UMEÅ UNIVERSITY

Making Fairness Actionable

Julian Alfredo Mendez

Licentiate Thesis, December 2024
Department of Computing Science
Umeå University
Sweden

Department of Computing Science
Umeå University
SE-901 87 Umeå, Sweden

julian.mendez@cs.umu.se

Paper I, Andrea Aler Tubella, Flavia Barsotti, Rüya Gökhan Koçer, and Julian Alfredo Mendez
Ethical implications of fairness interventions: what might be hidden behind engineering choices?.
In *Ethics and Information Technology*, 2022. DOI:10.1007/s10676-022-09636-z
Paper II, Andrea Aler Tubella, Dimitri Coelho Mollo, Adam Dahlgren Lindström,
Hannah Devinney, Virginia Dignum, Petter Ericson, Anna Jonsson, Timotheus Kampik,
Tom Lenaerts, Julian Alfredo Mendez, Juan Carlos Nieves.
ACROCPoLis: A Descriptive Framework for Making Sense of Fairness.
In *FAccT '23: Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, 2023. DOI:10.1145/3593013.3594059
Paper III, Julian Alfredo Mendez, Timotheus Kampik, Andrea Aler Tubella, Virginia Dignum.
A Clearer View on Fairness: Visual and Formal Representations for Comparative Analysis.
In *14th Scandinavian Conference on Artificial Intelligence, SCAI*, 2024.
DOI:10.3384/ecp208013
Paper IV, Julian Alfredo Mendez.
Soda: An Object-Oriented Functional Language for Specifying Human-Centered Problems.
In *arXiv*, 2023. DOI:10.48550/arXiv.2310.01961
Paper V, Julian Alfredo Mendez, Timotheus Kampik.
Can Proof Assistants Verify Multi-Agent Systems?
In *Proceedings of the 21st European Conference on Multi-Agent Systems, EUMAS*, 2024.

ISBN 978-91-8070-534-9 (print)
ISBN 978-91-8070-535-6 (digital)
ISSN 0348-0542
UMINF 24.12

Electronic version available at https://umu.diva-portal.org.
Printed by Scandinavian Print Group, Umeå, 2024.
Title image by Mostphotos.

Abstract

The opaque nature of machine learning systems has raised concerns about whether these systems can guarantee fairness. In addition, ensuring fair decision-making requires that multiple perspectives of fairness be considered. Currently, there is no agreement on the definitions, the facilitation of shared interpretation is difficult, and there is a lack of a unified formal language to describe them. Current definitions are implicit in the operationalization of systems, making them difficult to compare. In this thesis, we discuss how to make fairness actionable, providing concrete tools for that. We provide not only conceptual elements to model and abstract problems of fairness, but also a technical framework and a description language.

Sammanfattning

Att göra rättvisa handlingsbart

Den opaka naturen hos maskininlärningssystem väcker oro kring om dessa system kan garantera rättvisa. Dessutom kräver rättvis beslutsfattande att flera perspektiv på rättvisa beaktas. För närvarande finns det ingen enighet kring definitionerna, vilket försvårar delad tolkning, och det saknas ett enhetligt formellt språk för att beskriva dem. Nuvarande definitioner är inbyggda i hur systemen används, vilket gör dem svåra att jämföra. I denna avhandling diskuterar vi hur rättvisa kan göras praktiskt tillämpbar och tillhandahåller konkreta verktyg för detta. Vi erbjuder både konceptuella element för att modellera och abstrahera rättviseproblem samt en teknisk ram och ett beskrivningsspråk.

Acknowledgements

I would like to thank my supervisor, Virginia Dignum, for her experience and her support. I would like to thank my first co-supervisor, Andrea Aler Tubella, for the many years she spent with me, helping me, advising me, inspiring me, and teaching me how to grow as an independent researcher. I would like to thank my second co-supervisor, Timotheus Kampik, for his advice, his vision, his trust, and his perseverance, guiding me to mature and strengthen my research and to reach the final phase and concretion of my work.

I would like to thank my reference person, Juan Carlos Nieves Sanchez, who has helped me and guided me during my PhD course, and supported me when things did not go as originally planned. I would like to thank Helena Lindgren, for her support, her advice, and her trust. I would like to thank Andreas Brännström and other colleagues and former members of the Responsible AI group, who have given me important feedback and constructive observations.

I would like to thank Frank Drewes, for his teaching on how to write scientific papers, and for his role as PhD study coordinator, and to thank Kai-Florian Richter, for his support and guidance during my PhD studies. I would like to thank Human Resources, especially Helena Hjelm, Annakarin Resoluth, and Ennie Boberg, for their important support, I would like to thank Tomas Forsman and Mattias Åsander, who were always ready to help, and I would like to thank Erik Elmroth and Lena Kallin Westin, who successfully directed the Department of Computing Science during the difficult times of the recent pandemic.

I would like to thank the Wallenberg AI, Autonomous Systems and Software Program (WASP) and the Department of Computing Science, for their economic support to make this research possible.

I would like to thank my collaboration partners around the world, especially Ricardo Oscar Rodriguez and Maria Vanina Martinez, for their important advice and feedback. I would like to thank my friends of Club Euclides, especially Matilde Lalin, Malena Español, and Gabriela Jeronimo, who gave me useful advice on how to grow as a researcher. I would like to thank Pablo Haramburu and Monica Miura, who were with me in difficult times. I would also like to thank the anonymous reviewers who contributed with their time to peer-review my work.

I would like to thank my mother Julia, who has been a role model of perseverance, and my father Luis, who not only encouraged me to start a PhD in

Umeå, but also closely followed how my research results evolved and their contribution to society. I would like to thank my brother Fernando, who checked the mathematics in my drafts, my wife Silvia, who spent long hours proofreading the grammar in my drafts, and to my two children, Sophia and Daniel, who are the most motivating inspiration to develop technology to live in a fairer world.

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

List of papers

This thesis is based on the following papers.

Paper I Andrea Aler Tubella, Flavia Barsotti, Rüya Gökhan Koçer, and Julian Alfredo Mendez. Ethical implications of fairness interventions: what might be hidden behind engineering choices?. Ethics and Information Technology, volume 24, issue 1, article 12, Springer 2022. DOI:10.1007/s10676-022-09636-z

Paper II Andrea Aler Tubella, Dimitri Coelho Mollo, Adam Dahlgren Lindström, Hannah Devinney, Virginia Dignum, Petter Ericson, Anna Jonsson, Timotheus Kampik, Tom Lenaerts, Julian Alfredo Mendez, and Juan Carlos Nieves. ACROCPoLis: A Descriptive Framework for Making Sense of Fairness. In *FAccT '23: Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 1014–1025, 2023. DOI:10.1145/3593013.3594059

Paper III Julian Alfredo Mendez, Timotheus Kampik, Andrea Aler Tubella, and Virginia Dignum. A Clearer View on Fairness: Visual and Formal Representation for Comparative Analysis. In Florian Westphal, Einav Peretz-Andersson, Maria Riveiro, Kerstin Bach, and Fredrik Heintz, editors, *14th Scandinavian Conference on Artificial Intelligence, SCAI 2024*, pages 112-120. Swedish Artificial Intelligence Society, June 2024. DOI:10.3384/ecp208013

Paper IV Julian Alfredo Mendez. Soda: An Object-Oriented Functional Language for Specifying Human-Centered Problems. arXiv. DOI:10.48550/arXiv.2310.01961

Paper V Julian Alfredo Mendez, Timotheus Kampik. Can Proof Assistants Verify Multi-Agent Systems? *Presented at the 21st European Conference on Multi-Agent Systems, EUMAS 2024*[1].

[1] The submission will undergo another review cycle before inclusion in the post-proceedings.

Contributions

- Paper I: I am one of the four equally contributing authors. I participated in all the discussions, participated in the overall writing, and my main contribution was in the design and implementation of the testing software that proves the point of the paper.
- Paper II: I am one of many equally contributing authors. I was in the initial group that started the article and presented my ongoing research on formalizing fairness at our initial meeting. I contributed to the conceptual framework starting from different directions (e.g. logic-oriented, engineering-oriented). I also contributed to the example presented in the paper and the overall writing.
- Paper III: I am the main author in the development of the concept and execution of paper under supervision. This paper evolved from my research prior to Paper II, but combined its results, together with those in Paper IV.
- Paper IV: I am the sole author, especially in the design and conception, and the sole contributor to the underlying open source software, without substantial supervision.
- Paper V: I am the main author in the writing and the developer of the implementation and its formal proofs.

Contents

Chapter 1
Introduction

Artificial intelligence (AI) is a field of study whose results have been applied to automate decision-making processes, such as those related to job applications, credit scoring, and criminal justice. AI-based technology can be found in web search engines, recommendation systems, human speech processing, self-driving cars, generative tools, and strategy games. However, society has raised several concerns regarding AI, including issues of bias and fairness, as AI systems can inadvertently perpetuate existing prejudices present in their training data. There are also privacy concerns, as AI often requires large amounts of personal data to function effectively, and the lack of transparency in AI decision-making processes can make it difficult to understand and challenge outcomes. The potential for job displacement due to automation, the ethical use in surveillance and military applications, and the concerns about long-term impact of AI on human autonomy are also topics of concern, which go beyond the specific application in actionable fairness.

The prevalence of AI applications has significant implications [18] as the computational capabilities of machines progress. Fairness is hard to define, and even once having a definition of fairness, it is difficult to apply to different contexts. Making fairness actionable helps satisfy the need for society to assimilate and manage the growth of AI-based technologies, with a paramount emphasis on maintaining ethical values and responsibilities [19].

This thesis explores the challenges and limitations that exist when making fairness actionable. In particular, the four papers presented in this thesis are aimed at answering the following research questions:

1. What challenges are encountered when attempting to operationalize fairness? (Paper I and II)

2. What methods can be employed to make the formalization of fairness usable in real applications? (Paper II and III)

3. How can we move from a conceptual model of fairness to formalizations and instantiations? (Paper III, IV, and V).

To answer these questions, we analyze the characteristics of the challenges, tools available, our proposed tools, and limitations of our techniques.

1.1 Responsible AI

With the rise of AI-based technologies, responsible AI is a topic that demands increasing attention. AI is found in medicine, finance, law, security, and entertainment, to mention some of the fields that impact society. Image recognition can be used to detect cancerous cells in digital images, or for face recognition, for security purposes. Prediction can be used for the assessment of financial risk or criminal risk. Conversational bots can be used to provide relevant information with respect to specific questions, or for leisure. However, questions like "Should AI be applied for that purpose?", or even, "Should society develop that kind of technology?" are questions that society-wise are difficult to answer. These issues highlight the relevance of a functionally responsible use of AI technology.

We can define responsible AI as the practice of creating, using, and deploying AI and machine learning technologies in an ethical manner. It is a response to the growing influence of AI in various aspects of society, from healthcare and finance to law enforcement and transportation. It involves a set of principles and regulations intended to ensure that AI systems are designed and operated with a satisfactory understanding of their effects on people, society, and the environment.

Responsible AI comprises a number of interrelated concepts. One of the most visible of them is the concept of **fairness**. Fairness is mentioned directly or indirectly as one of the core principles for ethical AI. A comprehensive evaluation of fairness approaches is discussed in [5], which provides practical guidelines to mitigate bias in AI systems. This study is expanded in [55] by addressing ethical, legal, and social challenges, focusing on the interdisciplinary nature of fairness in AI. Further mitigation strategies are deepened in [23] with focus in fairness. In [26], a criticism of the existing AI ethics guidelines is given, and also recommendations for improvements to enhance their effectiveness. The United States Department of Defense (DoD) manifests the principles to consider when developing and deploying an AI system [17], and lists that AI systems should be responsible, **equitable**, traceable, reliable, and governable. Jakesch et al., in their review of worldwide principles, mention principles such as transparency, fairness, safety, accountability, privacy, autonomy, and performance [33]. In Section 1.2, we discuss how fairness in AI can be described.

1.2 Fairness in AI

The importance of fairness in machine learning and AI systems is widely accepted. There is an ongoing academic discussion about how to decide on the appropriate definition of fairness and how to address the conflict between fairness and model effectiveness. Fairness is often considered a property related to the allocation of resources in formal systems and is frequently studied in the context of distributed systems [24, 38]. Fairness is a key element in the development of ethical models in various contexts, such as healthcare, law, and

banking. This becomes more relevant considering the increasing awareness of how bias can be incorporated and amplified in AI models [44, 45]. Evaluating fairness from a modeling point of view requires understanding of how to identify and quantify the extent of unwanted bias, which may lead to prejudice and ultimately to discrimination.

Ensuring fairness given an already biased AI model or data set requires mitigating bias. Bias can be introduced for many reasons [44], such as data collection methods, feature measurement, and considered evaluation benchmarks. We can mention three methods to mitigate bias at different points in the processing of the data. *Pre-processing* involves modifying the training data before it is used to train the model [16]. The goal is to reduce or eliminate bias in the data itself [6]. *In-processing* compensates for bias during its processing, and can be used during model training by including changes to the objective function or by enforcing a constraint [6, 7]. *Post-processing* adjusts the results after the data have been processed [16], and the labels are reassigned based on a function during the post-processing phase [6, 7].

One of the relevant problems studied in the literature is fairness criteria for credit scoring. It is a very tangible use case where the quantification of attributes plays a decisive role, it is highly automated, and it can be algorithmically determined to some extent. The authors in [40] list algorithmic approaches to include fairness objectives in the development process of the machine learning model. They empirically compare various fairness processors in a profit-oriented credit scoring context using real-world data. Their goal is to provide a comprehensive overview and organization of fairness criteria and fairness processors that have recently been created and to assess their suitability for credit scoring through empirical testing. They include an implementation with the experiments they ran [41, 39]. The authors examine both in-processing and post-processing techniques that prioritize group fairness and individual fairness, respectively. The quantitative nature of the credit score is a relevant example of mechanical techniques to ensure fairness, as intended to achieve with the research questions.

1.3 Formalizing Fairness

Formalizing fairness can lead to a more transparent understanding of how to obtain fairer situations, which can benefit the individuals involved and the groups they represent. Although it is challenging to make it actionable, formalizing fairness and automating fairness verification [1, 2] is relevant, as is finding a suitable definition of fairness in mathematical terms. Several quantifiable definitions have been discussed [20, 27, 34, 37], encompassing several legal, philosophical, and social perspectives. Different interpretations and implementations of fairness may harm the groups they try to protect [15] or do not consider intersectionality, when an individual belongs to multiple underprivileged groups [37].

Some relevant definitions and interpretations of fairness must be considered, especially the distinction between *individual fairness* and *group fairness*. The former protects each individual by aiming at treating them similarly independently of the individual characteristics. The latter protects underprivileged groups with sensitive characteristics by aiming to have *parity on average* [14].

Example 1 (Group fairness vs. individual fairness)**.** Let us recap the definitions of individual fairness and group fairness. At an individual level, fairness can be defined as if two individuals are similar with respect to a particular property, they should receive similar outcomes. At a group level, fairness demands the existence of parity between different protected groups. These groups are defined by protected attributes, such as gender, skin color, and ethnicity, and statistical parity aims at ensuring equal outcomes for members of different protected groups.

These two concepts can be in conflict. For example, if two individuals with similar properties receive significantly different outcomes because they are members of different protected groups [9]. Metrics like equality of odds and equality of opportunity can be used to manage these conflicts. These metrics aim to ensure that equally qualified individuals across different groups are equally likely to receive the same outcome.

To put this into context, let us consider two individuals, Alice and Bob, who apply for a loan. Let us suppose that they have similar credit scores, incomes, and employment histories. In the context of individual fairness, a fair model should approve a loan for Alice if and only if the model approves the loan for Bob.

For group fairness, let us consider a property that should be independent of creditworthiness of an individual, for example, the year of birth. Let Group A and Group B be two groups of individuals born in even and odd years, respectively. If a model is fair with respect to group fairness and if 60 % of the applicants in Group A get their loan approved, then approximately 60 % of the applicants in Group B should get their loan approved.

In Paper III, we present an effective tool to check if instances comply with individual or group fairness, according to their metrics. Thus, the criteria can be expressed as configurations of a framework for instance checking.

A review of how fairness is defined in the field of machine learning for the purpose of prediction is discussed in [25]. The authors compare these definitions with the concepts of distributive justice in the social sciences. They also provide theoretical and empirical critiques of the ideas of fairness in the social sciences literature and assess their suitability for certain domains.

Game-theoretic approaches to formalizing fairness can be seen in [43], where the authors discuss how to obtain a fair exchange. The authors define a fair exchange when all parties get exactly the good they requested, or no good has been transferred at the end of the exchange. The cost of a trusted third party to guarantee fairness must also be included in the transaction. We present a different approach for recommendation systems in Example 2.

Example 2 (Fairness and recommendation systems)**.** In recommendation systems, a recommendation is given to the user as a suggestion to follow. In streaming services, news providers, and online advertising, recommendation systems use feedback to evaluate how successful a recommendation was. This framework works similarly to the multi-armed bandit (MAB) framework, where each arm is a possible choice.

A one-armed bandit is another name for a slot machine, a gambling machine that has a single handle and "steals" money from losing players. In probability theory and machine learning, the multi-armed bandit (MAB) problem is a problem in which one of multiple fixed choices are selected, the properties of each choice are only partially known at the beginning, and they may be better understood later. A characteristic of this problem is that choosing an arm does not affect the properties of the other arms. The notion of regret for the MAB framework is discussed in [4], and it can be applied to similar scenarios.

Let us assume that for options a, b, c in a given state e, there are different probabilities of an equally valuable reward for a, b, c. This can be expressed by an *average utility* u, for example $u(a) = 100$, $u(b) = 50$, and $u(c) = 0$.

In that case, let us consider the following strategies:

- "Optimal": it always obtains a reward by choosing the a option;
- "Fair": it chooses the reward considering frequency "For an interval I, eventually b and c, otherwise always a;
- "Fairness in proportion": "a twice as often as b, never c".

Let us specify the criteria above, given a finite set of options A, a finite sequence of events (e_i), $1 \leq i \leq n$, which we call an interval of events I, $e_i \in A$, a utility function, $u : A \to \mathbb{R}$, and $S : \mathcal{P}(I) \to \mathbb{R}$ defined as

$$S(I) = \sum_{1 \leq i \leq n} u(e_i)$$

For the "Optimal" criterion, the choice would always be to maximize u. We say that the criterion holds for I if and only if for any I', $S(I) \geq S(I')$. Notice that if more than one option is available, any option works.

This criterion can be checked locally, and each state can be checked independently of the other ones. We express it by saying that it holds for I if and only if for each i, $1 \leq i \leq n$, $e_i \in \{a \mid \forall x \in A : u(a) \geq u(x)\}$.

For the "Fair" criterion, we need a global check to ensure the compliance of the interval, and a local verification is not enough. We say that the criterion holds if and only if:

i. at least one of the options is chosen along the whole finite sequence I, and

ii. the utility is maximized, considering the previous restriction

An interval $I = (e_i)$, $1 \leq i \leq n$, satisfies the first part if

$$A \subseteq \{e_i \mid 1 \leq i \leq n\} \tag{1.1}$$

For the "Fairness in proportion" criterion, a local verification is also not enough. The idea behind this is that the frequency is proportionally related to the utility. We define the function $L : \mathcal{P}(I) \to \mathbb{N}_0$ as the length of an interval I, and $C : \mathcal{P}(I) \times A \to \mathbb{N}_0$, which counts the number of occurrences for each option $a \in A$, and each interval I. We work with a tolerance $\tau \in \mathbb{R}$ to define an approximate equality $\approx_\tau$, which says that for $x, y \in \mathbb{R}$, $x \approx_\tau y$ if and only if $\mid x - y \mid < \tau$. We say that the criterion holds for I if and only if

$$\frac{C(I,a)}{L(I)} \approx_\tau \frac{u(a)}{\sum_{b \in A} u(b)} \tag{1.2}$$

In Example 2, three strategies are discussed which would produce different outcomes, according to what is defined as fair. The definition of fairness is a choice to determine what is fair for the given context, and even formal abstractions can have many definitions of fairness that can be applied in different application areas. In the following, we discuss the challenges of implementing varied definitions of fairness.

1.4 Implementing Fairness

A document written in natural language can describe a rigorous definition of fairness, but implementing such definition requires technical details to be executed in real-world scenarios.

The implementation of fairness constraints plays a crucial role in operationalizing fairness for machine learning and AI systems. Fairness constraints refer to the specific requirements and criteria that algorithms and models must fulfill to ensure equitable outcomes for different demographic groups and individuals. These constraints can be integrated into the design of AI systems, where they drive the learning process to mitigate bias and discrimination. Implementing such constraints requires a balance between achieving fairness objectives and preserving the performance and utility of the model. The implementation of effective fairness constraints is essential for building AI systems that have ethical and legal standards that contribute to responsible AI systems.

Different fairness constraints from the existing literature have been adapted to economics markets [48]. The authors also analyze the effects of these constraints on those who benefit and those who are disadvantaged, as well as how much properties such as Pareto optimality, envy-freeness, and incentive compatibility are maintained.

Pareto optimality, also known as Pareto efficiency, is a concept in economics that describes a state where resources are allocated in such a way that it is impossible to make an individual better off without making at least another

individual worse off [53]. *Envy-freeness* is a criterion for fair division. It states that when resources are allocated between people with equal rights, each person should receive a share that they perceive to be at least as good as the share received by any other person [12, 8]. *Incentive compatibility* is a concept in game theory and economics that refers to a state where all participants can achieve the best outcome for themselves simply by selecting its natural action, which is the action that would maximize the performance of the participant if there were no competition [54].

Example 3 (Envy-freeness vs. equality)**.** Let us discuss a tangible example of envy-freeness. For that, we need to distribute a resource between different actors, with possibly different preferences for the possible different parts of the resource. Let us say that there is a small Neapolitan ice cream cake (the resource) that is to be completely distributed between Alice, Bob, and Charlie (the actors). The cake is made in three equally thick layers of vanilla, strawberry, and chocolate ice cream. A possible equal distribution of the cake would be that each of the actors received a third containing the same amount of the three flavors. This distribution would be equal, and it would also be envy-free, because all pieces are equivalent.

Let us assume that Alice, Bob, and Charlie express that they would eat only their favorite flavor, which is chocolate, vanilla, and strawberry, respectively. The three of them would actually eat a third of what they receive, and the above distribution is still envy-free. However, if the pieces contained different amounts of the three flavors, the actors may prefer some pieces instead of other ones. In this case, the distribution could be equal in quantity, but not envy-free. Conversely, if the layers of the cake were of different thicknesses, distributing only their favorite flavor to each actor would be envy-free, but it would not represent an equal distribution.

The framework presented in Paper III allows to configure different scenarios where the actors' preferences are taken into consideration when evaluating a fair distribution. In the framework, it is also possible to combine different preferences. Continuing with the example, the preference function could be more complex. For example, each actor could prefer to eat an amount of their favorite flavor only if what they would receive is not less than half what they would receive from a different flavor.

We can compare both scenarios by looking at their utility function. Let us assume that we are given a set of actors Ac and a set of resources R. In the case of equality, the utility function does not depend on the actor:

$$u : R \to \mathbb{R}$$

In the case of envy-freeness, we consider a utility function that depends on the actor. This utility function represents how good a distribution is with respect to the actor. The function can be defined as:

$$u : Ac \times R \to \mathbb{R}$$

The analysis in [48] notices that some constraints provide limited assurance in terms of who is advantaged or disadvantaged by their implementation. The authors explain how to algorithmically find equilibria for Fisher markets, which are markets where budget-constrained buyers compete for items that are scarce [21].

The automation of fairness verification is discussed in Paper III, where we build a pipeline that uses Pearson's correlation to detect false positives in a trained model. A similar construction can be designed to automatically detect deviations from the formulas provided by [48] in an online environment at runtime.

A different approach is discussed in [42], where the authors propose an operationalization of individual fairness that does not require a human definition of the distance metric. Instead of relying on traditional methods, the authors suggest new ways to acquire and use information about individuals who are equally worthy of attention to reduce the gap between social groups. They model this knowledge as a fairness graph and develop a unified Pairwise Fair Representation (PFR) of the data that captures both the data-driven similarities between people and the pairwise side information in the fairness graph.

The specifics and contexts where an AI system is deployed may require handling mutually incompatible definitions of fairness. This is considered in [49], where the authors discuss the AI fairness framework developed by LinkedIn, which distinguishes between equal treatment and equitable product expectations, rather than forcing a compromise between them. The article provides instructions for implementing equal AI treatment and is supported by principles demonstrated in a case study, examining the gender breakdown among different groups.

The authors' conclusions in [49] agree with our conclusions presented in Paper I, where we state that addressing AI fairness issues is complicated by the question of whether it is acceptable to use demographic information to reduce AI bias, and how to use it accurately. This is a consequence of the fact that privacy and fairness compete, one at the cost of the other [13]. The authors in [49] remark that it is essential to investigate alternative mitigation methods that do not require the use of demographic information, as that should be the preferred option. However, failing to address bias can have serious repercussions in the real world, and when making a decision on how to mitigate bias, one must consider the cost of not taking action.

The difficulties in measuring fairness are highlighted in [31], where the authors remark that computer systems often have unobservable properties that need to be inferred from measurements of observable properties. They attribute many of the harms associated with fairness in computational systems to discrepancies between the system and the environment. They demonstrate how some of these harms could be foreseen and, in some cases, reduced if a system is examined through the perspective of measurement modeling. They propose that the current discussions concerning fairness definitions are not actually about different operationalizations, but rather about different theoretical

interpretations of fairness. They show how measurement modeling can be used to get to the heart of these debates. This agrees with our position in Paper III, where we propose using a formal representation to agree on the understanding of fairness for a given scenario, rather than just a specific operationalization.

1.5 The Need for Meta-Models of Fairness

Determining whether something is actually fair requires a clear definition of what could be considered fair. Fairness is meaningful only related to the context in which it is considered, and in order to evaluate a definition of fairness, it is necessary to determine the parts involved. That was our purpose in Paper II, in a generic approach, and in Paper III, in a more technical approach, where we also provide a metamodel.

Although there are multiple interpretations of fairness in the literature, there has been limited research to link them together and compare them. This research gap presents an opportunity for our technique, which facilitates formalization and comparison. The prerequisites, causes, and goals to formalize human principles for AI are discussed in [32]. The authors base their studies on analyzing the concept of trust in AI, and they point out the negative consequences of excessive trust in AI. They present a model of trust based on trust between people (interpersonal trust), considering the vulnerability of the user and the ability to anticipate the impact of an AI model decision. If an AI model is highly accurate and assuming predictability works in favor of accuracy, the user can trust it. For example, an AI-based triage and diagnosis system, as proposed in [51], should have these properties. In contrast, if an AI system cannot do harm, its predictability becomes less relevant.

A formalization of 'contractual trust' and 'trustworthiness' is also included in [32], and it helps to define 'warranted' and 'unwarranted' trust. The authors, as in our formalization for fairness in Paper II and Paper III, have a formalization that is open to different configurations, and establish a connection between trust and explainable AI (XAI) using such formalization. Examples of critical situations where AI needs to be especially scrutinized include evading shutdown, hacking computer systems, running many AI copies, acquiring computation, attracting earnings and investments, hiring or manipulating human assistants, conducting AI research and programming, persuading and lobbying, hiding unwanted behavior, appearing aligned, and escaping containment, used in various domains like research and development, manufacturing and robotics, and autonomous weaponry.

Chapter 2

Discussion of the Included Papers in the Context of the Thesis

In this chapter, we discuss how the included papers are interrelated. We start by presenting a layered framework in Figure 2 which we use in the automation process to make fairness actionable.

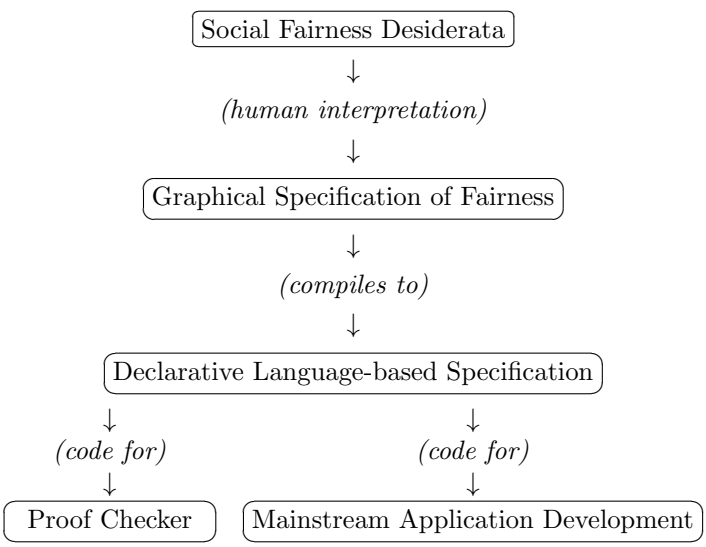


Figure 2.1: This depicts a layered framework that we propose to make fairness actionable, and the relation between these layers.

The top layer, *Social Fairness Desiderata*, refers to the desired principles or criteria that aim to ensure fairness in a social system, especially in the context of decision-making to achieve outcomes that are morally appropriate for the

individuals involved. In the realm of artificial intelligence, fairness desiderata might include avoiding population separation, or ensuring that interventions do not disproportionately benefit or harm specific groups [52]. These principles are mainly discussed in Paper I and Paper II.

Through human interpretation of the desiderata, it is possible to reach the second layer, the Graphical Specification of Fairness. This layer aims at being understood by a large population, even without a technical background. The schematic connection of conceptual boxes is conducive to a train of thought that describes formal rules. This graphical representation is sketched in Paper II and deeper developed in Paper III.

The graphical specification can be directly compiled to a Declarative Language-based Specification, which is the third layer. This layer is expected to be understood by a person with technical background, like a software developer or a data scientist. The declarative nature preserves the prioritization of clarity and readability over efficiency, providing higher reusability and maintainability. This is partially discussed in Paper III and mainly developed in Paper IV and Paper V.

The fourth layer is the layer closest to the machine in Figure 2. It is realized by two main outcomes: the formally verifiable pieces of code (Proof Checker) and the ready-to-integrate piece of code (Mainstream Application Development). As we discussed in Paper IV, this layer is mostly populated by a translation to Lean 4 as proof checker, and Scala 3 as mainstream software. Scala has the additional advantage of compiling to and accessing the Java Virtual Machine, and connecting to the vastly populated Java ecosystem.

2.1 Paper I

The paper is motivated by the need to ensure fairness in machine learning models. It discusses the difficulties of finding appropriate fairness definitions. It proposes that seemingly technical decisions, such as the choice of debiasing technique, encompass ethical implications. Thus, they have a different effect on different people, and therefore there are underlying ethical implications. This proposition is supported by experiments using state-of-the-art tools.

We used the German Credit Dataset [28] as a starting point, which contains personal data that is used to assess whether an individual can get a loan or not. Since we were unable to find a significant bias in that dataset, we introduced synthetic bias and created a slightly modified version of this dataset with controlled bias.

After adding bias, we used Fairlearn [56] and AIF 360 [30] to measure and mitigate data bias. The experiments [3] proved our point and we showed how adversarial debiasing, a technique of in-processing debiasing, and rejection option classification, a technique of post-processing debiasing, affected different data points, which in turn affect different individuals.

We can mention some of the debiasing techniques available in the literature[1]:

- **Adversarial debiasing** [57] is an in-processing technique that trains a classifier to maximize prediction accuracy and at the same time reduce an adversary's ability to determine protected attributes from the predictions.
- **Disparate impact remover** [22] is a preprocessing technique that adjusts featured values and increases group fairness while preserving rank-order within each group.
- **Prejudice remover** [36] is an in-processing technique that adds a discrimination-aware regularization component into the learning goal.
- **Reject option classification** [35] is a postprocessing technique that gives favorable outcomes to unpriviliged groups and unfavorable outcomes to priviliged groups within a confidence band around the decision boundary with the highest uncertainty.
- **Reweighing** [35] is a preprocessing technique that weights the examples in each ⟨ group, label ⟩ pairing differently to guarantee fairness before the classification process.

2.2 Paper II

This paper presents the ACROCPoLis framework to represent allocation with an emphasis on fairness aspects. Fairness is crucial for responsible AI systems, and a large number of frameworks and formal notions of algorithmic fairness exist for that. The framework is based on a conceptual analysis that is especially useful for comparing analogous situations, highlighting differences in various scenarios, and improving alignment with human values. Paper II introduces a grammar for fairness assessments by providing a shared vocabulary to make explicit the factors relevant within fairness statements, describing different situations and procedures, including the relevant relationships between them.

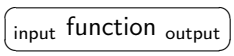
Making fairness actionable with the framework is not an immediate task. While the versatility of the entities we have introduced makes the framework highly flexible, its practical application in real world AI application may not be straightforward. The fine-tuning application of this model is left to a human expert, who interprets the meaning of the entities for a given scenario.

2.3 Paper III

This paper presents a notation and technology, the Tiles framework [46], that aims at facilitating the proper configuration of a definition of fairness. Tiles is a

[1] https://aif360.readthedocs.io/en/stable/modules/algorithms.html

highly configurable system for modeling a definition based on interconnectable blocks. These blocks are called *tiles*, and each tile has an identifier, an input, and an output, notated as follows:



The visual presentation of its blocks enhances readability, and each tile is functionally encoded in concise and maintainable code fragments. The main purpose of this notation is to have a direct presentation of a definition of fairness, to be able to compare different definitions of fairness, and to keep, nevertheless, a rigorous notation.

Although a variety of tiles can be shared in different fairness scenarios, new tiles need to be customized according to general guidelines, called the AcROMAgAt framework. This framework denotes the main entities participating in a fairness scenario, and determine the conditions to create and connect tiles.

The AcROMAgAt framework is based on a tuple which components are actors (Ac), resources (R), the outcome of a distribution (O), a set of measures (M), aggregation functions (Ag), and attribute functions (At), i.e.

$$F_c = \langle Ac, R, O, M, Ag, At \rangle.$$

A significant feature of Tiles is that each tile is a functional piece of code, rather than a procedural one. When describing rules, functional definitions are descriptive of *what they are* instead of *how they are computed*.

A limitation of Tiles is that there are no established tools for both generating and linking these tiles at the moment. Therefore, an expert needs the involvement of a developer to build the individual tiles and establish their connections.

2.4 Paper IV

Current technology has achieved a degree of complexity that creates a gap between humans and machines. To bridge this gap, it is necessary to provide tools for the transparency and reliability of systems. This is paramount when developing and using technologies that have relevant ethical consequences. A language that helps to bridge this gap needs to be a compromise between providing a high-level vision of a system that can let humans understand how a system works, and with a low-level precision provided by verified systems. In essence, this language builds a connection between what a system is supposed to do and what it is actually doing.

This paper presents Soda, a simplified object-oriented functional specification language that captures many of the virtues of the most popular programming languages and prevents many of their defects. By design, this language allows the creation of small functions, which are collected in classes, which in turn are grouped in packages. It has a small number of operands and basic types.

This language is directed to be easily understood, rather than easily programmed. Soda requires a certain level of maturity with respect to both object-oriented or modular programming and functional programming. The lack of proficiency in functional programming can pose challenges when encoding algorithms that involve cycles and mutability, such as the flood fill algorithm. Similarly, without a grasp of object-oriented or modular programming, classes may become cluttered with an abundance of functions or an excessive number of parameters, some of which could be more appropriately defined as constants within the classes.

Although the language itself safeguards against typical human errors, such as invoking undefined functions or providing incorrect data types, it remains vulnerable to deliberate misnaming of functions, as the naming convention serves as their primary reference. Furthermore, since no specific tools are provided to develop in Soda, step-by-step debugging needs to be done through its translation to Scala. On the one hand, Paper IV is technically more specific than Paper III, since it discusses in more detail and rigor the operationalization of the upper layer discussed in Paper III. On the other hand, Paper V applies concepts of Paper IV in an agent-based environment, but aiming at an execution incorporating formal verification.

2.5 Paper V

Transparency and resilience are valuable properties to empower humans in the relation to the technology they use, in particular, when this technology affects their values. The frameworks discussed in Paper II and Paper III, and the language discussed in Paper IV contribute to this purpose. Paper V adds a proof of concept of how transparency and resilience can be applied in practice.

Paper V presents the Soda language applied to verification of multi-agent systems (MAS). It shows how Soda can implement multi-agent systems, integrated into the mainstream software ecosystem, and including state-of-the-art proof checkers to verify pieces of code. The paper provides a brief introduction to Soda and its capabilities, as well as a simple example in which its interaction protocols are designed and verified with Soda.

Some of the important aspects to highlight are that verification is a highly challenging task. Every detail, regardless of how small it is, needs to be considered in order to ensure correctness of a piece of code. For example, accessing an element in a sequence a of size n at a given position i, notated $a(i)$ in Scala, or $a[i]$ in many imperative languages like Java, needs to ensure that the following holds $0 \leq i < n$. We envision this as a first step towards a complex practical tool. The usability of Soda's formal verification capabilities is subject to a suitable integration into reusable modules that include Lean proofs for even the simplest properties.

Chapter 3
Future Work

The following steps are directed to improve precision and demonstrate usefulness. We plan to apply the AcROMAgAt metamodel to additional fairness definitions to show that it is expressive enough and that it can still instantiate a wide variety of definitions. We plan to apply it to the YouTube algorithm of Paper II.

We would like to explore more real-world fairness definitions, placing more emphasis on group fairness or individual fairness, and to conduct further research on representing group fairness and individual fairness at a meta-level. To prove applicability on real-world cases, case studies would assess the practical relevance of AcROMAgAt. Therefore, we can apply AcROMAgAt to a comparison of conflicting fairness definitions in a real-world context. Our tools provide formal elements that can be used to verify properties or force constraints based on fairness definitions. For example, we could ensure that all available resources are allocated or that only available resources are allocated. We could formalize such properties and prove that they derive from the respective definitions of fairness. We plan to provide further advanced support for implementing and operationalizing metamodels of fairness, as well as for automated reasoning about fairness models.

We would like to continue the development of Tiles. The framework presented in Paper III is a proof-of-concept of how to compare different definitions of fairness. The approach of graphical boxes and diagrams gives a fast and accurate impression of what decisions made by an algorithm look like. Nevertheless, these blocks are constrained by previous designs.

Most of the diagrams we present for child care subsidy contain instances where actors are considered in a sequential manner. In other diagrams, streams represented by the connections between the blocks are easily split and zipped back, because the split produces streams of the same length. However, applying filters, like in the case of single guardians, may produce branches of different lengths. We would like to explore if connecting branches of different lengths can be done with another approach.

Tiles brings a transparent representation for humans, and we would like to explore deeper the potentials of formally proving the snippets implementing Tiles. Especially, to elicit patterns and structures that frequently occur in different blocks. Tiles has snippets programmed in Soda, designed to ex-

press directly their behavior in a clear and concise way. Formal verification of programs is very hard, but we believe that small-sized snippets can work as excellent examples to apply it.

The modular concept of Tiles can be applied to multi-agent systems, where problems can be solved by communicating multiple interacting intelligent agents. Agents can connect, for example, to central orchestration system written in Soda, where some of its pieces of code are formally verified.

The application domain of verified multi-agent systems is broad, and include diverse fields like economics and robotics. We would like to investigate how verified programs are being used or can be improved in the domain of online trading, and in particular, what challenges and opportunities exist when using Soda. As shown in Paper V, the possibilities are numerous, and we would like to have a web-based server with proven pieces of code, that can process online requests, and we could partially or completely verify the trading protocol.

Different programming languages have been used for multi-agent systems, and some of them are included in the Prolog family. We can mention AgentSpeak [50, 10] and Jason [11, 29], for example. Prolog, which has been applied for artificial intelligence, automated theorem proving, and computational linguistics, is widely used in academic research. Multiple languages have been designed based on Prolog, like λ-Prolog [47] which has polymorphic typing. We want to explore the proximity of Soda to the Prolog family, and evaluate how feasible is to use Soda in contexts where Prolog is used today.

The advent of Soda, a new programming language that can be compiled for Scala 3 and Lean 4, opens up a world of possibilities. The ability to have efficient code and prove code snippets brings a new level of rigor and precision to the field. The visual framework Tiles that can be compiled into Soda opens a promising view on how we approach coding. We can think of tools to produce code that is not only efficient and provable but also visually intuitive, bridging the gap between human interpretation and machine execution. This could lead to a significant advancement in software engineering, enabling us to have more accessible and reliable software. The next steps are aimed to: *i)* mature the provided tools so that they can be realistically applied to real-world problems and *ii)* provide a proof-of-concept of real-world applicability.

References

[1] A. Albarghouthi, L. D'Antoni, S. Drews, and A. V. Nori. Fairsquare: Probabilistic verification of program fairness. *Proc. ACM Program. Lang.*, 1(OOPSLA), Oct 2017. https://doi.org/10.1145/3133904.

[2] A. Albarghouthi and S. Vinitsky. Fairness-aware programming. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 211–219, New York, NY, USA, 2019. Association for Computing Machinery. https://doi.org/10.1145/3287560.3287588.

[3] A. Aler Tubella, F. Barsotti, R. G. Koçer, and J. A. Mendez. Ethical implications of fairness interventions: what might be hidden behind engineering choices?, 2022. https://gitlab.com/ing-umea/eit-ethical-implications.

[4] S. Barman, A. Khan, A. Maiti, and A. Sawarni. Fairness and welfare quantification for regret in multi-armed bandits. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):6762–6769, 6 2023. https://ojs.aaai.org/index.php/AAAI/article/view/25829.

[5] A. Bateni, M. C. Chan, and R. Eitel-Porter. Ai fairness: from principles to practice. *arXiv preprint arXiv:2207.09833*, 2022. https://arxiv.org/abs/2207.09833.

[6] R. K. E. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, S. Nagar, K. N. Ramamurthy, J. T. Richards, D. Saha, P. Sattigeri, M. Singh, K. R. Varshney, and Y. Zhang. AI fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *CoRR*, abs/1810.01943, 2018.

[7] R. Berk, H. Heidari, S. Jabbari, M. Joseph, M. Kearns, J. Morgenstern, S. Neel, and A. Roth. A convex framework for fair regression, 2017.

[8] H. S. Bin-Obaid and T. B. Trafalis. Fairness in resource allocation: Foundation and applications. In I. Bychkov, V. A. Kalyagin, P. M. Pardalos, and O. Prokopyev, editors, *Network Algorithms, Data Mining, and Applications*, pages 3–18, Cham, 2020. Springer International Publishing.

[9] R. Binns. On the apparent conflict between individual and group fairness, December 2019.

[10] R. H. Bordini, A. L. C. Bazzan, R. de O. Jannone, D. M. Basso, R. M. Vicari, and V. R. Lesser. Agentspeak(xl): efficient intention selection in bdi agents via decision-theoretic task scheduling. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3*, AAMAS '02, page 1294–1302, New York, NY, USA, 2002. Association for Computing Machinery. https://doi.org/10.1145/545056.545122.

[11] R. H. Bordini and J. F. Hübner. Jason: A java-based interpreter for an extended version of agentspeak. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 28–29. IEEE Computer Society, 2005. https://www.emse.fr/~boissier/enseignement/maop14/DOC/jason/Jason.pdf.

[12] M. Chakraborty, A. Igarashi, W. Suksompong, and Y. Zick. Weighted envy-freeness in indivisible item allocation. *ACM Trans. Econ. Comput.*, 9(3), Aug 2021.

[13] H. Chen, T. Zhu, T. Zhang, W. Zhou, and P. S. Yu. Privacy and fairness in federated learning: On the perspective of tradeoff. *ACM Comput. Surv.*, 56(2), 9 2023.

[14] A. Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, 5(2):153–163, 2017. PMID: 28632438.

[15] S. Corbett-Davies and S. Goel. The measure and mismeasure of fairness: A critical review of fair machine learning. *CoRR*, abs/1808.00023, 2018.

[16] B. d'Alessandro, C. O'Neil, and T. LaGatta. Conscientious classification: A data scientist's guide to discrimination-aware classification. *Big Data*, 5(2):120–134, Jun 2017. https://doi.org/10.1089%2Fbig.2016.0048.

[17] U. S. D. o. D. D. Defense Innovarion Board. Ai principles: Recommendations on the ethical use of artificial intelligence by the department of defense, 2019. https://media.defense.gov/2019/Oct/31/2002204459/-1/-1/0/DIB_AI_principles%20Supporting%20Document.pdf.

[18] V. Dignum. *Responsible artificial intelligence: how to develop and use AI in a responsible way*, volume 2156. Springer, 2019.

[19] V. Dignum, M. Baldoni, C. Baroglio, M. Caon, R. Chatila, L. Dennis, G. Génova, G. Haim, M. S. Kließ, M. Lopez-Sanchez, R. Micalizio,

J. Pavón, M. Slavkovik, M. Smakman, M. van Steenbergen, S. Tedeschi, L. van der Torre, S. Villata, and T. de Wildt. Ethics by design: Necessity or curse? In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, page 60–66, New York, NY, USA, 2018. Association for Computing Machinery. https://doi.org/10.1145/3278721.3278745.

[20] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 214–226, New York, NY, USA, 2012. Association for Computing Machinery. https://doi.org/10.1145/2090236.2090255.

[21] E. Eisenberg and D. Gale. Consensus of subjective probabilities: The pari-mutuel method. *The Annals of Mathematical Statistics*, 30(1):165–168, 1959.

[22] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268, 2015. https://doi.org/10.1145/2783258.2783311.

[23] E. Ferrara. Fairness and bias in artificial intelligence: A brief survey of sources, impacts, and mitigation strategies. *Sci*, 6(1), 2024. https://doi.org/10.3390/sci6010003.

[24] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '80, page 163–173, New York, NY, USA, 1980. Association for Computing Machinery. https://doi.org/10.1145/567446.567462.

[25] P. Gajane and M. Pechenizkiy. On formalizing fairness in prediction with machine learning, 2018.

[26] T. Hagendorff. The ethics of ai ethics: An evaluation of guidelines. *Minds and Machines*, 30:99–120, 2020. https://doi.org/10.1007/s11023-020-09517-8.

[27] M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, pages 3323–3331, 2016.

[28] H. Hofmann. Statlog (german credit data), 1994. https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data.

[29] J. F. Hübner and R. H. Bordini. Jason: a java-based interpreter for an extended version of agentspeak. https://github.com/jason-lang/jason.

[30] IBM. AIF 360. https://aif360.res.ibm.com.

[31] A. Z. Jacobs and H. Wallach. Measurement and fairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 375–385, New York, NY, USA, 2021. Association for Computing Machinery. https://doi.org/10.1145/3442188.3445901.

[32] A. Jacovi, A. Marasović, T. Miller, and Y. Goldberg. Formalizing trust in artificial intelligence: Prerequisites, causes and goals of human trust in ai. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 624–635, New York, NY, USA, 2021. Association for Computing Machinery. https://doi.org/10.1145/3442188.3445923.

[33] M. Jakesch, Z. Buçinca, S. Amershi, and A. Olteanu. How different groups prioritize ethical values for responsible ai, 2022.

[34] M. Joseph, M. J. Kearns, J. H. Morgenstern, and A. Roth. Fairness in learning: Classic and contextual bandits. In *NIPS*, 2016.

[35] F. Kamiran and T. Calders. Data preprocessing techniques for classification without discrimination. In *Knowledge and Information Systems*, volume 33, pages 1–33. Springer, 2012. https://link.springer.com/article/10.1007/s10115-011-0463-8.

[36] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer, 2012. https://link.springer.com/chapter/10.1007/978-3-642-33486-3_3.

[37] M. Kearns, S. Neel, A. Roth, and Z. S. Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning*, pages 2564–2572. PMLR, 2018.

[38] E. Kindler and W. van der Aalst. Liveness, fairness, and recurrence in petri nets. *Information Processing Letters*, 70(6):269–274, 1999. https://www.sciencedirect.com/science/article/pii/S0020019099000745.

[39] N. Kozodoi, J. Jacob, and S. Lessmann. https://github.com/kozodoi/Fair_Credit_Scoring.

[40] N. Kozodoi, J. Jacob, and S. Lessmann. Fairness in credit scoring: Assessment, implementation and profit implications. *European Journal of Operational Research*, 297(3):1083–1094, 2022. https://www.sciencedirect.com/science/article/pii/S0377221721005385.

[41] N. Kozodoi, J. Jacob, and S. Lessmann. Fairness in credit scoring: Assessment, implementation and profit implications. *European Journal of Operational Research*, 297(3):1083–1094, 3 2022. https://doi.org/10.1016/j.ejor.2021.06.023.

[42] P. Lahoti, K. P. Gummadi, and G. Weikum. Operationalizing individual fairness with pairwise fair representations. *Proceedings of the VLDB Endowment*, 13(4):506–518, Dec 2019. https://doi.org/10.14778%2F3372716.3372723.

[43] M. Lohr, K. Skiba, M. Konersmann, J. Jürjens, and S. Staab. Formalizing cost fairness for two-party exchange protocols using game theory and applications to blockchain (extended version), 2022.

[44] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *CoRR*, abs/1908.09635, 2019.

[45] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), Jul 2021. https://doi.org/10.1145/3457607.

[46] J. A. Mendez, T. Kampik, A. Aler Tubella, and V. Dignum. A Clearer View on Fairness: Visual and Formal Representation for Comparative Analysis. In F. Westphal, E. Peretz-Andersson, M. Riveiro, K. Bach, and F. Heintz, editors, *14th Scandinavian Conference on Artificial Intelligence, SCAI 2024*, pages 112–120. Swedish Artificial Intelligence Society, June 2024. https://doi.org/10.3384/ecp208013.

[47] D. Miller and G. Nadathur. Lambda Prolog, 1987. https://www.lix.polytechnique.fr/Labo/Dale.Miller/lProlog/.

[48] A. Peysakhovich, C. Kroer, and N. Usunier. Implementing fairness constraints in markets using taxes and subsidies, 2023.

[49] J. Quiñonero Candela, Y. Wu, B. Hsu, S. Jain, J. Ramos, J. Adams, R. Hallman, and K. Basu. Disentangling and operationalizing ai fairness at linkedin. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '23, page 1213–1228, New York, NY, USA, 2023. Association for Computing Machinery. https://doi.org/10.1145/3593013.3594075.

[50] A. S. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language. In W. Van de Velde and J. Perram, editors, *Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, number 1038 in Lecture Notes in Artificial Intelligence, pages 42–55, London, January 22–25 1996. Springer-Verlag. https://doi.org/10.1007/BFb0031845.

[51] S. Razzaki, A. Baker, Y. Perov, K. Middleton, J. Baxter, D. Mullarkey, D. Sangar, M. Taliercio, M. Butt, A. Majeed, A. DoRosario, M. Mahoney, and S. Johri. A comparative study of artificial intelligence and human doctors for the purpose of triage and diagnosis, 2018.

[52] T. Scantamburlo, J. Baumann, and C. Heitz. On prediction-modelers and decision-makers: why fairness requires more than a fair prediction model. *AI & SOCIETY*, March 2024.

[53] J. E. Stiglitz. Pareto optimality and competition. *The Journal of Finance*, 36(2):235–251, 1981.

[54] P. Toulis, D. C. Parkes, E. Pfeffer, and J. Zou. Incentive-compatible experimental design. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 285–302, New York, NY, USA, 2015. Association for Computing Machinery.

[55] S. Voeneky, P. Kellmeyer, O. Mueller, and W. Burgard, editors. *The Cambridge Handbook of Responsible Artificial Intelligence*. Cambridge University Press, 2022. https://doi.org/10.1017/9781009207898.

[56] H. Weerts, M. Dudík, R. Edgar, A. Jalali, R. Lutz, and M. Madaio. Fairlearn: Assessing and improving fairness of ai systems, 2023. https://fairlearn.org.

[57] B. H. Zhang, B. Lemoine, and M. Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340. ACM, 2018. https://dl.acm.org/doi/10.1145/3278721.3278779.

Paper I

Ethics and Information Technology (2022) 24:12
https://doi.org/10.1007/s10676-022-09636-z

ORIGINAL PAPER

Ethical implications of fairness interventions: what might be hidden behind engineering choices?

Andrea Aler Tubella[1] · Flavia Barsotti[2,3] · Rüya Gökhan Koçer[2] · Julian Alfredo Mendez[1]

Accepted: 20 January 2022 / Published online: 28 February 2022

Abstract
The importance of fairness in machine learning models is widely acknowledged, and ongoing academic debate revolves around how to determine the appropriate fairness definition, and how to tackle the trade-off between fairness and model performance. In this paper we argue that besides these concerns, there can be ethical implications behind seemingly purely technical choices in fairness interventions in a typical model development pipeline. As an example we show that the technical choice between in-processing and post-processing is not necessarily value-free and may have serious implications in terms of who will be affected by the specific fairness intervention. The paper reveals how assessing the technical choices in terms of their ethical consequences can contribute to the design of fair models and to the related societal discussions.

Keywords AI Ethics · Responsible AI · Fairness · Bias mitigation

Introduction

The increasing use of machine learning models in decision-making processes has been accompanied in recent years by a growing concern about potential ethical hazards, especially discrimination that such models may generate. Therefore, the need for designing ethical models leading to fair outcomes is now widely acknowledged. Accordingly, the development of methods to define, measure and ensure *fairness* in predictive models is rapidly growing. Many model debiasing techniques have been developed in order to *equalise* predictive outcomes in accordance with various statistical fairness definitions, with each technique offering advantages and trade-offs in terms of accuracy, use of sensitive data, compatibility with different families of models, or development stage. Thus, when developing a model, practitioners need to make some decisions regarding *how* and *when* to introduce fairness interventions. These decisions are often taken by considering technical and computational implications of the available alternatives (Green & Hu, 2018).

This study demonstrates that if these choices are based solely on engineering grounds, then relevant ethical considerations affecting human beings may be overlooked. As an example, we show that *when* and *how* exactly a fairness intervention is introduced into the model pipeline can seriously affect who benefits and who is excluded from the positive impact of such an intervention. The goal is to reveal the way in which such an ethical problem may emerge and be overlooked (or remain obscure) during the design of a fair model via an illustrative example. For this purpose we compare two approaches to fair model design, namely, *in-processing* (introducing fairness at training time) and

Disclaimer The opinions expressed in this paper are solely those of the author and do not necessarily represent those of her current or past employers. This disclaimer only applies to the author Flavia Barsotti.

✉ Andrea Aler Tubella
andrea.aler@umu.se

Flavia Barsotti
Flavia.Barsotti@ing.com; f.barsotti@uva.nl

Rüya Gökhan Koçer
Ruya.Kocer@ing.com

Julian Alfredo Mendez
julian.mendez@umu.se

[1] Department of Computing Science, Umeå University, Umeå, Sweden

[2] Strategy Office, ING Analytics, ING Bank, Amsterdam, The Netherlands

[3] Institute for Advanced Study (IAS), University of Amsterdam, Amsterdam, The Netherlands

post-processing (modifying already trained classifiers via decision-boundary variation).

We show that, while achieving the same levels of fairness and accuracy with both debiasing techniques, the individual predictions that are modified by each intervention are *significantly different*. This is because the same individual can be subject to a different classification outcome due to the interplay between specific individual characteristics and bias mitigation techniques. Our main conclusion is that in order to ensure that a model is designed ethically, it is necessary to scrutinize all decisions during the development process (e.g. especially those that appear to be engineering decisions).

The paper is organized as follows: Section Fairness and bias mitigation introduces fairness definitions together with the related ethical challenges identified in the literature, and provides an overview of bias mitigation interventions. Section Bias mitigation: ethical decisions behind engineering choices discusses the effects of alternative bias mitigation techniques and reports an experimental study (as illustrative example) in the field of credit risk loan application. Section Conclusion concludes by highlighting the importance of ethical decisions hidden behind engineering modelling choices and suggests future research directions. The appendix contains an overview of: i) an index measure we introduce at single data point level to assess the effect of debiasing and ii) features considered in the experimental study.

Fairness and bias mitigation

Fairness is one of the fundamental pillars underlying ethical model design in different contexts, e.g. health, legal and banking[1] are only a few. Given the growing knowledge on how bias can be introduced and amplified in models (Mehrabi et al., 2019), this paper focuses on the ethical implications connected to engineering choices when debiasing a model. The goal of building fair models is to prevent discrimination - direct or indirect - against individuals or groups based on specific sensitive characteristics. In the context of modeling, two aspects are particularly relevant: *how* to formally define fairness and *when* to enforce it. This section describes the importance of fairness definitions in Section Fairness definitions: the *how* and provides an overview of the main implications behind bias mitigation techniques in Section In-processing and post-processing: the *when*.

[1] The relevance of fairness for modelling is highlighted both by the European Commission in (EC, 2019) and by the European Banking Authority in (EBA, 2020).

Fairness definitions: the *how*

Assessing fairness from a modelling perspective requires determining *how* to detect and measure the magnitude of undesired bias that can potentially generate discrimination. For this purpose, the first decision is to choose a suitable fairness definition in mathematical terms. There are many quantifiable fairness definitions (Dwork et al., 2012; Hardt et al., 2016; Joseph et al., 2016; Kearns et al., 2018), capturing different legal, philosophical and social perspectives. Here, so long as one opts for fairness based on parity between different subgroups, there is often a trade-off between fairness and model accuracy: there might be cases where a model is classified as fair, based on a given fairness definition, at the cost of reduced model accuracy (Haas, 2020; Dwork et al., 2012). Therefore, joint implications of engineering and ethical decisions may generate a dilemma between: (i) having a model that is fair(er) but less accurate or (ii) opting for a biased but more accurate model. For this reason, critical research has shown how different interpretations and implementations of fairness may harm the groups they intend to protect (Corbett-Davies & Goel, 2018) or may also ignore the bias for subgroups that simultaneously belong to several protected groups (Kearns et al., 2018). Understanding the inherent trade-offs and implications behind a fairness definition is therefore crucial for organisations and practitioners to justify and trace their implementation choices (Binns, 2020). However, there is no generic rule to identify a-priori what is the best fairness metric in each single case, and several definitions of fairness are mutually exclusive (Kleinberg et al. 2016; Dwork et al. 2012; Chouldechova, 2017). The suitability of any given fairness definition needs to be determined on the basis of the societal norms and expectations regarding what is considered fair about the specific issue at stake. In this respect, taking into account the intended purpose of the model (i.e. provision of a public service or filing an indictment) is important. However, in the final analysis, the fundamentally context-dependent nature of what is considered *fair* about a given circumstance remains intact. The advisable approach is not to categorically select or discard a particular fairness definition a-priori: ethical insights from the social environment in which the model would be deployed are the key for this decision.

We can distinguish two broad categories of fairness definitions: *individual fairness* and *group fairness*. Individual fairness definitions aim to prevent harm to each single individual (e.g. data points in the sample), by ensuring that similar individuals would be treated similarly by the model regardless of the difference between their sensitive characteristics. Group fairness, on the other hand, aims to attain *parity on average* between subgroups such as men and women, that are defined based on a sensitive characteristic.

The illustrative example proposed in this paper is built by considering *predictive parity*[2] as *group fairness definition* introduced in (Chouldechova, 2017). This enables to show the ethical implications of operationalizing a given fairness definition within the model development pipeline. Selecting predictive parity is essentially a choice of convenience: the wider point we aim to raise is that pitfalls arising from the interplay between individual characteristics and mitigation techniques will arise regardless of the chosen definition.

In-processing and post-processing: the *when*

The introduction of unwanted bias in models can stem from a wide variety of reasons (Mehrabi et al., 2019): data collection methods, features' measurement, benchmarks for evaluation, relative size of different sub-groups, evolution of populations and behaviours over time (e.g. accumulated prejudices embedded in data) are few examples. In response to this wide spectrum of causes, there are multiple techniques to actively "de-bias" models according to different fairness definitions. We can distinguish three approaches which are differentiated by their "timing of intervention" within the model development pipeline: *pre-processing* methods focus on modifying the data itself, *in-processing* (or algorithm modification) methods include fairness metrics as an objective at training time, and *post-processing* methods[3] consist on taking a trained—possibly unfair—classifier and modifying its results to enforce fairness. The paper focuses on the last two, by stressing the link between bias mitigation and fairness metrics.

The choice of in-processing or post-processing is not tied to the specific fairness definition that one wishes to implement: both approaches can satisfy the same group fairness definition equally successfully. Thus, performing bias mitigation via in-processing or post-processing is often considered a pure engineering choice. In this light, the two approaches represent different answers to the question of *when* to introduce fairness interventions within the model development pipeline. Figure 1 provides a simple representation of this by reporting a specific sub-portion of the model development pipeline. As highlighted in the picture: (i) *in-processing* aims to mitigate bias *inside* the algorithm, before the model output is generated (e.g. *training* phase), ii) *post-processing* aims to mitigate bias *after* the algorithm produces its outcomes (e.g. *validation* phase).

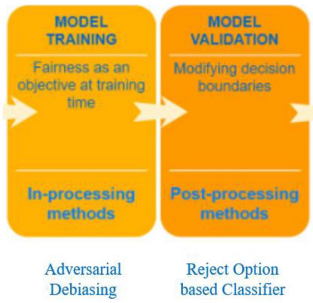


Fig. 1 Bias mitigation through the model development pipeline: in-processing vs post-processing. The plot depicts a sub-portion of the model development pipeline and highlights: i) fairness interventions for each phase (e.g. in-processing vs post-processing), (ii) specific instances for each phase (e.g. Adversarial Debiasing, Reject Option based Classifier) considered in the paper

The technical differences between in-processing and post-processing are widely acknowledged. In-processing methods allow practitioners to balance the trade-off between model performance and fairness by considering them jointly, but require opting for specific learning algorithms and applying fairness restrictions at an early stage. On the other hand, post-processing methods can often be used for any type of classifier, after fairness concerns have been identified: in this case, the control on the performance/fairness trade-off may be lower compared to in-processing, as the original classifier cannot be "re-learnt".

This paper focuses on the trade-off implied at single data point level by in-processing and post-processing solutions, highlighting how this might be linked to individual characteristics, due to the inherently different logic of the two approaches. As the outcomes for individual data points can be significantly different, this implementation choice can have strong societal and personal implications for different stakeholders. To take this decision in an informed, accountable and responsible manner, we therefore highlight the importance of exposing ethical decisions hidden behind engineering choices. We argue that understanding the trade-offs and implications of each method at an individual level is a necessary step towards responsible implementations of statistical fairness.

[2] Predictive parity requires the proportion of true positives within all positive predictions to be equal for all groups defined by the protected attribute. This prevents the model from having lower precision for underprivileged groups.

[3] In its essence, *post-processing* can be seen as a technique based on threshold differentiation across different groups, to take into account their specific characteristics. This is the approach we also consider in this paper for the illustrative example. From a more general perspective, the scientific literature also proposes post-processing methods that consider fine-tuning at single data point level depending on the magnitude of the errors (Kim et al., 2019).

Bias mitigation: ethical decisions behind engineering choices

The goal of this section is to show how different bias mitigation techniques might raise an ethical concern: alternative engineering implementations can imply a different treatment for the same data point (e.g. same person in the sample) depending on individual (and/or group) characteristics and features' correlation.

We focus the comparison on two specific instances of in-processing and post-processing implementations, respectively: (i) *Adversarial Debiasing* (AD), (ii) *Reject Option based Classifier* (ROC), occurring at different moments of the model development pipeline (Fig. 1). The two methods represent only an instance of many, which we consider as an example of how the choice of the bias mitigation technique brings with it ethical implications down the line.

Section Same person, different outcomes? discusses the theoretical overview of the impacts at single data point level deriving from alternative fairness interventions and Section Experimental study: credit risk loan application provides a study on real data in the context of a credit risk loan application.

The analysis has illustrative purposes, rather than exhaustive. In our view, the choice between these two debiasing approaches is a good example of a decision which often appears to be of solely technical nature. In reality, this choice brings relevant ethical implications that should not be overlooked.

Same person, different outcomes?

In-processing and post-processing methods achieve fairness through inherently different modifications to a classifier. On the one hand, in-processing requires incorporating a particular definition of fairness into the optimization process either directly as an additional constraint within the objective function or by means of adversarial learning. Both cases aim to optimize accuracy and fairness simultaneously. For this purpose they reduce the weight of those features which (implicitly or explicitly) give protected attribute information so as to render protected attribute information irrelevant for the predictions (Zafar et al., 2019; Donini et al., 2018; Komiyama et al., 2018). In their essence, in-processing methods try to make the protected attribute information conveyed by data points irrelevant for the classification outcome: the extent to which a given data point carries the protected attribute information in its features is crucial for determining how in-processing methods would affect that data point.

Conversely, since post-processing methods may only modify an already trained classifier, these methods are focused on selecting *which* predictions to modify to verify the desired definition of fairness (Hardt et al., 2016; Corbett-Davies et al., 2017). In its essence, post-processing can be seen as a form of threshold differentiation across different groups. This is also the approach we consider in the paper for the illustrative example: the extent to which a data point would be affected by post-processing is explicitly linked to group membership but does not require to carry on implicit information about it.

We show that these two distinct intervention choices (i.e. training time vs validation time depicted in Figure 1) can generate fundamentally different classifications for the same individual data point while operationalizing the same fairness definition. Therefore the choice between these two approaches is a good example of a decision which may appear to be of solely technical nature while having ethical implications.

In practice, since in-processing methods involve avoiding the use of protected attribute information embedded in several features in a latent form, the individual data points whose classification is modified are precisely those for which protected attribute information can be inferred from the correlations between their features. Consider the case of an in-processing intervention to enforce a notion of group fairness between Group A and Group B. Even though the information about group membership may be explicitly included in the dataset, in a real world dataset group membership could also be related to a number of other features, which can serve as *proxy variables*. Given the in-processing goal of reducing the weight of group membership on classification, the weight of all features which bear a strong correlation with group membership in the dataset will be reduced. What can we then expect at single data point level? Most of the data points that will have their prediction modified by the fairness intervention are those who exhibit features strongly correlated with the characterization of group membership in the dataset. In other words, those individuals who share many features with other individuals belonging to Group A or Group B as they are represented in the training set.

Let us now consider an equivalent fairness intervention at a post-processing stage. Since post-processing methods consider already trained classifiers, their focus is on modifying the classifications of specific inputs to satisfy fairness conditions. The set of inputs whose classification is modified is chosen in such a way that there are different classification thresholds for different classes of inputs. This set is different for each post-processing method (e.g. the points whose decision is modified can correspond to data points with low-confidence classifications, or to data points with a certain label or classification). In general,

the choice of the inputs who see their classification modified does not directly depend on the dataset features, but rather on the *decision threshold* of the original classifier[4]. Consequently, the data points who see their classification modified by post-processing interventions are completely pre-determined, regardless of whether they exhibit features strongly correlated to group membership in the dataset[5].

Adversarial Debiasing and reject option classification

This subsection discusses the comparison between alternative approaches (Figure 1) in terms of their potential ethical consequences by means of two common methods: (i) *Adversarial Debiasing (AD)* for in-processing (Zhang et al., 2018), (ii) *Reject Option based Classifier (ROC)* for post-processing (Kamiran et al., 2012).

Adversarial Debiasing is based on training two functions simultaneously: a *predictor* that assigns predictions to each input and an *adversary* that tries to guess the protected attribute information by using the outcome of the predictor. The objective of the predictor is to make accurate predictions while thwarting the adversary, meaning that the protected attribute cannot be guessed from the predictions. In this dynamic, making predictions independently from the protected attribute enables the predictor to succeed in both goals. Predictions are made in such a way that protected information implicitly embedded in the dataset are not betrayed. Thus the extent to which a given data point contributes to the emergence of such an implicit information pattern through its features is crucial for the way in which this point will be treated by AD.

Reject Option Classifier is based on the idea that bias is most likely to 'happen' close to the decision boundary, i.e. when classifications are most uncertain. Consequently, a strip around this boundary is marked and the classifications from the original model that fall into this critical region are modified according to a particular rule. The rule assumes that the protected attribute allows us to distinguish an underprivileged group and a privileged group (such as women and men defined by gender). All those data points belonging to the unprivileged group and fall into the critical region are given the desirable classification outcome whereas those data points in the critical region belonging the privileged group are given the undesirable classification outcome[6]. For

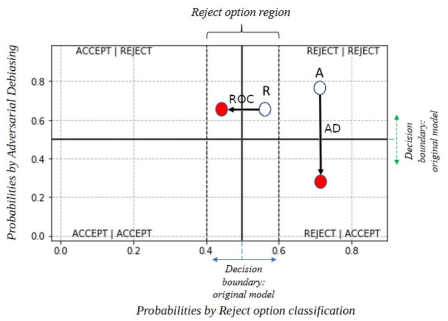


Fig. 2 Adversarial Debiasing (AD) vs Rejection Option Classifier (ROC) at single data point level. The probability predictions of the AD and ROC models are plotted against each other at single data point level. The *Y* axis reports the predicted risk score by the AD model, and the *X* axis reports the predicted score by the ROC classifier. Solid black lines represent the acceptance threshold at 0.5. Dotted black lines represent the boundaries of the critical region for ROC. Empty circles represent the initial position of each single data point. Red circles represent the position of each single data point resulting, respectively, from AD or ROC

all data points outside the critical region, the original classification attained by the model remains. As a consequence, privileged and underprivileged data points which are initially located in a narrow strip around the decision boundary are now, respectively, pushed above or below this boundary. In this case, the correlation between the features and the protected attribute that a given point exhibits do not matter directly; what matters is whether this point falls into the critical region and whether it belongs to the (un)privileged group regardless of whether this belonging can be detected by examining the other features.

Figure 2 illustrates this case with a plot. Two data points *A* and *R* corresponding to members of an underprivileged group are plotted on a space of predicted probabilities. The vertical axis reports the scores predicted by AD, while the horizontal axis reports the score determined by ROC. Solid black lines represent the acceptance threshold set equal to 0.5: a score below 0.5 means acceptance; a score above 0.5 implies rejection. To facilitate comparing the classification outcomes generated by ROC and AD, the plot is divided into four regions, namely *ACCEPT | REJECT* , *REJECT | REJECT* , *ACCEPT | ACCEPT*, *REJECT | ACCEPT*. Arrows in the plot indicate how the predictions for *A* and *R* are modified by debiasing interventions via AD or ROC respectively. The empty circles indicate the biased scores given to A and R by the initial (and biased) classifier; the red filled circles

[4] This is often tantamount to defining different classification thresholds for different groups.

[5] Notice that these correlations may however directly influence whether this data point belongs to the class of data points that see their classification modified.

[6] This holds for any classification model. In the case of a credit risk model which evaluates the risk in loan applications, the desirable classification would be "good creditworthiness/low risk score", implying loan granted.

Table 1 Attribute A9, *Personal status and sex*, from the German Credit Data dataset considered in the case study

Attribute A9	*Sex*	*Personal status*
A91	Male	Divorced/separated
A92	Female	Divorced/separated/married
A93	Male	Single
A94	Male	Married/widowed
A95	Female	Single

The table provides an overview about how Attribute A9 encodes the binary sensitive attribute "gender" together with marital status

indicate, for each data point, the corresponding debiased predictions resulting, respectively, from AD or ROC.

Post-processing via ROC modifies the classifications only for data points belonging to the critical region around the original decision boundary, marked by vertical dotted lines. Point *R* represents an underprivileged individual whose score is decreased below the 0.5 decision threshold by ROC. All data points representing underprivileged individuals situated in the critical region will have the same treatment. In contrast, AD intervention might impact data points in any area of the prediction space, even outside the critical region. Data point *A* might for example share many features in common with other underprivileged individuals belonging to the same group (training dataset), thus its prediction will be modified by AD. ROC will produce no impact on *A*, as it does not belong to the critical region. AD will impact *A* since it will reduce the weight of its features in the final classification. As a consequence, the same person may be affected disparately depending on the use of in- or post-processing.

This preliminary intuition shows how in-processing and post-processing methods achieve fairness through inherently different modifications to a classifier, producing impacts at single individual level that go beyond engineering aspects. Section 3.2 confirms the intuition via an experimental study in the context of a credit risk loan application built as illustrative example.

Experimental study: credit risk loan application

This experimental study presents and discusses the impacts generated at single data point level by AD and ROC when debiasing an originally biased classifier. The dataset we use for this study is based on the well-known German Credit Data[7] , which contains values for 20 attributes of 1000 loan applications. Attribute 9, named *Personal status and sex*, encodes gender together with marital status, as shown in Table 1. The groups we are considering in our fairness intervention are identified by the sensitive attribute "gender", as "female" (A92, A95) and "male" (A91, A93, A94). Similarly to Slack et al. (2020), we introduce *controlled bias* into the original dataset by creating a direct association between gender and creditworthiness[8]. For illustrative purposes, this experiment assumes the group with attribute "female" as the underprivileged group that is likely to suffer from bias (i.e. female, low credit score).

In the context of a credit risk loan application problem, we consider this case as an instance of the more general case of binary attribute and features' correlation.

We train three classifiers[9] to generate credit risk predictions: (i) a logistic regression model where the sensitive binary attribute gender is omitted from the dataset (ii) a corresponding "debiased" version of the model through AD, and (iii) a corresponding "debiased" version of the model through ROC. Note that the baseline logisitic regression model that is "debiased" through AD and ROC is biased despite the fact that we implement fairness through unawareness: it does not explicitly contain protected attribute information. For both "debiased" versions of the model, the case is built by considering *predictive parity* as fairness metric to optimise between groups given by the binary sensitive attribute gender[10] in the dataset.

Figure 3 shows the results of this experiment for a given logistic regression baseline model. Here the debiased risk scores obtained from AD and ROC 'corrections' are plotted against each other for the same set of data points considered as *validation set*, e.g. 300 data points. The vertical axis reports AD scores, and the horizontal axis reports ROC-scores. In both cases the decision threshold is 0.5: any person with a predicted probability above this boundary is considered 'too risky' (i.e. having low creditworthiness), thus the corresponding loan application will be rejected. Vertical dashed lines indicate the critical region considered by ROC.

[7] We use a slightly modified version of the dataset by introducing *controlled bias*. The description of the original dataset is available at: https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data). The code for this experiment is available from the following repository: https://gitlab.com/ing-umea/eit-ethical-implications.

[8] From a mathematical point of view, this is done by introducing a probabilistic relationship between a specific attribute and the final target classification. Let us suppose to have attribute A, and two categories A_1, A_2. Introducing controlled bias is done via a conditional statement: we associate a given probability $p \in [0, 1]$ to the target classification for individuals having A_1 and $(1 - p)$ for individuals having A_2. This is a way to simulate historical bias in a controlled environment. In practice, this implies establishing a deliberately 'low/high' probabilistic relationship between two binary variables, e.g. whether a given person is female (male) and her (his) creditworthiness. The Appendix discusses the scenarios considered in this specific experiment and their impacts.

[9] The three classifiers are trained on the same (randomly chosen) 700 points of our dataset and tested on the remaining 300 points.

[10] Recall that gender attribute is represented via "Attribute A9", *Personal status and sex* in German Credit Data as reported in Table 1. We directly refer to gender for easiness of exposition.

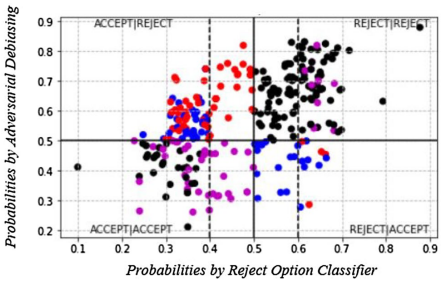


Fig. 3 Credit risk loan application. The probability predictions of the AD and ROC interventions (on the same baseline logistic regression model) are plotted against each other for the same data points (e.g. *validation set*, 300 data points). The vertical axis reports the predicted score by the AD model, and the horizontal axis reports the predicted score by the ROC classifier. Solid black lines represent the acceptance *decision threshold* at 0.5. Dotted black lines represent the boundaries of the critical region for ROC. Black and blue circles correspond to data points with "male" attribute; red and purple circles correspond to data points with "female" attribute. The plot reports the classification results based on AD and ROC for the 300 data points in the *validation set*. Within this set, 104 data points have the "female" attribute, and 196 "male" attribute. For 186 out of 300 data points (47 "female" attribute and 139 "male" attribute) AD and ROC agree in the classification outcome. Regarding the remaining 114 data points for which the two methods disagree in the classification, we have: 88 data points (53 "female" attribute, 35 "male" attribute) rejected by AD but accepted by ROC, and 26 data points (4 "female" attribute and 22 "male" attribute) accepted by AD but rejected by ROC

To facilitate the comparison of the classification outcomes generated by ROC and AD, the plot is divided into four regions with the same logic considered for Figure 2. To highlight how the two fairness interventions differ, the focus of our attention is on the *ACCEPT* | *REJECT* (top-left) and *REJECT* | *ACCEPT* (bottom-right) regions. In the first one, we see the data points whose risk score would imply acceptance from ROC but rejection from AD; in the second one, we see the data points whose risk score would imply rejection from ROC but acceptance from AD. In these two regions, data points represented in blue correspond to "male" attribute, while in red to "female" attribute. In the regions where both AD and ROC classification models agree (e.g. *ACCEPT* | *ACCEPT*, *REJECT* | *REJECT*), black data points represent "male" attribute and purple points "female" attribute. Notice that, within the critical region for ROC, only data points associated to "female" attribute are linked to acceptance, and only data points with "male" attribute to rejection. Both ROC and AD achieve equivalent levels of fairness and accuracy[11]. However, their effect on single data points is quite different. Indeed, their final classifications disagree for a large number of individuals, as depicted in the *ACCEPT* | *REJECT* and *REJECT* | *ACCEPT* regions.

Impacts of Adversarial Debiasing and Reject Option based Classifier at individual level

To compare the impacts of AD vs ROC classification outcomes at single data point level, we introduce Index[12] $t(s_i)$ to measure how "common" the features of a single data point s_i are when compared to the data points in the dataset belonging to the same underprivileged group. The index $t(s_i)$ is higher when s_i has many characteristics in common with the other data points in the same group and smaller if s_i has few common features. The experimental results show significantly different average Index values, namely $t(s_i)$, for the data points where AD and ROC imply a switch in the classification outcome. These averages are computed over the number of n data points in that specific region and are, respectively, $\bar{t}(s_i) = 1.85$ in the *ACCEPT* | *REJECT* region (standard deviation $sd = 0.526$, data points $n = 53$) and $\bar{t}(s_i) = 3.81$ in the *REJECT* | *ACCEPT* region (standard deviation $sd = 0.746$, data points $n = 4$). The two averages proved to be significantly different ($p_{value} = 0.012$, $t = -5.15$) from each other from a statistical point of view. This result suggests that, on average: i) debiasing via AD tends to ignore the circumstances of individuals who do not reflect the most represented characteristics of the underprivileged group in the dataset, whereas ii) debiasing via ROC alters the classification outcome of all individuals belonging to the critical region and does not make any further selection linked to feature commonality. This observation is robust w.r.t. different implementations and dataset changes. This case study reveals that the evidence remains the same when we change the size of the rejection region or artificially introduce a bigger bias into the dataset through causal relationships. There are interesting directions to explore via a deeper and extensive technical analysis which is beyond the scope of the present paper. The experimental study built on this credit risk loan application case aims to raise awareness that the choice of bias mitigation via in-processing or post-processing has societal and ethical implications. This engineering choice can impact *who* is most affected by the fairness intervention. Implied by the nature of in-processing, the individuals who are likely to see their classification outcome

[11] A deeper analysis on the trade-off between fairness and accuracy is beyond the scope of this paper. Results show no material gap and, starting from this observation, the study rather focuses on impacts at single individual level in terms of classification outcome to shed light on ethical implications.

[12] The appendix contains the explanation of the Index $t(s_i)$ components, provides a toy example on artificial data and the technical details of the comparison on German Credit Data.

switching from rejected to accepted are those sharing features with the majority of the members of the underprivileged group represented in the dataset. Conversely, since post-processing methods rely on modifying the decision threshold, the individuals who are likely to see their classification outcome switching from rejected to accepted are the ones close to the original decision threshold. Deciding in favour of in-processing or post-processing bias mitigation techniques thus implies different impacts on different groups of underprivileged individuals. This choice should not be considered purely through an engineering lens, but should rather take into account also the importance of ethical decisions, embedding a combination of factors such as deployment context, legal constraints, potential harm to specific group of stakeholders.

Conclusion

Building *fair* models is not an easy task. At the same time, it is important to acknowledge that building fair models cannot be reduced to a purely engineering problem. Designing and developing models, embedding or not machine learning techniques, might require the need of specific modelling choices that naturally imply trade-offs between engineering and ethical decisions. The goal of this paper is to stress the importance of ethical decisions potentially hidden behind modeling choices and their impacts at single individual level by focusing on group fairness and debiasing techniques. The empirical analysis discussed in the paper should be considered as a counterfactual evidence to showcase the overlooked impacts of engineering decisions in individual predictions. We shed light on this specific issue by stressing the importance of getting to such decisions in an informed and responsible way. Each decision should be explainable, traceable and justified, considering the implications it might have on the individuals and who will be impacted by it (e.g. as for in-processing vs post-processing). Understanding the consequences brought by implementation choices is therefore a step forward in moving beyond the computational lens and considering fairness through a wider societal and democratic perspective (Green & Hu, 2018).

Our contribution shows that identifying *how* and *when* to tackle the bias mitigation issue in a model development pipeline is not a value-free choice. Echoing practitioner's calls for comparisons and assessments of the ethical implications and side effects of different mitigation strategies (Holstein et al., 2019), we offer a characterisation of the individual data points that are impacted by in-processing and post-processing interventions, to be considered in the societal debate (e.g. which interventions are desired). It is not clear or obvious from an ethical point of view which subgroup should be prioritized in debiasing operations; those who reflect characteristic correlations in a dataset or those who do not reflect any such pattern but lie within a certain distance of the decision threshold. This choice is to be seen as context dependent and require profound reflection. Our goal is therefore not to provide a generic solution to this challenge, but to point out that this decision is impactful and should not be over- looked. In other words our aim is to show that there are substantive ethical decisions embedded into the choice between in-processing and post-processing; and ignoring this context is an ethical oversight. As an example, when considering intersectionality, we can identify implications for people at the intersection of several protected classes. Depending on their representation in the dataset, intersectional groups might be "targeted"or "overlooked" by the intervention (e.g. in-processing intervention via AD may fail to consider intersectional groups if not well-represented in the dataset, whereas debiasing via ROC alters the classiffication outcome for all individuals belonging to the critical region without making any further selection linked to feature commonality). Indeed, the difficulty of incorporating intersectionality in fairness methods is well-known in the literature (Kearns et al., 2018; Chouldechova & Roth, 2018).

Our contribution contains an important message: it is prudent to avoid making engineering choices solely on the basis of purely technical grounds. It is fundamental to ensure that no ethical choice remains unnoticed. The illustrative case discussed in the paper provides one full explanatory example supporting this advice. The results of the experimental study provide evidence that are robust w.r.t. different implementations and dataset changes (e.g. different size of the rejection region or different bias artificially introduced). The paper demonstrates how the translation of technical engineering questions into ethical decisions can concretely contribute to the design of fair models. At the same time, assessing the impacts of the resulting classification can have implications for the specific context of the original problem. A research direction we are currently exploring is extending the analysis to a broader setting and assess the robustness of different fairness interventions w.r.t. causal relationships between attributes.

Appendix: Index at individual level, Toy example and German Credit Data analysis

Index at individual level

Index $t(s_i)$ introduced in Section Experimental study: credit risk loan application attributes a single non-negative real number to any data point s_i belonging to the unprivileged class. The index is built based on dot product operator as follows. Let us consider:

Table 2 Toy example: single individual data points and binary features. The Table reports the overview of the binary features per each individual in the sample

Individual s_i	Feature 1	Feature 2	Feature 3
A	1	0	1
B	1	0	1
C	1	0	0
D	0	1	0
E	0	0	1
F	0	0	0
G	1	1	1

- R: dataset with m columns and n rows. Columns are all binary features capturing absence or presence of a characteristic. All values in R are either 1 (presence) or 0 (absence). All n observations are individuals belonging to the unprivileged class.
- S_i: row vector of dimension m characterizing a single data point s_i in terms of all the binary features. S is a subset of R. Row i in dataset S is represented by S_i.
- T: vector of dimension m, whose generic element j contains the sum of all n elements belonging to R and associated to the binary feature j.
- TB: vector of dimension m containing average values $\frac{T}{n}$ over the population of the underprivileged group. Each generic entry j in TB increases in magnitude if the particular binary feature j is common among the underprivileged group members.

The index $t(s_i)$ is defined as the dot product

$$t(s_i) = S_i.TB, \tag{1}$$

where S_i captures the characteristics of the single individual compared to the group and TB captures the characteristics of the group. The result of the dot product is a real value $t(s_i)$ which increases when individual s_i shares more characteristics with the majority of the unprivileged group members in the dataset. Conversely, the index decreases if s_i has few common features with the majority of the unprivileged group members in the dataset.

Toy example

This toy example illustrates how to compute Index $t(s_i)$ on a specific dataset. This example considers a case with a sample of $n = 7$ individuals belonging to a group based on the protected attribute gender. For each generic individual s_i in the sample, there are $m = 3$ binary features. Table 2 reports the overview of the series of binary features observed for each individual. As we can see from the table, individual *A* shares: i) *Feature 1* with individuals *B*, *C*, *G*; ii) *Feature 3* with individuals *B*, *E*, *G*. In other words, individual *A* has some characteristics in common with three other individuals in terms of *Feature 1* and - also - with three individuals in terms of *Feature 3*. On the other hand, individual *D* has characteristics in common only with individual *G* as they both share *Feature 2*. Individual *F* has no shared characteristics with any other individual in the group. By looking at the table, we can argue that, based on this set of binary features, *F* has no commonality with the other individuals in the group.

Table 3 Toy example: Index $t(s_i)$ values. The Table reports the values of Index $t(s_i)$ given in Eq. (1) computed for each individual considered in the toy example. Table 2 reports the binary features per each individual in the sample

Individual s_i	Index $t(s_i)$
A	1.1429
B	1.1429
C	0.5714
D	0.2857
E	0.5714
F	0.0000
G	1.4286

Table 3 reports the values of Index $t(s_i)$ computed for all the individuals in the group.

German Credit Data Analysis

The experiment performed on German Credit Data considers two distinct sets of data points, namely S^A, S^B (associated to different subsets of R). Index $t(s_i)$ given in Eq. (1) is computed for each data point in each subset and then $\bar{t}(s_i)$ averages the single values to a unique measure at subgroup level (S^A, S^B). The use of t-test with unequal sample size enables to compare the mean values at group level to determine whether these two sets of data points have statistically different index scores.

Table 4 reports the binary features used for the index computation in the experimental study on German Credit Data. They represent the entries of R dataset in our experimental study[13].

To add *controlled bias* in the dataset, we artificially alter the classification outcomes for gender attribute "female", as mentioned in Section Experimental study: credit risk loan application. We introduce *controlled bias* via a probabilistic relationship (conditional statement) assigning the value 'defaulted' to any given data point with "female" attribute with probability 60%. Thus, while in the original dataset the correlation between "female" attribute and 'default' is

[13] The interested reader can refer to https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data) for a full description of the single instances for each feature.

Table 4 Features' description. The table reports the overview of the binary features considered for the purpose of this experimental study on German Credit Data

Features' Description
Property (A12)
Savings account/bonds (A6)
Loan Purpose (A4)
Credit history (A3)
Housing (A15)
Job (A17)
Other installment plans (A14)
Other debtors / guarantors (A10)

−0.007, this stochastic treatment increases the correlation to approximately 0.18. As a result, the classifier will have an increased likelihood of attributing a low creditworthiness score to data point with "female" attribute. We train three models (logistic regression, Adversarial Debiasing, reject option based classification) first on a *training set* of size 700 and then used the trained models to generate the outcomes presented in the main text which is the *validation set* of size 300.

Acknowledgements The research reported in this work was partially supported by the EU H2020 ICT48 project "Humane AI Net" under contract # 952026. The support is gratefully acknowledged. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The authors would like to thank Dilhan J. Thilakarathne for bringing the team together and facilitating the research leading to this paper.

Funding Open access funding provided by Umea University.

References

Binns, R. (2020). On the apparent conflict between individual and group fairness. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 514–524.

Chouldechova, A. (2017). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data, 5*(2), 153–163.

Chouldechova, A., & Roth, A. (2018). The frontiers of fairness in machine learning. arXiv arXiv:1810.08810

Corbett-Davies, S., & Goel, S. (2018). The measure and mismeasure of fairness: A critical review of fair machine learning. arXiv preprint arXiv:180800023

Corbett-Davies, S., Pierson, E., Feller, A., Goel, S., & Huq, A. (2017). Algorithmic decision making and the cost of fairness. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, vol Part, F1296*, 797–806. https://doi.org/10.1145/3097983.3098095

Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J., Pontil, M. (2018). Empirical risk minimization under fairness constraints. In *Advances in Neural Information Processing Systems, Neural information processing systems foundation* (Vol. 2018-December, pp. 2791–2801). arXiv:1802.08626

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., & Zemel, R. (2012). Fairness through awareness. In *ITCS 2012 - Innovations in Theoretical Computer Science Conference* ACM Press, New York, New York, USA, pp. 214–226. https://doi.org/10.1145/2090236.2090255, http://dl.acm.org/citation.cfm?doid=2090236.2090255.

EBA. (2020). *EBA report on big data and advanced analytics*. European Banking Authority: Tech. rep.

EC. (2019). *Ethics guidelines for trustworthy AI*. European Commission: Tech. rep.

Green, B., & Hu, L. (2018). The myth in the methodology: Towards a recontextualization of fairness in machine learning. In *Proceedings of the machine learning: the debates workshop*.

Haas, C. (2020). The price of fairness—A framework to explore trade-offs in algorithmic fairness. In *40th International Conference on Information Systems*, ICIS 2019, Association for Information Systems.

Hardt, M., Price, E., & Srebro, N. (2016). Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, pp. 3323–3331

Holstein, K., Wortman Vaughan, J., Daumé, H., Dudik, M., & Wallach, H. (2019). Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery*. New York, NY, USA, CHI '19, p 1–16, https://doi.org/10.1145/3290605.3300830

Joseph, M., Kearns, M.J., Morgenstern, J.H., & Roth, A. (2016). Fairness in learning: Classic and contextual bandits. In *NIPS*.

Kamiran, F., Karim, A., & Zhang, X. (2012). Decision theory for discrimination-aware classification. In *Proceedings—IEEE International Conference on Data Mining, ICDM*, pp. 924–929. https://doi.org/10.1109/ICDM.2012.45.

Kearns, M., Neel, S., Roth, A., & Wu, Z.S. (2018). Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning, PMLR*, pp. 2564–2572.

Kim, M., Ghorbani, A., & Zou, J. (2019). Multiaccuracy: Black-box post-processing for fairness in classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 247–254.

Kleinberg, J., Mullainathan, S., & Raghavan, M. (2016). Inherent trade-offs in the fair determination of risk scores. arXiv preprint arXiv:160905807.

Komiyama, J., Takeda, A., Honda, J., & Shimao, H. (2018). Nonconvex optimization for regression with fairness constraints. In *35th International Conference on Machine Learning, ICML 2018, PMLR* (Vol .6, pp. 4280–4294). http://proceedings.mlr.press/v80/komiyama18a.html.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2019). A survey on bias and fairness in machine learning. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 180–186.

Zafar, M.B., Valera, I., Gomez-Rodriguez, M., & Gummadi, K.P. (2019). Fairness constraints: A flexible approach for fair classification. Tech. rep., Max Planck Institute for Software Systems, http://fate-computing.mpi-sws.org/.

Zhang, B.H., Lemoine, B., & Mitchell, M. (2018). Mitigating unwanted biases with adversarial learning. In *AIES 2018—Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 335–340, arXiv:1801.07593.

Paper II

ACROCPoLis: A Descriptive Framework for Making Sense of Fairness

Andrea Aler Tubella*
andrea.aler@umu.se
Umeå University
Sweden

Dimitri Coelho Mollo
dimitri.mollo@umu.se
Umeå University
Sweden

Adam Dahlgren Lindström
dali@cs.umu.se
Umeå University
Sweden

Hannah Devinney
hannahd@cs.umu.se
Umeå University
Sweden

Virginia Dignum
virginia.dignum@umu.se
Umeå University
Sweden

Petter Ericson
pettter@cs.umu.se
Umeå University
Sweden

Anna Jonsson
aj@cs.umu.se
Umeå University
Sweden

Timotheus Kampik
tkampik@cs.umu.se
Umeå University
Sweden
SAP Signavio
Germany

Tom Lenaerts
Tom.Lenaerts@ulb.be
Université Libre de Bruxelles
Vrije Universiteit Brussel
Belgium
University of California, Berkeley
CA,USA

Julian Alfredo Mendez
julian.mendez@cs.umu.se
Umeå University
Sweden

Juan Carlos Nieves
jcnieves@cs.umu.se
Umeå University
Sweden

ABSTRACT

Fairness is central to the ethical and responsible development and use of AI systems, with a large number of frameworks and formal notions of algorithmic fairness being available. However, many of the fairness solutions proposed revolve around technical considerations and not the needs of and consequences for the most impacted communities. We therefore want to take the focus away from definitions and allow for the inclusion of societal and relational aspects to represent how the effects of AI systems impact and are experienced by individuals and social groups. In this paper, we do this by means of proposing the ACROCPoLis framework to represent allocation processes with a modeling emphasis on fairness aspects. The framework provides a shared vocabulary in which the factors relevant to fairness assessments for different situations and procedures are made explicit, as well as their interrelationships. This enables us to compare analogous situations, to highlight the differences in dissimilar situations, and to capture differing interpretations of the same situation by different stakeholders.

*All authors contributed equally to this research. Authors listed alphabetically

FAccT '23, June 12–15, 2023, Chicago, IL, USA

ACM ISBN 979-8-4007-0192-4/23/06.
https://doi.org/10.1145/3593013.3594059

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Algorithmic fairness; socio-technical processes; social impact of AI; responsible AI

ACM Reference Format:
Andrea Aler Tubella, Dimitri Coelho Mollo, Adam Dahlgren Lindström, Hannah Devinney, Virginia Dignum, Petter Ericson, Anna Jonsson, Timotheus Kampik, Tom Lenaerts, Julian Alfredo Mendez, and Juan Carlos Nieves. 2023. ACROCPoLis: A Descriptive Framework for Making Sense of Fairness. In *2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT '23), June 12–15, 2023, Chicago, IL, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3593013.3594059

1 INTRODUCTION

Fairness is a fundamental aspect of justice, and central to a democratic society [51]. It is therefore unsurprising that justice and fairness are at the core of current discussions about the ethics of the development and use of AI systems. Given that people often associate fairness with consistency and accuracy, the idea that our decisions as well as the decisions affecting us can become fairer by replacing human judgment with automated, numerical systems, is appealing [1, 17, 25]. Nevertheless, current research and journalistic investigations have identified issues with discrimination, bias and lack of fairness in a variety of AI applications [42].

Algorithmic fairness has been framed as a newly emerging area that studies how to mitigate discrimination in automated decision-making, providing opportunities to improve fairness in AI applications [20]. Research in algorithmic fairness, or AI fairness, has produced a number of frameworks and formal notions of fairness in AI [5, 26], many of which are mutually incompatible. There is to date no agreement on the relative strengths and weaknesses of such notions, nor on the appropriate scope for their application. Furthermore, as Birhane indicates [7]: "many of the 'solutions' put forward (1) revolve around technical fixes and (2) do not center individuals and communities that are disproportionately impacted". While technical and formal approaches to fairness remain an active area of research, actual progress is stalled as they insufficiently address the reasons as to why the AI systems were introduced in the first place [43]. Moreover, they typically fail to take into consideration many of the socio-technical factors that are relevant for a satisfying assessment of the fairness of using AI systems in given situations. There is therefore a need to broaden the lens on fairness [4, 7], taking the focus away from formal definitions, and allowing for the inclusion of societal and relational aspects to represent how the effects of AI systems affect and are experienced by individuals and social groups [18].

In light of these considerations, it is central to try and find tools that tame the complexity of fairness descriptions and models, so as to allow multidisciplinary insights and stakeholder participation that can lead to actionable solutions for the use of algorithms in socially responsible decision-making. In this paper, we propose such a tool: the ACROCPoLis framework to represent allocation processes with an emphasis on modeling socio-technical and contextual aspects that are relevant to fairness. For the purposes of this framework, our conception of allocation is very broad, including not only material resources such as food, housing, water, or capital, but also immaterial social constructions such as free time, job posts, legal rights or societal recognition. Consequently, a wide range of situations can be assessed. The framework provides a shared language in which the factors relevant to fairness assessments for different situations and procedures can be made explicit, including the main entities involved and the relevant interconnections between them.

The goal of this contribution is to redirect the focus to the fact that AI systems are part of wider, complex socio-technical processes where a variety of stakeholders play important roles. The stakeholders can for example be social and political forces, as well as technical constraints. With a general framework to represent such processes, it becomes possible (1) to compare analogous situations, (2) to highlight the relevant differences in dissimilar situations, and, in cases of conflicting fairness assessments, (3) to capture differing interpretations of the same situation by different stakeholders – thus organizing the discussion and pointing to the relevant points of disagreement between the parts. As such, our proposed framework for fairness is preparatory to ethical assessments of fairness, describing the stakeholders, their roles, their mutual relations, and the stakes and values at play, while remaining neutral on normative questions.

Although the origin and focus of the here proposed framework are on processes that include AI as part of the decision-making, we see such systems not as independent, neutral technological products disconnected from the context in which they are devised and applied, but rather as artifacts or tools that humans use to shape and enforce social, political, and economic structures. The framework is meant to provide an analytical tool that allows a richer appreciation of the complexity involved in fairness assessments by identifying in precise ways the relevant actors, their power to influence the process and the broader context that have all interacted to yield the observed outcomes. Our overall aim is to initiate an essential discussion on what fairness means within the AI community, which aspects are essential to examine, and which systemic factors one may be overlooking.

2 BACKGROUND AND MOTIVATION

How to best understand fairness is heavily contested in the many areas of study in which the notion is used – this includes philosophy, political science, social science, and AI and data ethics. Traditional formal operationalizations of fairness stem from fields outside of AI, such as economics. For example, a commonly studied problem in that field is the fair division of goods [46].. The questions that these fields of research address, however, are typically dependent on specific formal models, which makes them hard to adapt to a broader societal context.

Within AI, fairness has been identified as a core principle in a myriad of guidelines and standards focused on the production of responsible and trustworthy AI systems [11, 22]. In this context, fairness is specifically tied to non-discrimination, bias and harm reduction. Thus, the field of AI fairness focuses particularly on the notion of unfair bias. This can be seen in multiple applications: from detecting undesirable word associations in natural language processing (e.g., associating the word "doctor" directly with male pronouns) to undesirable associations of features with prediction outcomes in predictive systems (e.g. skin color with criminality). Fairness in AI also includes the idea that AI applications should be robust across populations, with similar accuracy and error rates across subgroups (e.g. considering equal representation of people with different socio-economic status in medical predictive systems [57]). This can be summarized as avoiding unfair bias in terms of outcomes, benefits and harms.

Thus, assessing fairness from a technical perspective requires determining how to define and measure undesired bias in terms of specific characteristics, attributes, and outcomes. This is often taken to imply a requirement to quantify all these aspects. Recent years have seen the introduction of several fairness definitions [21, 29, 33, 34], capturing different legal, philosophical, and social perspectives. However, there are arguably elements of bias and unfairness that resist quantification, requiring the use of qualitative or mixed methods in order to fully understand the dynamics at play [16, 28, 38, 41].

Several statistical operationalizations of fairness, often called "definitions", have been developed, defended, and criticized, each hinging on different statistical features of the predictions produced by algorithms as criterial for fairness: [30], for instance, identifies 13 different operationalizations (see also [13] and [45]). Further complicating the issue, analyses have shown that most of such operationalizations are mutually incompatible, and thereby cannot be simply added to improve algorithmic fairness [37, 44]. In many

cases, trade-offs need to be made between optimizing algorithms for fairness and achieving the social goods which are supposed to be produced by these algorithms [14].

Such operational approaches to fairness nonetheless constitute a considerable advancement, and provide useful tools for responsible AI designers. However, even setting aside the problems of mutual incompatibility and optimization-performance trade-offs, the fact that they rely purely on quantifiable, statistical measures of fairness is itself problematic. Critical approaches emphasize that fairness is multi-dimensional and that purely quantitative bias definitions and de-biasing methods may lead to new biases, and may be unable to deal with intersectionality (where effects of biases are greater-than-additive, or cause "double binds" [15]). More generally, these definitions tend to ignore or downplay the complexities of social and political contexts, including their complex and mutable dynamics, which makes relying on past data to drive future decisions deeply problematic [2, 8, 23, 24, 34, 39]. Moreover, by seeing algorithmic fairness as a purely technical problem, there is a risk of ignoring the value-laden choices that must be made in the design of algorithms (e.g., what fairness measure to use and what attributes to protect), in how they are applied (e.g., in which social contexts, for what aims, etc.), which individuals and groups should have a say in shaping the algorithms and their applications (e.g., engineers in private companies, democratically elected politicians, affected groups, etc.), and whether algorithmically assisted decision-making is morally acceptable at all in specific cases [12, 27, 31, 32, 52, 56].

Following these criticisms, recent proposals have called for more robust frameworks within the field of AI focusing on "studying up" [4, 7, 43], i.e. moving beyond the technical, and including crucial aspects of "power, historical inequalities and epistemological standpoints" [43]. Such undertaking is relevant in the case of fairness, given the diversity of perspectives and models for fairness. The need for a descriptive framework is thereby not a matter of establishing a single agreed understanding of fairness, but rather to provide the means to represent more accurately the elements and relations that should underlie assessments of the fairness of a process and/or situation, and that can help reveal where disagreements lie when conflicting fairness assessments are defended by different parties. In this sense, the present work is similar to the framework put forward in *hard choices in AI* [19], which delves into a wider set of socio-technical challenges beyond fairness, but likewise aims to bring to the surface disputes and differing views between different Actors involved in the implementation and use of AI systems.

3 A FRAMEWORK FOR DESCRIBING AND ANALYZING FAIRNESS: ACROCPOLIS

The goal of our framework is to provide a shared language to specify and parse descriptions of situations and processes that are considered "fair" or "unfair" (collectively: *fairness statements*). In particular, we attempt to extract specific features in the description that relate to *why* and *how* the situation or process is perceived as fair or unfair, and to help express this in a clear and consistent fashion for a variety of different situations and processes. By means of a common framework, we provide a structured description of the different components of fairness statements, and as such support

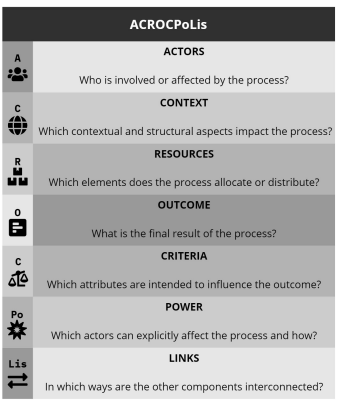


Figure 1: The components of the ACROCPoLis framework, summarized.

the understanding of why different Actors may differ in the fairness assessments they provide. A common framework also enables a schematic representation of fairness issues in studies on the consequences of AI applications, and enables standardised annotation of fairness-relevant components in meta-studies in AI and ML.

According to our proposed framework, ACROCPoLis, a process can be decomposed into seven components: **A**ctors, **C**ontext, **R**esources, **O**utcome, **C**riteria, **Po**wer, and **Li**nk**s**, depicted in Figure 1. Briefly, *Actors* are the agents of the situation/process under study, whether they are individuals, groups of people, institutions or a combination thereof. *Context* captures the relevant contextual and structural factors that bear on the specific situation/process, as well as information about what the situation or process involves. Fairness processes and situations often involve reasoning about the (re)distribution of specific *Resources*, which we take in a broad sense to include not just material items such as money and food, but also increased representation, compensation, agency, legal rights, acknowledgment, etc. The redistribution of such Resources are one type of *Outcome*. Another Outcome type is decision, for instance regarding access to social benefits, hiring, and in legal contexts. *Criteria* contains whatever attributes are used to influence the Outcome, and *Power* describes the ability of each Actor to influence the system, including the nature of the influence. The *Links* highlight important connections between categories and concepts in the described system, in particular those connections that are not obvious from the previous description.

The choice of categories responds to the growing literature calling for a multi-disciplinary and socio-technical view of processes that involve AI systems. Considering a broad category of Actors allows for including companies and organizations, which often hold accountability for the Outcomes of the process [50], as well

as communities and individuals affected by a given process, whose perspectives are key to fairness assessments [7]. Following this human-centric perspective, it is crucial to include aspects of the Context to allow for discussion of contextual and structural factors that are often the root cause of unfair processes, through perpetuation or exacerbation of existing inequalities [10, 36].

In terms of algorithmic fairness, the focus is often placed on allocation processes [55], where Resources are being distributed. We consider Resources in a wide sense, including "labels","scores" or even "error rates", but also aspects such as recognition or representation. This allows for modeling part of the Outcomes of a decision process as a reallocation of Resources, for which different definitions of fairness can apply. As our focus is on cases in which AI systems are integrated in socio-technical processes, a natural category to include is the Outcome of the process being considered. The effects of Outcomes on the Actors, the Context and Resource allocation play a key part in assessing the (un)fairness of a process. When describing a process in terms of fairness, a focus is often on the attributes of the Actors and the Context that explicitly influence the process [55]. Identifying these aspects, which we name Criteria, allows for an explicit characterization of what concretely produces the Outcomes observed.

A final critical choice in the framework we propose is to explicitly include a category on Power. This category aims to make explicit which agents have direct influence, i.e. Power, over the process, as well as to describe how much agency (or lack thereof) each Actor possesses. Being explicit on this aspect is critical for a sociological view of processes [4, 43], since it is crucially intertwined with issues of fairness (who holds the power? [7]) and accountability.

Describing the above categories is, in many cases, not sufficient to assess or describe the fairness of a process. Indeed, fairness assessments often hinge on the relationships between components. For this reason, Links, denoting such components, are an integral part of the framework.

The next sections detail the different elements of ACROCPoLis. As a running example we use the well-known COMPAS case, where an AI decision support system ostensibly evaluated the recidivism risk of court defendants based on various demographic data. In an exposé by ProPublica, the system was found to yield different results based on race, in effect recommending harsher penalties and higher bail payments for Black people than white people [42]. In particular, the perspectives of both Northpointe (the company that built the COMPAS system) and ProPublica will be described, as well as the factors these Actors took as most relevant, revealing the differences in the fairness assessments they provided.

3.1 The Components

Actors. When describing a process as being (un)fair, there is always a set of agents that make and/or are subjected to certain decisions and/or allocations of Resources. In particular, such a description considers these agents as *moral agents*, with their circumstances and feelings being worthy of consideration in assessing the fairness of a situation, and their decisions and actions having moral weight. As such, current AI agents are unlikely to be considered Actors under this framework, as current automated systems are generally not considered moral agents. However, it is not uncommon for groups or institutions to be considered moral agents in and of themselves, and as such, they should be represented as Actors in this framework. Furthermore, Actors are not simply identifiers but are differentiated by having certain attributes, which can be used to study and describe a large number of Actors' involvement in a process by way of grouping them by their various attributes.

For our running example, both Northpointe and ProPublica consider the relevant Actors in the system to be Northpointe themselves as designers of the COMPAS system, the policymakers who approved its use, the judges who used the system to inform their decisions, and the defendants who were subjected to it.

Context. Although it is possible to talk abstractly about various conceptions of fairness, most fairness statements concern some specific (type of) process embedded in some Context. In particular, we take this category to contain both the specific context that the fairness statement concerns, and any of its aspects connected to the (re)distribution of Power and Resources, including structural and population-level aspects. The concept Context also includes the sociotechnical systems active in the process, relevant aspects of the dynamics of such systems, as well as other aspects of the world that are relevant for making assessments about the (un)fairness of the process.

In relation to the COMPAS system, ProPublica and Northpointe have different views on important parts of what Context to include in the description of the system. In particular, while existing inequalities are acknowledged by both, the impact of racism on the higher arrest and conviction rates of Black people in the training data is particularly important to ProPublica's description of the situation. Conversely, an important part of the Context for Northpointe is that the limited Resources available to the court motivate using the COMPAS system as a time-saving measure in order to process more court cases faster.

Resources. Our conception of Resources is very broad, including not only material resources such as food, housing, water, or capital, but also immaterial social constructions, such as dignity, free time, and legal rights. Moreover, we also include concepts such as societal recognition and agency within a structure as Resources. This category therefore includes any element that can be seen as being (re)distributed by the process. In most cases, the description of a process as fair or unfair relates to some distribution of Resources in this broad sense, either with the Resource distribution impacting the Outcome of the process, or that the process, through redistribution, leads to, counteracts, or perpetuates some existing (un)fair distribution of Resources. .

The Resources relevant to the description, for both Northpointe and ProPublica are Resources taken from, on the one hand, incarcerated people like "bail", "time not in prison", and social status in general. On the other hand, Resources are also taken from the state or society at large like "state funds" as applied to jails, prisons, courts, and the costs of social measures for recidivism prevention. In this wider view, the issues of private prisons, prison labor and prisoner exploitation also become relevant, not only due to higher (absolute) incarceration rates leading to a redistribution of Resources from the public into private hands, but also due to the fact that uneven incarceration rates between groups can lead to exploitation of the incarcerated group, redistributing the fruits

of their labor and, quite often, their savings and future earnings (through debt), to beneficiaries in other groups.

Outcome. When describing fairness in processes, the Outcome is clearly relevant. Outcomes can be modeled as a (re)distribution of some Resource, a (re)definition of a Power relation, or the introduction or removal of Power or agency invested in some Actor. It is also possible to model Outcomes on the group level, and across different timeframes; a process run once, or in a perfect society, may be perfectly fair, but under existing societal structures and run over many iterations, it may entrench or introduce unfair (dis)advantages or distribute Resources unfairly. This category therefore includes changes, effects or results that are direct consequences of the process being analyzed.

Both Northpointe and ProPublica have an individualist view on the Outcomes of the system, focusing on each bail judgment and prison term for individual defendants. However, both also look at the process from a statistical, iterated perspective, comparing the Outcomes over many individual runs. In particular, a critical Outcome is the *rate of false positives* in the group of Black defendants, which is higher than the same statistic for white defendants. To clarify the notion of Resource distribution with this example, COMPAS redistributes among the US population Resources such as freedom, political participation, perceived social value, economic opportunities, and the like, in light of the sentencing decisions it contributes to.

Criteria. When describing a process in terms of fairness, a focus is often on the attributes of the Actors and the Context that explicitly influences the process. These denote the "causality" behind the system, insofar as they capture the criteria underpinning the decisions that produce Outcomes. As such, the criteria form a connection from the state of the world (as defined by the Actors, their attributes, and the Context the process is enacted within) to the Outcome of the process. Criteria may be explicit or implicit, intended by the Actors in exercising their Power, or unintended.

In the case of COMPAS, the Criteria behind the process consist of i) the various types of demographic and judicial data that COMPAS employs in its algorithm; ii) the sentencing guidelines used by judges; iii) the instructions about how judges are to use COMPAS; and iv) the actual use of such guidelines and instructions. These Criteria are agreed upon by both ProPublica and Northpointe. It is notable that in both views, the race of defendants should not be part of the Criteria.

Power. Power describes the explicit ability of an Actor to affect the process being described. This category aims to make concrete which agents have a direct influence, i.e. power, over the process, as well as to describe how much agency (or lack thereof) each Actor possesses. In particular, an Actor can have Power over various aspects of the Context in which the Actors exist, as well as over the Criteria by which a fairness process is conducted or evaluated. Additionally, Actors may have Power over other Actors, and thus second-order Power over any aspects that those Actors control.

As noted in the Actors section, various Actors in the system were able to control specific parts, though generally Northpointe was keen on downplaying the amount of control they could enact through their design of the COMPAS system, and ProPublica moreover was clear that public officials had the Power to limit the type of information that the system could be provided with. Both parties agreed that the judge was the one who ultimately had the Power to use or ignore the COMPAS system in making their judgments. For ProPublica, however, COMPAS retains an undue degree of Power in influencing judges' decisions, who may be more prone to follow the outputs of the system, despite instructions not to give too much weight to them in informing their decisions.

3.2 The Links

Describing the various interconnections between components of the system is a key part of any application of ACROCPoLis. The Links can be understood as shown in Figure 2: Some certain Power belongs to the Actors that can use it, and the Power can modify the Context that the Actors exist in. The Context and the Actors both influence the Criteria while the Power modifies the Criteria – the Criteria in their turn determine the Outcome. An Outcome can update the Power, and also redistribute the Resources that are determined by the Context and used, accessed or generated by the Actors.

While the Links indicated in Figure 2 are intended to serve as a good basis for discussion and analysis, they may need to be further expanded to fully describe the connections an analyst considers to be relevant to assess the fairness of the process under study. For example, Actors may have Power over other Actors, either directly or through the existence of hierarchical power structures in the Context, which would give these Actors second-order access to the Powers that their subordinates have, or there may be more indirect connections between Actors, Context and Outcome than is appropriate to model as a criterion, but that is nevertheless relevant for assessing the fairness of the process. These more fine-grained Links may be added to the ACROCPoLis analysis on a case-by-case basis.

As mentioned under Outcome, an important additional connection in the use of the COMPAS system is that between the race of the defendant and the COMPAS assessment (in particular the rate of false positives), which also often influences the final decision by the judge. In their descriptions of the situation, both Northpointe and ProPublica agree that such a difference existed, though they differ on whether it is relevant or not.

3.3 The Evaluation

The above framework is, to a certain extent, purely descriptive. It picks out relevant aspects of processes and situations under analysis, highlighting those components that are central to the assessment of the process as being fair or not. A crucial part of any fairness analysis is, however, normative: one needs to decide whether the situation, as presented, is fair or not, and why. What Links are missing? What aspects of Context and Power are ignored? Are there conflicts between what the Criteria actually are and what they were intended to, or should, be?

Such an evaluation can be obtained by comparing ACROPoLis analyses based on the perspectives of stakeholders with diverse and potentially diverging viewpoints. Ideally, the ambition is to present an integrated analysis that is as complete as possible. In this way, conflicts and disagreements about the nature of the different

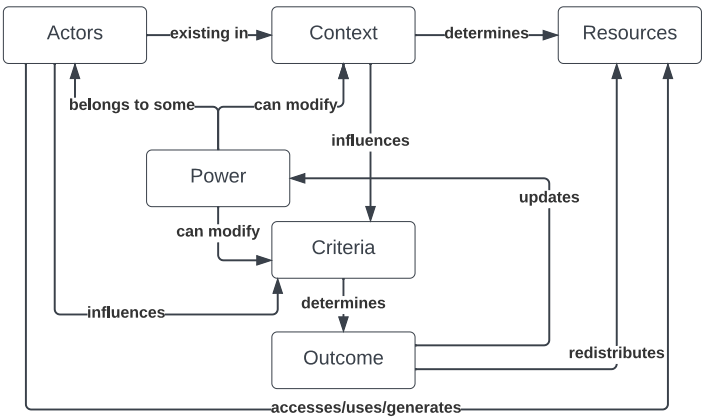


Figure 2: A schematic overview of the components and links of the ACROCPoLis framework

components of the process or situation can be brought to surface, facilitating a clearer assessment of the case and of the contested points, opening the way for compromise and/or consensus building. The final result can then be accompanied by a joint concluding assessment with respect to the fairness of the outcome and, in case actions to address fairness issues are supposed to follow, by a counterfactual "future state" ACROPoLis proposal that suggests how these issues can potentially be addressed.

These features of ACROCPoLis can be clearly seen in the dispute between Northpointe and ProPublica. In defending the fairness of the system, Northpointe pointed to the existing differences in recidivism rate between Black and white defendants as an explanation for why the rate of false positives are expected to be higher for Black people than for white people, purely on the basis of statistical and demographic considerations. COMPAS, therefore, would be correctly mirroring historical and demographic patterns, which it was not intended to change. To this, ProPublica objected that COMPAS perpetuates patterns of unequal treatment of the Black population by the justice system, seeing COMPAS as partly responsible not only for failing to ameliorate, but also for worsening such unequal Outcome.

In other words, the disagreement between Northpointe and ProPublica is largely due to the different views each party takes both to the Power COMPAS exercises, the kinds of Criteria that such a system should employ, and thus the Outcomes it should produce.

Figure 2 summarises, with a degree of simplification, the COMPAS case study examined above, following the framework provided by ACROCPoLis. Figure 3 shows the resulting instantiation.

3.4 Using the framework

Beside describing in precise ways the fairness-relevant elements and relationships in situations and processes, ACROCPoLis can also be used to identify the crucial factors leading to unfair outcomes, and the most appropriate loci for intervention. For instance, in the COMPAS example, it becomes clear that the Context of systemic racism against Black people in the US strongly influences the Outcomes, despite the fact that the Criteria were supposed to be neutral toward defendants' race. Thus, an intervention to ameliorate the situation could involve modifying the Criteria to counter, instead of mirroring, such systemic and historical inequality of treatment between groups in the US population.

Analogously, defendants have diminished autonomy and Power insofar as judges' decisions are influenced by the groups they are seen to belong to, risking to downplay the individual history and circumstances of defendants in motivating the decisions. Such considerations put pressure on any system that uses historical demographic data to help decide the fate of individuals, suggesting that a larger societal discussion may be needed when evaluating whether AI technologies such as COMPAS should be used by governments.

4 YOUTUBE RECOMMENDATION SYSTEM: A CASE STUDY

To further illustrate the use of ACROCPoLis, let us examine recent work on the search and recommendation algorithms of the video platform YouTube, and how they are analyzed in terms of fairness in the literature. YouTube's algorithms are proprietary (closed-sourced and not documented in a transparent manner from a user or public perspective). This opacity has triggered studies into the features of the algorithms as well as the consequences of their operation within the platform, based on publicly accessible facts [3, 53].

Two such studies are examined here: the first study focuses on examining the question of whether the YouTube recommendation algorithm favors a small subset of the video content available on

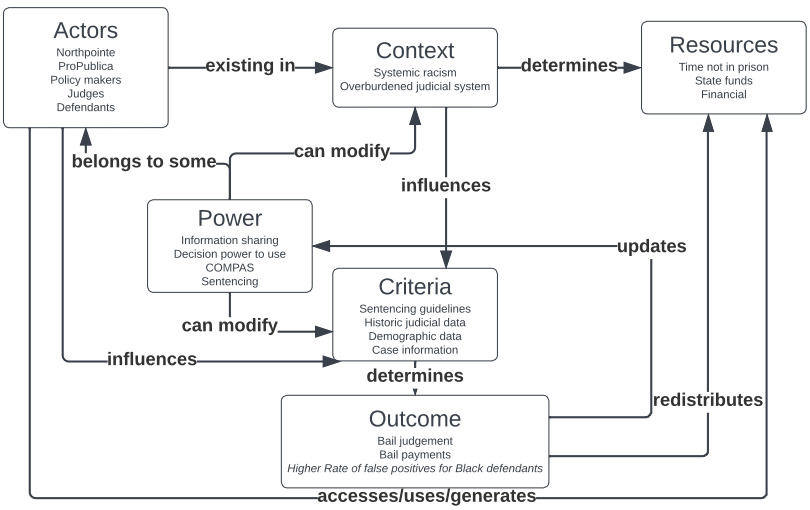


Figure 3: The COMPAS case study visualized using our framework, based on the components identified in [42]. We can use the Links between them to identify where the process lacks fairness.

the platform [35]; and the second examines whether YouTube's search algorithm is biased toward video content associated with specific positions on the United States political spectrum [40].

Kirdemir et al. [35] investigate the structure of recommendation networks, and probabilistic distribution of video recommendations from a given node in the network. Recommendation graphs are constructed based on eight different real-world scenarios (seed video sources) to then apply a stochastic approach and observe PageRank [47, 48] distributions over such graphs. The study found that a small fraction of the nodes in the recommendation network received the large majority of connections, suggesting that YouTube's recommendation algorithm is biased toward a small number of content items available on the platform.

Lutz et al. [40] examine political bias in YouTube's recommendation and search algorithms from a US-centered, binary (right-leaning vs left-leaning) perspective. Search and recommendations were studied in separate experiments. We will be concerned exclusively with the study on the search algorithm. For that experiment, the authors scraped the 200 top search results for a variety of politically-charged terms in the US public debate, using four YouTube accounts with different viewing profiles create for the experiment. The top search results for each term and each profile were evaluated for their political leaning. The study found that left-leaning content items were significantly more likely to appear among the three top search results.

Do these studies indicate unfair biases in YouTube's search and recommendation algorithms? Let us use the ACROCPoLis framework to examine these cases in detail, starting from the identification of the relevant components.

In Table 1, the ACROCPoLis framework is used to describe the fairness-relevant components extracted from both articles. In addition to identifying the components playing a role in each paper, let us take a closer look at the **Links** that are relevant for assessing the fairness of the situations examined in the studies at hand, and what they reveal about the limitations of the analyses proposed therein.

In [35], the focus is on the possibility that the YouTube recommendation algorithm is biased toward certain sorts of content, thus generating a pattern in which users are ultimately pushed towards a small fraction of the content available on the platform (Content Bias). The Actors identified by the paper, i.e., content creators, content consumers, and YouTube itself, are seen on the background of YouTube's revenue model (Context). The platform revenue's model is largely organized around advertisement revenue, which is shared between the company and the content creators. Thereby, YouTube

Table 1: ACROCPoLis analysis of the YouTube recommendation and search algorithms, based on results from two different papers, one focused on content bias, one on political bias. Items between parentheses are neglected in the papers, but need to be included in a full ACROCPoLis analysis

	Content Bias [35]	**Political Bias [40]**
A	Content creators, content consumers, YouTube, (policymakers)	Content creators, content consumers, YouTube, (policymakers)
C	Competition between content creators for user views; abundance of content items; YouTube's revenue model	Population of platform users with diverse positions in the political spectrum; social and legal protections of free speech
R	Time spent on the platform by users; views per content item	Time spent on the platform by users; views per content item
O	Users following platform recommendations lead to the same small group of content items; 'winner takes all' effect	Users more exposed to left-leaning content items in top search results in the US
C	Proprietary recommender algorithm, criteria unknown	Proprietary search ranking algorithm, criteria unknown
Po	Content creators: produce content and add it to the platform (Content consumers: choose what content items to view and engage with, and for how long) YouTube: generate content recommendations	Content creators: produce political content and add it to the platform (Content consumers: choose what content items to view and engage with, and for how long) YouTube: generates ranked search results
Lis	See remainder of this section.	

have an interest in maximizing the time spent by consumers on the platform, thus increasing the amount of advertisement they are exposed to; while content creators have an interest in maximizing the number of platform users that view the content items they produce, thus increasing the share of advertisement revenue they receive from the platform (Resources).

Interestingly, the paper seems to place most of the Power in the hands of YouTube itself, insofar as the platform is responsible for the recommendation algorithm that increases the visibility and accessibility of certain content items rather than others. The ACROCPoLis framework, however, invites an examination as well of the Power, if any, that the other Actors possess. First, content creators arguably have an important amount of Power in shaping the allocation and distribution of the relevant Resources (time spent and views by content consumers). Indeed, it is at least in part the attractiveness and interest of the content items they produce that attract consumers to YouTube to start with. Moreover, content creators may aim at creating content items that they believe will attract a larger number of users, in light of their own knowledge about societal trends and public interest in different subject matters. Finally, content creators may partially reverse-engineer some of the Criteria used by YouTube's recommendation algorithm in order to increase the likelihood of appearing as recommended content to the population of consumers they think might be more attracted to their content items (a sort of Recommender Engine Optimization). In consequence, content creators should arguably not be seen as passive, powerless Actors in the situation under examination. While their interests partially overlap with those of the platform, they also partially differ, insofar as content creators compete among themselves for the available Resources, while YouTube is mostly interested in overall content consumption on the platform.

Similarly, and perhaps more strikingly, content consumers in the platform are treated as purely passive Actors in the paper. Indeed, the methods used in the research simulate consumers that fully follow the deliverances of the recommender algorithm (with some noise added), thus being largely deprived of any Power in deciding what content to consume, and for how long. This is an oversimplification that the authors acknowledge, but it risks neglecting relevant Links that are central to the assessment of the fairness of the outputs of YouTube's recommender algorithm. Content consumers have at least some degree of autonomy in selecting what content items to consume, and which recommendations to follow or to ignore. Importantly, they may thus influence the Criteria embodied in the recommender algorithm itself. It is indeed in the interest of YouTube to generate recommendations that mirror the type of content that consumers may want to engage with, and it is hence to be expected that the algorithm responds to the patterns of consumer preference in the platform. Moreover, the large amount of data that Alphabet, the owner of YouTube, possesses about each user of its services allows considerable personalization of content recommendations. However, as the authors admit, studying the influence of personalization on the recommendations produced by YouTube is very challenging. In brief, through the Power content consumers possess by means of their consumption preferences, they likely influence the Criteria (i.e., the features used by the recommendation algorithm) that lead to the observed Outcomes (i.e., recommendations that encourage consumers to view a small fraction of the content items available).

The above considerations suggest that an assessment of whether or not the content bias found by the researchers is unfair or else requires a fuller picture of the Power the relevant Actors possess, and how they shape the Criteria that produce the content bias found by the study. It may well be, for example, that no unfairness is involved, say, if what the recommendation algorithm does is to successfully draw users toward content items that they are more likely to want to consume. Given the varying quality of the content items available, the variety of subject matters treated in such content items, and social trends that help determine what is or is not in the 'public eye', it is arguably expected that relatively few content items are taken to be particularly interesting by content consumers. In other words, YouTube's recommender algorithm might be appropriately promoting content items with higher relevance, interest, and quality. Importantly, we are not claiming that this is the case. Our claim is merely that the paper leaves underdefined several ACROCPoLis components and Links that are crucial to a satisfying assessment of the fairness of the situation being examined.

Analogous considerations apply to [40]. The result of the paper that concerns us here is the finding that YouTube's search algorithm shows a bias toward placing more left-leaning content among the top three search results on the platform than would be expected in light of the relative representation of political leanings among content items. One central methodological choice in the paper is that of inferring the distribution of political leanings across the YouTube user base in the US from the political leanings found in the sample of content items examined in the study. Under the ACROCPoLis framework, this methodological choice is far from inconsequential, since it involves a conflation between two different Actor categories that are non-overlapping and that have partially incompatible interests and Power, namely content creators, and content consumers. Content creators are plausibly interested in maximizing the viewership of their content items, and content consumers are interested in consuming the content items that they find most interesting, entertaining, or informative. While this suggests that content creators should aim at satisfying consumer preferences, the goals of content creators can also involve other considerations, such as popularizing minority positions, spreading relatively neglected ideas and ideologies, and the like. In other words, it is plausible that the political leanings expressed by content producers are not an appropriate measure of the political leanings of content consumers.

As the ACROCPoLis analysis makes clear, content creators and content consumers are different Actors, who possess different kinds and degrees of Power in shaping the search ranking algorithm (Criteria) and thus the Outcomes. Thereby, the study results could at most be used to argue that there is an unfair bias in top 3 search results in the US under a uniform distribution understanding of fairness (i.e., equal exposure to be given to each political leaning, regardless of representativeness in the population). It is doubtful, however, that uniform distribution is the appropriate approach to fairness to be used in this case, as it would also require YouTube to give space to extreme political positions that have little representation in the general population.

As with [35], this study does not take into consideration the role of content consumers in shaping the search ranking algorithm that YouTube employs. As the authors admit, this is a limitation of the study, for if the political bias found is mostly due to consumer preferences, it is debatable that such a bias is unfair at all, rather than being an expression of the Power exercised by content consumers over the search ranking algorithm (and thus over YouTube itself) by means of their autonomous consumption choices. Both papers, moreover, do not go into the role that policymakers have in shaping the Criteria, and thus the Outcomes. This neglect is arguably justified, insofar as, at least in solidly democratic countries, policy interventions are mostly concerned with illegal and harmful content, while free speech guarantees make it so that policymakers have little Power when it comes to influencing the Criteria and Outcomes involving legal content items.

Importantly, the considerations above are not meant to diminish the value of the studies examined. They are merely intended as illustrations of how the ACROCPoLis framework can help furnish a fuller picture of what pieces of information are needed in order to provide more strongly substantiated fairness assessments of specific situations and processes. Indeed, the ACROCPoLis analyses above point out limitations in the studies examined, thus revealing further research questions that need to be explored to complement their findings.

Finally, it should not have escaped the reader's attention that a glaring gap in the foregoing examination, and in the studies themselves, is the Criteria component. This is due to the fact that YouTube's algorithms are proprietary and closed, making it so that the Power that different Actors exercise in shaping it, as well as how the algorithms lead to the observed outcomes, is only partially inferrable by input-output testing of the platform. Moreover, it is likely that YouTube's algorithms are constantly in flux, with fixes and tweaks introduced by the company to improve their working in light of the platform's interests and the regulatory requirements it is called by policymakers to respect. This makes it so that a fully adequate fairness assessment of YouTube, by the lights of ACROCPoLis, cannot currently be produced. As an aside, it is worthwhile to point out that the counterfactual scenario in which YouTube's algorithms are made openly available, either by the company or by successful reverse-engineering, would importantly change the ACROCPoLis analyses. After all, it is plausible that in such a situation more of the Power would move to the hands of content creators, hence changing the Outcomes produced (potentially in undesirable and/or unfair ways).

5 CONCLUDING REMARKS

With ACROCPoLis, we have proposed a common, uniform model to represent and analyze fairness statements. In this paper, we highlight the potential for practical use with an analysis of the well-known COMPAS case, as well as analyses of two recent studies focused on biases in YouTube. Still, ACROCPoLis is merely a starting point on the road toward operationalizing algorithmic fairness modeling. Further applications are needed to assess, validate and improve ACROCPoLis in order to resolve potential weaknesses, and to formulate extensions as well as refinements to better catch conceptual nuances that may be of substantial practical relevance.

ACROCPoLis is mostly a descriptive framework that supports the identification of the relevant aspects to take into consideration in fairness assessments. A crucial part of any fairness analysis is, however, normative: one needs to decide whether the situation, as presented, is fair or not, and why. In particular, it is necessary to identify which fairness questions or situations do not fit into the framework as it stands. Of particular importance is the identification of aspects that are relevant for the specification of fairness assessments that ACROCPoLis may be leaving out. Our next step in this direction is to apply ACROCPoLis to the domain of law and legal reasoning [54], which has a long tradition of developing frameworks and theories that facilitate fair and "ethical" decision-making, and whose application is central to fundamental societal challenges such as the facilitation of democracy and human rights.

Further work is needed to address the following issues (this list is non-exhaustive, but intended to serve as a starting point for future research and dialog across disciplines and stakeholders):

- Is the framework flexible enough to maintain its adequacy as AI-based technology and associated fairness concerns evolve?
- To what extent does ACROCPoLis support interdisciplinary communication and public discourse, considering that

conceptions of fairness and how fairness should be modeled differ substantially across fields and communities?
- What are the challenges around operationalizing the framework at scale in real-world socio-technical systems?
- Does ACROCPoLis support procedural notions of fairness [49], in which the fairness of the *process* is put into focus, i.e., where procedural aspects of a decision matter just as much or more than the consideration of its inputs and outputs?
- How can ACROCPoLis accommodate the potential future scenario of AI systems that attain the status of moral agents?
- Is a framework such as ACROCPoLis sufficient, and useful, to handle questions "beyond" fairness, in ways that support cross-disciplinary communication?

Addressing these issues will require a cross-disciplinary and participatory approach, in which the aim of a shared understanding of how to interpret concrete fairness situations is central. A key idea for future work is to apply ACROCPoLis at the design stage of intelligent systems. By modeling what role the system will play within a wider process, the clarity of the framework allows us to pose key questions about its context, purpose and effects, including whether an AI system is needed in the first place. These considerations have been dubbed "hard choices" [19], pointing to an "overlap between design decisions and major sociotechnical challenges", which frameworks such as ACROCPoLis can help clarify. An interdisciplinary view is essential for this purpose.

In the context of real-world applications, the use of ACROCPoLis by different domain experts should be studied. In this sense, we envision a participatory framework in which the ACROCPoLis framework can be filled by participants according to their vision and expertise, and serves as a shared language to describe which are the key aspects to analyse the fairness of a scenario. There may be a need to formulate extensions as well as gradual refinements to better catch conceptual nuances that may be of substantial practical relevance. This highlights key requirements for the use of the framework: the application of ACROCPoLis requires a substantial time investment, as well as domain expertise from multiple perspectives. We do not see this as a limitation: both factors are necessities for a comprehensive perspective on fairness at the design stage.

As particular future study objectives, we aim to assess to what extent ACROCPoLis can, in the context of complex sociotechnical systems *i)* discover fairness problems that otherwise would remain hidden; *ii)* help improve the alignment of different stakeholders with respect to fairness; *iii)* affect system change that leads to an (objective or perceived) increase in fairness. Such studies can draw upon a range of well-established social science and information systems research methods. However, we recommend caution during study design and execution; in particular, we want to highlight that attempts to quantify the efficacy of ACROCPoLis either generally or in a particular context can be dangerous and may contradict the design intention of the framework.

Moreover, we must point to the central role of modelers in the fairness debate. Indeed, model creators have the power to frame how fairness is assessed, and thus the outcome of fairness evaluations. For example, Power (or proxies thereof, such as money) may be considered a Resource (which implies it can be distributed), or it may be modeled as an Actor's attribute, which may obfuscate the possibility of redistributing power as a potential solution to an unfair situation. Importantly, a risk that needs to be further examined is whether frameworks like ACROCPoLis may lead to the neglect of relevant perspectives that are distant from the ones of the modelers themselves.

In this sense, a particularly challenging aspect is the consideration of latent interests or unorganized groups. There may be outcomes or resources that do not relate to any particular agent groups, such as the erosion of democracy in the case of YouTube. Similarly, unorganized stakeholder groups which may be affected by a system but do not constitute a "group" in terms of shared interests, values or attributes may be overlooked by the requirement to name Actors. This kind of shortcoming is sometimes bridged by including open participation in some steps of the design (e.g. evaluation [9]), which could be done in the setting of ACROCPoLis as well by asking the wider public for comment on different ACROCPoLis models of the same situation.

Finally, we contend that the conceptual assessment and refinement of ACROCPoLis is in fact more important than its mathematical formalization or its implementation in an IT system-executable format: societal validation must precede technical verification, or the creation of a verifiable specification. Moreover, there is a growing debate about whether fairness is sufficient as a criterion when it comes to addressing algorithmic harms, with scholars calling for an orientation towards justice and dismantling oppressive structures [6, 12, 31].

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their thoughtful feedback, which has led us to substantially expand the discussion of ACROCPoLis in this paper.

REFERENCES

[1] Axel Abels, Tom Lenaerts, Vito Trianni, and Ann Nowé. 2021. Dealing with Expert Bias in Collective Decision-Making. *arXiv preprint arXiv:2106.13539* (2021).
[2] Andrea Aler Tubella, Flavia Barsotti, Rüya Gökhan Koçer, and Julian Alfredo Mendez. 2022. Ethical implications of fairness interventions: what might be hidden behind engineering choices? *Ethics and Information Technology* 24, 12 (2022). https://doi.org/10.1007/s10676-022-09636-z
[3] Jane Arthurs, Sophia Drakopoulou, and Alessandro Gandini. 2018. Researching youtube. , 3–15 pages.
[4] Chelsea Barabas, Colin Doyle, JB Rubinovitz, and Karthik Dinakar. 2020. Studying up: Reorienting the Study of Algorithmic Fairness around Issues of Power. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (Barcelona, Spain) *(FAT* '20)*. Association for Computing Machinery, New York, NY, USA, 167–176. https://doi.org/10.1145/3351095.3372859
[5] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2019. *Fairness and Machine Learning*. fairmlbook.org. http://www.fairmlbook.org
[6] Cynthia L. Bennett and Os Keyes. 2020. What is the Point of Fairness? Disability, AI and the Complexity of Justice. *SIGACCESS Access. Comput.* (mar 2020), 1 pages. https://doi.org/10.1145/3386296.3386301
[7] Abeba Birhane. 2021. Algorithmic injustice: a relational ethics approach. *Patterns* 2, 2 (2021), 100205. https://doi.org/10.1016/j.patter.2021.100205
[8] Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (Technology) is Power: A Critical Survey of "Bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 5454–5476. https://doi.org/10.18653/v1/2020.acl-main.485
[9] Christine Boshuijzen-van Burken, Niek Mouter, Shannon Spruit, and Lotte Fillerup. 2023. *Value Sensitive Design Meets Participatory Value Evaluation for Autonomous Systems in Defence*. Technical Report. EasyChair.
[10] Joy Buolamwini and Timnit Gebru. 2018. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Conference on Fairness, Accountability and Transparency, FAT 2018, 23-24 February 2018, New York, NY, USA*

(Proceedings of Machine Learning Research, Vol. 81), Sorelle A. Friedler and Christo Wilson (Eds.). PMLR, 77–91. http://proceedings.mlr.press/v81/buolamwini18a.html

[11] Raja Chatila, Kay Firth-Butterfield, and John C Havens. 2018. *Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems Version 2.* Technical Report. University of Southern California Los Angeles. https://apps.dtic.mil/sti/pdfs/AD1170922.pdf

[12] Marika Cifor, Patricia Garcia, T.L. Cowan, Jasmine Rault, Tonia Sutherland, Anita Say Chan, Jennifer Rode, Anna Lauren Hoffmann, Niloufar Salehi, and Lisa Nakamura. 2019. Feminist Data Manifest-No. https://www.manifestno.com/

[13] Sam Corbett-Davies and Sharad Goel. 2018. The Measure and Mismeasure of Fairness: A Critical Review of Fair Machine Learning. https://doi.org/10.48550/ARXIV.1808.00023

[14] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. Algorithmic Decision Making and the Cost of Fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. https://doi.org/10.1145/3097983.3098095

[15] Kimberle Crenshaw. 1991. Mapping the Margins: Intersectionality, Identity Politics, and Violence against Women of Color. *Stanford Law Review* 43, 6 (1991), 1241–1299. https://doi.org/10.2307/1229039

[16] Catherine D'Ignazio and Lauren F. Klein. 2020. *Data Feminism* (1 ed.). MIT Press.

[17] Virginia Dignum. 2021. The myth of complete ai-fairness. In *International Conference on Artificial Intelligence in Medicine*. Springer, 3–8.

[18] Virginia Dignum. 2022. Relational Artificial Intelligence. *arXiv preprint arXiv:2202.07446* (2022).

[19] Roel Dobbe, Thomas Krendl Gilbert, and Yonatan Mintz. 2021. Hard choices in artificial intelligence. *Artificial Intelligence* 300 (2021), 103555.

[20] Mateusz Dolata, Stefan Feuerriegel, and Gerhard Schwabe. 2022. A sociotechnical view of algorithmic fairness. *Information Systems Journal* 32, 4 (2022), 754–818.

[21] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. 2012. Fairness through awareness. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, Shafi Goldwasser (Ed.). ACM, 214–226. https://doi.org/10.1145/2090236.2090255

[22] EUCommission. 2018. ETHICS GUIDELINES FOR TRUSTWORTHY AI. *High-Level Expert Group on Artificial Intelligence* (2018). Issue December. https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-aihttps://ec.europa.eu/digital-single-market/en/high-level-expert-group-artificial-intelligence

[23] Sina Fazelpour and Zachary C. Lipton. 2020. Algorithmic Fairness from a Non-ideal Perspective. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. ACM. https://doi.org/10.1145/3375627.3375828

[24] Sina Fazelpour, Zachary C. Lipton, and David Danks. 2021. Algorithmic Fairness and the Situated Dynamics of Justice. *Canadian Journal of Philosophy* (oct 2021), 1–17. https://doi.org/10.1017/can.2021.24

[25] Elias Fernández Domingos, Inês Terrucha, Rémi Suchon, Jelena Grujić, Juan C Burguillo, Francisco C Santos, and Tom Lenaerts. 2022. Delegation to artificial agents fosters prosocial behaviors in the collective risk dilemma. *Scientific Reports* 12, 1 (2022), 1–12.

[26] Stefan Feuerriegel, Mateusz Dolata, and Gerhard Schwabe. 2020. Fair AI: Challenges and Opportunities. *Business & Information Systems Engineering* 62 (05 2020). https://doi.org/10.1007/s12599-020-00650-3

[27] Sorelle A. Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. 2021. The (Im)possibility of fairness. *Commun. ACM* 64, 4 (apr 2021), 136–143. https://doi.org/10.1145/3433949

[28] Seraphina Goldfarb-Tarrant, Rebecca Marchant, Ricardo Muñoz, Mu– Muñoz Sánchez, Mugdha Pandya, and Adam Lopez. 2021. Intrinsic Bias Metrics Do Not Correlate with Application Bias. In *Proceedings ofthe 59th Annual Meeting ofthe Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics (ACL), 1926–1940. https://doi.org/10.18653/v1/2021.acl-long.150 arXiv:2012.15859

[29] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of Opportunity in Supervised Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 3315–3323. https://proceedings.neurips.cc/paper/2016/hash/9d2682367c3935defcb1f9e247a97c0d-Abstract.html

[30] Brian Hedden. 2021. On statistical criteria of algorithmic fairness. *Philosophy & Public Affairs* 49, 2 (mar 2021), 209–231. https://doi.org/10.1111/papa.12189

[31] Anna Lauren Hoffmann. 2021. Even When You Are a Solution You Are a Problem: An Uncomfortable Reflection on Feminist Data Ethics. *Global Perspectives* 2, 1 (03 2021). https://doi.org/10.1525/gp.2021.21335 arXiv:https://online.ucpress.edu/gp/article-pdf/2/1/21335/462728/globalperspectives_2021_2_1_21335.pdf 21335.

[32] Gordon Hull. 2022. Dirty Data Labeled Dirt Cheap: Epistemic Injustice in Machine Learning Systems. *SSRN Electronic Journal* (2022). https://doi.org/10.2139/ssrn.4137697

[33] Matthew Joseph, Michael Kearns, Jamie Morgenstern, and Aaron Roth. 2016. Fairness in Learning: Classic and Contextual Bandits. https://doi.org/10.48550/ARXIV.1605.07139

[34] Michael Kearns, Aaron Roth, and Saeed Sharifi-Malvajerdi. 2019. Average individual fairness: algorithms, generalization and experiments.

[35] Baris Kirdemir, Joseph Kready, Esther Mead, Muhammad Nihal Hussain, and Nitin Agarwal. 2021. Examining Video Recommendation Bias on YouTube. In *Advances in Bias and Fairness in Information Retrieval*, Ludovico Boratto, Stefano Faralli, Mirko Marras, and Giovanni Stilo (Eds.). Springer International Publishing, Cham, 106–116. https://doi.org/10.1007/978-3-030-78818-6_10

[36] Svetlana Kiritchenko and Saif M Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. *arXiv preprint arXiv:1805.04508* (2018).

[37] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. 2016. Inherent Trade-Offs in the Fair Determination of Risk Scores. https://doi.org/10.48550/ARXIV.1609.05807

[38] Susan Leavy. 2018. Gender bias in artificial intelligence: The need for diversity and gender theory in machine learning. In *Proceedings - International Conference on Software Engineering*. Association for Computing Machinery, New York, New York, USA, 14–16. https://doi.org/10.1145/3195570.3195580

[39] Lydia T. Liu, Sarah Dean, Esther Rolf, Max Simchowitz, and Moritz Hardt. 2018. Delayed Impact of Fair Machine Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 3150–3158. https://proceedings.mlr.press/v80/liu18c.html

[40] Michael Lutz, Sanjana Gadaginmath, Natraj Vairavan, and Phil Mui. 2021. Examining Political Bias within YouTube Search and Recommendation Algorithms. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. 1–7. https://doi.org/10.1109/SSCI50451.2021.9660012

[41] Melissa McCradden, Shalmali Joshi, Mjaye Mazwi, and James A Anderson. 2020. When Your Only Tool Is A Hammer: Ethical Limitations of Algorithmic Fairness Solutions in Healthcare Machine Learning. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. Association for Computing Machinery, 109. https://doi.org/10.1145/3375627.3375824

[42] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *ACM Comput. Surv.* 54, 6, Article 115 (jul 2021), 35 pages. https://doi.org/10.1145/3457607

[43] Milagros Miceli, Julian Posada, and Tianling Yang. 2022. Studying Up Machine Learning Data: Why Talk About Bias When We Mean Power? *Proc. ACM Hum.-Comput. Interact.* 6, GROUP, Article 34 (jan 2022), 14 pages. https://doi.org/10.1145/3492853

[44] Thomas Miconi. 2017. The impossibility of "fairness": a generalized impossibility result for decisions. https://doi.org/10.48550/ARXIV.1707.01195

[45] Shira Mitchell, Eric Potash, Solon Barocas, Alexander D'Amour, and Kristian Lum. 2021. Algorithmic Fairness: Choices, Assumptions, and Definitions. *Annual Review of Statistics and Its Application* 8, 1 (mar 2021), 141–163. https://doi.org/10.1146/annurev-statistics-042720-125902

[46] Hervé Moulin. 2019. Fair Division in the Internet Age. *Annual Review of Economics* 11, 1 (2019), 407–441. https://doi.org/10.1146/annurev-economics-080218-025559 arXiv:https://doi.org/10.1146/annurev-economics-080218-025559

[47] Lawrence Page. 1998. Method for node ranking in a linked database. https://patents.google.com/patent/US6285999B1/en

[48] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Stanford InfoLab. http://ilpubs.stanford.edu:8090/422/ Previous number = SIDL-WP-1999-0120.

[49] Nick Pearce. 2007. Rethinking fairness. *Public Policy Research* 14, 1 (2007), 11–22. https://doi.org/10.1111/j.1744-540X.2007.00458.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1744-540X.2007.00458.x

[50] Inioluwa Deborah Raji, Andrew Smart, Rebecca N. White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. 2020. Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (Barcelona, Spain) *(FAT* '20)*. Association for Computing Machinery, New York, NY, USA, 33–44. https://doi.org/10.1145/3351095.3372873

[51] John Rawls. 2004. A theory of justice. In *Ethics*. Routledge, 229–234.

[52] Andrew D. Selbst, Danah Boyd, Sorelle A. Friedler, Suresh Venkatasubramanian, and Janet Vertesi. 2019. Fairness and Abstraction in Sociotechnical Systems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM. https://doi.org/10.1145/3287560.3287598

[53] Jonathan Stray, Alon Y. Halevy, Parisa Assar, Dylan Hadfield-Menell, Craig Boutilier, Amar Ashar, Lex Beattie, Michael D. Ekstrand, Claire Leibowicz, Connie Moon Sehat, Sara Johansen, Lianne Kerlin, David Vickrey, Spandana Singh, Sanne Vrijenhoek, Amy X. Zhang, McKane Andrus, Natali Helberger, Polina Proutskova, Tanushree Mitra, and Nina Vasan. 2022. Building Human Values into Recommender Systems: An Interdisciplinary Synthesis. *CoRR* abs/2207.10192 (2022). https://doi.org/10.48550/arXiv.2207.10192 arXiv:2207.10192

[54] Harry Surden. 2020. The ethics of artificial intelligence in law: Basic questions. *Forthcoming chapter in Oxford Handbook of Ethics of AI* (2020), 19–29.

[55] Sahil Verma and Julia Rubin. 2018. Fairness definitions explained. In *Proceedings - International Conference on Software Engineering* (New York, NY, USA). IEEE Computer Society, 1–7. https://doi.org/10.1145/3194770.3194776
[56] Pak-Hang Wong. 2020. Democratizing Algorithmic Fairness. *Philosophy & Technology* 33, 2 (jun 2020), 225–244. https://doi.org/10.1007/s13347-019-00355-w
[57] Jie Xu, Yunyu Xiao, Wendy Hui Wang, Yue Ning, Elizabeth A Shenkman, Jiang Bian, and Fei Wang. 2022. Algorithmic fairness in computational medicine. *EBioMedicine* 84 (2022), 104250.

Paper III

A Clearer View on Fairness: Visual and Formal Representations for Comparative Analysis

Julian Alfredo Mendez*
0000−0002−7383−0529
julian.mendez@cs.umu.se

Timotheus Kampik
0000−0002−6458−2252
tkampik@cs.umu.se

Andrea Aler Tubella
0000−0002−8423−8029
andrea.aler@upc.edu

Virginia Dignum
0000−0001−7409−5813
virginia@cs.umu.se

Department of Computing Science, *Umeå University*, Umeå, Sweden

***Abstract*—The opaque nature of machine learning systems has raised concerns about whether these systems can guarantee fairness. Furthermore, ensuring fair decision making requires the consideration of multiple perspectives on fairness. At the moment, there is no agreement on the definitions of fairness, achieving shared interpretations is difficult, and there is no unified formal language to describe them. Current definitions are implicit in the operationalization of systems, making their comparison difficult. In this paper, we propose a framework for specifying formal representations of fairness that allows instantiating, visualizing, and comparing different interpretations of fairness. Our framework provides a meta-model for comparative analysis. We present several examples that consider different definitions of fairness, as well as an open-source implementation that uses the object-oriented functional language SODA.**

***Index Terms*—Responsible artificial intelligence · Ethics in artificial intelligence · Formal representation of fairness**

I. INTRODUCTION

A key challenge in ensuring or assessing fairness is the heterogeneity of perspectives on fairness, because there is no canonical definition of what is fair and what is not. In particular, fairness is not a "one-size-fits-all"-problem: there is no unique operationalizable definition of fairness. In fact, research in various areas of formal definitions of fairness has increased considerably [15]. In the machine learning community, different frameworks have been presented to quantify fairness in classification [3], [5]. Even if fairness can be seen as "the absence of prejudice or favoritism towards an individual or group based on its inherent or acquired characteristics" [29], different criteria can be used to determine fairness of decisions, and many of them should be specifically formulated to be clear to those involved. Determining what is fair varies between cultures [10], and even within the same culture, different individuals can perceive fairness differently [13].

Agreeing on a particular notion of fairness or facilitating an understanding of the diversity of perspectives on fairness can avoid conflicts. A structured discussion and analysis of fairness requires a framework for specifying and comparing perspectives on fairness to enable the elicitation of differences and ultimately desiderata that stakeholders can agree on. Although agreements on the interpretation of fairness or other societal values are complex, a growing number of approaches are being proposed at both theoretical and practical levels, particularly following the Design for Values methods [35], [16], [36].

This paper uses the ACROCPoLis framework [2], which provides a shared vocabulary for fairness assessments, making explicit the relevant factors and their relations. This allows for comparison of similar situations, highlighting differences in dissimilar situations, and capturing different interpretations by different stakeholders. This framework is the underpinning to obtain an applicable framework for operationalizing fairness by:

i. introducing Tiles (Transparent, Intuitive, Logical, Ethical, and Structured), a visual specification language especially tailored for fairness definitions;
ii. presenting a formal meta-model and examples of fairness definitions using Tiles; and
iii. providing an implementation of Tiles in an object-oriented functional language.

The remaining sections are structured as follows. Section II provides an overview of the state of the art, and in particular of challenges regarding the formalization of fairness. Then, Section III provides an informal conceptualization of fairness (drawing from existing research) and introduces a formal meta-model for fairness, as well as Tiles, the corresponding approach to implementation and visualization of fairness models. Formalization and implementation are illustrated using several simple examples in Section IV. Finally, we conclude the article with a discussion of related work and an outline of future research directions in Section V.

II. BACKGROUND

While fairness is a crucial societal concept, its definition, even in a specific context, is typically subjective. For example, when a state provides childcare subsidies to a family, a "fair" distribution may be colloquially defined in the following ways, among others:

- per child, every family receives the same amount of subsidies;
- per child, subsidies depend on family income, i.e., the amount of subsidies increases with decreasing income;

* Corresponding author. The authorship order is by relative overall contributions to the manuscript.

- per child, subsidies depend on family income and the number of older siblings, i.e., the amount of subsidies per child increases with an increasing number of children.

Each option may be considered fair; one cannot objectively stipulate that one option is necessarily "fairer" than the other. Different communities may have different opinions about what a fair childcare subsidy is [8]. For example, the province of Manitoba, Canada, considers these relevant factors: family income, number and age of the children, number of days required for care, and reason for care [21]. Similarly, the Australian Government publishes a structure diagram of how some factors weigh on the allocation of the childcare subsidy, especially income [12], as the subsidy rate is lowered, in stages, as family income increases, and reaches zero for families with an annual income of or above 352,453 AUD (in 2019-2020).

Comparing different scenarios is a complex task, especially for those who are not specialized in the topic. Thus, a formal diagram can help visualize the differences between criteria of two different countries, or the same country at different points in time. However, creating a system to design such diagrams is challenging, as informal descriptions carry the risk of inconsistencies and flawed modeling. This risk may be reduced if we are able to categorize the different fairness scenarios and provide pre-built consistent blocks to model them. Each block works as a logical unit that is small enough to be fully understood, but powerful enough to require only a few blocks for a standard diagram.

Two prominent categories of *scenarios* pertaining to fairness are *resource allocation scenarios* and *scoring scenarios*. Given a group of individuals, resource allocation scenarios focus on how to find an optimal allocation of a fixed amount of resources [24]. The value of resources is abstracted by a *utility function*, which is a function that gives a comparable value to resources. The utility function may represent qualities or quantities, such as money, time, weight, and size. Implementing fairness in resource allocation is a challenging task because fairness and efficiency are competing objectives [6]. The Gini index [19], [20] and the points on the Lorenz curve [17], [18] are well-known approaches to fairness in resource allocation scenarios and provide frequently used measures for wealth (in)equality in a macroeconomic context.

Scoring scenarios focus on how fair a scoring of a group of individuals is based on their individual attributes. Individuals receive a score based on their attributes, abstracted by a *scoring function*, which is a function that gives a comparable score to individuals with respect to some aspect. This score may assess the likelihood that an individual is able to repay a loan or is a good fit for a particular job position.

To check whether the scoring function itself is fair with different individuals, we could use a counterfactual check [25], especially considering that protected attributes, such as gender, ethnic origin, social status, age, and sexual orientation, can be "noisy", and produce unfair scoring [30]. However, removing or exchanging protected attributes could have limitations, as attributes often contain confounding factors and correlations that are difficult to disentangle or even detect. We consider the scenarios presented in [26] as a reference to identify common real-world scenarios, where machine learning-based decision making is used. We compare the scenarios in Table I.

Other scenarios include insurance policy prediction [38], income prediction [28], equal opportunity policies for health care [33], teacher evaluation and promotion [9], online recommendation [23], and university ranking [27], [34].

With the rise of data science and machine learning in recent years, research interest in statistical notions of fairness has increased. Here, the most prominent examples are *group fairness* and *individual fairness* [11]:

- *Group fairness* intuitively stipulates that groups that are separated by protected properties (such as gender) are to be treated in the same manner, i.e. that outcomes must not differ, given everything else is equal between the groups.
- *Individual fairness* intuitively stipulates that individuals that are similar given their non-protected properties should be treated in a similar manner.

Recent works attempt to reconcile the supposed conflict between group and individual fairness, but also call into question the sufficiency of the statistical measures that operationalize the concepts, and in particular individual fairness. For example, claims of individual fairness can also exacerbate existing biases that may then be reflected in the selection of desirable, non-protected properties [14]. Furthermore, decisions made to mitigate bias are not value-free [1].

Still, tools for operationalizing fairness, such as IBM's *AI Fairness* 360 [4], Google's *What-if* tool [39], and Microsoft's *Fairlearn* [7], depend on these highly specific statistical formalizations that reflect group or individual fairness notions. They also assume that high-quality data is available in a rather unambiguous context that allows for the societally beneficial operationalization of fairness using these notions. Considering the recent academic discourse on the diversity and heterogeneity of fairness definitions that are needed to facilitate nuanced analysis and ultimately outcomes that are societally desirable [2], [14], it is striking that there are no formal meta-models of fairness that can instantiate a broad range of fairness definitions and scenarios from different points of view.

III. Formalization and Representation

Since our objective is to introduce an implementable and ultimately operationalizable approach to instantiate and compare context-dependent fairness definitions, our fairness formalization is grounded in conceptual approaches to fairness of societal relevance. As observed in the previous section, fairness typically pertains to decisions or actions that are made based on the attributes of specific agents or groups thereof. Each decision or action has a resource allocation or score as an outcome. Decisions or actions can be abstract, e.g., the execution of an action can be seen as assigning a score or as the use of a resource. Somewhat reflecting this intuition, we previously introduced ACROCPoLis, a conceptual framework for making sense of fairness [2].

TABLE I
COMPARISON OF REAL-WORLD SCENARIOS.

Scenario	Relevant Attributes (Input)	Outcome (Output)
Job hiring	affiliation, education level, job experience, IQ score, age, gender, address	a decision and/or a score
Granting loans	credit history, purpose of the loan, loan amount requested, employment status, income, marital status, gender, age, address, housing status, and credit score	decision and/or score
College admission	institutions previously attended, SAT scores, extracurricular activities, GPAs, test scores, interview score	decision or score
Criminal risk assessment	number of arrests, type of crime, address, employment status, marital status, income, age, housing status	score and decision
Child maltreatment prediction	contemporaneous and historical information for children and caregivers	score (likelihood) and decision
Health care	disease (chronic conditions) prediction include vital signs, blood test, sociodemographic data, education, health insurance, home ownership, age, race, address	score (likelihood)
Facial analysis	face (image)	decision

ACROCPoLis identifies six entities that are general to model fairness scenarios: *Actors*, *Context*, *Resources*, *Outcome*, *Criteria*, and *Power*, as well as the *Links* connecting them. In order to make the ACROCPoLis framework usable, we made decisions on the formalization, which required a trade-off between simplicity and generality. In our approach, we consider Actors, Context, Resources, and Outcome, and we add Measure, Aggregation, and Attribute, as we describe in Table II. We encode Criteria, Power, and Links indirectly in the other entities. Criteria are the explicit or implicit aspects needed to make a decision, affect, or justify the outcome. We interpret Power as an attribute of actors, which could be indirectly used from the Context. Links are the relations included in the attributes and in the aggregations.

This section introduces our formal meta-model of fairness and explains how the meta-model can be applied to instantiate fairness scenarios, with the notation that we provide.

A. Meta-model

Our meta-model requires two sets: I, which is a non-empty set of identifiers, and M, which is a non-empty set of measures. For the set of identifiers I, we also require a relation '$\leq$' that is a *total order*. This means that, for every $a_1, a_2, a_3 \in I$,

1) $a_1 \leq a_1$ (reflexive);
2) if $a_1 \leq a_2$ and $a_2 \leq a_3$, then $a_1 \leq a_3$ (transitive);
3) if $a_1 \leq a_2$ and $a_2 \leq a_1$, then $a_1 = a_2$ (antisymmetric);
4) $a_1 \leq a_2$ or $a_2 \leq a_1$ (strongly connected).

Some data types that could implement I are a set of strings with alphabetical order, or a set of integers with a 'less than or equals to' relation, or any other possibly infinite set with a total order.

For the set of measures M, we require it to be a subset of the real numbers $\mathbb{R}$ enriched with a distinguished element `NaN` ('Not a Number'), with the usual total order '$\leq$' for $\mathbb{R}$, and basic operations, like addition, subtraction, multiplication, and division. M could be implemented by a floating point data type [22]. In fact, `NaN` is a particular value of numeric data types, such as the floating point number, and captures cases where operations on floating point are undefined, e.g., when dividing by 0.

Once I and M are defined, we can identify a specific fairness scenario, which we call a *context*, and we just use an identifier $c \in I$ to refer to this. We do not need more structural information regarding the context, because all the relevant information of the context is in fact in other components of the tuple. Similarly to the case of the context, we identify the actors and resources by their identifiers, allowing functions on them to provide relevant information about them. The set of actors is Ac and the set of resources is R, and both are subsets of I, i.e. $Ac \subseteq I$ and $R \subseteq I$. We also require that there are no common identifiers in both sets, and that both do not contain c, i.e. $Ac \cap R = \emptyset$ and $c \notin Ac$, $c \notin R$.

Up to this point, we have defined the basic sets of identifiers (I) and measures (M), and some relevant elements of I, such as the context c, the elements of Ac and the elements of R. With these defined, we can define a set of attributes, which we call At. This set is in fact a finite set of functions f that take an identifier in Ac or R, and return either another identifier in Ac or R, or a measure in M. To denote this, we define $Fun(A, B)$ as the set of functions from A to B:

$$Fun(A, B) := \{f \mid f : A \to B\}.$$

Then, we require that the following holds:

$$\begin{aligned} At \subseteq\ & Fun(Ac, Ac) \cup Fun(R, Ac) \cup \\ & Fun(Ac, R) \cup Fun(R, R) \cup \\ & Fun(Ac, M) \cup Fun(R, M). \end{aligned}$$

We define the set of aggregation functions as a finite and possibly empty set Ag that contains only functions that can operate on any finite sequence of elements in either identifiers in Ac, identifiers in R, or measures in M, and return a single element of the same set as the domain. This can be denoted as follows. Let $Agg_n(A)$ be defined as the set of functions in sequences of elements of A of length n to an element of A, denoted by:

$$Agg_n(A) := \{f \mid f : A^n \to A\},$$

TABLE II
ENTITIES

Entity	Meaning	Relation to ACROCPoLis
Actor	is an individual or organization that participates in the fairness scenario, either by receiving resources, distributing resources, or affecting the distribution of resources.	the same as *Actor*
Context	is an entity that contains relevant contextual and structural factors in a fairness scenario.	the same as *Context*
Resource	is a measurable element to be distributed to the actors involved in a fairness scenario.	the same as *Resource*
Outcome	is the association between actors and resources in a fairness scenario.	the same as *Outcome*
Measure	is the space of quantities and qualities to measure and compare attributes of context, actors, and resources.	part of *Links*
Aggregation	is the space of functions to combine quantities and qualities and preserve them as measures.	part of *Links*
Attribute	is the space of concrete relevant features of an actor, a resource, or the context, especially reflecting a quantity or a quality.	part of *Links*, covering *Power*

where A^n denotes the n-ary Cartesian power of A. Then, we say that:

$$Ag \subseteq \bigcup_{k\in\mathbb{N}} (Agg_k(Ac) \cup Agg_k(R) \cup Agg_k(M))$$

We can define the outcome O of a scenario of fairness as a finite possibly empty set of pairs, each pair called an *assignment*, where each actor receives one resource. We can denote this as $O \subseteq \{\langle a, r\rangle \mid a \in Ac, r \in R\}$. This outcome is to be evaluated to determine whether it is fair or not according to the definition of fairness defined by human evaluators.

Given that the components are defined above and assuming that Ac, R, O, M, Ag, and At are all pairwise disjoint, we can define the tuple for a given scenario of fairness as:

$$F_c = \langle Ac, R, O, M, Ag, At\rangle.$$

We name the whole framework above AcROMAgAt. Note that I is only indirectly mentioned through its relevant elements, namely c, the elements in Ac, and the elements in R.

B. Steps to identify the entities

As described above, resource allocation scenarios are intended to allocate limited resources among actors. To identify the abstract components in this kind of scenario, we want to model whether a particular resource allocation satisfies the needs of actors according to our definition of fairness. To illustrate our definitions, we consider the entities involved in modeling a childcare subsidy scenario.

The first step is to recognize the *actors*, the *resources*, and the *context*. It might be the case that, for a given scenario, some actors are not visible or not clearly identifiable, but we focus on those receiving the resources in a particular context. In the case of the childcare subsidy scenario, each actor would be a family, the resources would be the amount paid, and the context the name of the country or territory where the subsidy is being considered.

We can then recognize the *attributes* of actors and resources that are relevant in the given context. As we learn from the requirements, some attributes would be the income of the family, the number of children, and their ages. Attributes for the resources could be the amount paid, and the currency. The *outcome* can be defined considering actors and resources, and the *measures* are those quantities and qualities that emerge from the attributes. The outcome represents how much is given to each family. Lastly, we identify *aggregations* to combine quantities and qualities and compare them. Aggregations can be seen as a collection of utility functions that help express qualities and quantities as functions of basic values. For example, if a family receives multiple childcare subsidies instead of one, an aggregation function can ensure that the total amount does not exceed the established cap per family.

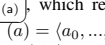
In the case of scoring scenarios, the steps are analogous, but there is an emphasis on the role played by the attributes, since the score is what is being scrutinized for fairness. As in the case of resource allocation, context attributes provide the required additional information, such as historical information. At first, we could consider scoring as the allocation of an unlimited resource, but it is a limited resource in some cases, as when choosing a candidate for a job interview, or when it is used in an examination that is later normalized among all results to follow a statistical distribution. We consider scoring the allocation of an infinite abstract resource. Intuitively, there may be an overlap between scoring and resource allocation, e.g., if school grades must follow a pre-specified distribution; in our interpretation, this is *not* a scoring scenario, because the resource is finite (given a finite set of actors). The two scenario categories are not disjoint. The same problem could be modeled as a resource allocation scenario or a scoring scenario, depending on what features are more predominant or relevant for the particular use.

C. Fairness pipelines with Tiles

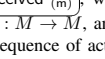
For modeling AcROMAgAt fairness scenarios, we introduce Tiles, which is a system to define rules based on the composition of building blocks (*tiles*). To demonstrate how Tiles work, we assume an abstract fairness scenario $F_c = \langle Ac, R, O, M, Ag, At\rangle$. Each tile has an identifier or function, an input, and an output, depicted as follows:

Tiles can be connected to create a *composite tile*, where the output of one tile is the input of another. They can be seen as compositions of tiles. They are connected using *connection ports* (the inputs and outputs of the function), and in some cases, a tile may have multiple input connection ports and/or multiple output connection ports. A tile with multiple input ports can be interpreted as a function with multiple parameters, or similarly of just one parameter which is a tuple of multiple ports. A tile with multiple output ports, instead, is interpreted as the replication of the output of the tile seen as a function. Multiple ports are denoted using commas, i.e. $(a_0), (a_1)$ denotes two ports of one sequence each, where both possibly empty sequences have the exact same number of elements. This allows us to re-write it as a sequence of pairs $(\langle a_0, a_1 \rangle)$.

A *pipeline* is a special case of a composite tile, which has a *starting* tile and an *ending* tile. The starting tile does not have an input, and the ending tile has a single value as output, which is usually a Boolean value. An *unfold* tile generates a sequence from a single value, for example, if given the number n, it creates a sequence of n elements. A *fold* tile generates a value from a sequence, for example, if it computes the sum of all the elements in a sequence. When configuring a pipeline, each tile can use *contextual information* and the *outcome* O all along the pipeline. The contextual information and the outcome remain constant with respect to the pipeline.

Let us see how AcROMAgAt fairness scenarios are represented by Tiles. *Actors* can be represented by the tile 

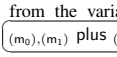
, which returns a sequence of actors, denoted by (a), i.e. $(a) = \langle a_0, ..., a_{n-1} \rangle$, where each $a_i \in Ac$, and for $1 \leq i < j \leq |Ac|$ and $a_i, a_j \in Ac$, we have $a_i \neq a_j$. This sequence is sorted by identifier.

Based on the sequence of actors, we can define a tile that retrieves the *resource* for each actor. This is achieved by the tile 

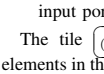
, which, given an *aggregation* function $\sigma \in Ag$, $\sigma : M \to M$, and an *attribute* $p \in At$, for each a in the input sequence of actors, returns a *measure* m such that:

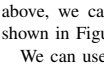
$$m = \sigma\ (\{p(r)\ \mid\ \langle a, r \rangle \in O\}).$$

To avoid verbosity in the tiles, we use the following notation conventions.

- We use a variable of a type to denote the type or the variable, depending on the context. For example, in the case of a for Ac, a can denote the type Ac or a variable of type Ac.
- We denote $(\cdot)$ as the sequence type and its elements. For example, (a) is a sequence of actors.
- We use a without index to denote an element of the sequence.
- When dealing with multiple ports, the variables in the input ports are independent from the variables in the output ports. For example, in 

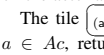
, the m_0 in the output port can be different from the m_0 in the input port.

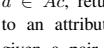
The tile 

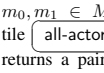
is true if and only if all the elements in the input sequence are equal. With the tiles defined above, we can define the tile 

as a pipeline as shown in Figure 1.

We can use similar definitions to encode equity, where actors receive resources according to their need, which depends on the actor and on the context, but not on the given resource.

The tile 

is a function that, for each actor $a \in Ac$, returns the need (measure) $m \in M$ with respect to an attribute $p \in At$. The tile 

, given a pair of sequences, returns true if and only if for $m_0, m_1 \in M$, each pair m_0, m_1 verifies $m_0 \geq m_1$. The tile 

works similarly to

, but returns a pair of sequences, where each pair duplicates the same actor, for parallel processing. Figure 2 shows how we encode equity.

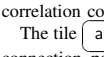
We see how we distinguish connections between tiles by giving subindices to their connecting variables, regardless of the fact that a_0 and a_1 are the same actor.

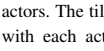
A tile pipeline, such as the one in Figure 2, can intuitively be seen as a directed acyclic graph, where the tiles are the vertices, the starting tiles are the source vertices, the ending tiles are the sink vertices, the edges are the connections between tiles, and the edge direction is implicit by connecting the output of one tile to the input of another.

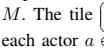
D. Tiles *for scoring scenarios*

Based on Table I, we provide tiles centered on statistical approaches for scoring scenarios. In Figure 3, we present one possible pipeline of tiles to determine whether there is a correlation between an attribute and the performance of a prediction on individuals. Finding a correlation between values does not ensure causality, but it can serve as an indicator to detect possible unfair situations.

We assume that there is a threshold such that the values m above that threshold are positive and those below are negative. Alternatively, the implementation of these tiles could abstract such a threshold by returning Boolean values true or false. Without loss of generality, we assume that m is 0 for false and 1 for true. We use these values to calculate the Pearson correlation coefficient [37].

The tile 

is a tile that allows for three connection ports and produces three identical sequences of actors. The tile 

takes a sequence of actors, with each actor $a \in Ac$, returns the predicted values with respect to an attribute $p \in At$ as a sequence of measures $m \in M$. The tile 

takes a sequence of actors, with each actor $a \in Ac$, and returns the actual values with respect to an attribute $p \in At$ as a sequence of measures, $m \in M$. In the case of the prediction of recidivism, the *prediction* can be taken from the data two years before the evaluation and the *results* from what actually happened. Both sequences are

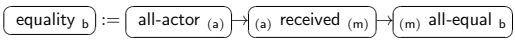


Fig. 1. Pipeline for equality: it is defined with three tiles, one producing actors, then a tile that retrieves what each actor receives, and the last one that checks whether all received the same.

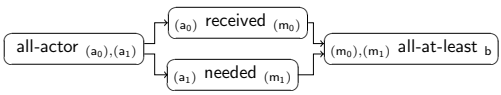


Fig. 2. Representation of equity using Tiles. The first tile on the left creates the sequence of actors that are processed in parallel, but respecting the order, by two tiles. These tiles return how much an actor received and how much the actor needs. The last tile on the right compares both values.

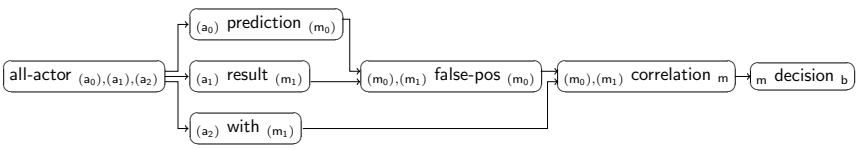


Fig. 3. Example of a configured correlation pipeline to measure the bias on false positives. The tile on the left creates triples of actors. The three branches are the original prediction on an actor ('prediction'), the actual result of an actor ('result'), and if the actor has a given property ('with'). With the original prediction and the actual result, the false positives are calculated. This, together with the characteristic of a property, is given to compute the correlation. Ultimately, we find the decision of whether there is a significant bias based on the correlation.

combined to estimate false positives, which is done by the tile (m0),(m1) false-pos (m).

The tile (m0),(m1) false-pos (m), given a pair (m_0, m_1), $m_0, m_1 \in M$, returns 1 if the pair is a false positive, and 0 otherwise. A false positive is that the prediction is 1 and the actual value is 0. The tile (m0),(m1) false-neg (m) returns 1 if the pair is a false negative, and 0 otherwise. A false negative is that the prediction is 0 and the actual value is 1. (m0),(m1) true-pos (m) and (m0),(m1) true-neg (m) are analogous, but return 1 if given (m_0, m_1), $m_0 = m_1$, and 0 otherwise. The tile (a) with (m) retrieves from all actors an attribute p, for example, the skin color. Binary attributes can be encoded with 0 and 1 to compute the correlation.

The tile (m0),(m1) correlation m computes a correlation coefficient for the subsets filtered by attributes with respect to the score. We chose the Pearson correlation coefficient, but other correlations can be used in this diagram, as long as they respect the same input/output ports. The Pearson correlation is defined, for a sample of size n, for x_i, y_i $(1 \leq i \leq n)$ individual sample points, for $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, the sample arithmetic mean, and the same for $\bar{y}$ as follows:

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

A final tile m decision b makes the decision about whether the correlation is acceptable. For example, some arbitrary categorization could define the ranges $(0, 0.3]$ as weak correlation, $(0.3, 0.5]$ as moderate correlation, and $(0.5, 1]$ as strong correlation.

E. Implementation

Tiles can be configured for specific scenarios. Each configuration should be implemented in a more fine-grained language. Considering such a configuration, we believe that the language in which Tiles can be configured should have good readability, although this is a property that is difficult to measure. We chose SODA [32], [31] because it is an object-oriented functional language, especially designed to describe, analyze, and model human-centered problems. The tiles used in the examples are summarized in Table III, and we provide an open source implementation of them at https://julianmendez.github.io/tiles .

F. Assumptions

We assume that the information we have is *consistent*, that the resources have either a *utility function* or a *score*, and that we are provided with *complete information* of the outcome, which means that we know exactly what each actor receives. In practice, we may need to detect that a system is not fair before analyzing all assignments. Nevertheless, we can still model the problem for a particular instance at a particular point in time.

Finally, another assumption is that each tile is decidable, and that the complexity of the whole pipeline does not impede the execution possibility. Although we provide the elements to check fairness and also examples, we do not state if the

TABLE III
SUMMARY OF ACROMAGAT TILES USED IN THE EXAMPLES.

Generic Tile	**Meaning**
(α) all-satisfy p b	Given a sequence of objects of type α, it returns true if and only if all the elements satisfy property p.
$(\alpha_0),(\alpha_1)$ f(α_0, α_1) (α)	Given a pair of sequences of two objects of the same type α, it returns a sequence of objects of the same type, resulting from applying the function f to both elements of the pair. If the parameters are omitted, the order is as expected. For example, for measures, $(m_0),(m_1)$ plus (m) denotes that each element m in the output sequence is computed by applying the function plus (+) to two measures, i.e. $m = m_0 + m_1$.
(α) p? (α)	Given a sequence of objects of type α, it returns a possibly empty sequence of objects of the same type such that all of them satisfy the property p.
all-actor (a)	Returns a sorted sequence of actors (a), where each $a \in Ac$ occurs exactly once.
(a) received (m)	Given a sequence of actors (a), with $a \in Ac$, it returns a sequence of measures (m), $m \in M$, such that each m is the aggregated value using the aggregation function σ applied to the set produced by the resource attribute p, based on the outcome O.
(m) all-equal b	Given a sequence of measures (m), $m \in M$, it returns true if all values are equal.
Customized Tile	**Meaning**
(a) needed (m)	Given a sequence (a), for each $a \in Ac$, and the attribute $p \in At$, it returns a sequence of measures (m), where each $m \in M$ has the need of that actor with respect to p.
$(m_0),(m_1)$ all-at-least b	Given a pair of sequences $(m_0),(m_1)$, where each $m_0, m_1 \in M$, it returns true if for all pairs, $m_0 \geq m_1$, and it returns false otherwise.
(a) prediction (m)	Given a sequence of actors (a), it returns a sequence of measures (m), such that for each actor $a \in Ac$, for a measure $m \in M$, it holds that $m = 1$ if based on the outcome O the prediction with respect to an attribute $p \in At$ is positive, and $m = 0$ if it is negative.
(a) result (m)	Given a sequence of actors (a), it returns a sequence of measures (m), such that for each actor $a \in Ac$, for a measure $m \in M$, it holds that $m = 1$ if based on contextual information in c, the result with respect to an attribute $p \in At$ was positive, and $m = 0$ if it was negative.
$(m_0),(m_1)$ false-pos (m)	Given a pair of sequences $(m_0),(m_1)$, where each $m_0, m_1 \in M$, it returns a sequence of measures (m), $m \in M$, such that $m = 1$ if the value of $m_0 = 1$ and $m_1 = 0$, and $m = 0$ otherwise.
(a) with (m)	Given a sequence of actors (a), $a \in Ac$, it returns a sequence of measures $m \in M$ containing the characteristic value: 1 for those actors that have the attribute p and 0 otherwise.
$(m_0),(m_1)$ correlation m	Given a pair of sequences of measures, $(m_0),(m_1)$, where each $m_0, m_1 \in M$, it returns a single value $m \in M$, which is the Pearson correlation coefficient.
m decision b	Given a correlation measure $m \in M$, it returns true if and only if the correlation is considered significant.

elements we provide can model all possible fairness definitions or if it is feasible to model all possible fairness definitions.

IV. EXAMPLE

Let us consider an example to which the Tiles framework can be applied. For that, we go back to the childcare subsidy scenario. For the purpose of this scenario, a family has one or more parents or (legal) guardians, who are responsible for one or more children. Guardians may receive different childcare subsidies depending on the definition of fairness used. Some possible criteria for the amount of money that each family could receive are listed here:

- (no subsidy) no subsidy is given to any family (Figure 4);
- (per child) give to all families the same amount for each child (Figure 5);
- (per family) give the same amount of money to each family, regardless of the number of children (Figure 6);
- (single guardian) give the subsidy when the family has only one guardian (Figure 7).

In our diagrams, each actor is a family (as defined in this scenario). Some of the properties of a family are:

- number of adults: a positive integer (1 or more);
- number of children: a positive integer (1 or more);
- a (yearly) income: a non-negative integer (0 or more).

These properties are considered contextual information and do not change across the pipeline. The resource is money for the childcare subsidy, and it is represented by a non-negative integer. The measures are then non-negative integers.

V. CONCLUSION

In this paper, we have presented a formal meta-model for instantiating definitions of fairness, supported by a visualization approach and a proof-of-concept implementation. We envision the presented work as a step towards making differences

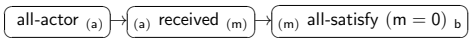


Fig. 4. Pipeline for no subsidy. The tile on the left provides all actors. The tile in the middle computes how much resource each actor received. The tile on the right checks that all resources are equal to 0.

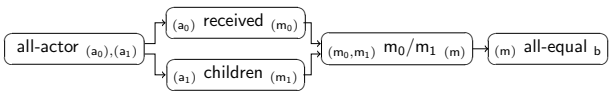


Fig. 5. Representation of "per child" using Tiles. The tile on the left provides actors, which are divided in two branches. The upper branch computes how much each actor (a family) has received and the lower branch how many children the family has. Both values are zipped back to compute the division. Note that we assume that each family has at least a child, but otherwise, if the number of children is 0, the division would be computed as `NaN`.

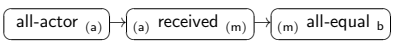


Fig. 6. Representation of "per family" using Tiles. This is equivalent to a standard equality pipeline where each actor receive exactly the same amount of resource.

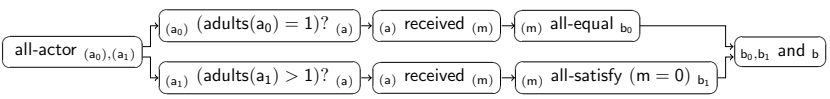


Fig. 7. Representation of "single guardian" using Tiles. This pipeline has two main branches. The upper branch accepts only families with one adult, i.e. single-parent/guardian families. The lower branch accepts all remaining families. It is worth noting that the sequences in both branches may have different number of elements and cannot be zipped back. On the other hand, the Boolean computation is combined with the 'and' tile, on the right.

between approaches to fairness in a given context explicit and qualitatively comparable.

For the next steps, our aim is to validate the framework and to expose it to domain experts and decision-makers that work on fairness-related specifications, for example, in the context of organizational and public policies, in order to elicit guidelines for practical use.

Future research can extend our work primarily in two directions. One direction from a formal perspective is to define axioms/principles for fairness scenarios. These may be related to the expected behavior of the underlying functions. For example, in a resource allocation scenario, an outcome function should exactly allocate the initially specified resources without "creating" or "wasting" any resources. Beyond that, one may specify principles that constrain subjective aspects of fairness scenarios, for instance, to gauge whether different formalizations of the same real-world scenario agree on a shared set of fundamental ideas. From an applied perspective, we aim to further advance our toolkit to define, visualize, and compare fairness definitions so that it is more accessible to practitioners such as analysts working on policy and process design, or decision automation, for example, by developing a visual interface to connect the tiles and automatically generate the source code.

Acknowledgements: This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] Aler Tubella, A., Barsotti, F., Koçer, R.G., Mendez, J.A.: Ethical implications of fairness interventions: what might be hidden behind engineering choices? Ethics and Information Technology **24**(1), 12 (Feb 2022). https://doi.org/10.1007/s10676-022-09636-z, https://doi.org/10.1007/s10676-022-09636-z

[2] Aler Tubella, A., Coelho Mollo, D., Dahlgren Lindström, A., Devinney, H., Dignum, V., Ericson, P., Jonsson, A., Kampik, T., Lenaerts, T., Mendez, J.A., Nieves, J.C.: ACROCPoLis: A Descriptive Framework for Making Sense of Fairness. In: Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency. p. 1014–1025. FAccT '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3593013.3594059, https://doi.org/10.1145/3593013.3594059

[3] Barocas, S., Selbst, A.D.: Big Data's Disparate Impact. California Law Review **104**(3), 671–732 (2016). https://doi.org/10.15779/Z38BG31, http://www.jstor.org/stable/24758720

[4] Bellamy, R.K.E., Dey, K., Hind, M., Hoffman, S.C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilovic, A., Nagar, S., Ramamurthy, K.N., Richards, J.T., Saha, D., Sattigeri, P., Singh, M., Varshney, K.R., Zhang, Y.: AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. IBM J. Res. Dev. **63**(4/5), 4:1–4:15 (2019). https://doi.org/10.1147/JRD.2019.2942287, https://doi.org/10.1147/JRD.2019.2942287

[5] Berk, R., Heidari, H., Jabbari, S., Kearns, M., Roth, A.: Fairness in Criminal Justice Risk Assessments: The State of the Art (2017). https://doi.org/10.48550/ARXIV.1703.09207, https://arxiv.org/abs/1703.09207

[6] Bin-Obaid, H.S., Trafalis, T.B.: Fairness in Resource Allocation: Foundation and Applications. In: Bychkov, I., Kalyagin, V.A., Pardalos, P.M., Prokopyev, O. (eds.) Network Algorithms, Data Mining, and

Applications. pp. 3–18. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-37157-9_1
[7] Bird, S., Dudík, M., Edgar, R., Horn, B., Lutz, R., Milan, V., Sameki, M., Wallach, H., Walker, K.: Fairlearn: A toolkit for assessing and improving fairness in AI. Microsoft, Tech. Rep. MSR-TR-2020-32 (2020)
[8] Busemeyer, M.R., Goerres, A.: Policy feedback in the local context: analysing fairness perceptions of public childcare fees in a german town. Journal of Public Policy **40**(3), 513–533 (2020). https://doi.org/10.1017/S0143814X18000491
[9] Chalfin, A., Danieli, O., Hillis, A., Jelveh, Z., Luca, M., Ludwig, J., Mullainathan, S.: Productivity and Selection of Human Capital with Machine Learning. American Economic Review **106**(5), 124–27 (May 2016). https://doi.org/10.1257/aer.p20161029, https://www.aeaweb.org/articles?id=10.1257/aer.p20161029
[10] Dator, J., Pratt, D., Seo, Y.: What Is Fairness?, pp. 19–34. University of Hawai'i Press (2006). https://doi.org/10.2307/j.ctv3zp081.6, http://www.jstor.org/stable/j.ctv3zp081.6
[11] Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R.S.: Fairness through awareness. In: Goldwasser, S. (ed.) Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012. pp. 214–226. ACM (2012). https://doi.org/10.1145/2090236.2090255, https://doi.org/10.1145/2090236.2090255
[12] Australian Institute of Family Studies, A.G.: Understanding the Child Care Subsidy (2024), https://aifs.gov.au/research/research-snapshots/understanding-child-care-subsidy
[13] Finkel, N.J., Harré, R., Rodriguez Lopez, J.L.: Commonsense Morality Across Cultures: Notions of Fairness, Justice, Honor and Equity. Discourse Studies **3**(1), 5–27 (2001). https://doi.org/10.1177/1461445601003001001, https://doi.org/10.1177/1461445601003001001
[14] Fleisher, W.: What's Fair about Individual Fairness? In: Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society. p. 480–490. AIES '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3461702.3462621, https://doi.org/10.1145/3461702.3462621
[15] Franklin, J.S., Bhanot, K., Ghalwash, M., Bennett, K.P., McCusker, J., McGuinness, D.L.: An Ontology for Fairness Metrics. In: AIES 2022 - Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society. pp. 265–275. Association for Computing Machinery, Inc (7 2022). https://doi.org/10.1145/3514094.3534137
[16] Friedman, B., Kahn, P., Borning, A.: Value Sensitive Design: Theory and Methods. University of Washington technical report **2**, 12 (2002), https://dada.cs.washington.edu/research/tr/2002/12/UW-CSE-02-12-01.pdf
[17] Gastwirth, J.L.: A General Definition of the Lorenz Curve. Econometrica **39**(6), 1037–1039 (1971). https://doi.org/10.2307/1909675, http://www.jstor.org/stable/1909675
[18] Gastwirth, J.L.: The Estimation of the Lorenz Curve and Gini Index. The Review of Economics and Statistics **54**(3), 306–316 (1972). https://doi.org/10.2307/1937992, http://www.jstor.org/stable/1937992
[19] Gini, C.: Sulla misura della concentrazione e della variabilita dei caratteri. Atti del Reale Istituto Veneto di Scienze, Lettere ed Arti **73**, 1203–1248 (1914), https://cir.nii.ac.jp/crid/1573105974129324928
[20] Gini, C.: Measurement of Inequality of Incomes. The Economic Journal **31**(121), 124–125 (03 1921). https://doi.org/10.2307/2223319, https://doi.org/10.2307/2223319
[21] Government, M.: Child Care Subsidy (2024), https://www.gov.mb.ca/education/childcare/families/childcare_subsidies.html
[22] IEEE: IEEE Standard for Floating-Point Arithmetic. IEEE Std 754-2008 pp. 1–70 (2008). https://doi.org/10.1109/IEEESTD.2008.4610935
[23] Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender systems: an introduction. Cambridge University Press (2010)
[24] Katoh, N., Ibaraki, T.: Resource Allocation Problems, pp. 905–1006. Springer US, Boston, MA (1998). https://doi.org/10.1007/978-1-4613-0303-9_14, https://doi.org/10.1007/978-1-4613-0303-9_14
[25] Kusner, M.J., Loftus, J., Russell, C., Silva, R.: Counterfactual fairness. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30, pp. 4067–4077. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf
[26] Makhlouf, K., Zhioua, S., Palamidessi, C.: On the Applicability of Machine Learning Fairness Notions. SIGKDD Explor. Newsl. **23**(1), 14–23 (may 2021). https://doi.org/10.1145/3468507.3468511, https://doi.org/10.1145/3468507.3468511
[27] Marope, P.T.M., Wells, P.J., Hazelkorn, E., et al.: Rankings and accountability in higher education: Uses and misuses. Unesco (2013)
[28] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A Survey on Bias and Fairness in Machine Learning (2019). https://doi.org/10.48550/ARXIV.1908.09635, https://arxiv.org/abs/1908.09635
[29] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A Survey on Bias and Fairness in Machine Learning. ACM Comput. Surv. **54**(6) (jul 2021). https://doi.org/10.1145/3457607, https://doi.org/10.1145/3457607
[30] Mehrotra, A., Celis, L.E.: Mitigating Bias in Set Selection with Noisy Protected Attributes. In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. p. 237–248. FAccT '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3442188.3445887, https://doi.org/10.1145/3442188.3445887
[31] Mendez, J.A.: Soda (2020), https://julianmendez.github.io/soda
[32] Mendez, J.A.: Soda: An Object-Oriented Functional Language for Specifying Human-Centered Problems (2023). https://doi.org/10.48550/arXiv.2310.01961
[33] Moreno-Ternero, J.D.: On the design of equal-opportunity policies. Investigaciones económicas **31**(3), 351–374 (2007)
[34] O'Neil, C.: Weapons of math destruction. How Big Data Increases Inequality and Threatens Democracy. Crown (2016)
[35] Pigmans, K., Dignum, V., Doorn, N.: Group proximity and mutual understanding: measuring onsite impact of a citizens' summit. Journal of Public Policy **41**(2), 228–250 (2021)
[36] de Reuver, M., van Wynsberghe, A., Janssen, M., van de Poel, I.: Digital platforms and responsible innovation: expanding value sensitive design to overcome ontological uncertainty. Ethics and Information Technology **22**, 257–267 (2020)
[37] Rodgers, J.L., Nicewander, W.A.: Thirteen Ways to Look at the Correlation Coefficient. The American Statistician **42**(1), 59–66 (1988). https://doi.org/https://doi.org/10.2307/2685263, http://www.jstor.org/stable/2685263
[38] Shrestha, Y.R., Yang, Y.: Fairness in Algorithmic Decision-Making: Applications in Multi-Winner Voting, Machine Learning, and Recommender Systems. Algorithms **12**(9) (2019). https://doi.org/10.3390/a12090199, https://www.mdpi.com/1999-4893/12/9/199
[39] Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F.B., Wilson, J.: The What-If Tool: Interactive Probing of Machine Learning Models. IEEE Trans. Vis. Comput. Graph. **26**(1), 56–65 (2020). https://doi.org/10.1109/TVCG.2019.2934619, https://doi.org/10.1109/TVCG.2019.2934619

Paper IV

Soda: An Object-Oriented Functional Language for Specifying Human-Centered Problems*

Julian Alfredo Mendez[0000−0002−7383−0529]

Department of Computing Science, Umeå University, Sweden
julian.mendez@cs.umu.se

Abstract. We present Soda (Symbolic Objective Descriptive Analysis), a language that helps to treat qualities and quantities in a natural way and greatly simplifies the task of checking their correctness. We present key properties for the language motivated by the design of a descriptive language to encode complex requirements on computer systems, and we explain how these key properties must be addressed to model these requirements with simple definitions. We give an overview of a tool that helps to describe problems in an easy way that we consider more transparent and less error-prone.

Keywords: Responsible artificial intelligence · Functional languages · Object-oriented languages · Human-centered programming languages

1 Introduction

Understanding how artificial intelligence (AI) agents work can be challenging because AI algorithms are complex and their reasoning opaque. Although transparency is often seen as a requirement, realistically, it might not always be possible, for example, due to privacy or security concerns, whereas the need to ensure that a system operates within moral bounds remains. At the same time, validation and verification procedures highly depend on the specific contextual interpretations that have been employed to ground abstract principles (e.g., fairness or privacy) into the concrete functionalities of an agent [1].

Verification can be a difficult or unfeasible task, and even when achieved, its specifications can be difficult to understand. Verification is only as strong as the assumptions that underlie the specification, which means that specifying assumptions and analyzing these specifications is crucial for verification [7]. Given that AI mostly operates in environments that are at best partially known by the system designer, and that properties are often discussed at a high level of abstraction by stakeholders without a background in formal languages, specification languages need to be easily understood by different stakeholders.

From a technical point of view, unit tests are crucial in ensuring the quality of a piece of software [36]. Test-driven development (TDD), which is a software development process in which developers create test cases together with the main development,

* This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

has been shown to be more productive than other more conventional development techniques [12]. We propose going one step further by introducing theorems together with the code, which in turn becomes more reliable.

Our contribution is to present Soda [1] (Symbolic Objective Descriptive Analysis), a new descriptive language based on widely adopted concepts such as modeling with classes and functional definitions. In this context, by descriptive language, we mean a language based on descriptions rather than processes, and that seems closer to a specification language rather than an implementation language. The set of basic constructs suffices to model complex requirements, but is small enough to be immediately understood by a larger group of stakeholders.

In this paper, we address the following research question which is composed of two parts:

RQ: Can we design a **descriptive language** to encode requirements on AI systems such that:

- **RQ.1:** the language or a fragment is **formally verifiable**, and
- **RQ.2:** it is easily integrated into **state-of-the-art technology**?

In the following sections, we show our approach to this question.

2 Language Description

In this section, we present key properties for the language motivated by **RQ**, and we explain how these key properties must be addressed to model requirements with simple definitions.

2.1 Key Properties

One of the key properties of the language is that it should be used to formalize requirements in an intuitive way, and one of the most effective ways to do it is to use types. We want the language to be *statically typed*, because static typing simplifies type checking, which in turn helps prevent formalization errors. To avoid errors caused by side effects [21], we make variables *immutable*, which are closer to their standard mathematical interpretation.

We want to relate qualities and quantities to describe complex conditions. We expect to use the very same language to model a problem containing hierarchies, like a resource access monitoring agent, and a problem containing measures, like a price monitoring agent. To do this, the language has to be *expressive* and *general*.

Standard computer languages include a considerable number of reserved words and basic types, which usually hinder understanding. More reserved words usually bring more combinations and nuances to their use, and to simplify its comprehension, it is convenient for the language to have a *small* set of constructs.

Alongside the aforementioned properties, we want the language to be used to evaluate effectively whether properties hold. For that, the language should be *easy to prototype*, and its prototypes should be *human-level efficient*, which means efficient enough for its expected use.

[1] `https://julianmendez.github.io/soda`

2.2 Main Constructs

The main constructs are presented to ensure the requirements. We have chosen constructs that look similar to those used in popular programming languages.

Since Soda is *statically typed*, it needs a type definition construct. Let us name it ':' (colon). The syntax is $x : A$, meaning that x is of type A.

Due to *immutability*, we want to define constants and functions, but not variables in the computer science sense. In the following, we mention variables in the mathematical sense when referring to lambda expressions. To define a constant or a function, we need a construct. Let us name it '=' (equals sign). The notation is

$$f(x_1 : A_1) \ldots (x_n : A_n) : A = e$$

where f is the *function name*, each x_i $(1 \leq i \leq n)$ is a parameter of type A_i, and e is an expression of type A. A function f without parameters is called a *constant*. A function can be called using named parameters with the ':=' symbol. For example, $f(x : Int)(y : Int) : Int$ can be invoked as $f(x := 0)(y := 1)$.

Most current programming languages include lambda expressions, which are anonymous functions based on lambda calculus [4]. We have included lambda expressions in Soda because they have been highly adopted. The notation

$$\texttt{lambda}\ x \longrightarrow f(x)$$

corresponds to $(\lambda x).f(x)$ or $\lambda x \rightarrow f(x)$ in the literature. Since we work with typed lambda calculus, the type needs to be specified when it cannot be inferred, and we denote it by $\texttt{lambda}\ (x : A) \longrightarrow f(x)$.

Since the language is *expressive* and *general*, it includes standard operations from mathematics, such as '+', '-', '*', '/', for arithmetic, and '`not`', '`and`', '`or`' for logic, with the usual meaning. Logic functions are evaluated with lazy evaluation. Therefore, when a `and` b is evaluated, a is evaluated first, and if a is already false, b is not evaluated. Analogously for `or`, if the first value is already true. Lazy evaluation can also be used to compute functions that otherwise would be undefined, and this is done by using defined parts on the left to prevent undefined parts on the right. If the computations have no side effects, the result of computing with or without lazy evaluation is exactly the same, but the time needed is not.

Note that the language can define recursive functions over finite structures, as mainstream purely functional programming languages do. This gives an expressive power that is enough to model human-understandable constraints.

To define piecewise functions we have '`if-then-else`' structures, with the notation

$$\texttt{if}\ b\ \ \texttt{then}\ e_1\ \ \texttt{else}\ e_2$$

where b is a Boolean expression, and e_1 and e_2 are expressions of the same type. The interpretation is standard, and the result is e_1 if b is true, and e_2 otherwise.

The pattern matching construct, called '`match-case`', has the format:

$$\texttt{match}\ x\ \ \texttt{case}\ p_1 \Longrightarrow e_1\ \ \ldots\ \ \texttt{case}\ p_n \Longrightarrow e_n$$

where x is a variable to match, p_i are patterns, and e_i are expressions, for $1 \leq i \leq n$. The type of the match structure is the most specific supertype of the e_i expressions. The p_i patterns could be of different types, but they should be constructors that possibly contain construction variables. In fact, we use pattern matching to use extractors for object deconstruction, which can also be used for type checking. Although the notation `if-then-else` could be defined as a pattern matching structure, we decided to keep it because it is more concise and more universally recognizable.

We find it relevant to highlight that the constructs we present in this section are meant to create small functions, which improves readability [21], and to require the use of function names in intermediate computations to create accurate documentation for specifications.

2.3 Types and Classes

Humans classify concepts into categories using features that help describe and reason. In Soda, we use types and classes to model objects that have attributes. They help us model ethical values like privacy and fairness, especially in relation to regulations.

There are some differences in view of whether it is convenient to have classes being instantiated by objects (like in Java [13]), or whether it is better to have modules, where functions are imported (like in Haskell [34]). We compromise between these two options because the objects created with the classes of Soda are immutable, and they work as namespaces for modules or to specify how to retrieve attributes in objects.

We distinguish between a type and a class as is usual in the literature [2]: an object has a *type*, and the type describes what the object can do, but not how, and, in contrast, a *class* provides the implementation for an object. As most designers and programmers are familiar with object-oriented programming, we choose to build classes with a construct called '`class`'. We adhere to the Open-Close Principle, where "software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification" [22]. Classes can be extended with the '`extends`' construct

```
class A
extends B_1 ... B_n
 d_1
 ...
 d_m
end
```

where class A extends classes $B_1, \ldots, B_n$, and $d_1, \ldots, d_m$ are constant or function definitions.

It is also possible to define *interfaces*, which are types that contain only declarations of constants and functions, without specifying what they contain. These declarations are in a block with the word '`abstract`' as follows

```
class A
abstract  f_1 ... f_n
end
```

where each f_i is a constant or function declaration.

Each class has a default type constructor, which is named the same as the class with an extra underscore ('_') as suffix. The abstract elements in a class need to be given as parameters to instantiate a class. For example,

```
class Pair

  abstract
    fst : Int
    snd : Int

end
```

can be instantiated with `Pair_ (1) (2).`

Since we want to be able to apply design patterns and the language is statically typed and object-oriented, we need a way to refer to the instance itself, and the construct is '`this`'.

We allow type parameters for classes and functions to have polymorphism. To denote the type parameter, we use square brackets '[]'. In class declarations we specify that the parameter is of type `Type`. For example, a parameterized pair could be defined as

```
class Pair [A : Type] [B : Type]

  abstract
    fst : A
    snd : B

end
```

and then instatiated as `Pair_ [Int] [Int] (1) (2).`

In addition, it is possible to declare upper and lower type bounds. An upper type bound is denoted with the word '`subtype`' or '`<:`' as found in the literature, and the lower type bound is the word '`supertype`' or '`>:`'. For example

$$\texttt{class}\ C\ \ [A\ \texttt{subtype}\ B]\ \ \texttt{extends}\ D$$

indicates the declaration of class C parameterized with type A, which is a subtype of B, and C itself extends D. Subtyping in this language follows the Liskov Substitution Principle [20]. To organize the classes, we group them in *packages*, so that a package is just a collection of classes. We can declare that a class belongs to a package using the word '`package`'. In addition, the word '`import`' helps to bring classes from other packages.

The language can be translated to other languages by including the so-called directives, which are specific to the target languages. The word '`directive`' defines a piece of code that is considered only in specific translations and ignored in others.

3 Discussion and Implementation

3.1 Integration with Scala

To prototype the specification, we translate it into the Scala [27] code. We use type checking and type inference provided by Scala. Each class is translated into a Scala `trait`, which is open for extension, each constant definition is translated to a lazy value (`lazy val`), and each function definition to a `def`, as well as each abstract constant or function declared with `abstract`. The default type constructor is a `case class` that extends the original trait. Scala case classes provide constructors and extractors for pattern matching and are not extensible. The `if-then-else` and `match-case` are very similar to what is provided in Scala. The supporting data types are the same provided by Scala, such as `String`, `Int`, `Double`, `Option`, and `Seq`.

The prototype can be run on the Java Virtual Machine (JVM), which is multiplatform and optimized for concurrent execution, and it can therefore use JVM libraries, so that, for example, a monitor can communicate with an AI agent interface. Although libraries can produce side effects, this can be easily controlled by the import commands, which define exactly the classes that are being used. On the one hand, a purely functional specification will not include any reference to those libraries. On the other hand, it is possible to connect an agent employing the corresponding JVM libraries. This dual use of the JVM brings the right amount of flexibility needed for a practical use, without losing control on critical parts.

The translation to Scala comprises all the nuances of the language. Type parameters, in Soda `[A : Type]`, are also provided in Scala `[A]`, as they are in Kotlin [17] and Java `<A>`, and Lean `(A : Type)`.

3.2 Integration with Lean

Lean [18] is a theorem prover and programming language based on the calculus of constructions with inductive types. Part of Soda can be translated into Lean to prove correctness of Soda snippets. The types in Lean are not the same as those in Scala, and the JVM libraries cannot be used, but some core purely functional pieces of code in Soda can be proven correct by using Lean.

The class definitions in Soda define three things that in Lean must be defined separately: a *type*, a *namespace*, and a *constructor*. Every Soda class defines a namespace, in Lean `namespace`. Some classes contain parametric internal values defined with `abstract` in Soda. These classes are translated into Lean as `class`. They include a default constructor, which has the same name as the class ending with an underscore, and the fields, all after the Lean `where`. Lean already provides extractors needed for pattern matching. The default equality given in Soda is achieved by deriving (`deriving`) in Lean from `DecidableEq` (or from `BEq`).

As for Scala, function definition in Soda is translated to a `def` at the beginning in Lean, `if-then-else` in Soda is identical in Lean, and a `match-case` structure in Soda is a `match-with-`⇒ structure in Lean. The structure

$$\texttt{match}\ x\ \ \texttt{case}\ p_1 \Longrightarrow e_1\ \ \ldots\ \ \texttt{case}\ p_n \Longrightarrow e_n$$

is translated as

$$\texttt{match}\ x\ \texttt{with}\ \mid p_1 \Rightarrow e_1\ \ \ldots\ \mid p_n \Rightarrow e_n$$

For the case of package management (in Soda `package` and `import`) and self-instances (in Soda `this`) are not supported at the moment, and neither are the `subtype` and `supertype` type bounds.

Some basic types in Lean are stricter than in Soda or Scala, and there is not always a direct mapping from Soda to Lean. However, it is possible to create a specific mapping with `directive`, which allows adding a mapping for a type, and including Lean theorems with their proofs.

3.3 Undefined States and Termination

Soda does not use exceptions. This is because they correspond to an imperative feature, as they are used to interrupt the evaluation of a function and perform a jump (*goto*) to the point where they are caught, assuming that they are caught at all.

This also implies that designs in this language need to be careful, considering edge cases and properly managing them when building objects. If exceptions are thrown from JVM objects, they can be caught by underlying types in Scala (e.g. `Try`), and then managed as objects.

Aside from the possibility of integrating JVM libraries, the language itself is highly expressive. There is no limit for self-recursion, which brings advantages and disadvantages. We provide an annotation to force tail recursion (`@tailrec`) to avoid stack overflow. There are two prominent functions that can be easily defined in this language. The first function is *range*, which generates the first natural numbers. The second function is *fold*, which applies a cumulative operation on a sequence. Both functions are sufficient to define the most common operations on sequences. Since both functions above always terminate, they are a convenient substitute for full recursion, preventing possible infinite recursion.

3.4 Related Work

As mentioned above, the language aims to have a highly readable formal language for humans. In the design of the language, we consider the good properties of some programming languages. The main features we look for are:

- the specification is intended to be read and understood by a human (which is the most relevant point);
- everything defined is done only once, in one place (to avoid confusion due to partial definitions);
- objects are immutable (because one of the key properties is immutability);
- classes cannot be modified, but they can be extended (because of the Open-Close Principle [22]).

One of the properties that we adopted was the *functional notation*. For that, we evaluated three categories: the Lisp style, the ML (Meta Language) [23] style, and the Haskell style. In the Lisp category we can mention Clojure [14], which has a large community and can be integrated with the JVM. In the Haskell category we can mention Haskell, which is a de-facto standard in the functional programming community. In the same category, we also have Agda [26] and Idris [3], which can be used as proof assistants. In the ML category we can mention OCaml [19], which is a very efficient implementation of ML, Coq [32], which is a proof assistant with a very reduced core, and Lean [18], which is an efficient proof assistant. We wanted to avoid the excessive use of parentheses as in the Lisp style because it is less readable for non-experts. We wanted to avoid the definitions of partial functions, which is the standard notation in the Haskell style, in order to reduce undefined functions. We considered the ML style to be the most appropriate for the language, which induces definitions of total functions with a moderate number of parentheses.

Another property we considered was *readability*. For that we looked at Python [28, 33] and Prolog [5, 35]. Python is a language that is popular among scientists without a computer science background and is directed at a broad range of ages. For example, some young students start with Scratch [24] and then transition to Python or JavaScript [9], while [31], a minimalist dialect of Lisp, has been used to teach at university courses.

Prolog is also widely accepted in the scientific community. For example, 2APL (A Practical Agent Programming Language) [6] is a programming language for multi-agent systems consisting of individual agents that may share and access external environments. 2APL integrates the declarative and imperative style by connecting declarative beliefs and goals with events and plans. As later developments based on 2APL we can mention N-2APL (Norm-Aware Agent Programming Language), the 2OPL (Organisation Programming Language), and a framework for norm-aware multi-agent systems [8] that integrates the programming languages. Although Prolog is a logic language and the interpreter itself operates in a different way, the pieces of code created in Prolog are similar to those in purely functional languages.

Last but not least, we sought for a good *functional object-oriented integration*. This was provided by Scala, which has a thriving community of purely functional and object-oriented backgrounds. Scala also provides an advanced type inference system and can compile to JVM bytecode.

Table 1 contains a summary of the properties we searched for. We tested the efficiency of the programming languages mentioned above on a computer with an Intel Core i5-8350U CPU (1.70 GHz) with 8 cores, running on Linux 6.2.0 from the Ubuntu 22.04 LTS distribution. For all programming languages, we wrote a piece of code that was either a built-in cycle (like in Clojure, Prolog, and Python) or a recursion with tail recursion. We tried different powers of 10, and indicated the largest number of repetitions fitting in 30 seconds. This value is not meant to be a global benchmark to compare the languages or their implementations, but rather to show how Soda is well-positioned in terms of efficiency.

Employing formal proofs instead of only empirical evidence (like unit tests) gives a stronger reliability on multi-agent systems. This has been addressed in a develop-

Table 1. Properties of related languages.

lang.	version	A	B	C	D	E	F
Agda	2.6.3	Yes	Yes	No	Yes	No	10^9
Clojure	1.11.1	Yes	No	No	No	Yes	10^9
Coq	8.16.1	Yes	Yes	No	Yes	No	10^7
Haskell	8.8.4	Yes	Yes	No	Yes	No	10^8
Idris2	0.6.0	Yes	Yes	No	Yes	No	10^{10}
Lean	4.0.0	Yes	Yes	No	Yes	No	10^8
OCaml	4.14.1	Yes	No	Yes	Yes	No	10^9
Prolog	8.4.2	No	No	No	No	No	10^8
Python	3.10.6	No	No	Yes	No	No	10^8
Scala	3.3.1	Yes	No	Yes	Yes	Yes	10^{10}
Soda	0.19.0	Yes	Yes	Yes	Yes	Yes	10^{10}

Columns: **A.** dominantly or only functional // **B.** purely functional in its core syntax // **C.** full object-oriented notation // **D.** statically typed // **E.** JVM integration // **F.** number of repetitions in 30 s

ment [16] where a verification framework for the GOAL agent programming language [15] has been formalized in Isabelle/HOL [25], which is a proof assistant based on higher-order logic. We also became interested in Scallina [10,11], which is a tool for translating from Coq to Scala. As for languages for verification, we can also mention Dafny [30] and F* [29].

4 Conclusion

We present Soda, a language used to model constraints in AI systems. Our main motivation for Soda is to model complex requirements that need to be easily understood by humans. Furthermore, it can be used to model and prototype other types of constraints. In addition, Soda can be efficiently prototyped in an optimized multiplatform state-of-the-art technology, like the JVM, and some pieces of code can be verified in Lean.

We give an overview of a tool that helps to describe problems in a way that we consider more transparent and less error-prone. Although writing descriptions in this style could require more effort than with standard imperative languages, the effort to fully comprehend those descriptions is significantly smaller, thanks to the reduced number of constructs. This language is also conducive to writing better designs since each function explains a piece of code and works as a running documentation. In addition, the computer gives extra verification by checking and inferring the types.

In future work, we would like to expand the Lean translator to handle more nuances of Soda. Since understandability is a key feature of Soda, we would like to conduct a case study in which stakeholders can read descriptions and corroborate the readability of the language.

References

1. Aler Tubella, A., Theodorou, A., Dignum, F., Dignum, V.: Governance by Glass-Box: Implementing Transparent Moral Bounds for AI Behaviour. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 5787–5793. International Joint Conferences on Artificial Intelligence Organization (7 2019). https://doi.org/10.24963/ijcai.2019/802, https://doi.org/10.24963/ijcai.2019/802
2. Atkinson, M., DeWitt, D., Maier, D., Bancilhon, F., Dittrich, K., Zdonik, S.: The Object-Oriented Database System Manifesto. In: KIM, W., NICOLAS, J.M., NISHIO, S. (eds.) Deductive and Object-Oriented Databases, pp. 223–240. North-Holland, Amsterdam (1990). https://doi.org/10.1016/B978-0-444-88433-6.50020-4
3. Brady, E.: Idris (2007), https://www.idris-lang.org
4. Church, A.: An unsolvable problem of elementary number theory. American Journal of Mathematics **58**, 345–363 (1936)
5. Colmerauer, A., Kowalski, R.: Prolog (1972)
6. Dastani, M.: 2APL: a practical agent programming language. Autonomous Agents and Multi-Agent Systems **16**, 214–248 (6 2008). https://doi.org/10.1007/s10458-008-9036-y
7. Dennis, L.A., Fisher, M., Lincoln, N.K., Lisitsa, A., Veres, S.M.: Practical verification of decision-making in agent-based autonomous systems. Automated Software Engineering **23**(3), 305–359 (2016). https://doi.org/10.1007/s10515-014-0168-9
8. Dybalova, D., Testerink, B., Dastani, M., Logan, B.: A Framework for Programming Norm-Aware Multi-agent Systems. In: Balke, T., Dignum, F., van Riemsdijk, M.B., Chopra, A.K. (eds.) Coordination, Organizations, Institutions, and Norms in Agent Systems IX. pp. 364–380. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-07314-9_20
9. Eich, B.: JavaScript (1995), https://github.com/tc39/ecma262
10. El Bakouny, Y., Mezher, D.: Scallina: Translating Verified Programs from Coq to Scala. In: Ryu, S. (ed.) Programming Languages and Systems. pp. 131–145. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-030-02768-1_7
11. El Bakouny, Y., Mezher, D.: The Scallina Grammar. In: Massoni, T., Mousavi, M.R. (eds.) Formal Methods: Foundations and Applications. pp. 90–108. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-030-03044-5_7
12. Erdogmus, H., Morisio, M., Torchiano, M.: On the effectiveness of the test-first approach to programming. IEEE Transactions on Software Engineering **31**(3), 226–237 (2005). https://doi.org/10.1109/TSE.2005.37
13. Gosling, J.: Java (1995), https://www.oracle.com/java/
14. Hickey, R.: Clojure (2007), https://clojure.org
15. Hindriks, K.V.: Programming Rational Agents in GOAL. In: El Fallah Seghrouchni, A., Dix, ü., Dastani, M., Bordini, R.H. (eds.) Multi-Agent Programming: Languages, Tools and Applications. pp. 119–157. Springer US, Boston, MA (2009). https://doi.org/10.1007/978-0-387-89299-3_4, https://doi.org/10.1007/978-0-387-89299-3_4
16. Jensen, A.B.: Towards Verifying GOAL Agents in Isabelle/HOL. In: Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART. pp. 345–352. INSTICC, SciTePress (2021). https://doi.org/10.5220/0010268503450352, https://www.scitepress.org/Papers/2021/102685/102685.pdf
17. JetBrains: Kotlin (2011), https://kotlinlang.org
18. Leonardo de Moura - Microsoft Research: Lean (2013), https://github.com/leanprover/lean

19. Leroy, X., Vouillon, J., Doligez, D., Rémy, D., Suárez, A.: OCaml (1996), https://ocaml.org
20. Liskov, B.H., Wing, J.M.: A Behavioral Notion of Subtyping. ACM Trans. Program. Lang. Syst. **16**(6), 1811–1841 (Nov 1994). https://doi.org/10.1145/197320.197383, https://doi.org/10.1145/197320.197383
21. Martin, R.C.: Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall (2009)
22. Meyer, B.: Object-Oriented Software Construction. Prentice Hall (1988)
23. Milner, R., et al.: ML (Meta Language) (1973)
24. MIT Media Lab: Scratch (2007), https://scratch.mit.edu
25. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL - A Proof Assistant for Higher-Order Logic. LNCS **2283** (2002)
26. Norell, U., et al.: Agda (2007), https://wiki.portal.chalmers.se/agda/
27. Odersky, M.: Scala (2004), https://www.scala-lang.org
28. Peters, T.: The Zen of Python (8 2004), https://peps.python.org/pep-0020/
29. Research, M., Inria: F* (2011), https://www.fstar-lang.org
30. Rustan, K., Leino, M.: Dafny (2009), https://dafny.org/
31. Steele, G.L., Sussman, G.J.: Scheme (1975), https://www.scheme.org
32. The Coq Development Team: Coq (1989), https://coq.inria.fr
33. van Rossum, G.: Python (1991), https://www.python.org
34. Wadler, P., et al.: Haskell (1990), https://www.haskell.org
35. Wielemaker, J.: SWI-Prolog (1987), https://www.swi-prolog.org
36. Zhu, H., Hall, P.A.V., May, J.H.R.: Software Unit Test Coverage and Adequacy. ACM Comput. Surv. **29**(4), 366–427 (dec 1997). https://doi.org/10.1145/267580.267590, https://doi.org/10.1145/267580.267590

Paper V

Can Proof Assistants Verify Multi-Agent Systems?

Julian Alfredo Mendez[0000−0002−7383−0529],
Timotheus Kampik[0000−0002−6458−2252]

Umeå University, Sweden
{julian.mendez,tkampik}@cs.umu.se

Abstract. This paper presents the Soda language for verifying multi-agent systems. Soda is a high-level functional and object-oriented language that supports the compilation of its code not only to Scala, a strongly statically typed high-level programming language, but also to Lean, a proof assistant and programming language. Given these capabilities, Soda can implement multi-agent systems, or parts thereof, that can then be integrated into a mainstream software ecosystem on the one hand and formally verified with state-of-the-art tools on the other hand. We provide a brief and informal introduction to Soda and the aforementioned interoperability capabilities, as well as a simple demonstration of how interaction protocols can be designed and verified with Soda. In the course of the demonstration, we highlight challenges with respect to real-world applicability.

Keywords: Engineering Multi-Agent Systems · Formal Verification · Proof Automation

1 Introduction

Multi-agent systems (MAS) are systems composed of multiple interacting intelligent agents, working towards joint or conflicting goals. MAS have emerged as an approach for modeling and implementing complex distributed systems, because MAS-based systems can solve problems that are very difficult or impossible to solve using monolithic architectures. Due to their dynamic and distributed nature, MAS are notoriously difficult to test. At the same time, ensuring desirable behavior in MAS components is crucial to verify that the autonomy that agents are granted does not lead to safety or compliance issues. Consequently, testing and verification of MAS has emerged as a substantial branch of MAS engineering research. Here, most research focuses either on finite state model checking [4] or–presumably less frequently–on testing in the software engineering sense (as in *test-driven development*) [1]. In this paper, we take a novel path and present a prototypical approach where parts of a MAS are implemented in Soda, a high-level programming language that can be compiled to Scala for integration with a mainstream programming ecosystem, as well as to Lean, a proof assistant and

programming language that applies calculus of constructions [13]. The formal verification of the program code can then be executed in Lean.

To illustrate the approach and demonstrate its feasibility, we provide a simple running example of interaction protocol verification with SODA and Lean, which we introduce conceptually below. In our example, we want to implement an interaction protocol for a platform on which multiple agents buy and sell items. We assume that the transactions are safe and conducted by a mediator that is trusted by all involved agents and can transfer possession of items and money between the agents. In the following, we provide an example interaction sequence that can serve as an archetype for the protocol that our MAS should enact.

1. A seller C notifies the mediator A to advertise a certain item R at a price P.
2. The mediator A informs all agents that the item is available for sale.
3. While no buyer has shown interest, C can remove the ad.
4. If a buyer B wants to buy R and has money P to pay for it, the buyer notifies A.
5. When A is notified of the purchase, it removes the ad, and it transactionally transfers the item R from C to B, and the amount of money P from B to C.

In our approach, we use lists to store items. As a proof of concept, we prove a technical but important property:

Property. *Changing an item in the list does not change the size of the list.*

This property is important, but it is usually taken for granted. Proving these kinds of properties in a proof assistant like Lean requires substantial effort because lemmas and tactics require maturity to be employed. All cases need to be covered, and the resulting object after the execution must always be defined. In the particular case of our functions, we work with immutable lists. Our lists are thread-safe, which allows them to run in parallel execution, and follow the standard construction in functional languages. They are defined as being either an empty list or an element prepended to a list. This makes them memory-efficient because replacing an arbitrary position requires reconstructing the list until the given position.

As we can see, the property we prove looks simplistic from an intuitive software engineering perspective. Still, we chose to prove this property as a minimal demonstrating example, due to the following reasons: *i)* even such simple facts are somewhat effortful to prove with proof assistants such as Lean; *ii)* our example is sufficient for highlighting practical challenges, while also serving as a proof-of-concept with respect to purely technical feasibility.

In the remainder of this paper, we give a brief overview of research on testing and verifying MAS (Section 2) to then introduce SODA and its interoperability capabilities, both with the Scala/Java ecosystem and with the Lean proof assistant (Section 3). We then continue our running example to demonstrate how agent interaction protocols can be specified and verified with SODA (Section 4).

Subsequently, we discuss in detail the formal proofs (Section 5). We show with practical experiments that the example is efficient (Section 6). After that, we discuss our work and its limitations, and conclude with a future outlook (Section 7).

2 Why Formal Verification for MAS?

Software verification ensures that specific components meet specification requirements. We can compare software verification with software validation, since both are software evaluation processes. The former focuses on determining whether a software artifact meets the conditions with respect to its broader purpose, imposed at the beginning of a development phase, whereas the latter determines whether the requirements are satisfied at the end of a development phase. The verification of agents and MAS has been subject to extensive study over the years. Here, *verification* can either be seen from an applied software engineering perspective, i.e., *testing* without "hard" guarantees, or as the formal analysis (*formal verification*) of MAS or components thereof.

Jason [5] and SARL [16] are specialized multi-agent programming languages, with Jason focusing on logical reasoning and SARL on dynamic, distributed environments. Jason is an interpreter for an extended version of AgentSpeak(L) [15], a BDI (Belief-Desire-Intention) agent-oriented logic programming language. Jason provides a platform for developing multi-agent systems with customizable features, especially for environments requiring logical reasoning and decision-making. SARL is a general-purpose agent-oriented programming language designed to handle concurrency, distribution, interaction, decentralization, reactivity, autonomy, and dynamic reconfiguration. It integrates well with the Janus[8] platform for distributed multi-agent systems.

Although agent testing is certainly a relevant research domain, e.g., in the context of technologies for agent-oriented programming [1,2], a well-informed assumption is that for MAS, testing is insufficient because of agents' and MAS' highly dynamic behavior [19], in particular if contrasted with mainstream software components in server-client architectures. Because mainstream software testing approaches are often deemed insufficient for engineering MAS, researchers have devised a range of formal approaches to verify agents or other components of MAS. These approaches typically rely on model checking techniques [4,7,10]. Still, due to their dynamic nature and the resulting combinatorial explosion of the number of models [19], applying model checking to MAS can be similarly intractable as attempting to achieve substantial coverage in traditional tests. Also, model-checking approaches are typically limited to particular abstractions and modalities; for example, TLA+ prominently focuses on temporal reasoning [9]. In contrast, proof assistants such as Coq [3], Isabelle [14], and Lean [6] allow for more generic formal analyses, and thus they can potentially enable more flexible verification of agents and MAS. However, these tools are not straightforwardly applicable to the formal analysis of program code written in languages that engineers use to specify behavior at the "business logic"(or *knowledge*) level. This

may explain why proof assistants have not yet been comprehensively studied in the context of MAS verification, at least (to the best of our knowledge) not as tools that can directly analyze executable specifications of MAS components.

The work we present in this paper is a step towards overcoming this obstacle, by bridging the gap between mainstream software ecosystems in which MAS can be engineered and tools for proof automation, demonstrating basic feasibility by implementing a simple example.

3 Designing MAS with Soda

Before we demonstrate Soda's verification capabilities, let us provide a brief overview of its syntax (Subsection 3.1) and the technologies the language is built upon (Subsection 3.2).

3.1 Soda syntax

Soda [11][1] is a statically typed functional language with object-oriented notation. Its name is an acronym for *Symbolic Objective Descriptive Analysis*, and it was designed to be descriptive by being declarative and easy to read. It combines the rigor of purely functional languages like Lean, with the modern technology integration of languages like Scala, which in turn is a bridge to the Java Virtual Machine (JVM) ecosystem. Soda provides not only the expressive power for the type of transactions we have in the example, but also the elements to include Lean proofs for the definitions. It also allows for interconnection with complex packages provided by JVM libraries and frameworks, and it has a simple and direct way to write functions, similar to mainstream languages like Scala or Python.

Soda follows the functional style of ML[12][2], and it has a small set of constructs. Most of them can be placed in one of the following categories: function-related constructs and class-related constructs. It uses the basic types provided by Scala, such as Boolean, Int, Float, and String.

The function-related constructs are meant to define functions. It is possible to say that a variable x is of a certain type A with $x : A$. The functions are then defined in the usual way; e.g., $f(x : Int) : Int = x + 1$ defines the function $x + 1$. Lambda expressions are also written as expected: the function above can be defined using the lambda expression $f : Int = \texttt{lambda}\ x \longrightarrow x + 1$. The construct `if-then-else` allows for the definition of piecewise functions. As in other ML-styled functional languages, the construct `match-case-`$\Longrightarrow$ is used for pattern matching. The following function returns the maximum of two integers:

$$max\ (x : Int)\ (y : Int) : Int = \texttt{if}\ x > y\ \texttt{then}\ x\ \texttt{else}\ y$$

The class-related constructs are meant to define new types, to group functions in classes, and to group classes in packages. The construct `class-end` defines a

[1] For a more detailed technical report that describes the Soda language, see [11].
[2] ML stands for *Meta Language*, but the acronym is rarely spelled-out.

class from the beginning to the end. It groups functions in a module, and it defines a type, a default constructor, and a default equality function. A class can extend other classes with the construct `extends`. If a class needs parameters to instantiate an object, these parameters can be defined with the construct `abstract`. A class can depend on parameterized types, which are parameters belonging to `Type` inside square brackets (`[ ]`). Classes are grouped in packages with the `package` reserved word, and `import` is used to import classes from different packages.

3.2 Technology behind Soda

In a figurative way, we can say that the Soda syntax lies in the intersection of Scala version 3 syntax and Lean version 4 syntax[3], and they all fall under the umbrella of ML-styled programming languages. Soda code needs to be compiled in order to be executed. The main compiler converts the Soda source code into Scala source code. Since it is a source-to-source compiler of similar source code, this compiler is a *transpiler*, and we also call it a *translator*. We claim that the translation into Scala is sound, in the sense that all correct code in Soda is translated into correct code in Scala. It is also possible to import Java Virtual Machine (JVM) libraries into Soda, using `import`, but it requires a manual check from the developer to ensure that the imported libraries keep the code purely functional.

Soda can be partially translated into Lean. This means that it is possible to prove properties of Soda code in Lean. The construct `directive` can include embedded pieces of code of a specific target language. For example, a proof would be considered in a translation to Lean, but it would be ignored in a translation to Scala. The proof has to be actually written in Lean, based on definitions provided in Soda. Soda's source code is available for download at https://github.com/julianmendez/soda.

4 Soda for MAS Verification

Soda provides a small set of constructs to create code that can be executed in Scala 3 and proven in Lean 4. On the one hand, Lean is not only a powerful programming language, but also a proof assistant. Its mathematical library, mathlib[4], contains a significant number of mathematical theorems and their proofs, which can serve as a template to prove properties of agents. On the other hand, Scala is a very flexible language, with an efficient implementation connected to the JVM libraries, which allows for multi-platform execution. It is important to note that Lean proofs cannot cover the use of JVM libraries, since these are not specified for Lean.

[3] Scala 3 and Lean 4 have improved their syntax compared to Scala 2 and Lean 3 respectively.

[4] https://github.com/leanprover-community/mathlib4

We have specified the example of Section 1 in Soda. We describe the basic types, available functions, and some properties that the system should hold. Although the example is intended to be a proof of concept, it already contains the main components for a full-scale example. All agents are identified with a unique number, which is associated with an account number controlled by the mediator. Thus, the mediator can transfer money between the accounts of the participants. The mediator can also transfer items, which are also identified with unique numbers. The items have properties like their owner and price. The available functions are intended to perform the main operation between the agents. There are functions to advertise an item, to change its price, and to sell it to another agent. Since Soda is purely functional, the functions do not change the state of a market, but create new instances of the market. The actual communication between the agents is done through a library used in the Scala translation.

In order to enable the verification of the desired behavior, we make use of *invariants*. Invariants are logical assertions that are always valid during execution. They are helpful resources to prove that a system behaves according to predefined expectations. As Lean can be used to prove theorems, the Lean translation can serve to prove the preservation of invariants. The two invariants which we are interested in are the following:

- No item is created or destructed after modifying the state of an item;
- No account is created or destructed after modifying the state of an item.

The choice for representation is not trivial. Different ways of representing the market lead to a number of derived challenges. Sometimes, more efficient performance is achieved by adding some redundancy, but it makes it more difficult to prove invariants. Similarly, the definitions of the functions require consideration, as shown in Figure 1 where an intuitive recursive definition of the length of a list (`length_def`) should be equivalent to a tail recursive definition of the length of a list (`length_fl`). We prove that both definitions are equivalent, allowing us to use the intuitive definition for the proofs and the tail recursive definition for the execution.

4.1 Package Structure

The project is available (open-source) at https://github.com/julianmendez/market. It is designed to be compiled using sbt [5] to be executed in the JVM environment. A Bash script uses the Soda binary to translate the Soda files, including those proofs that can be verified with Lean. The project file structure follows the standard sbt/Maven layout. The project contains three packages:

- `core`, containing the core data types, the market, its functionality and the proofs;
- `parser`, containing a YAML parser; and

[5] https://www.scala-sbt.org

```
_tailrec_foldl [A : Type] [B : Type] (list : List [A] ) (current : B)
    (next : B -> A -> B) : B =
  match list
    case Nil ==> current
    case (head) :: (tail) ==>
      _tailrec_foldl [A] [B] (tail) (next (current) (head) ) (next)

foldl [A : Type] [B : Type] (list : List [A] ) (initial : B)
    (next : B -> A -> B) : B =
  _tailrec_foldl [A] [B] (list) (initial) (next)

length_fl [A : Type] (list : List [A] ) : Nat =
  _tailrec_foldl [A] [Nat] (list) (0) (
    lambda (accum : Nat) --> lambda (_elem : A) --> accum + 1)

_tailrec_length [A : Type] (list : List [A] ) (accum : Nat) : Nat =
  match list
    case Nil ==> accum
    case (_head) :: (tail) ==>
      _tailrec_length [A] (tail) (accum + 1)

length_tr [A : Type] (list : List [A] ) : Nat =
  _tailrec_length [A] (list) (0)

length_def [A : Type] (list : List [A] ) : Nat =
  match list
    case Nil ==> 0
    case (_head) :: (tail) ==> length_def [A] (tail) + 1
```

Fig. 1. The function `_tailrec_foldl` is a tail-recursive auxiliary function of `foldl`, which is a left fold (*foldl*) function for parameterized types. `length_fl`, `length_tr`, and `length_def` syntactically different but semantically equivalent ways of defining the length of a list of an arbitrary type A in SODA. The tail-recursive definitions (`length_fl`, `length_tr`) outperform the naive definition (`length_def`).

– `main`, containing the entry point class, which supports the console execution.

Firstly, the main core is a package that contains the classes to model the market. The main operations are:

– **deposit** *user amount* : works as an update-insert (upsert) function to declare a user and to put money in the linked account.
– **assign** *item user* : works as an upsert function to declare an item and to establish its owner.
– **price** *item amount* : has the double functionality of assigning the price of an item for sale, if the value is positive, or hiding it, if the value is zero.

– **sell** *item user* : transactionally transfers the possession of an item to a specified user, and money equivalent for the price from the new user's to the old user's account.

Secondly, the project has a package to read YAML files. A list of operations simulates the interaction that occurs in a MAS scenario. The parser creates a list of operations, which can later be executed on an instance of market.

The last relevant package is the *main* package, which contains the entry point and is intended to receive and parse parameters from the console.

The parser package and the core packages contain their counterpart testing classes. The parser test package has two layers of parsing: the syntactic layer, which is implemented using SnakeYAML[6], a Java YAML parser, and the semantic layer, where the operation objects are instantiated. The core test package instantiates a market and applies the parsed operations to it.

Before we start modeling our *market* example, we need to introduce two fundamental preliminaries that are crucial to enable formal verification: natural number support (Subsection 4.2) and list wrappers (Subsection 4.3).

4.2 Preliminary 1: Support for Natural Numbers

One of the challenges that we need to face is how we handle natural numbers in Soda. We use the standard notation in the literature where natural numbers are used as a synonym of non-negative integers, i.e. including the number 0. While Lean has natural numbers as part of its core (`Nat`), Scala does not have natural numbers as a type. In Soda, we implement the natural numbers based on a type `Nat`, and two constructors: `0` and `Succ_`. We define its translation as shown in Figure 2.

```
directive scala
type Nat = Int
object Succ_ {
  def apply (n : Int) : Int = n + 1
  def unapply (n : Int) : Option [Int] =
    if (n <= 0) None else Some (n - 1)
}

directive lean
notation "Succ_" => Nat.succ
```

Fig. 2. This table shows the Soda definition of `Nat`, to be translated to Scala and to Lean.

A good definition of natural numbers is crucial in order to accomplish two things: proofs and calculations. A recursive definition is needed to provide proofs in

[6] https://bitbucket.org/asomov/snakeyaml-engine

Lean, especially those involving recursive definitions such as the length of a list. However, some calculations may need to be time efficient, even with larger natural numbers.

In functional languages with ML syntax, the deconstruction is often done with pattern matching. While the Lean syntax allows us to use this type of syntax, in Scala this does not work off-the-shelf. The main issue is that our natural numbers in Scala are in fact `Int` objects, which makes addition of two numbers very efficient, and `Succ_` is a construction external to the objects. For that reason, we define the function *monus*1, which subtracts 1 from a given positive number until this number reaches 0. This is defined in Figure 3. The function *monus*1 is based on the function *monus*, denoted by $\dot{-}$, which handles subtraction in natural numbers. For two natural numbers a and b, it is defined as: $a \dot{-} b = \max(a - b, 0)$.

```
monus1 (index : Nat) : Nat =
  match index
    case Succ_ (k) ==> k
    case _otherwise ==> 0
```

Fig. 3. This table shows the definition of *monus1*, which for a positive number, it returns the previous number, and for other numbers, it returns 0.

4.3 Preliminary 2: List Wrappers

After having defined `Nat`, we define a type we name `ListWrapper`. This is actually not a list itself, but a toolbox that works as a wrapper for lists, containing methods and their proofs. Let us see how this type is constructed. Nil is the constructor for an empty list, :: is the list constructor, and Nat is a type for non-negative integers.

The function *foldl* (from *fold left*), as it is called in other functional programming languages like Haskell, applies a combining function to an iterable structure, a finite list in our case, and starts with an initial value. One relevant property of *foldl* is the possibility of implementing it as a tail-recursive function, which in turn can be optimized by a virtual machine or a compiler as a cycle. In other words, *foldl* is a purely functional approach to execute certain types of cycles. In addition to its efficient execution, *foldl* ensures termination on finite structures, like the list structure. In Figure 1, we can see `foldl`, which is an implementation of *foldl* that uses an auxiliary tail recursive function called `_tailrec_foldl`.

Although *foldl* is a useful resource for efficient and well-founded recursion, it becomes more complex when it is involved in proofs. Because of that, in our proofs, we provide (traditionally) recursive definitions, in addition to the tail-recursive *foldl* definitions.

SODA lists allow for access to any arbitrary position. In Scala, accessing at position i of a list a of type T would be notated $a(i)$, which is already a value of type T. However, when we need to access an element inside a proof, we need to ensure that i is in range, and thus that $a(i)$ is well defined. We present this in Figure 4, where we show how we access an element of a list. The function `get` visits all the elements of a list until it reaches the specified position. If that happens, it returns the content at that position (`Some (elem)`), otherwise, it returns `None`.

```
_tailrec_get_def [A : Type] (list : List [A] ) (index : Nat) (current
    : Nat) : Option [A] =
  match list
    case Nil ==> None
    case (head) :: (tail) ==>
      if current == index
      then Some (head)
      else _tailrec_get_def [A] (tail) (index) (Succ_ (current) )

get [A : Type] (list : List [A] ) (index : Nat) : Option [A] =
  _tailrec_get_def [A] (list) (index) (0)
```

Fig. 4. The function `get` retrieves an element safely, without throwing an exception if the index is out of range. This is necessary to ensure that the retrieved element is well defined.

Since SODA lists are immutable, modifying the state of a market requires creating new lists. This can be written in Scala for a mutable structure as $a(i) = e$, which updates the content of a list a at position i, by assigning it the value e. In Figure 5, we show the definition of `set_def`, and in Section 5, we prove that it preserves the length of the list, i.e. it does not create or remove elements.

```
set_def [A : Type] (list : List [A] ) (index : Nat) (element : A) :
    List [A] =
  match list
    case Nil ==> Nil
    case (head) :: (tail) ==>
      if index == 0
      then (element) :: (tail)
      else (head) :: (set_def [A] (tail) (monus1 (index) ) (element) )
```

Fig. 5. The function `set_def` is a tail recursive function that creates a new list with an element updated at a given index.

4.4 Modeling the Market

As we can see in Figure 6, money is modeled as an integer. An item is modeled as a pair ⟨owner, price⟩. A market is a pair containing the owner's money and the items. Conventionally, we say that an item is advertised if and only if its price is greater than 0. In the context of this example, money is modeled as a natural number (`Nat`), but in reality, we could allow users to have debt and use an integer instead.

```
class Money = Int

class Item

  abstract
    owner : Nat
    price : Money

end

class Market

  abstract
    accounts : List [Money]
    items : List [Item]

end
```

Fig. 6. A market is a structure composed of smaller structures.

The methods for manipulating a market or an item are separated in a module called `MarketMod`. This design style differs from the traditional object-oriented approach, where the methods pertaining to the market should be in the market class itself. We chose this design as a compromise in the integration of SODA, Scala, and Lean. While Scala uses a traditional object notation as in other object-oriented programming languages, Lean uses syntactic sugar to access functions inside modules, as in other functional programming languages. Since the SODA translators do not support the notational translation between the two paradigms, we place the functions in modules, and use classes as structures.

5 Formal Proofs

As we discuss above, the virtue of SODA resides in connecting efficient implementations and formal proofs in the same language. We decided to show a simple property: "Changing an element in a list does not modify its length". This property allows ensuring that a transaction does not change the length of a list. This

could be reformulated as: "Creating a list with `set_def` returns a list of the same length as the original".

5.1 Defining the Length of a List

To prove the goal, we use the definition of length given in Figure 1. Although this definition looks clear, it is not tail recursive, and for that reason it could be considered a naive definition of the length of a list. For example, in an average computer configuration, this function cannot compute the length of a list of 10,000 elements, since it would produce a stack overflow during execution. However, this definition greatly simplifies the proofs. A more applicable function to compute the length of a list is `length_tr`, which uses tail recursion. Its implementation is shown in Figure 1.

Figure 7 shows Theorem `len_tr_eq_len_def`, which states that the tail recursive function is equivalent to the naive definition. The proof uses Lemma `len_tr_accum`, which helps in the interpretation of the extra parameter of `_tailrec_length`.

Although fully understanding the proof requires understanding of Lean 4, we sketch how the proof is structured. Let us focus on Lemma `len_tr_accum` in Figure 7, which proves that the *accum* parameter accumulates the computed length. To achieve this, we apply an inductive strategy to the list (`induction list with`). If the list is empty (`nil`), it rewrites the definitions. If the list is not empty (`cons head tail ih`), it applies some definitions and uses the inductive hypothesis for the given parameters. Notice that the property has a value *accum*, which we can instantiate to different values to use the induction hypothesis.

This theorem is in fact shorter than the lemma, and it is also proven by applying induction. Since the equivalence does not have parameters, we first add them (`funext A list`). As before, the induction has two cases. The `nil` case is a direct rewriting of the definition. For the inductive step (`cons head tail ih`), we use the lemma and then the induction hypothesis.

Since we have proven that both functions are equivalent, we can use the efficient function (`length_tr`) for the definitions, and the naive function (`length_def`) for the proofs.

5.2 Updating a List

We have auxiliary functions and lemmas to prove that `set_def` does not change the number of elements. Figure 8 shows a property on `monus1` stating that applying it to the successor of a number returns the original number. In Figure 9, we can see the proof that `set_def` preserves the length of the list, and the proof is provided by induction on the list. For the base case of the empty list (`nil`), it suffices to provide the definitions of `set_def` and `length_def`. For the inductive case (`cons head tail ih`), after using the definitions of `set_def` and `length_def`, we split the proof into two cases based on the value of the index. If it is at the beginning, (`zero`), the definition of `monus1` is used. Otherwise (`succ`

```
directive lean
theorem
  len_tr_accum (A : Type) (list : List (A) )
    : forall (accum : Nat) ,
      _tailrec_length (A) (list) (accum) = _tailrec_length (A) (list)
          (0) + accum := by
    induction list with
    | nil =>
      intro n
      rewrite [_tailrec_length, _tailrec_length, Nat.zero_add]
      rfl
    | cons head tail ih =>
      intro n
      rewrite [_tailrec_length, _tailrec_length]
      rewrite [ih (1)]
      rewrite [ih (n + 1)]
      rewrite [Nat.add_assoc]
      rewrite [Nat.add_comm 1]
      rfl

directive lean
theorem
  len_tr_eq_len_def
    : length_tr = length_def := by
  funext A list
  rewrite [length_tr]
  induction list with
  | nil =>
    rewrite [_tailrec_length, length_def]
    rfl
  | cons head tail ih =>
    rewrite [_tailrec_length, len_tr_accum]
    rewrite [ih]
    rewrite [length_def]
    rfl
```

Fig. 7. Proof of Theorem `len_tr_eq_len_def`: it states that `length_tr` = `length_def`. This means that the tail recursive definition of length of a list is equivalent to the naive definition. The proof uses Lemma `len_tr_accum`.

k), after using the definitions of `monus1` and `length`, we employ the induction hypothesis.

6 Experiments

To test the efficiency of the market application, we run a series of experiments. The purpose of the experiments is to determine the viability of the example.

```
directive lean
theorem
  monus1_succ
    : forall (index : Nat),
      monus1 (Nat.succ (index)) = index := by
  intro idx
  rewrite [monus1]
  simp
```

Fig. 8. Lemma proving that *monus*1 works as the inverse of successor.

```
directive lean
theorem
  len_set (A : Type) (list : List (A)) (element : A)
    : forall (index : Nat),
      length_def (A) (set_def (A) (list) (index) (element) ) =
          length_def (A) (list) := by
  induction list with
  | nil =>
    intro idx
    rewrite [set_def, length_def]
    rfl
  | cons head tail ih =>
     intro idx
     rewrite [set_def, length_def]
     cases idx with
     | zero =>
       rewrite [monus1]
       rewrite [Nat.zero_eq]
       rfl
     | succ k =>
       rewrite [monus1]
       simp
       rewrite [length_def]
       rewrite [ih]
       rfl
```

Fig. 9. Theorem and proof that proves that `set_def` does not change the length of the provided list.

As a prerequisite, we have developed a tool that generates test instances. Each instance depends on three parameters: the number of users, the number of items, and the number of transactions.

Each test instance is a YAML file that follows some patterns. At the beginning, it contains the instructions to add the user accounts, with some amount of

money. After that, it adds the items assigned to some users, with some non-zero price. Finally, it adds all sell transactions together with a price change. This price change is needed because the market puts an item as not-for-sale after each transaction is successful.

We fix some values, and we assume that each user has, on average, 8 items to sell. We conducted two types of experiments, which are shown in Table 1. On the left, we choose an arbitrary batch of 65536 (2^{16}) transactions and we test the system with an exponentially growing number of users and items. On the right, we fix the number of users and items as 2048 and 16384 respectively, and we test the instances with an exponentially growing number of transactions. The experiments were run on a Linux Ubuntu 22.04.5 LTS computer equipped with 8 Intel cores i5-8350U CPU at 1.70 GHz and 32 GiB of RAM. The Bash script to run the experiments is provided in the project repository.

Table 1. The elapsed time in seconds of different synthetically generated instances. The columns show: the number of users, the number of items, the number of transactions, the elapsed time in seconds, and the number of transactions per second.

users	items	transac	time (s)	tr/s
1	8	65536	1.10	59578
2	16	65536	1.13	57996
4	32	65536	1.13	57996
8	64	65536	1.26	52013
16	128	65536	1.40	46811
32	256	65536	1.48	44281
64	512	65536	1.84	35617
128	1024	65536	2.32	28248
256	2048	65536	3.55	18461
512	4096	65536	5.79	11319
1024	8192	65536	10.41	6295
2048	16384	65536	19.89	3295

users	items	transac	time (s)	tr/s
2048	16384	256	4.35	59
2048	16384	512	5.05	101
2048	16384	1024	4.95	207
2048	16384	2048	4.87	421
2048	16384	4096	5.28	776
2048	16384	8192	6.36	1288
2048	16384	16384	8.23	1991
2048	16384	32768	13.14	2494
2048	16384	65536	19.89	3295
2048	16384	131072	36.67	3574
2048	16384	262144	68.49	3827
2048	16384	524288	126.88	4132

From Table 1, we conclude that the measurements show the effect of the overhead starting time of the application, as the first rows present very similar values. Another result is that the growth in the number of users and items seems to have a linear impact on the execution time. Regarding the transactions, once the number of users and the number of items are fixed, the required time appears to be linear with respect to the number of transactions. And, most importantly, with an average of around 4000 transactions per second, we demonstrate the viability of our example: the results indicate that an application of a multi-agent environment with formally verified pieces of code can also be efficient to run in production.

7 Conclusion

In this paper, we have demonstrated the applicability of SODA to the design of (so far simple) MAS components that can be integrated into the Scala and Java technology ecosystem, while also supporting formal verification with the proof assistant Lean. We claim that the use of proof assistants like Lean for MAS verification is a relevant research direction, as it allows for a more abstract and flexible formal analysis of MAS. However, given our demonstration, it is clear that more work is required to support the verification of MAS in a meaningful manner that is useful for software engineers. To move towards applicability, we hope to further advance this research in the following ways: *i)* by expanding proof-of-concept implementations to more complex scenarios that require the verification of additional fundamental MAS abstractions, such as agents' reasoning loops, providing reusable abstractions for verification; *ii)* by providing a more detailed analysis of the advantages and disadvantages of using proof assistants instead of model checkers for MAS verification; *iii)* by studying the applicability of our formal verification approach to real-world problems, such as the assurance of fairness properties in complex socio-technical systems. As an independent research direction, we consider it relevant to investigate whether and to what extent existing agent programming languages, such as protocol-based languages [17,18], can be extended or integrated to allow formal verification with proof assistants.

References

1. Amaral, C.J., Hübner, J.F., Kampik, T.: TDD for AOP: test-driven development for agent-oriented programming. In: Agmon, N., An, B., Ricci, A., Yeoh, W. (eds.) Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023. pp. 3038–3040. ACM (2023). https://doi.org/10.5555/3545946.3599165, https://dl.acm.org/doi/10.5555/3545946.3599165
2. Amaral, C.J., Kampik, T., Cranefield, S.: A Framework for Collaborative and Interactive Agent-oriented Developer Operations. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. AAMAS '20, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2020)
3. Barras, B., Boutin, S., Cornes, C., Courant, J., Filliatre, J.C., Gimenez, E., Herbelin, H., Huet, G., Munoz, C., Murthy, C., et al.: The Coq proof assistant reference manual: Version 6.1 (1997)
4. Bordini, R.H., Dennis, L.A., Farwer, B., Fisher, M.: Automated Verification of Multi-Agent Programs. In: 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008), 15-19 September 2008, L'Aquila, Italy. pp. 69–78. IEEE Computer Society (2008). https://doi.org/10.1109/ASE.2008.17, https://doi.org/10.1109/ASE.2008.17
5. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason, vol. 15. John Wiley & Sons (2007)

6. De Moura, L., Kong, S., Avigad, J., Van Doorn, F., von Raumer, J.: The Lean theorem prover (system description). In: Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25. pp. 378–388. Springer (2015)
7. Dennis, L.A., Fisher, M., Webster, M.P., Bordini, R.H.: Model checking agent programming languages. Automated Software Engineering **19**(1), 5–63 (2012). https://doi.org/10.1007/s10515-011-0088-x, https://doi.org/10.1007/s10515-011-0088-x
8. Galland, S., Gaud, N., Rodriguez, S., Hilaire, V.: Janus: another yet general-purpose multiagent platform. In: Proceedings of 7th Agent-Oriented Software Engineering Technical Forum (TFGASOSE-10) (2010)
9. Lamport, L.: Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley (2002), http://research.microsoft.com/users/lamport/tla/book.html
10. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: an open-source model checker for the verification of multi-agent systems. International Journal on Software Tools for Technology Transfer **19**(1), 9–30 (2017). https://doi.org/10.1007/s10009-015-0378-x, https://doi.org/10.1007/s10009-015-0378-x
11. Mendez, J.A.: Soda: An Object-Oriented Functional Language for Specifying Human-Centered Problems (2023). https://doi.org/10.48550/arXiv.2310.01961
12. Milner, R.: A theory of type polymorphism in programming. Journal of Computer and System Sciences **17**(3), 348–375 (1978). https://doi.org/10.1016/0022-0000(78)90014-4, https://www.sciencedirect.com/science/article/pii/0022000078900144
13. de Moura, L., Kong, S., Avigad, J., van Doorn, F., von Raumer, J.: The Lean Theorem Prover (System Description). In: Felty, A.P., Middeldorp, A. (eds.) Automated Deduction - CADE-25. pp. 378–388. Springer International Publishing, Cham (2015)
14. Paulson, L.C.: Isabelle: A generic theorem prover. Springer (1994)
15. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: Van de Velde, W., Perram, J.W. (eds.) Agents Breaking Away. pp. 42–55. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
16. Rodriguez, S., Gaud, N., Galland, S.: SARL: a general-purpose agent-oriented programming language. In: 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). vol. 3, pp. 103–110. IEEE (2014)
17. V., S.H.C., Singh, M.P., Chopra, A.K.: Kiko: Programming Agents to Enact Interaction Models. In: Agmon, N., An, B., Ricci, A., Yeoh, W. (eds.) Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023. pp. 1154–1163. ACM (2023). https://doi.org/10.5555/3545946.3598758, https://dl.acm.org/doi/10.5555/3545946.3598758
18. V., S.H.C., Singh, M.P., Chopra, A.K.: Mandrake: Multiagent Systems as a Basis for Programming Fault-Tolerant Decentralized Applications. In: Agmon, N., An, B., Ricci, A., Yeoh, W. (eds.) Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023. pp. 1218–1220. ACM (2023). https://doi.org/10.5555/3545946.3598765, https://dl.acm.org/doi/10.5555/3545946.3598765
19. Winikoff, M., Cranefield, S.: On the testability of BDI agent systems. IJCAI International Joint Conference on Artificial Intelligence pp. 4217–4221 (2015)