ABSTRACT

Bias in decisions made by modern software is becoming a common and serious problem. We present Themis, an automated test suite generator to measure two types of discrimination, including causal relationships between sensitive inputs and program behavior. We explain how Themis can measure discrimination and aid its debugging, describe a set of optimizations Themis uses to reduce test suite size, and demonstrate Themis’ effectiveness on open-source software. Themis is open-source and all our evaluation data are available at http://fairness.cs.umass.edu/. See a video of Themis in action: https://youtu.be/brB8wkaUesY.

ACM Reference format:

https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Software plays an important role in making decisions that shape our society. Software decides what products we are led to buy [28], who gets financial loans [35], what a self-driving car does, which may lead to property damage or human injury [18], how medical patients are diagnosed and treated [39], and who gets bail and which criminal sentence [3]. Unfortunately, there are countless examples of bias in software. Translation engines inject societal biases, e.g., “She is a doctor” translated into Turkish and back into English becomes “He is a doctor” [13]. YouTube is more accurate when automatically generated closed captions for videos with male than female voices [41]. Facial recognition systems often underperform on female and black faces [24]. In 2016, Amazon software decided not to offer same-day delivery to predominantly minority neighborhoods [27]. And the software US courts use to assess the risk of a criminal repeating a crime exhibits racial bias [3].

Bias in software can come from learning from biased data, implementation bugs, design decisions, unexpected component interactions, or societal phenomena. Thus, software discrimination is a challenging problem and addressing it is integral to the entire software development cycle, from requirements elicitation, to architectural design, to testing, verification, and validation.

Even defining what it means for software to discriminate is not straightforward. Many definitions of algorithmic discrimination have emerged, including the correlation or mutual information between inputs and outputs [42], discrepancies in the fractions of inputs that produce a given output [11, 20, 46, 48] (known as group discrimination [17]), or discrepancies in output probability distributions [26]. These definitions do not capture causality and can miss some forms of discrimination.

To address this, our recent work developed a new measure called causal discrimination and described a technique for automated fairness test generation [17]. This tool demonstration paper implements that technique for the group and causal definitions of discrimination in a tool called Themis v2.0 (building on an early prototype [17]). This paper focuses on the tool’s architecture, test suite generation workflow, and efficiency optimizations (Section 2), and its user interface (Section 3). Section 4 places Themis in the context of related research and Section 5 summarizes our contributions.

2 THEMIS: AUTOMATED FAIRNESS TEST GENERATION

Figure 1 describes the Themis architecture and fairness test-suite generation workflow. Themis consists of four major components: input generator, cache, error-bound confidence calculator, and discrimination score calculator. Themis uses the input schema of the system under test to generate test suites for group or causal discrimination. Themis generates values for non-sensitive attributes randomly or according to a priori distributions [26]. These definitions do not capture causality and can miss some forms of discrimination.

Even defining what it means for software to discriminate is not straightforward. Many definitions of algorithmic discrimination have emerged, including the correlation or mutual information between inputs and outputs [42], discrepancies in the fractions of inputs that produce a given output [11, 20, 46, 48] (known as group discrimination [17]), or discrepancies in output probability distributions [26]. These definitions do not capture causality and can miss some forms of discrimination.

To address this, our recent work developed a new measure called causal discrimination and described a technique for automated fairness test generation [17]. This tool demonstration paper implements that technique for the group and causal definitions of discrimination in a tool called Themis v2.0 (building on an early prototype [17]). This paper focuses on the tool’s architecture, test suite generation workflow, and efficiency optimizations (Section 2), and its user interface (Section 3). Section 4 places Themis in the context of related research and Section 5 summarizes our contributions.
With more than two races, the number of possible executions grows exponentially with the number of input attributes being tested for discrimination. However, the group and causal discrimination definitions are monotonic: if software discriminates over threshold \( \theta \) with respect to a set of attributes \( X \), then the software also discriminates over \( \theta \) with respect to all supersets of \( X \) (see Theorems 4.1 and 4.2 and their proofs in [17]). This allows Themis to prune its test input space. Once Themis discovers that software discriminates against \( X \), it can prune testing all supersets of \( X \).

Further, causal discrimination always exceeds group discrimination with respect to the same set of attributes (see Theorem 4.3 and its proof in [17]) so Themis can prune its test input space when measuring both kinds of discrimination: If software group discriminates with respect to a set of attributes, it must causally discriminate with respect to that set at least as much.

These observations and their formal proofs allow Themis to employ a provably sound pruning strategy (Algorithm 3 in [17]).

**Adaptive sampling.** Themis approximates group and causal discrimination scores through sampling done via iterative test generation (recall Figure 1). Sampling in Themis is adaptive, using the ongoing score computation to determine if a specified bound of error with a desired confidence level has been reached. Themis generates inputs uniformly at random using an input schema, and maintains the proportion of samples evidencing discrimination, computing the bound of error for that proportion.

**Test caching.** Themis may generate repetitive tests: tests relevant to group discrimination are also relevant to causal discrimination, and tests relevant to one set of attributes can also be relevant to another set. This redundancy in fairness testing allows Themis to exploit caching to reuse test results without re-executing tests.

### 3 USING THEMIS TO DISCOVER AND DEBUG DISCRIMINATION

Themis is a standalone application written in Python. Themis is open-source: http://fairness.cs.umass.edu/. This paper describes the Themis user interface: Themis engine, input generator, and error-bound confidence calculator. Themis uses to reduce test suite size. Applying these optimizations to real-world software (see Section 3), reduced test suite sizes by, on average, 2,849 times for group discrimination and 148 times for causal discrimination [17]. The more software discriminates, the greater the reduction in test suite size.

**Sound pruning.** The number of possible executions grows exponentially with the number of input attributes being tested for discrimination. However, the group and causal discrimination definitions are monotonic: if software discriminates over threshold \( \theta \) with respect to a set of attributes \( X \), then the software also discriminates over \( \theta \) with respect to all supersets of \( X \) (see Theorems 4.1 and 4.2 and their proofs in [17]). This allows Themis to prune its test input space. Once Themis discovers that software discriminates against \( X \), it can prune testing all supersets of \( X \).

Further, causal discrimination always exceeds group discrimination with respect to the same set of attributes (see Theorem 4.3 and its proof in [17]) so Themis can prune its test input space when measuring both kinds of discrimination: If software group discriminates with respect to a set of attributes, it must causally discriminate with respect to that set at least as much.

These observations and their formal proofs allow Themis to employ a provably sound pruning strategy (Algorithm 3 in [17]).

**Adaptive sampling.** Themis approximates group and causal discrimination scores through sampling done via iterative test generation (recall Figure 1). Sampling in Themis is adaptive, using the ongoing score computation to determine if a specified bound of error with a desired confidence level has been reached. Themis generates inputs uniformly at random using an input schema, and maintains the proportion of samples evidencing discrimination, computing the bound of error for that proportion.

**Test caching.** Themis may generate repetitive tests: tests relevant to group discrimination are also relevant to causal discrimination, and tests relevant to one set of attributes can also be relevant to another set. This redundancy in fairness testing allows Themis to exploit caching to reuse test results without re-executing tests.

### 3 USING THEMIS TO DISCOVER AND DEBUG DISCRIMINATION

Themis is a standalone application written in Python. Themis is open-source: http://fairness.cs.umass.edu/. This paper describes the Themis user interface: Themis engine, input generator, and error-bound confidence calculator. Themis uses to reduce test suite size. Applying these optimizations to real-world software (see Section 3), reduced test suite sizes by, on average, 2,849 times for group discrimination and 148 times for causal discrimination [17]. The more software discriminates, the greater the reduction in test suite size.

**Sound pruning.** The number of possible executions grows exponentially with the number of input attributes being tested for discrimination. However, the group and causal discrimination definitions are monotonic: if software discriminates over threshold \( \theta \) with respect to a set of attributes \( X \), then the software also discriminates over \( \theta \) with respect to all supersets of \( X \) (see Theorems 4.1 and 4.2 and their proofs in [17]). This allows Themis to prune its test input space. Once Themis discovers that software discriminates against \( X \), it can prune testing all supersets of \( X \).

Further, causal discrimination always exceeds group discrimination with respect to the same set of attributes (see Theorem 4.3 and its proof in [17]) so Themis can prune its test input space when measuring both kinds of discrimination: If software group discriminates with respect to a set of attributes, it must causally discriminate with respect to that set at least as much.

These observations and their formal proofs allow Themis to employ a provably sound pruning strategy (Algorithm 3 in [17]).

**Adaptive sampling.** Themis approximates group and causal discrimination scores through sampling done via iterative test generation (recall Figure 1). Sampling in Themis is adaptive, using the ongoing score computation to determine if a specified bound of error with a desired confidence level has been reached. Themis generates inputs uniformly at random using an input schema, and maintains the proportion of samples evidencing discrimination, computing the bound of error for that proportion.

**Test caching.** Themis may generate repetitive tests: tests relevant to group discrimination are also relevant to causal discrimination, and tests relevant to one set of attributes can also be relevant to another set. This redundancy in fairness testing allows Themis to exploit caching to reuse test results without re-executing tests.

### 3 USING THEMIS TO DISCOVER AND DEBUG DISCRIMINATION

Themis is a standalone application written in Python. Themis is open-source: http://fairness.cs.umass.edu/. This paper describes the Themis user interface: Themis engine, input generator, and error-bound confidence calculator. Themis uses to reduce test suite size. Applying these optimizations to real-world software (see Section 3), reduced test suite sizes by, on average, 2,849 times for group discrimination and 148 times for causal discrimination [17]. The more software discriminates, the greater the reduction in test suite size.

**Sound pruning.** The number of possible executions grows exponentially with the number of input attributes being tested for discrimination. However, the group and causal discrimination definitions are monotonic: if software discriminates over threshold \( \theta \) with respect to a set of attributes \( X \), then the software also discriminates over \( \theta \) with respect to all supersets of \( X \) (see Theorems 4.1 and 4.2 and their proofs in [17]). This allows Themis to prune its test input space. Once Themis discovers that software discriminates against \( X \), it can prune testing all supersets of \( X \).

Further, causal discrimination always exceeds group discrimination with respect to the same set of attributes (see Theorem 4.3 and its proof in [17]) so Themis can prune its test input space when measuring both kinds of discrimination: If software group discriminates with respect to a set of attributes, it must causally discriminate with respect to that set at least as much.

These observations and their formal proofs allow Themis to employ a provably sound pruning strategy (Algorithm 3 in [17]).

**Adaptive sampling.** Themis approximates group and causal discrimination scores through sampling done via iterative test generation (recall Figure 1). Sampling in Themis is adaptive, using the ongoing score computation to determine if a specified bound of error with a desired confidence level has been reached. Themis generates inputs uniformly at random using an input schema, and maintains the proportion of samples evidencing discrimination, computing the bound of error for that proportion.

**Test caching.** Themis may generate repetitive tests: tests relevant to group discrimination are also relevant to causal discrimination, and tests relevant to one set of attributes can also be relevant to another set. This redundancy in fairness testing allows Themis to exploit caching to reuse test results without re-executing tests.
Themis: Automatically Testing Software for Discrimination

Themeis view of group discrimination results.

Themeis view of causal discrimination results.

Currently, Themeis handles categorical inputs, such
test suite sizes by, on average, 2,849 times for group discrimination and 148 times for causal discrimination.

4 RELATED WORK

Discrimination shows up in many software applications, e.g., advertisements [40], hotel bookings [28], and image search [23]. Meanwhile, software is entering domains in which discrimination could result in serious negative consequences, including criminal justice [3], finance [35], and hiring [38]. Software discrimination may occur unintentionally, e.g., as a result of implementation bugs, as an unintended property of self-organizing systems [4, 6, 8, 9], as an emergent property of component interaction [5, 7, 10, 25], or as an automatically learned property from biased data [11, 12, 19–22, 46–48].
Themis focuses on two measures of discrimination, group and causal. Group discrimination is a generalization of the Calders-Verwer (CV) score \[12\], used frequently in prior work on algorithmic fairness, particularly in the context of fair machine learning \[11, 20, 46, 48\]. Many other definitions exist. One defines discrimination by observing that a “better” input is never deprived of the “better” output \[14\]. That definition requires a domain expert to create a distance function for comparing inputs. Causal discrimination \[17\] (which Themis measures) goes beyond prior work by measuring causality \[37\]. Fairness in machine learning research is similarly moving toward causal measures of discrimination, e.g., counterfactual fairness \[26\]. FairML \[1\] uses orthogonal projection to co-perturb attributes, which can mask some discrimination, but find discrimination that is more likely to be observed in real-world scenarios.

FairTest \[42\] uses manually written tests to measure four kinds of discrimination scores: the CV score and a related ratio, mutual information, Pearson correlation, and a regression between the output and sensitive inputs. By contrast, Themis generates tests automatically and also measures causal discrimination.

Reducing discrimination in machine learning classifiers \[11, 20, 46, 48\] is important work that is complementary to ours. We focus on measuring discrimination via software testing, not developing methods for removing it. Themis can be used to manually debug discrimination bugs and thus remove discrimination. Work on formal verification of non-discrimination \[2\] is similarly complementary to our testing approach.

Causal relationships in data management systems \[16, 29, 30\] can help explain query results \[32\] and debug errors \[43–45\] by tracking and using data provenance \[31\]. For software systems can help explain query results \[32\] and debug errors \[43–45\] by tracking and using data provenance \[31\]. For software systems\[24\], self-adapting reliability as an emergent property of distributed systems' software architectures. In International Workshop on Engineering Fault Tolerant Systems (EFTS), pages 38–43, Dubrovnik, Croatia, September 2007.


