An Information-Flow Perspective on Algorithmic Fairness

Samuel Teuber, Bernhard Beckert

Karlsruhe Institute of Technology, Am Fasanengarten 5, 76131 Karlsruhe
{teuber,beckert}@kit.edu

Abstract

This work presents insights gained by investigating the relationship between algorithmic fairness and the concept of secure information flow. The problem of enforcing secure information flow is well-studied in the context of information security: If secret information may "flow" through an algorithm or program in such a way that it can influence the program's output, then that is considered insecure information flow as attackers could potentially observe (parts of) the secret. There is a strong correspondence between secure information flow and algorithmic fairness: if protected attributes such as race, gender, or age are treated as secret program inputs, then secure information flow means that these "secret" attributes cannot influence the result of a program. While most research in algorithmic fairness evaluation concentrates on studying the impact of algorithms (often treating the algorithm as a black-box), the concepts derived from information flow can be used both for the analysis of *disparate treatment* as well as *disparate impact* w.r.t. a structural causal model.

In this paper, we examine the relationship between quantitative as well as qualitative information-flow properties and fairness. Moreover, based on this duality, we derive a new quantitative notion of fairness called *fairness spread*, which can be easily analyzed using quantitative information flow and which strongly relates to counterfactual fairness. We demonstrate that off-the-shelf tools for information-flow properties can be used in order to formally analyze a program's algorithmic fairness properties, including the new notion of fairness spread as well as established notions such as demographic parity.

1 Introduction

The problem of enforcing secure information flow is well-studied in the context of information security and allows to rigorously analyze whether secret information can "flow" through an algorithm potentially influencing its result. A wide spectrum of methods and tools is available to analyze and quantify to what extent a given program satisfies information-flow properties (Lampson 1973; Denning 1975, 1976; Darvas, Hähnle, and Sands 2005; Smith 2009; Beckert et al. 2013; Klebanov 2014; Ahrendt et al. 2016; Biondi et al. 2018). This paper investigates the relationship between algorithmic fairness and information flow by building upon our initial findings from (Beckert, Kirsten, and Schefczyk 2022; Teuber and Beckert 2023a). An extended version of this paper can be found on arXiv (Teuber and Beckert 2023b). We demonstrate that there is a strong correspondence between qualitative as well as quantitative information-flow properties and algorithmic fairness: if protected attributes such as race, gender, or age are treated as secret program inputs, then secure information flow means that these "secret" attributes cannot influence the result of a program. Our approach is thus a white-box analysis of the algorithm and analyzes the decision-making *process* rather than only it's outcome. This allows us to find disparities in treatment even for cases where these disparities are not immediately observable in a statistical evaluation of the algorithm's outcome.

A large body of research, see e.g. (Beutel et al. 2019; Taskesen et al. 2021; Hardt, Price, and Srebro 2016; Mitchell et al. 2021), studies the disparate impact of algorithms using statistical metrics such as Equalized Odds or Equality of Opportunity (Hardt, Price, and Srebro 2016), Demographic Parity or certain forms of Individual Fairness. Many of these approaches consider the algorithm as a black-box, as they do not take into account the algorithm's decision mechanism, but only its outcome. While there exist some approaches for the white-box analysis of algorithms, e.g. (Albarghouthi et al. 2017; Ghosh, Basu, and Meel 2021, 2022; Bastani, Zhang, and Solar-Lezama 2019), they are fairness-specific and do not consider the relationship between information flow and fairness discussed here. Another direction of related research analyzed information flow on the level of software architecture (Ramadan et al. 2018, 2019) rather than program-level to mitigate fairness risks. That research is in some ways orthogonal to our work as it does not consider program code analysis and furthermore does not investigate the concrete correspondence between information flow and algorithmic fairness presented here.

The structure of this paper is as follows: After providing the necessary background on program analysis and causality (Section 2), we demonstrate and discuss the relationship between multiple flavors of qualitative information flow and demographic parity – including options to allow for certain justified exceptions to fairness guarantees (Section 3). Section 4 then extends the relation between information flow and fairness from the qualitative into the quantitative world. To this end, we derive a new quantitative notion of fairness

```
1 def c1(group, score):
2   return (group != 0)
```

(a) Group Decision

```
1 def c2(group, score):
2   return (score>=8)
```

(b) Score Decision

```
1 def c3(group, score):
2   if (group >= 6):
3     return (score >= 8)
4   else:
5     return (score >= 6)
```

(c) Mixed Decision

Figure 1: Three exemplary credit decision algorithms

called *Fairness Spread*, which can be computed using quantitative information-flow analysis tools. In Sections 3 and 4 we assume the statistical independence of protected group attributes and unprotected attributes which initially limits our approach to disparate treatment alone. Section 5 demonstrates how this limitation can be lifted in the qualitative and quantitative approach by integrating causal models into our approach. Finally, Section 6 demonstrates the applicability of our approach on two case studies.

Throughout this paper, we will demonstrate our ideas using the three exemplary programs shown in Figure 1: All programs make a loan allocation decision based on a protected group attribute (`group`) and an unprotected variable containing the credit score (`score`). As group attribute, we assume a protected category such as gender, rage or age. The algorithms vary in their use of the two variables: While Figure 1a is clearly biased as its decision is solely dependent on the group attribute, the other two decision algorithms use the credit score as well (Figure 1c) or exclusively (Figure 1b). There exist a variety of off-the-shelf tools for the analysis of programs w.r.t. information-flow properties. For example, we will be using the interactive theorem prover KeY (Ahrendt et al. 2005, 2016), which features a (partially) automated analysis technique for qualitative information-flow properties, the automated information-flow analysis tool Joana (Graf, Hecker, and Mohr 2013; Snelting et al. 2014) and the tool counterSharp (Teuber and Weigl 2021) which allows the sound, bit-precise quantiative analysis of C programs using propositional model counting.

2 Preliminaries

Program Analysis This work analyzes decision procedures or more specifically it concerns with program code analysis. As the algorithm's inputs, we consider a protected group attribute $g \in \mathcal{G}$ and an independent, unprotected input $u \in \mathcal{U}$. The output of the algorithm is a decision $d \in \mathcal{D}$ (e.g., whether a person is awarded a credit or not). For a given group and unprotected attribute, we will usually describe the algorithm as a deterministic function P that returns a decision $d = P(g, u)$. Note, that this restriction is not necessarily relevant for practical use: A probabilistic algorithm can be turned into a deterministic algorithm by adjusting the domain of the unprotected attribute $\mathcal{U}$. We assume $\mathcal{G}$ and $\mathcal{U}$ to be finite which is a reasonable assumption for realistic programs: Even for the case where inputs are technically continuous they can be quantized with a user-defined precision.

We now consider the random variables $G \in \mathcal{G}$ and $U \in \mathcal{U}$, which we assume to be independent. Section 5 will demonstrate how we can extend our approach to model correlations between unprotected and protected variables. By entering G and U into P, we obtain the random variable $D = P(G, U)$ which is dependent on the function P as well as G and U. The probability $\Pr[D = d]$ that D takes on a certain value $d \in \mathcal{D}$ is then given by:

$$\sum_{(g,u)\in\mathcal{G}\times\mathcal{U}} \Pr[P(g,u) = d] \Pr[G = g] \Pr[U = u].$$

By analyzing the (conditional) distribution of the random variable D, we are able to analyze properties such as the probability that an individual of group $g \in \mathcal{G}$ obtains an outcome $d \in \mathcal{D}$.

Causal Models Section 5 will use causal models and counterfactuals to extend our approach to dependent distributions of G and U. To this end, we briefly introduce causal models and the notion of counterfactual fairness following the presentations of (Pearl 2009) and (Kusner et al. 2017):

Definition 1 (Causal Model)**.** *We define a* causal model $C = (V, U, F)$ *to consist of a set* $V = \{V_1, \ldots, V_n\}$ *of modeled variables, a set* B *of latent background variables for which we assume no causal mechanism, and a set* $\{f_1, \ldots, f_n\}$ *of functions defining the variables* V_i*, i.e.* $V_i = f_i\left(\mathrm{pa}_i, B_{pa_i}\right)$ *where* $\mathrm{pa}_i \subseteq V \setminus \{V_i\}$ *and* $B_{pa_i} \subseteq B$*.*

Here, the variables in pa_i are the *parents* of V_i and we assume that the computation of all the V_i from background variables in B has the structure of a directed acyclic graph. By abuse of notation, we will sometimes refer to B as a single random variable. Given a causal model C, a variable V_i and a value x, we can perform an *intervention* $C_{V_i \leftarrow x}$ resulting in a modified causal model C' that is the same as C except for f_i which is set to $f_i = x$. For our purposes, we can interpret C as a deterministic program computing all $V_i \in V$ when provided with values for the background variables B. Given a decision procedure $P(G, U)$ and a causal model C containing variables G, U we denote as $\hat{P}_C(B)$ the program which takes B as input, computes G and U through C and then returns $P(G, U)$. Moreover, we denote with $\hat{P}_C(g, B)$ the program which computes G and U through $C_{G \leftarrow g}$ and returns the result of P. Using this notation, we can now define counterfactual fairness as follows:

Definition 2 (Counterfactual Fairness)**.** *A program* P *with inputs* G *and* U *is counterfactually fair with respect to a causal model* C *iff, for any* $g_1, g_2 \in \mathcal{G}, u \in \mathcal{U}, d \in \mathcal{D}$*, it holds that:* $\Pr\left[\hat{P}_C(g_1, B) = d \,\middle|\, U = u, G = g_1\right] = \Pr\left[\hat{P}_C(g_2, B) = d \,\middle|\, U = u, G = g_1\right]$

It is worth noting, that in this scenario we condition on $G = g_1$ in *both* cases as B must be conditioned w.r.t. to the factual outcome (not the counterfactual).

3 Qualitative Information Flow and Demographic Parity

Information Flow is a well-studied concept originally introduced for software security analysis, see, e.g., (Lampson 1973; Denning 1975; Darvas, Hähnle, and Sands 2005;

Smith 2009; Beckert et al. 2013; Klebanov 2014; Ahrendt et al. 2016; Biondi et al. 2018). Consider a program P for checking whether a user entered the right password. Such a program will have two inputs: The correct password and the password entered by the user. Ideally, we would want that such a program will not leak any information about the correct password. An information-flow analysis checks whether information about the secret password is leaked by the program resp. how much information is leaked. A "bad" program that, for example, returns the full secret password to the user would be said to contain insecure information flow. To this end, we distinguish between variables with *high* security status (e.g., the correct password) and variables with *low* security status (e.g., the user's input and the program's return value). In its most basic form, insecure information flow is defined as follows: a program contains insecure information flow iff there exist two high security status values h, h' and a low security status value l such that the program returns different outputs for the inputs (h, l) and (h', l). That definition is sufficient for many applications. But, even a "good" password checker will leak some information about the password and have insecure information flow in the above sense because the output depends on whether the entered password is correct and, thus, depends on the secret. Therefore, there are various refined flavors of information flow that take into account such considerations. From a formal methods perspective, information flow is a *hyperproperty*: Where regular program properties talk about a single program run and analyze whether it satisfies a given specification, hyperproperties make statements about multiple program runs and their releation to each other. Due to the large research body on information flow, there exist many tools for analysing information-flow hyperproperties.

In the following, we demonstrate that information flow provides an interesting framework for the analysis of Algorithmic Fairness. We analyze what information about a protected group variable is leaked to the low security decision variable. Thus, the group variable G is assigned high security status while the unprotected variable U as well as the decision D are assigned low security status. A crucial assumption in this as well as the following section is the independence of the random variable describing the group $G \in \mathcal{G}$ and the random variable describing unprotected properties $U \in \mathcal{U}$. In Section 5, we demonstrate that we are nonetheless able to analyze the case of correlated variables as well when provided with a model of their dependence.

3.1 Unconditional Information Flow and Fairness

We first define the basic information flow property. We call this property *Unconditional Noninterference* and it represents the strictest variant of secure information flow:

Definition 3 (Unconditional Noninterference, see e.g. (Ahrendt et al. 2016))**.** *A program P satisfies* unconditional noninterference *iff for all $g_1, g_2 \in \mathcal{G}$ and for all $u \in \mathcal{U}$ it holds that $P(g_1, u) = P(g_2, u)$.*

This definition forbids any and all information flow between G and the outcome D of the program. The notion of Unconditional Noninterference is similar to fairness through unawareness (Grgic-Hlaca et al. 2016). However, our approach decidedly allows the program P to *process* an input from $\mathcal{G}$ as long as it does not influence the outcome (see, e.g., the example in Section 6.1). There is also a correspondence of Unconditional Noninterference to counterfactual fairness (Kusner et al. 2017), but that concept typically uses the rich framework of causality to reason about the relations between G and U – a relation we will intentionally only discuss in Section 5 in order to distinguish disparate treatment and impact. There is furthermore a close relation to Demographic Parity (Dwork et al. 2012; Mitchell et al. 2021):

Lemma 1 (Unconditional Noninterference implies Demographic Parity)**.** *Let P be a program and let G, U be distributed arbitrarily, but independently. If P satisfies unconditional noninterference, then the outcome of P satisfies* demographic parity*, i.e., for all $d \in \mathcal{D}, g_1, g_2 \in \mathcal{G}$ it holds that* $\Pr[P(G,U) = d | G = g_1] = \Pr[P(G,U) = d | G = g_2]$

See proof in extended version. We now analyze the example programs shown in Figure 1 with respect to Unconditional Noninterference: For the case of Figure 1a, we observe that for all u it holds that $\texttt{c1}(0, u) \neq \texttt{c1}(1, u)$. Thus, this first program as well as Figure 1c do not satisfy unconditional noninterference – contrary to the program in Figure 1b which ensures demographic parity.

The great benefit of the correspondence between information flow and fairness is that we can use information-flow tools for fairness analyses. Accordingly, we can use methods such as the program verification tool KeY (Ahrendt et al. 2016) to prove that the program in Figure 1b satisfies unconditional noninterference and, thus, demographic parity.

Note, that unconditional noninterference is a strictly stronger property than demographic parity and that there may be cases where an algorithm satisfying demographic parity does not satisfy noninterference: When disparate treatment does not lead to disparate impact in terms of the binary outcome, a program not satisfying unconditional noninterference may still yield demographic parity in its results and the disparity will initially be invisible when analyzed statistically. At the same time, a program could equally be considered unfair in this case due to the decision making process that uses a protected attribute. Appendix A.5 demonstrates this on a concrete example.

In practice, it will not always be possible to conform to the strict standard of unconditional noninterference. For example, in the case of a password checker we will have to leak the information whether or not the user entered the right password which would violate unconditional noninterference. Similarly, there may be constraints which make it impossible to conform to unconditional noninterference in the fairness scenario. This raises the question in which ways one can relax this property while still retaining certain fairness guarantees. To this end, we can proceed in three manners:

1. We provide *restricted categories*, i.e., subgroups of all individuals, according to which an algorithm may discriminate while ensuring that discrimination only happens along these categories (restricted information flow, see Section 3.2).

2. We provide conditions under which a *declassification* of group attributes is allowed for cases where this is morally and/or legally justified (conditional information flow, see Appendix A.1).
3. We analyze information flow as a quantitative rather than a qualitative property (quantitative information flow, see Section 4).

3.2 Restricted Information Flow and Fairness

To make the allowed types of discrimination explicit, one can rely on the notion of *Restricted Information Flow* where we provide categories within which all individuals must be treated equally. To this end, we introduce a *Restricted Classification*:

Definition 4 (Restricted Classification)**.** *A* Restricted Classification *is a function $R : \mathcal{G} \times \mathcal{U} \to \mathcal{R}$ mapping each combination (g, u) to a class r of a finite set $\mathcal{R}$.*

Based on this notion of restricted classification, we can introduce *Restricted Information Flow*:

Definition 5 (Restricted Information Flow)**.** *Let R be some restricted classification. A program P satisfies* Restricted Information Flow *for R iff, for all $g_1, g_2 \in \mathcal{G}$ and all $u \in \mathcal{U}$:*

$$R(g_1, u) = R(g_2, u) \;\; \textit{implies} \;\; P(g_1, u) = P(g_2, u) \;\; .$$

We can relate this notion of information flow with a conditional variant of demographic parity:

Lemma 2. *Let P be a program and G, U be distributed arbitrarily, but independently. If P satisfies Restricted Information Flow for a restricted class R, then, for each $r \in \mathcal{R}$, the outcome of P satisfies conditional demographic parity with respect to the condition $R(G, U) = r$, i.e., for all $r \in \mathcal{R}$ and all $d \in \mathcal{D}, g_1, g_2 \in \mathcal{G}$:*

$$\begin{aligned} & \Pr[P(G,U) = d \mid G = g_1, R(G,U) = r] \\ = \; & \Pr[P(G,U) = d \mid G = g_2, R(G,U) = r] \end{aligned}$$

See proof in extended version. For example, for the program in Figure 1c, we can define a restricted classification $R(g, u) : \mathcal{G} \times \mathcal{U} \to \{\texttt{true}, \texttt{false}\}$ which maps to categories as follows: $(g, u) \mapsto (6 \leq u \wedge u < 8 \wedge g \geq 6)$. Then, Restricted Information Flow holds for Figure 1c with respect to R. A possible intuition behind this restriction is the idea that when an individual close to retirement age applies for a loan, it may be a justified disparate treatment that the loan is only granted if the individual can afford higher credit rates to ensure a timely redemption before retirement — which is only possible for individuals with a higher credit score. Thus, it may be justified to consider an individual's group (in this case their age in decades) if their credit score is in between the regular threshold and the threshold for near-retirement loans[1]. We can analyze and prove the Restricted Information Flow property using the interactive theorem prover KeY. Restricted information flow makes the limitations of a fairness guarantee explicit and thus allows an audit of the classification along which individuals are disparately treated. Providing auditors with the classification function R can represent the starting point of a debate on whether the classification is justified and chosen correctly.

In cases where we only require a fairness guarantee limited to a subset of individuals, we place all individuals of that subset into one category while placing all other individuals into singleton categories each only containing a single possible input. A more intuitive version of this approach (*Conditional Information Flow*) is discussed in Appendix A.1.

4 Quantitative Information Flow and Fairness

Quantitative Information Flow, see e.g. (Smith 2009), transforms the qualitative security analysis of software into a metric for a software's security. A good quantitative metric for information flow will satisfy two requirements: 1. Its minimum (optimal value) is reached if the program satisfies unconditional noninterference. 2. A less secure program will have a higher value of the metric. Quantitative Information Flow can be used to compare different program variants (to analyze which version is more secure) or to provide a bound on the likelihood of a breach. One well-established technique for the quantitative analysis of information flow is the metric of *min-entropy* (Smith 2009). Its value is based on the concept of *Vulnerability*, which corresponds to the probability that an attacker who observes the low inputs and the low output can correctly guess a variable of high security status in a single try. Transferring that notion to the context of fairness, we define Conditional Vulnerability as follows:

Definition 6 (Conditional Vulnerability)**.** *For a program P and random independent variables G, U, we define the* Conditional Vulnerability *$V(G \mid P(G,U), U)$ of G w.r.t. $P(G,U)$ and U as follows:*

$$\sum_{u \in \mathcal{U}} \sum_{d \in \mathcal{D}} \max_{g \in \mathcal{G}} \big(\Pr[P(G,U) = d, U = u] \cdot \Pr[G = g \mid P(G,U) = d, U = u] \big)$$

We demonstrate this definition on a small example from the security context: Consider a password checker for a correct password $G \in \mathcal{G} = \{1, 2, 3\} = \mathcal{U}$. The program returns 1 if the entered password $U \in \mathcal{U}$ was correct and 0 otherwise. If an attacker enters a password $U \in \mathcal{U}$, there are two possibilities: Either the password is correct in which case P returns 1 and the attacker now knows the secret G or the password is incorrect and the attacker has a 50% chance of guessing the password correctly. Thus, the Conditional Vulnerability of G with respect to $P(G,U)$ and uniform U, i.e., the likelihood of an attacker guessing the correct password after 1 try, is:$1/3 * 1 + 2/3 * 1/2 = 2/3$. Conditional Vulnerability can be easily computed using off-the-shelf tools (see Appendix A.2). Below, we discuss whether Conditional Vulnerability is indeed a suitable metric for an algorithm's fairness. We consider this question for binary decisions, i.e., $|\mathcal{D}| = 2$. Similarly as for security, there are two requirements that a fairness metric should satisfy: 1. The metric

[1]This is meant as an example and does not make a claim as to whether or not this practice is indeed legally or morally justified.

returns its minimum (optimal) value if the program satisfies (un)conditional demographic parity; 2. A less fair program has a higher value of the metric.

4.1 Naive Approach

Unfortunately, a naive approach where Conditional Vulnerability is used as a fairness metric does not satisfy the above two requirements. This is, because $V(G \mid P(G,U),U)$ measures two properties at the same time: On the one hand, we measure how much information a program leaks about the protected variable G. On the other hand, we also measure how easy it is to guess G based on its probability distribution. Thus, for Conditional Vulnerability, it does not make a difference whether one can easily guess G due to information leakage in the program or due to the distribution over G. To this end, consider the following example:

Example 1 (Arbitrarily unfair program for constant Conditional Vulnerability)**.** *Let there be two groups $\mathcal{G} = \{0,1\}$. For simplicity, we assume $\mathcal{U} = \{0\}$ for the unprotected attribute. As before, we consider two possible outcomes, i.e., $\mathcal{D} = \{0,1\}$. Let 1 be an advantaged majority group and let $\mu_1 = \Pr[G=1]$. We now consider the value of $V(G \mid P(G,U),U)$ which can be computed as:* $\max_{g\in\mathcal{G}}(\Pr[G=g]\cdot\Pr[P(G,0)=0 \mid G=g]) +$ $\max_{g\in\mathcal{G}}(\Pr[G=g]\cdot\Pr[P(G,0)=1 \mid G=g])$. *Consider the case where $\min_{d\in\mathcal{D}}\Pr[P(1,0)=d] \geq \frac{1-\mu_1}{\mu_1}$. Then, no matter which output d is chosen by P for $g=0$, the product $\mu_1 \cdot \Pr[P(1,0)=d]$ will always be larger than $(1-\mu_1)\cdot\Pr[P(0,0)=d]$. Consequently, the formula above is equal to μ_1. Thus, for highly uneven distributions of G, the value of V is independent of P and, in particular, independent of the behavior of P for $G=0$. For example, in a setup where $\Pr[G=1] = 0.98$, we could have an (unfair) program P such that $\Pr[P(1,0)=1] \leq 1-(1-0.98)/0.98 \approx 0.979$ and $\Pr[P(0,0)=1] = 0$, which would be indistinguishable from a (less unfair) program P' with $\Pr[P'(0,0)=1] > 0$. The reason for the indistinguishability is the high probability of $G=1$ which turns $G=1$ into the best guess for an attacker independently of P's outcome. Due to this, P's decision for $G=0$ does not influence V.*

This example demonstrates an extreme case of the abovementioned problem that Conditional Vulnerability does not make a distinction as to why G is easy to guess, which is obviously an an undesirable property for a fairness metric as it violates the second requirement from above. This problem stems from the security context in which Conditional Vulnerability was originally defined: When considering the amount of information leaked about a secret variable, it is a justified assumption that an attacker (whose likelihood of success we are trying to measure) will use public information on the probability distribution of the secret. In this regard, however, the fairness setting differs from the security setting: When measuring fairness, we consider information leakage of the group attribute problematic independently of the underlying group distribution. For example, a biased credit algorithm would be deemed inacceptable independently of whether it discriminates against a group making up 10% or 50% of the population. Therefore, we propose to measure the Conditional Vulnerability of a program under the assumption that G is distributed uniformly. If the measurement of V is performed in a world with a uniform distribution of G, we can use it to compute a metric that we call the *Fairness Spread* S corresponding to a variant of demographic parity that is weighted by U. In the following section, we are going to demonstrate that this metric satisfies the requirements from above. Furthermore, we are going to prove that Fairness Spread is independent of G's distribution which allows us to assume G's uniformity when computing S even for cases where G is not in fact uniformly distributed.

Program	$\lvert\mathcal{U}\rvert$	Count	V	S
`c1`	10	20	0.2	1.0
`c2`	10	10	0.1	0.0
`c3`	10	12	0.12	0.2
`c3-non-uniform`[2]	100	130	0.13	0.3

Table 1: Results of quantitative analysis on examples in Figure 1: Each analysis took less than 3 seconds of computation time, count corresponds to $\lvert\{(u,d)\in\mathcal{U}\times\mathcal{D} \mid \exists g\in\mathcal{G}.d=P(g,u)\}\rvert$, V and S correspond to conditional vulnerability and fairness spread.

4.2 The Fairness Spread Metric

To improve on the naive approach (see previous section), we propose an information-flow-based, quantitative fairness metric. We call our metric Fairness Spread and initially introduce it for the case of independent G and U (this restriction is lifted in Section 5):

Definition 7 (Fairness Spread)**.** *For independent variables $G\in\mathcal{G}$, $U\in\mathcal{U}$, $\lvert\mathcal{D}\rvert = 2$ and a program P we define the* fairness spread *$S(G,U,P)$ as follows where $\mu(g,u) = \Pr[P(G,U)=1 \mid G=g, U=u]$:*

$$\sum_{u\in\mathcal{U}} \Pr[U=u]\cdot \max_{g_1,g_2\in\mathcal{G}} (\mu(g_1,u)-\mu(g_2,u))$$

In essence, Fairness Spread quantifies the weighted disparity in treatment between the most advantaged and most disadvantaged groups for each possible assignment of U and is thus related to demographic parity. Fairness Spread is also related to a quantitative metric called Preference (Borca-Tasciuc et al. 2023), but differs from it by accounting for an individual's counterfactual outcome instead of comparing group conditional probability weight in absolute terms. Note, that this metric satisfies our requirements from above: $S(G,U,P) = 0$ implies demographic parity, and if an algorithm further diverges from demographic parity for the most advantaged/disadvantaged group $S(G,U,P)$ increases. Fairness Spread has an additional nice property, namely that it is independent of the distribution over G:

Lemma 3 (Independence of G)**.** *Let $G_1, G_2 \in \mathcal{G}$ be two random variables with independent arbitrary distributions*

[2]For the analysis of a non-uniformly distributed U we resized $\mathcal{U}$ to 100 to more easily model the non-uniform distribution.

with non-zero probabilities for all $g \in \mathcal{G}$. Then, for any program P and any independent random variable $U \in \mathcal{U}$: $S(G_1, U, P) = S(G_2, U, P)$.

This matches the intuition that discrimination of a group is undesirable irrespective of the group's population share. By fixing G to be uniformly distributed when computing $V(G \mid P(G,U), U)$ the obtained value can be mapped exactly to the notion of Fairness Spread as seen in Theorem 1.

Theorem 1 (Conditional Vulnerability measures Fairness Spread)**.** *Let $G \in \mathcal{G}$ and $U \in \mathcal{U}$ be independent random variables and let G be uniformly distributed. Then:*

$$S(G, U, P) \quad = \quad |\mathcal{G}| \cdot V(G \mid P(G,U), U) - 1$$

See proof in extended version. Since we already discussed the suitability of Fairness Spread as a metric for fairness, Theorem 1 provides a sensible formalism to measure an algorithm's fairness. Furthermore, Lemma 3 enables us to compute S even in setting where G is not uniformly distributed thus ensuring a wide applicability.

4.3 Application to Examples

We can now quantify how fair resp. unfair the programs from Figure 1 are with respect to the metric of Fairness Spread. To this end, we analyzed all three programs for a setting with 10 groups (e.g., age ranges) and a credit score of U between 1 and 10. Initially, we assumed a uniform distribution of credit scores. In addition, we analyzed `c3` (Figure 1 (b)) using a wrapper program that adjusted the probability of a credit score between 6 to 7 to be 30% (instead of 20% in the uniform case). All analyses are summarized in Table 1 and were computed using the counterSharp tool which allows the quantitative analysis of C-programs. Recent advances in (approximate) Model Counting (Chakraborty, Meel, and Vardi 2021; Yang and Meel 2023) and quantitative program analysis show that this technique scales even to programs with more than 1000 LOC (Teuber and Weigl 2021). The computed results match our previously developed intuition for the three programs: While `c1` is entirely unfair due to its decision-making solely based on `group`, the program `c3` is less fair than `c2`, but fairer than `c1`, due to its limited use of the `group` variable. In the table's last line, we see that `c3` becomes more unfair according to S if a larger share of the population have a credit score of 6 or 7. That makes sense because in this case a larger share of individuals with `group >= 6` will be denied credit in comparison to younger individuals with the same score.

5 Modeling Dependent Variables

The previous sections assume the independence of G and U, which is in general a strong – if not too strong – assumption for fairness analyses. This raises the question of how our approach can be extended to the analysis of disparate impact in the presence of dependencies between G and U. Fortunately, as our approach analyzes programs, we can also analyze the structural equations of a causal model. This observation leads to the following lemma for binary classifiers where $\hat{P}_C$ is the program composed of P and a structural model C as defined in Section 2:

Lemma 4 (Coincidence of Fairness Spread and Counterfactual Fairness)**.** *Let P be some program with $|\mathcal{D}| = 2$ and C some causal model over background variables B. Then, P is counterfactually fair iff $S\left(G, B, \hat{P}_C\right) = 0$.*

See proof in extended version. As a consequence of Lemma 4, a binary decision procedure P is counterfactually fair iff $\hat{P}_C(G, B)$ satisfies unconditional noninterference allowing the application of the entire machinery of Section 3. We have furthermore once again satisfied the first of the two requirements for a fairness metric outlined in Section 4, as a fairness spread of 0 coincides with counterfactual fairness. Addressing the second requirement (a larger value implies more unfairness), we can analyze the probability of a random individual having a counterfactual. To this end, we define a difference function which, given a concrete assignment to the background variables $B = b$ and a program P, provides the maximal deviation of counterfactuals. This can be defined as $\mathrm{Diff}_C(P, b) = \max_{g \in \mathcal{G}} \left| \hat{P}_C(b) - \hat{P}_C(g, b) \right|$. Note, that for binary decisions Diff_C is 0 or 1. Using this function, we can compute the probability of a random individual having a deviating counterfactual, i.e., $\Pr[\mathrm{Diff}_C(P, B) = 1]$ which we can bound through Fairness Spread:

Lemma 5. *Let P be some program and C some causal model over background variables B, and let $|\mathcal{D}| = 2$. Then: $\Pr[\mathrm{Diff}_C(P, B) = 1] \leq S\left(G, B, \hat{P}_C\right)$. And for $|\mathcal{G}| = 2$, the two terms are equal.*

See proof in extended version. Note, that the probability term bounded by Lemma 5 is – just like S – independent of G's distribution as it merely talks about the existence of a deviating counterfactual w.r.t. any other group. As an example, consider the program in Figure 1c: We previously established that the program is less fair than Figure 1b according to Fairness Spread. We will now reevaluate the fairness of this program assuming a causal model. To this end, we assume `score` is provided externally and computed based on knowledge about `income` and a client's `zipCode` (encoded as a ranking) which we cannot observe. The causal model C can be formalized as follows where we assume that different age groups live in different regions:[3]

$$\begin{aligned} \texttt{group} &\coloneqq B_1 \\ \texttt{income} &\coloneqq B_2 \\ \texttt{zipCode} &\coloneqq \textbf{if} \quad (\texttt{group} \geq 6) \quad B_3 \quad \textbf{else} \quad B_4 \\ \texttt{score} &\coloneqq \texttt{income} + \texttt{zipCode} \end{aligned}$$

We now encode the causal model as a program and analyze the programs `c2` and `c3` using the tooling described in Section 4.[4] The analysis shows that you are more likely to

[3]The causal models presented in this paper are just examples. We do not claim that the presented causalities do in fact exist.

[4]We assume $B_1 \sim \mathcal{U}_d(0, 9)$, $B_2 \sim \mathcal{U}_d(0, 9)$, $B_3 \sim \mathcal{U}_d(-1, 5)$, $B_4 \sim \mathcal{U}_d(-3, 3)$ where $\mathcal{U}_d$ is the discrete uniform distribution.

have a deviating counterfactual when processed by c2 than when processed by c3: While we obtain a Fairness Spread of 0.23 for c3, the Fairness Spread of c2 is 0.27 (analyses were performed in less than 2s using counterSharp). This result remains the same when, only considering the groups group $= 5$ and group $= 6$. Intuitively, this can be understood as follows: If your age attribute is 5 and your data is processed by c2 you have a 27% chance that you have a counterfactual with a deviating result for group $= 6$. On the other hand, when your data is processed by c3 this probability drops to 23%, indicating that c3 is fairer than c2 w.r.t. the above causal model. This is due to the (unfair) influence of group on score which c2 (partially) counteracts. This demonstrates that analyses assuming independence are not always sufficient and can even lead to misleading results (indicated by the difference to the results in Table 1, where c2 is fairer than c3).

Through the strong relation of Conditional Vulnerability and Counterfactual Fairness expressed in Lemmas 4 and 5 we can seamlessly apply qualitative and quantitative information flow analyses in contexts with dependent variables by suitably encoding their corresponding causal model.

While in the example above the disparate treatment based on zipCode leads to a disparate impact which is deemed inacceptable, there may be cases in where such disparities are justified. In such cases, an extension called path-dependent counterfactual fairness (Kusner et al. 2017; Loftus et al. 2018) can be used which is equally supported by our Information-Flow based approach (see Appendix A.3).

6 Case Studies

We applied our approach to two case-studies, showcasing both the qualitative (Section 6.1) as well as the quantitative (Section 6.2) approach[5]: First, we show how our approach can be used to analyze fairness in the context of German Tax computation. Moreover, we demonstrate our causal quantitative analysis on the computation of car insurance premiums. More applications can be found in Appendix A.5.

6.1 Analysis of German Tax Computation

German employers are required by law to deduct an employee's wage tax from their salary before payout. To this end, it is necessary that employers can reliably compute an employee's wage tax. Errors in this computation could lead to payouts of parts of the wage being delayed for over a year (in case the employee files a tax statement) or never being paid out. For that purpose, the German government provides a PDF file containing 26 pages of flowcharts outlining the computation of the german wage tax (Federal Ministry of Finance 2022b) as well as a Java implementation (Federal Ministry of Finance 2022a). This software consists of 1.5k LOC and solely operates on a global state while reusing many intermediate variables. As Germany has a church tax, which this Java program also computes, it requires a person's religious affiliation as input (passed as an integer). This raises the question of whether (e.g., through a program error) a person's religious affiliation also influences the wage tax (rather than only influencing the church tax). Notably, this property cannot be checked through exhaustive testing as the program has 35 inputs (21 of which can be arbitrary integers). Therefore, we analyzed the Java programs from the years 2015–2023 using the information flow analysis tool Joana (Graf, Hecker, and Mohr 2013; Snelting et al. 2014) and were able to show that there is no disparate treatment in the wage tax computation. Note, that this analysis was fully automated and demonstrates the scalability of this approach.

6.2 Analysis for Car Insurance Decisions

This case study is adapted from (Kusner et al. 2017), who analyzed the example of a car insurance company increasing premiums if an individual registers a red car. Instead of color we consider engine power, which in our causal model is influenced both by gender and aggressiveness of the driver. The probability of an accident is solely determined by aggressiveness. The causal model can be summarized as follows where the insurance company wants to increase the premium whenever accident > 12:

$$\begin{aligned} \texttt{gender} &:= B_1 \\ \texttt{aggressive} &:= B_2 \\ \texttt{engine} &:= 3 * \texttt{gender} + 8 * \texttt{aggressive} \\ \texttt{accident} &:= 20 * \texttt{aggressive} \end{aligned}$$

Assuming that aggressive cannot be observed directly, the insurance company has the choice of either building a linear classifier only considering the observable variable engine or considering both engine and gender (see Appendix A.4). We quantitatively analyzed the two programs implementing the two possibilities w.r.t. the above causal model and find that the algorithm that takes both engine and gender into account is counterfactually fair while the classifier only considering engine has a Fairness Spread of 0.36.[6] This is, because considering gender removes the aggressiveness independent imbalance introduced through engine power. Note, that the latter algorithm does not process the group variable. But, using causal modeling and information flow, we can still find analyze its impact.

7 Discussion and Conclusion

This paper demonstrates numerous interconnections between the fields of Information Flow and Algorithmic Fairness. In particular, our work demonstrates that pre-existing Information Flow approaches can be used to analyze fairness questions. We present an approach for the analysis of concrete program code instead of abstract models or empirical evaluations – even taking into account variable dependencies through the use of causal modeling. We believe that the ability to analyze a decision procedure's actual code is one of the major strengths of the approach presented in this paper. Some potential limitations and ethical considerations of our approach are discussed in Appendix A.5 In future work, we plan to expand the tooling available for the analysis of information flow to ML systems allowing for the analysis of such decision making systems in a manner similar to classical program code.

[5] Artifact available at (Teuber and Beckert 2023c)

[6] This assumes $B_1 \sim \mathcal{U}_d(0,1), B_2 \sim \mathcal{U}(0,1)$

Ethical Statement

The techniques outlined in this work allow for the analysis of complex, intricate fairness properties on the program code level. This approach has multiple advantages: 1. By starting from a formal fairness specification, the fairness properties an algorithm is supposed to possess are made explicit and auditable by computer scientists as well as experts from other fields. This is particularly beneficial if justified exceptions lead to conditional or restricted fairness as the exceptions are made explicit. 2. By providing means to formally prove that a program conforms to a given formal fairness specification, this approach can provide a guarantee which goes beyond the guarantees attainable through purely empirical or statistical analysis; 3. Through its distinction between disparate treatment and impact, information-flow analysis can help in uncovering and mitigating biases in an algorithm's code.

Nonetheless, employing this technique brings certain hazards. In particular, a software engineer may be tempted to use off-the-shelf information flow-analysis tools to put a "seal" on their software without fully understanding or publicizing the assumptions used in a proof of noninterference, i.e., what specification the program is actually proven to satisfy. Therefore, we recommend that an information-flow-based assessment of a software's fairness should be publicized in combination with all assumptions for and limitations to this fairness guarantee. This makes such an assessment auditable and ensures that third parties can check whether assumptions hold (e.g., that the conditions of conditional fairness are actually justified) or critique potentially misguided use of program-analysis-based fairness guarantees.

Acknowledgements

This work was supported by funding from the pilot program Core-Informatics of the Helmholtz Association (HGF). It was furthermore supported by KASTEL Security Research Labs. We thank Dr. Alexander Weigl and Dr. Michael Kirsten for helpful discussions and the anonymous reviewers for their feedback on earlier drafts of this paper.

References

Ahrendt, W.; Baar, T.; Beckert, B.; Bubel, R.; Giese, M.; Hähnle, R.; Menzel, W.; Mostowski, W.; Roth, A.; Schlager, S.; and Schmitt, P. H. 2005. The KeY tool. *Softw. Syst. Model.*, 4(1): 32–54.

Ahrendt, W.; Beckert, B.; Bubel, R.; Hähnle, R.; Schmitt, P. H.; and Ulbrich, M., eds. 2016. *Deductive Software Verification - The KeY Book - From Theory to Practice*, volume 10001 of *LNCS*. Cham: Springer. ISBN 978-3-319-49811-9.

Albarghouthi, A.; D'Antoni, L.; Drews, S.; and Nori, A. V. 2017. FairSquare: probabilistic verification of program fairness. *Proc. ACM Program. Lang.*, 1(OOPSLA): 80:1–80:30.

Bastani, O.; Zhang, X.; and Solar-Lezama, A. 2019. Probabilistic verification of fairness properties via concentration. *Proc. ACM Program. Lang.*, 3(OOPSLA): 118:1–118:27.

Beckert, B.; Bruns, D.; Klebanov, V.; Scheben, C.; Schmitt, P. H.; and Ulbrich, M. 2013. Information Flow in Object-Oriented Software. In Gupta, G.; and Peña, R., eds., *Logic-Based Program Synthesis and Transformation, 23rd International Symposium, LOPSTR 2013, Madrid, Spain, September 18-19, 2013, Revised Selected Papers*, volume 8901 of *LNCS*, 19–37. Cham: Springer.

Beckert, B.; Kirsten, M.; and Schefczyk, M. 2022. Algorithmic Fairness and Secure Information Flow (Extended Abstract). In Heitz, C.; Hertweck, C.; Viganò, E.; and Loi, M., eds., *European Workshop on Algorithmic Fairness (EWAF '22), Lightning round track*.

Beutel, A.; Chen, J.; Doshi, T.; Qian, H.; Woodruff, A.; Luu, C.; Kreitmann, P.; Bischof, J.; and Chi, E. H. 2019. Putting Fairness Principles into Practice: Challenges, Metrics, and Improvements. In Conitzer, V.; Hadfield, G. K.; and Vallor, S., eds., *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, AIES 2019, Honolulu, HI, USA, January 27-28, 2019*, 453–459. New York, NY, USA: Association for Computing Machinery.

Biondi, F.; Enescu, M. A.; Heuser, A.; Legay, A.; Meel, K. S.; and Quilbeuf, J. 2018. Scalable Approximation of Quantitative Information Flow in Programs. In Dillig, I.; and Palsberg, J., eds., *Verification, Model Checking, and Abstract Interpretation - 19th International Conference, VMCAI 2018, Los Angeles, CA, USA, January 7-9, 2018, Proceedings*, volume 10747 of *LNCS*, 71–93. Cham: Springer.

Borca-Tasciuc, G.; Guo, X.; Bak, S.; and Skiena, S. 2023. Provable Fairness for Neural Network Models using Formal Verification. In Alvarez, J. M.; Fabris, A.; Heitz, C.; Hertweck, C.; Loi, M.; and Zehlike, M., eds., *Proceedings of the 2nd European Workshop on Algorithmic Fairness, Winterthur, Switzerland, June 7th to 9th, 2023*, volume 3442 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Chakraborty, S.; Meel, K. S.; and Vardi, M. Y. 2021. Approximate Model Counting. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, 1015–1045. IOS Press.

Darvas, Á.; Hähnle, R.; and Sands, D. 2005. A Theorem Proving Approach to Analysis of Secure Information Flow. In Hutter, D.; and Ullmann, M., eds., *Security in Pervasive Computing, Second International Conference, SPC 2005, Boppard, Germany, April 6-8, 2005, Proceedings*, volume 3450 of *LNCS*, 193–209. Cham: Springer.

Denning, D. E. 1976. A Lattice Model of Secure Information Flow. *Commun. ACM*, 19(5): 236–243.

Denning, D. E. R. 1975. *Secure information flow in computer systems.* Ph.D. thesis, Purdue University.

Dwork, C.; Hardt, M.; Pitassi, T.; Reingold, O.; and Zemel, R. S. 2012. Fairness through awareness. In Goldwasser, S., ed., *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, 214–226. New York, NY, USA: Association for Computing Machinery.

Federal Ministry of Finance. 2022a. Lohn- und Einkommenssteuerrechner: Service fuer Entwickler.

Federal Ministry of Finance. 2022b. Programmablaufpläne für den Lohnsteuerabzug 2023.

Ghosh, B.; Basu, D.; and Meel, K. S. 2021. Justicia: A Stochastic SAT Approach to Formally Verify Fairness. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 7554–7563. AAAI Press.

Ghosh, B.; Basu, D.; and Meel, K. S. 2022. Algorithmic Fairness Verification with Graphical Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9): 9539–9548.

Graf, J.; Hecker, M.; and Mohr, M. 2013. Using JOANA for Information Flow Control in Java Programs - A Practical Guide. In *Proceedings of the 6th Working Conference on Programming Languages (ATPS'13)*, Lecture Notes in Informatics (LNI) 215, 123–138. Springer Berlin / Heidelberg.

Grgic-Hlaca, N.; Zafar, M. B.; Gummadi, K. P.; and Weller, A. 2016. The case for process fairness in learning: Feature selection for fair decision making. In *NIPS symposium on machine learning and the law*, 2, 11. Barcelona, Spain.

Hardt, M.; Price, E.; and Srebro, N. 2016. Equality of Opportunity in Supervised Learning. In Lee, D. D.; Sugiyama, M.; von Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 3315–3323.

Klebanov, V. 2014. Precise quantitative information flow analysis - a symbolic approach. *Theor. Comput. Sci.*, 538: 124–139.

Kusner, M. J.; Loftus, J. R.; Russell, C.; and Silva, R. 2017. Counterfactual Fairness. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 4066–4076.

Lampson, B. W. 1973. A Note on the Confinement Problem. *Commun. ACM*, 16(10): 613–615.

Loftus, J. R.; Russell, C.; Kusner, M. J.; and Silva, R. 2018. Causal Reasoning for Algorithmic Fairness. *CoRR*, abs/1805.05859.

Mitchell, S.; Potash, E.; Barocas, S.; D'Amour, A.; and Lum, K. 2021. Algorithmic Fairness: Choices, Assumptions, and Definitions. *Annual Review of Statistics and Its Application*, 8(1): 141–163.

Pearl, J. 2009. Causal inference in statistics: An overview. *Statistics Surveys*, 3: 96–146.

Ramadan, Q.; Ahmadian, A. S.; Jürjens, J.; Staab, S.; and Strüber, D. 2019. Explaining Algorithmic Decisions with respect to Fairness. In Becker, S.; Bogicevic, I.; Herzwurm, G.; and Wagner, S., eds., *Software Engineering and Software Management, SE/SWM 2019, Stuttgart, Germany, February 18-22, 2019*, volume P-292 of *LNI*, 161–162. GI.

Ramadan, Q.; Ahmadian, A. S.; Strüber, D.; Jürjens, J.; and Staab, S. 2018. Model-based discrimination analysis: a position paper. In Brun, Y.; Johnson, B.; and Meliou, A., eds., *Proceedings of the International Workshop on Software Fairness, FairWare@ICSE 2018, Gothenburg, Sweden, May 29, 2018*, 22–28. New York, NY, USA: Association for Computing Machinery.

Smith, G. 2009. On the Foundations of Quantitative Information Flow. In de Alfaro, L., ed., *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5504 of *LNCS*, 288–302. Cham: Springer.

Snelting, G.; Giffhorn, D.; Graf, J.; Hammer, C.; Hecker, M.; Mohr, M.; and Wasserrab, D. 2014. Checking Probabilistic Noninterference Using JOANA. *it - Information Technology*, 56: 280–287.

Taskesen, B.; Blanchet, J. H.; Kuhn, D.; and Nguyen, V. A. 2021. A Statistical Test for Probabilistic Fairness. In Elish, M. C.; Isaac, W.; and Zemel, R. S., eds., *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, 648–665. New York, NY, USA: Association for Computing Machinery.

Teuber, S.; and Beckert, B. 2023a. Formally Verified Algorithmic Fairness Using Information-Flow Tools. In Alvarez, J. M.; Fabris, A.; Heitz, C.; Hertweck, C.; Loi, M.; and Zehlike, M., eds., *Proceedings of the 2nd European Workshop on Algorithmic Fairness, Winterthur, Switzerland, June 7th to 9th, 2023*, volume 3442 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Teuber, S.; and Beckert, B. 2023b. An Information-Flow Perspective on Algorithmic Fairness. *CoRR*, abs/2312.10128.

Teuber, S.; and Beckert, B. 2023c. samysweb/AAAI24-Fairness: AAAI24 Artefact. 10.5281/zenodo.10385360.

Teuber, S.; and Weigl, A. 2021. Quantifying Software Reliability via Model-Counting. In Abate, A.; and Marin, A., eds., *Quantitative Evaluation of Systems - 18th International Conference, QEST 2021, Paris, France, August 23-27, 2021, Proceedings*, volume 12846 of *LNCS*, 59–79. Cham: Springer.

Yang, J.; and Meel, K. S. 2023. Rounding Meets Approximate Model Counting. In Enea, C.; and Lal, A., eds., *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II*, volume 13965 of *Lecture Notes in Computer Science*, 132–162. Springer.