoning: Science, Technology, Values*. Ed. by Magnani, L. and Nersessian, N. J. New York, NY: Springer US, 2002, pp. 59–74. ISBN: 978-1-4615-0605-8. DOI: 10.1007/978-1-4615-0605-8_4. URL: https://doi.org/10.1007/978-1-4615-0605-8_4.

[69] Guardo, A. "Semantic Dispositionalism and Non-Inferential Knowledge". In: *Philosophia* 42.3 (2014), pp. 749–759.

[70] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. "A Survey of Methods for Explaining Black Box Models". In: *ACM Comput. Surv.* 51.5 (2019), 93:1–93:42. DOI: 10.1145/3236009. URL: https://doi.org/10.1145/3236009.

[71] Haas, J. de and Houkes, W. "Can't Software Malfunction?" In: *Metaphysics* 9.1 (2025), pp. 1–15. DOI: 10.5334/met.165.

[72] Harding, S. G. *The feminist standpoint theory reader: Intellectual and political controversies*. Psychology Press, 2004.

[73] Hirota, Y., Nakashima, Y., and Garcia, N. "Quantifying Societal Bias Amplification in Image Captioning". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 13440–13449. DOI: 10.1109/CVPR52688.2022.01309.

[74] Holm, S. "The problem of phantom functions". In: *Erkenntnis* 82.1 (2017), pp. 233–241.

[75] Houkes, W. and Vermaas, P. *Technical Functions: On the Use and Design of Artefacts*. Dordrecht: Springer, 2010.

[76] Humbatova, N., Jahangirova, G., Bavota, G., Riccio, V., Stocco, A., and Tonella, P. "Taxonomy of real faults in deep learning systems". In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. ICSE '20. Seoul, South Korea: Association for Computing Machinery, 2020, 1110–1121. ISBN: 9781450371216. DOI: 10.1145/3377811.3380395. URL: https://doi.org/10.1145/3377811.3380395.

[77] Humphreys, P. "Extending Ourselves: Computational Science, Empiricism, and Scientific Method". In: *Oxford University Press* (2004).

[78] Hurshman, C. "Artefacts and intervention: a persistence theory of artefact functions". In: *Synthese* 202.5 (2023), pp. 1–28.

[79] Hurshman, C. "Do opaque algorithms have functions?" In: *Synthese* 204.3 (2024), pp. 1–26.

[80] Ignatiev, A., Cooper, M. C., Siala, M., Hebrard, E., and Marques-Silva, J. "Towards Formal Fairness in Machine Learning". In: *Principles and Practice of Constraint Programming: 26th International Conference, CP 2020, Louvain-La-Neuve, Belgium, September 7–11, 2020, Proceedings*. Louvain-la-Neuve, Belgium: Springer-Verlag, 2020, 846–867. ISBN: 978-3-030-58474-0. DOI:

10 . 1007 / 978 - 3 - 030 - 58475 - 7 _ 49. URL: https://doi.org/10.1007/978-3-030-58475-7_49.

[81] Illari, P. and Floridi, L. *The Philosophy of Information Quality*. Springer International Publishing, 2014.

[82] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. *Adversarial Examples Are Not Bugs, They Are Features*. 2019. arXiv: 1905.02175 [stat.ML]. URL: https://arxiv.org/abs/1905.02175.

[83] Jackson, S. J. "Rethinking Repair". In: *Media technologies: Essays on communication, materiality, and society* (2014), pp. 221–39.

[84] Jacobs, A. Z. and Wallach, H. "Measurement and Fairness". In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '21. Virtual Event, Canada: Association for Computing Machinery, 2021, 375–385. ISBN: 9781450383097. DOI: 10 . 1145 / 3442188 . 3445901. URL: https://doi.org/10.1145/3442188.3445901.

[85] James, G., Witten, D., Hastie, T., Tibshirani, R., and Taylor, J. *An Introduction to Statistical Learning*. Springer International Publishing, 2023. ISBN: 9783031387470. DOI: 10.1007/978-3-031-38747-0. URL: http://dx.doi.org/10.1007/978-3-031-38747-0.

[86] Jarrahi, M. H., Memariani, A., and Guha, S. "The Principles of Data-Centric AI". In: *Commun. ACM* 66.8 (July 2023), 84–92. ISSN: 0001-0782. DOI: 10.1145/3571724. URL: https://doi.org/10.1145/3571724.

[87] Jonsson, B. et al. "Probabilistic Extensions of State Transition Systems". In: *Formal Methods in System Design* 18 (2001), pp. 1–30.

[88] Kang, Q. and Tay, W. P. "Sequential Multi-Class Labeling in Crowdsourcing: A Ulam-Renyi Game Approach". In: *Proceedings of the International Conference on Web Intelligence*. WI '17. Leipzig, Germany: Association for Computing Machinery, 2017, 245–251. ISBN: 9781450349512. DOI: 10 . 1145 / 3106426 . 3106446. URL: https://doi.org/10.1145/3106426.3106446.

[89] Karimi, A.-H., Schölkopf, B., and Valera, I. "Algorithmic Recourse: from Counterfactual Explanations to Interventions". In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '21. Virtual Event, Canada: Association for Computing Machinery, 2021,

353–362. ISBN: 9781450383097. DOI: 10.1145/3442188.3445899. URL: https://doi.org/10.1145/3442188.3445899.

[90] Kawamoto, Y. "An epistemic approach to the formal specification of statistical machine learning". In: *Software and Systems Modeling* 20.2 (2020), 293–310. ISSN: 1619-1374. DOI: 10.1007/s10270-020-00825-2. URL: http://dx.doi.org/10.1007/s10270-020-00825-2.

[91] Kripke, S. *Wittgenstein on Rules and Private Language – An Elementary Exposition*. Oxford: Blackwell, 1981.

[92] Kügelgen, J. von, Bhatt, U., Karimi, A.-H., Valera, I., Weller, A., and Scholkopf, B. "On the Fairness of Causal Algorithmic Recourse". In: *ArXiv* abs/2010.06529 (2020). URL: https://api.semanticscholar.org/CorpusID:222310477.

[93] Kusch, M. *A Sceptical Guide to Meaning and Rules: Defending Kripke's Wittgenstein*. Montreal: McGill-Queen's University Press, 2006.

[94] Kusner, M., Loftus, J., Russell, C., and Silva, R. "Counterfactual fairness". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, 4069–4079. ISBN: 9781510860964.

[95] Lehman, M. M. "Laws of Software Evolution Revisited". In: *European Workshop on Software Process Technology*. 1996. URL: https://api.semanticscholar.org/CorpusID:16925060.

[96] Leonelli, S. "Learning from Data Journeys". In: *Data Journeys in the Sciences*. Springer, 2020, pp. 1–24.

[97] Leonelli, S., Rappert, B., and Davies, G. "Data Shadows: Knowledge, Openness, and Absence". In: *Science, Technology, & Human Values* 42.2 (2017), pp. 191–202. DOI: 10.1177/0162243916687039. eprint: https://doi.org/10.1177/0162243916687039. URL: https://doi.org/10.1177/0162243916687039.

[98] Lin, C. K. and Jackson, S. J. "From Bias to Repair: Error as a Site of Collaboration and Negotiation in Applied Data Science Work". In: *Proc. ACM Hum.-Comput. Interact.* 7.CSCW1 (Apr. 2023). DOI: 10.1145/3579607. URL: https://doi.org/10.1145/3579607.

[99] Lingel, J. "The digital remains: Social media and practices of online grief". In: *The Information Society* 29.3 (2013), pp. 190–195. DOI: 10.1080/01972243.2013.777311.

[100] Liu, X. and Lorini, E. "A unified logical framework for explanations in classifier systems". In: *Journal of Logic and Computation* 33.2 (2023), pp. 485–515. DOI: 10.1093/logcom/exac102.

[101] Lundberg, S. M. and Lee, S.-I. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30 (2017).

[102] López, G. and Núñez, D. "Markov Chains and Probabilistic Process Algebra". In: *Theoretical Computer Science* 321 (2004), pp. 1–20.

[103] Malevé, N. "On the data set's ruins". In: *AI & Society* 36.4 (2021), pp. 1117–1131. DOI: 10.1007/s00146-020-01093-w.

[104] Manganini, C. "A Sceptical Paradox for Computational Artefacts". In: *Philosophical Inquiries* (forthcoming).

[105] Manganini, C., Anna Corsi, E., and Primiero, G. "Data speak but sometimes lie: A game-theoretic approach to data bias and algorithmic fairness". In: *International Journal of Approximate Reasoning* 190 (2026), p. 109608. ISSN: 0888-613X. DOI: https://doi.org/10.1016/j.ijar.2025.109608. URL: https://www.sciencedirect.com/science/article/pii/S0888613X2500249X.

[106] Manganini, C. and Primiero, G. "Reasoning With and About Bias". In: *Perspectives on Logics for Data-driven Reasoning*. Ed. by Hosni, H. and Landes, J. Cham: Springer Nature Switzerland, 2024, pp. 127–154. ISBN: 978-3-031-77892-6. DOI: 10.1007/978-3-031-77892-6_7. URL: https://doi.org/10.1007/978-3-031-77892-6_7.

[107] Manganini, C. and Primiero, G. "Defining Formal Validity Criteria for Machine Learning Models". In: *Philosophy of Science for Machine Learning: Core Issues and New Perspectives*. Ed. by Durán, J. M. and Pozzi, G. Cham: Springer Nature Switzerland, 2026, pp. 295–312. ISBN: 978-3-032-03083-2. DOI: 10.1007/978-3-032-03083-2_14. URL: https://doi.org/10.1007/978-3-032-03083-2_14.

[108] McDowell, J. "Wittgenstein on following a rule". In: *Synthese* 58 (1984), pp. 325–364.

[109] McGinn, C. *Wittgenstein on Meaning – An Interpretation and Evaluation*. Oxford-New York: Blackwell, 1987.

[110] McLaughlin, P. *What Functions Explain: Functional Explanation and Self-Reproducing Systems*. Cambridge: Cambridge University Press, 2000.

[111] Medina, J. *The epistemology of resistance: Gender and racial oppression, epistemic injustice, and the social imagination*. Oxford University Press, 2013.

[112] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. *A Survey on Bias and Fairness in Machine Learning*. 2022. arXiv: 1908.09635 [cs.LG].

[113] Menon, A. K. and Williamson, R. C. "The cost of fairness in binary classification". In: *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*. Ed. by Friedler, S. A. and Wilson, C. Vol. 81. Proceedings of Machine Learning Research. PMLR, 2018, pp. 107–118. URL: https://proceedings.mlr.press/v81/menon18a.html.

[114] Messeri, L. and Crockett, M. J. "Artificial intelligence and illusions of understanding in scientific research". In: *Nature* 627 (2024), pp. 49–58. DOI: 10.1038/s41586-024-07146-0. URL: https://doi.org/10.1038/s41586-024-07146-0.

[115] Miller, A. "Rule-Following Skepticism". In: *Routledge Companion to Epistemology*. Ed. by Bernecker, S. and Pritchard, D. London: Routledge, 2010, pp. 441–451.

[116] Montagna, F., Marini, C., and Simi, G. "Product logic and probabilistic Ulam games". In: *Fuzzy Sets and Systems* 158.6 (2007). The Logic of Soft Computing, pp. 639–651. ISSN: 0165-0114. DOI: https://doi.org/10.1016/j.fss.2006.11.007. URL: https://www.sciencedirect.com/science/article/pii/S0165011406005148.

[117] Mundici, D. "Ulam games, Łukasiewicz logic, and AF C*-algebras". In: *Fundamenta Informaticae* 18.2-4 (1993), pp. 151–161.

[118] Mundici, D. et al. "The logic of Ulam's game with lies". In: *Knowledge, Belief and Strategic Interaction, Cambridge Studies in Probability, Induction, and Decision Theory*. Cambridge University Press, 1992, pp. 275–284.

[119] Noriega-Campero, A., Bakker, M. A., Garcia-Bulle, B., and Pentland, A. S. "Active Fairness in Algorithmic Decision Making". In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (2018). URL: https://api.semanticscholar.org/CorpusID:52896535.

[120] Olson, D., Meyerson, J., Parsons, M. A., Castro, J., Lassere, M., Wright, D. J., Arnold, H., Galvan, A. S., Hswe, P, Nowviskie, B., Russell, A., Vinsel, L., and Acker, A. *Information Maintenance as a Practice of Care*. June 2019. DOI: 10.5281/zenodo.3236410. URL: https://doi.org/10.5281/zenodo.3236410.

[121] Pearl, J. and Mackenzie, D. *The Book of Why. The New Science of Cause and Effect*. New York: Basic Books, 2018. ISBN: 978-0-465-09760-9.

[122] Pelc, A. "Searching Games with Errors—Fifty Years of Coping with Liars". In: *Theor. Comput. Sci.* 270.1–2 (Jan. 2002), 71–109. ISSN: 0304-3975. DOI: 10.1016/S0304-3975(01)00303-6. URL: https://doi.org/10.1016/S0304-3975(01)00303-6.

[123] Piccinini, G. *Physical Computation: A Mechanistic Account*. Reprint. Oxford: Oxford University Press, 2018.

[124] Preston, B. "Why is a wing like a spoon? A pluralist theory of function". In: *The Journal of Philosophy* 95.5 (1998), pp. 215–254.

[125] Preston, B. *A Philosophy of Material Culture: Action, Function, and Mind*. 1st ed. London: Routledge, 2012.

[126] Primiero, G. "A Taxonomy of Errors for Information Systems". In: *Minds and Machines* 24 (Aug. 2014), pp. 249–273. DOI: 10.1007/s11023-013-9307-5.

[127] Primiero, G. "Information in the philosophy of computer science". In: *The Routledge Handbook of Philosophy of Information*. Ed. by Floridi, L. 1st ed. London: Routledge, 2016, pp. 90–106.

[128] Primiero, G. *On the Foundations of Computing*. Oxford: Oxford University Press, 2019.

[129] Putnam, H. "The mental life of some machines". In: *Mind, Language and Reality, Philosophical Papers, Vol. 2*. Cambridge: Cambridge University Press, 1975, pp. 408–428.

[130] Quaresmini, C. and Primiero, G. "Data Quality Dimensions for Fair AI". In: *Proceedings of the 2nd Workshop on Fairness and Bias in AI co-located with 27th European Conference on Artificial Intelligence (ECAI 2024), Santiago de Compostela, Spain, October 20th, 2024*. Ed. by Calegari, R., Dignum, V., and O'Sullivan, B. Vol. 3808. CEUR Workshop Proceedings. CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-3808/paper12.pdf.

[131] Rathkopf, C. and Heinrichs, B. “Learning to Live with Strange Error: Beyond Trustworthiness in Artificial Intelligence Ethics”. In: *Cambridge Quarterly of Healthcare Ethics* 33.3 (2024), 333–345.

[132] Räz, T. “Group Fairness: Independence Revisited”. In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT ’21. Virtual Event, Canada: Association for Computing Machinery, 2021, 129–137. ISBN: 9781450383097. DOI: 10.1145/3442188.3445876. URL: https://doi.org/10.1145/3442188.3445876.

[133] Rényi, A. “On a problem of information theory”. In: *A MAGYAR TUDOMÁNYOS AKADÉMIA MATEMATIKAI KUTATÓ INTÉZETÉNEK KÖZLEMÉNYEI* 6.4 (1961), pp. 515–516.

[134] Ribeiro, M. T., Singh, S., and Guestrin, C. “" Why should i trust you?" Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[135] Russell, A. L. and Vinsel, L. “After innovation, turn to maintenance”. In: *Technology and Culture* 59.1 (2018), pp. 1–25.

[136] Seshia, S. A., Desai, A., Dreossi, T., Fremont, D., Ghosh, S., Kim, E., Shivakumar, S., Vazquez-Chanlatte, M., and Yue, X. “Formal Specification for Deep Neural Networks”. In: *Proceedings of the International Symposium on Automated Technology for Verification and Analysis (ATVA)*. 2018, pp. 20–34.

[137] Shagrir, O. “Why we view the brain as a computer”. In: *Synthese* 153.3 (2006), pp. 393–416.

[138] Shanker, S. G. *Wittgenstein and the Turning Point in the Philosophy of Mathematics*. Amsterdam: Amsterdam University Press, 1987.

[139] Shanker, S. G. “Wittgenstein versus Turing on the nature of Church’s thesis”. In: *Notre Dame Journal of Formal Logic* 28.4 (1987).

[140] Shanker, S. G. *Wittgenstein’s Remarks on the Foundations of AI*. London: Routledge, 1998.

[141] Shumailov, I., Zhao, Y., Mullins, R., Papernot, N., and Anderson, R. “The Curse of Recursion: Training on Generated Data Makes Models Forget”. In: *Proceedings of the 40th International Conference on Machine Learning (ICML)*. 2023. URL: https://arxiv.org/abs/2305.17493.

[142] Simkute, A., Luger, E., Evans, M., and Jones, R. *"It is there, and you need it, so why do you not use it?" Achieving better adoption of AI systems by domain experts, in the case study of natural science research*. 2024. arXiv: 2403.16895 [cs.HC]. URL: https://arxiv.org/abs/2403.16895.

[143] Sjoding, M. W., Dickson, R. P., Iwashyna, T. J., Gay, S. E., and Valley, T. S. "Racial bias in pulse oximetry measurement". In: *New England Journal of Medicine* 383.25 (2020), pp. 2477–2478.

[144] Sprevak, M. "Kripke's paradox and the Church–Turing thesis". In: *Synthese* 160.2 (2008), pp. 285–295.

[145] Sprevak, M. "Triviality arguments about computational implementation". In: *The Routledge Handbook of the Computational Mind*. Routledge, 2018, pp. 175–191.

[146] Spurrett, D. "On hostile and oppressive affective technologies". In: *Topoi* 43 (2024), pp. 821–832.

[147] Stabler, E. P. "Kripke on functionalism and finite automata". In: *Synthese* 70 (1987), pp. 1–22.

[148] Steinert, S. "Maintenance of Value and the Value of Maintenance". In: *Maintenance and Philosophy of Technology*. Taylor & Francis, 2024.

[149] Sullivan, E. and Kasirzadeh, A. "Explanation Hacking: The Perils of Algorithmic Recourse". In: *Philosophy of science for machine learning: Core issues and new perspectives*. Ed. by Durán, J. M. and Pozzi, G. Springer, forthcoming.

[150] Summers, C. and Dinneen, M. J. "Nondeterminism and Instability in Neural Network Optimization". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Meila, M. and Zhang, T. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 9913–9922. URL: https://proceedings.mlr.press/v139/summers21a.html.

[151] Suresh, H. and Guttag, J. V. "A Framework for Understanding Sources of Harm throughout the Machine Learning Life Cycle". In: *Equity and Access in Algorithms, Mechanisms, and Optimization* (2019).

[152] Termine, A., Primiero, G., and D'Asaro, F. A. "Modelling Accuracy and Trustworthiness of Explaining Agents". In: *Logic, Rationality, and Interaction - 8th International Workshop, LORI 2021, Xi'ian, China, October 16-18, 2021, Proceedings*. Ed. by Ghosh, S. and Icard, T. Vol. 13039. Lecture Notes in Computer Science. Springer, 2021, pp. 232–245. DOI: 10.1007/978-3-030-88708-7_19. URL: https://doi.org/10.1007/978-3-030-88708-7_19.

[153] Turner, R. "Specification". In: *Minds and Machines* 21.2 (2011), pp. 135–152.

[154] Turner, R. *Computational Artefacts: Towards a Philosophy of Computer Science*. London: Springer, 2018.

[155] Turner, R. "Computational Artefacts: the Things of Computer Science". In: *Philosophy & Technology* 33 (2019), pp. 357–367.

[156] Turner, R. "Computational Intention". In: *Studies in Logic, Grammar and Rhetoric* 63.1 (2020), pp. 19–30.

[157] Ulam, S. M. "Adventures of a Mathematician". In: *Mathematics*. Chapman and Hall/CRC, 2019, pp. 193–200.

[158] Ustun, B., Spangher, A., and Liu, Y. "Actionable Recourse in Linear Classification". In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. FAT* '19. Atlanta, GA, USA: Association for Computing Machinery, 2019, 10–19. ISBN: 9781450361255. DOI: 10.1145/3287560.3287566. URL: https://doi.org/10.1145/3287560.3287566.

[159] Venkatasubramanian, S. and Alfano, M. "The philosophical basis of algorithmic recourse". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. FAT* '20. Barcelona, Spain: Association for Computing Machinery, 2020, 284–293. ISBN: 9781450369367. DOI: 10.1145/3351095.3372876. URL: https://doi.org/10.1145/3351095.3372876.

[160] Verma, S. and Rubin, J. "Fairness Definitions Explained". In: *Proceedings of the International Workshop on Software Fairness*. FairWare '18. Gothenburg, Sweden: Association for Computing Machinery, 2018, 1–7. ISBN: 9781450357463. DOI: 10.1145/3194770.3194776. URL: https://doi.org/10.1145/3194770.3194776.

[161] Vogelsang, A. and Borg, M. "Requirements Engineering for Machine Learning: Perspectives from Data Scientists". In: *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. 2019, pp. 245–251. DOI: 10.1109/REW.2019.00050.

[162] Vygotsky, L. S. *Language and Thought*. Cambridge, MA: MIT Press, 1962.

[163] Warren, J. "Killing Kripkenstein's Monster". In: *Nous* 54.2 (2020), pp. 257–289.

[164] Whitehead, M., Ali, R., Carrol, E., Holmes, C., and Kee, F. *Equity in Medical Devices: Independent Review*. Tech. rep. 2023. URL: https://assets.publishing.service.gov.uk/media/65e89e9e62ff48001a87b2d8/equity-in-medical-devices-independent-review-report-web-accessible.pdf.

[165] Wick, M., Panda, S., and Tristan, J.-B. "Unlocking fairness: a trade-off revisited". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[166] Wilson, G. M. "Semantic Realism and Kripke's Wittgenstein". In: *Philosophy and Phenomenological Research* 58.1 (1998), pp. 99–122.

[167] Wittgenstein, L. *Philosophical Investigations*. 3rd ed. [1968]. Oxford: Basil Blackwell, 1953.

[168] Wolpert, D. H. and Macready, W. G. "No Free Lunch Theorems for Optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 67–82.

[169] Wright, L. *Teleological Explanations: An Etiological Analysis of Goals and Functions*. Berkeley: University of California Press, 1976.

[170] Wu, X. and Zhang, X. "Automated Inference on Criminality using Face Images". In: *ArXiv* abs/1611.04135 (2016). URL: https://api.semanticscholar.org/CorpusID:8149177.

[171] Young, M. T. "Maintenance". In: *The Routledge handbook of the philosophy of engineering*. Routledge, 2020, pp. 356–368.

[172] Young, M. T. "Now you see it (now you don't): Users, maintainers and the invisibility of infrastructure". In: *Technology and the City: Towards a Philosophy of Urban Technologies*. Springer, 2021, pp. 101–119.

[173] Öhman, C. J. and Watson, D. "Are the dead taking over Facebook? A Big Data approach to the future of death online". In: *Big Data & Society* 6.1 (2019), p. 2053951719842540. DOI: 10.1177/2053951719842540. eprint: https://doi.org/10.1177/2053951719842540. URL: https://doi.org/10.1177/2053951719842540.