Stream-Based Monitoring of Algorithmic Fairness

Jan Baumeister(✉), Bernd Finkbeiner, Frederik Scheerer, Julian Siber, and Tobias Wagenpfeil

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
{jan.baumeister, finkbeiner, frederik.scheerer, julian.siber, tobias.wagenpfeil}@cispa.de

Abstract. Automatic decision and prediction systems are increasingly deployed in applications where they significantly impact the livelihood of people, such as for predicting the creditworthiness of loan applicants or the recidivism risk of defendants. These applications have given rise to a new class of *algorithmic-fairness* specifications that require the systems to decide and predict without bias against social groups. Verifying these specifications statically is often out of reach for realistic systems, since the systems may, e.g., employ complex learning components, and reason over a large input space. In this paper, we therefore propose stream-based monitoring as a solution for verifying the algorithmic fairness of decision and prediction systems at runtime. Concretely, we present a principled way to formalize algorithmic fairness over temporal data streams in the specification language RTLola and demonstrate the efficacy of this approach on a number of benchmarks. Besides synthetic scenarios that particularly highlight its efficiency on streams with a scaling amount of data, we notably evaluate the monitor on real-world data from the recidivism prediction tool COMPAS.

1 Introduction

Machine learning is used to automate an increasing number of critical decisions pertaining to people's opportunities in areas such as loan or job application [19], healthcare [41], and criminal sentencing [5]. It is of vital interest that these decision and prediction systems adhere to societies' shared values and, hence, they should in particular not discriminate against members of protected social groups, e.g., based on attributes such as gender or perceived ethnicity. Since the machine-learned systems are trained from historical data, they often inherit the historical human bias present in these data sets. So far, this usually was revealed by posterior analyses after the systems have been deployed for years [37,38] and hence have already produced harmful results. In this paper, we propose stream-based monitoring of algorithmic fairness properties as a way to alleviate this situation, and to significantly reduce the impact unfair decisions have on people that are subject to learned decision and prediction systems. Unlike static verification of these systems, which is often intractable due to their complex learning components and large input space, monitors are lightweight and can be deployed alongside the machine-learned systems, raising awareness once they are

sufficiently sure of unfair behavior. While this does not avert all decisions made by an unfair system, we show empirically that it can still significantly reduce the number of decisions made by an unfair system by alerting practicioners early.

1.1 Motivating Example

As a motivating example, we consider the COMPAS tool developed by Northpointe [37]. COMPAS predicts the recidivism risk of defendants in criminal trials in order to assist judges in, e.g., setting bond amounts or in sentencing during trial. Hence, the system gives a prediction on how likely a person is to commit a(nother) crime, and this prediction has a direct impact on criminal sentencing. A retrospective investigation by ProPublica into the predictions by COMPAS during 2013 and 2014 in Broward County, Florida, revealed that the tool is significantly biased against defendants perceived as black [5]. For instance, the false positive rate for black defendants was found to be significantly higher than for white defendants, i.e., black defendants were more likely classified with a high risk of re-offending, without actually committing a crime in the near future. The ultimate vision of our work is that, instead of such a posterior analysis of algorithmic fairness, runtime monitors are deployed that assess the fairness of decision and prediction systems during their execution, in order to raise awareness of unfair treatment early and in this way mitigate unproportional harm put on groups due to an unfair bias. To illustrate our monitoring approach, we will consider a simplified version of the risk assessment setting as shown in Figure 1. This table shows a number of events describing an execution of the COMPAS system that is defined on data streams such as `event` or `id`. For example, the first row describes that on the 2nd of January 2013, an individual of group `A` was screened via COMPAS and assessed to have a high risk of recidivism. A (simplified) algorithmic-fairness specification compares certain conditional probabilities associated with the different groups:

date	event	id	group	risk	...
2013-01-02	SCREEN	0	A	HIGH	...
2013-01-02	SCREEN	1	B	LOW	...
2013-01-03	RECID.	1	-	-	...
2013-01-03	SCREEN	2	B	HIGH	...
...	...	...	...	...	...

Fig. 1: Data streams for an example of recidivism risk assessment with COMPAS [37].

$$\left| \mathbb{P}(\texttt{HIGH} \mid \texttt{A}, \texttt{RECIDIVISM}) - \mathbb{P}(\texttt{HIGH} \mid \texttt{B}, \texttt{RECIDIVISM}) \right| \leq \epsilon \ .$$

This condition states that the probability of a re-offending member of group `A` to be labeled as high-risk is not too far (less than ϵ) from the probability of a re-offending member of group `B` to be labeled as high-risk. Hence, it compares the *true positive rates* between the two groups. In this paper, we show how we can use the stream-based monitoring language RTLola to process such data streams and in this way analyze the algorithmic fairness of their underlying system in real-time. The main idea is to automatically partition the stream events into independent trials and to construct RTLola specifications that estimate the conditional probabilities associated with algorithmic-fairness specifications.

1.2 Outline and Contributions

The challenges in our stream-based setting are twofold: First, we observe only a single execution of the system but require a larger number of independent trials to reliably estimate the conditional probabilities. Second, the independent trials and also the fairness definitions contain a real-time component. We address the first challenge in Section 3 by defining a principled way to extract individual trials based on a predefined dependence relation between stream events. In Section 4, we then describe how this can be implemented in the specification language RTLola. We show how we can estimate conditional probabilities over these trials with RTLola, and address the second challenge: RTLola naturally supports reasoning about real-time events, and hence we can use it to collect stream events that are spread throughout time and calculate their relative delay, which allows us to express certain intricacies of algorithmic-fairness specifications, such as an upper time bound between relevant events. We evaluate this RTLola compilation in a case study including both synthetic and real-world benchmarks. For the former, we present a benchmark generator that models application scenarios at a company and a seminar assignment at a university. In both cases, we can easily scale, e.g., the number of applicants, which serves as a stress test for our implementation and allows a thorough comparison with more traditional approaches based on databases. We show that RTLola significantly outperforms database approaches, which suggests that stream-based monitoring is the tool of choice for settings with high data throughput. Moreover, synthetic benchmarks allow us to set a ground truth for the fairness of the decision system, and we show that our monitoring approach can detect unfair systems without raising too many false alarms on fair systems. As a real-world benchmark, we consider the aforementioned recidivism prediction tool COMPAS [5]. Unlike the synthetic benchmarks, this is also an example of a prediction system, such that more complex specifications become relevant. We show that RTLola is able to express these specifications succinctly and effectively alert to unfairness in the prediction system early. All experiments can be found in Section 5.

Contributions. To summarize, we make the following contributions:

- We formalize the estimation of probabilities from single executions in stream-based monitoring.
- We implement RTLola monitors that allow the monitoring of a wide range of algorithmic-fairness specifications from the literature.
- We present a generator for constructing challenging benchmarks related to algorithmic fairness in job application and university admission.
- We perform an extensive experimental evaluation on these synthetic benchmarks, as well as on a real-world data set from the COMPAS tool.

1.3 Related Work

Efforts of the machine learning community generally aim more at improving the fairness of learned models than rigorously verifying it [35]. Three categories of

mechanisms stand out, namely Pre-Processing [30,22], In-Processing [1,31], and Post-Processing [38,17]. Our work on monitoring algorithmic fairness is an orthogonal effort that allows us to audit learned systems even when their training process cannot be influenced, as we treat the learned system as a black box. We present a general approach based on RTLola and encode popular fairness properties, such as *equalized odds*. These techniques can also be used to encode other fairness properties such as *equal opportunity* [26] or *counterfactual fairness* [32].

Related to our effort of verifying and testing fairness, a variety of different approaches in the formal methods community exist: Udeshi et al. [43] propose an automated and directed testing technique to generate discriminatory inputs for machine learning models. FairTest [42] is a framework for specifying and testing algorithmic fairness. A similar approach is given by Bastiani et al. [7] by using adaptive concentration inequalities to design a scalable sampling technique for providing fairness guarantees. Albarghouthi et al. [3] transform fairness properties as probabilistic program properties and develop an SMT-based technique to verify fairness of decision-making programs. Albarghouthi and Vinitsky [4] propose a white-box monitoring technique based on adding annotations in a program, but they cannot reason about temporal properties, unlike our approach. To certify individual fairness, Rouss et al. [39] introduce a local property that coincides with robustness within a particular distance metric. Another approach is to repair biased decision systems with a program repair technique [2]. Teuber and Beckert [40] have made an intriguing connection between secure information-flow and algorithmic fairness, and use information-flow tools for verifying fairness of white-box programs. Henzinger et al. propose monitoring of *probabilistic specification expressions (PSEs)* [27] and extensions [29] for monitoring algorithmic fairness properties [28]. Baum et al. [8] combine monitoring and input generation for a probabilistic falsification technique aimed at individual fairness. Cano et al. [14] propose fairness shields that combine monitoring and enforcement of fairness properties. In our work, we show that it is possible to use the widely studied formalism of stream-based monitoring languages [9] to go even further by additionally considering temporal aspects of fairness such as delays between relevant events. Notably, this is possible without any pre-processing of the stream-events that may be needed for, e.g., PSEs, as this is already handled by stream-based monitoring languages. These languages predate the PSE approach by decades [18] and have already proven useful in diverse areas such as unmanned aircraft [10,11] and network monitoring [21]. We use RTLola in this paper, but the general ideas may also be adapted to other stream-based languages, such as TeSSLa [16] or Striver [25].

The usage of opaque machine-learning models in high-stake scenarios has sparked scholarly debate on its ethics [13,34], as well as extensive governmental regulation [6,15]. Given that these models promise to be more accurate [33] and ultimately even more impartial than human decision makers, there seems to be a clear trend toward further adoption. As we show here, RTLola can be a useful tool for alleviating unintended negative side effects of this trend by promoting effective monitoring of the decision system during deployment.

2 Preliminaries

We briefly recall the necessary background on probability theory, algorithmic fairness and stream-based monitoring with RTLola.

2.1 Probability Theory

A *probability space* is a tuple $(\Omega, \mathcal{E}, \mathbb{P})$, where Ω is a *sample space* and $\mathcal{E}$ is a σ-*algebra* over Ω, i.e., we have $\emptyset \in \mathcal{E}$, $A \in \mathcal{E} \implies \bar{A} \in \mathcal{E}$, and $A_0, A_1, \ldots \in \mathcal{E} \implies \bigcup_{i=0}^{\infty} A_i \in \mathcal{E}$. Finally, $\mathbb{P}$ is a *probability measure* $\mathcal{E} \to \mathbb{R}$, i.e., a non-negative function with $\mathbb{P}(\Omega) = 1$, $\mathbb{P}(\emptyset) = 0$ that satisfies countable additivity: For any sequence of pairwise disjoint events $A_0, A_1, \ldots \in \mathcal{E}$ we have that $\mathbb{P}(\bigcup_{i=0}^{\infty} A_i) = \Sigma_{i=0}^{\infty} \mathbb{P}(A_i)$. A *random variable* is a function $X : (\Omega, \mathcal{E}) \to (\Gamma, \mathcal{V})$ that maps elements of the sample space Ω to some set Γ equipped with the σ-algebra $\mathcal{V}$, such that $X^{-1}(B) \in \mathcal{E}$ for all $B \in \mathcal{V}$. Such an X induces a probability measure on $(\Gamma, \mathcal{V})$ as $\mathbb{P}(B) = \mathbb{P}(X^{-1}(B))$ for all $B \in \mathcal{V}$. Lastly, the *conditional probability* of some $A \in \mathcal{E}$ given some $B \in \mathcal{E}$ is defined as $\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$.

2.2 Algorithmic Fairness

Algorithmic fairness is an umbrella term for several specifications that have recently been put forward for decision and classification systems [38]. The general idea is to compare the probabilities of certain good and bad events between social groups, e.g., we may require the probability of a loan request being accepted conditioned on the applicant being in group A to be not too far from the same probability conditioned on the applicant being in group B. In the following, we introduce the fairness specifications considered in this work, as they have been proposed in the literature. Note that these pure definitions do not consider timing issues such as the good outcome being obtained within a certain bound.

The simplest fairness specification is *demographic parity*, which requires the probabilities of good outcomes conditioned on the different groups to differ no more than the predefined parameter ϵ.

Definition 1 (Demographic Parity [20]). *A decision system for the binary decision A satisfies demographic parity, iff*

$$\big|\, \mathbb{P}(A = 1 \mid G = 1) - \mathbb{P}(A = 1 \mid G = 0) \,\big| \leq \epsilon \ .$$

The value of G represents the group a person belongs to (e.g., male or female), while A indicates the positive outcome. Demographic parity ensures that positive outcomes are assigned to the two groups at a similar rate, but it does not consider background factors that may be relevant to assess the fairness of a system. For instance, men and women may apply unproportionally to different departments of a university, such that the admission process of the university appears to be unfair while the same processes of the individual departments are fair.[1] If the

[1] This observation has been termed *Simspon's Paradox* by Colin Blyth [12].

existence of such confounding variables is known, it may be more appropriate to use a fairness measure such as *conditional statistical parity*.

Definition 2 (Conditional Statistical Parity [17]). *A decision system for the binary decision A satisfies conditional statistical parity, iff*

$$\left|\mathbb{P}(A = 1 \mid L = 1, G = 1) - \mathbb{P}(A = 1 \mid L = 1, G = 0)\right| \leq \epsilon \ .$$

Conditional statistical parity states, similar to demographic parity, that people from different groups should have an equal probability of positive outcomes. Additionally, it further conditions the probability on other legitimate factors L, e.g., confounding variables such as the department that students apply to. These factors have to be determined a priori and are based on the background knowledge of the specifier.

While the above parity measures define a notion of fairness for decision systems with binary outcomes, many fairness issues arise also for prediction systems that, e.g. classify the recidivism risk of defendants in criminal trials [5]. Fairness of such systems is more accurately described by comparing the true and false positive rates between groups, as done by the *equalized-odds* fairness measure.

Definition 3 (Equalized Odds [26]). *A prediction system $\hat{Y}$ for the outcome Y satisfies equalized odds, iff*

$$\left|\mathbb{P}(\hat{Y} = 1 \mid G = 1, Y = 0) - \mathbb{P}(\hat{Y} = 1 \mid G = 0, Y = 0)\right| \leq \epsilon \ \textit{and}$$
$$\left|\mathbb{P}(\hat{Y} = 1 \mid G = 1, Y = 1) - \mathbb{P}(\hat{Y} = 1 \mid G = 0, Y = 1)\right| \leq \epsilon \ .$$

Here, $\hat{Y}$ describes the predicted value, while Y is the true value of an outcome to be predicted. Hence, the equalized-odds measure requires the differences between the false positive rates (FPR) and the differences between the true positive rates (TPR) of all pairs of groups to be within a predefined bound ϵ.

2.3 Stream-based Monitoring with RTLola

In this work, we use the stream-based specification language RTLola to monitor the previously described fairness definitions. RTLola uses stream-equations to translate streams of input data to output streams and trigger conditions that describe violations of the specification. We illustrate the RTLola language with a small example and refer for more details to [9,10,23].

Example 1 (RTLola Example).

```
1 input user_id : UInt64, value : Int64
2 output amount(user)
3     spawn with user_id
4     eval when user_id = user with value + amount(user).last(or: 0)
5 trigger @value amount.aggregate(over_instances: all, using:
      max).defaults(to: 0) > 500 "Upper Limit Violation"
6 trigger @1Hz value.aggregate(over: 1s, using: count) > 5 "Too many
      transactions"
```

The specification declares two input streams describing a transaction to a user: The input stream `user_id` encodes a unique identifier for each user and `value` represents the amount. Next, the output stream `amount` sums up the values per user using parameterization. With parameterization, the output stream describes a set of instances and the specification can refer to each instance with the parameter, in this example the parameter `user`. The spawn declaration describes when a new instance is added to this set, in our case for every new `user_id`. The eval declaration describes for each instance when a new value is computed with the `when`-clause and the computation of this value with the `with`-clause. Here, each instance of the `amount` stream is computed when the `user_id` is equal to the instance parameter and the new value is computed as the sum of the previous value of this instance and the current value of the `value`-stream. The first trigger then aggregates over all instances of the `amount` stream, takes the maximum value, and compares this value against a threshold. Since in theory this access could fail, we need to provide a default value. If this condition is true, the generated monitor for this specification emits the corresponding trigger message. The second trigger checks the number of transactions over the last five seconds, illustrating the real-time capabilities of RTLola.

The semantics of RTLola is defined over a collection of timed data streams and intuitively checks whether the values in the collection correspond to the computed values for the stream equation. Additionally, it validates that the time is monotone.

Definition 4 (Data Streams). *A collection of timed data streams $\omega \in \mathbb{W}$ over a set of input streams $\mathit{ID}^{\uparrow}$ and output streams $\mathit{ID}^{\downarrow}$ is the combination of a StreamMap and a TimeMap.*

$$\begin{aligned} \mathit{Stream} &:= \mathit{InstanceID} \to \mathit{Time} \to \mathbb{V}_{\bot} \\ \mathit{StreamMap} &:= \mathit{ID}^{\uparrow} \uplus \mathit{ID}^{\downarrow} \to \mathit{Stream} \\ \mathit{TimeMap} &:= \mathit{Time} \to \mathbb{R} \\ \mathbb{W} &:= \mathit{StreamMap} \times \mathit{TimeMap} \end{aligned}$$

Figure 2 gives an intuition on the data stream representation based on the specification in Example 1. The *TimeMap* is a total function from the discrete timestamps, indicated at the top of the figure, to a real-time value. In the examples, the first three events arrive at the timestamps 0.6, 0.8, and 2.4. Given $\omega = (\mathit{streams}, \mathit{times}) \in \mathbb{W}$, we use $\omega(t) := \mathit{times}(t)$ to get the real-time value of a discrete timestamp $t \in \mathit{Time}$. The *StreamMap* assigns each stream identifier and instance to an infinite sequence of optional values, where $\bot$ indicates that the stream instance does not produce a value. In our example, $\omega(\mathit{user\_id})(\top)$ represents the infinite sequence of the input stream, and $\omega(\mathit{user\_id})(\top)(1)$ returns the value of the input stream at time 1. Note that we use $\top$ as the instance identifier if the stream is not parameterized, i.e., only one stream instance exists in the *StreamMap*. In contrast, the output stream `amount` is parameterized, such that different instances (e.g., $\omega(\mathit{amount})(1)$) exist. Formally, infinite sequences are represented by total functions, and we define

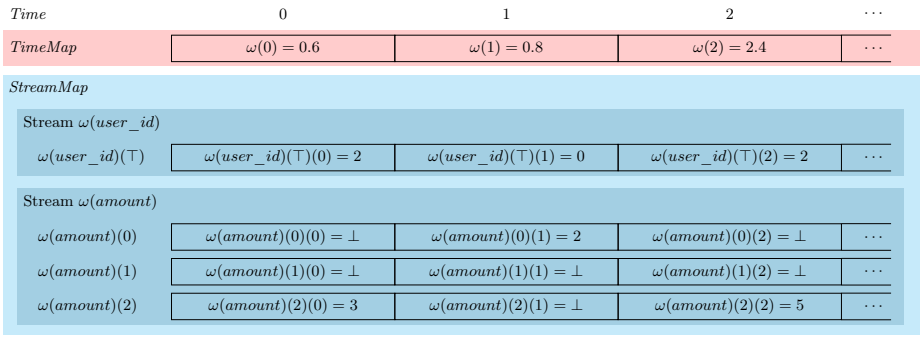


Fig. 2: The data streams exemplified on the specification from Example 1.

the access functions $\omega(sid) := \mathit{streams}(sid)$ for the stream $sid \in \mathtt{ID}^{\uparrow} \uplus \mathtt{ID}^{\downarrow}$, $\omega(sid)(i) := \mathit{streams}(sid)(i)$ to access stream instances $i \in \mathit{InstanceID}$ of stream sid, and $\omega(sid)(i)(t)$ for the stream instance value at discrete timestamp t.

The set of *stream events* of ω is defined as $\mathit{Events}(\omega) := \{(r, f) \mid \forall sid \in \mathtt{ID}^{\uparrow} \uplus \mathtt{ID}^{\downarrow}, i \in \mathit{InstanceID}.\, \omega(sid)(i)(\omega^{-1}(r)) = f(sid)(i)\} \subseteq \mathbb{E}_{\omega} := \mathbb{R} \times (\mathtt{ID}^{\uparrow} \uplus \mathtt{ID}^{\downarrow} \to \mathit{InstanceID} \to \mathbb{V}_{\perp})$. Hence, $\mathbb{E}_{\omega}$ denotes the set of all conceivable stream events over the datatypes defined by ω, while $\mathit{Events}(\omega)$ denotes the concrete events appearing in ω. For our example above, $\mathit{Events}(\omega)$ would map each real-time timestamp to the corresponding column in the figure.

3 Statistical Estimates from Data Streams

In this section, we outline the formal background for our RTLola specifications that estimate algorithmic fairness properties. We first describe how to extract multiple samples from a single execution of our system, we then describe how to use random variables to describe fairness properties in this setting, and lastly how we estimate the probability of events over these random variables.

3.1 Extracting Independent Trials from Data Streams

The central challenge in our setting is that we observe only a single execution of the system under scrutiny but want to perform a statistical estimation that naturally gets more accurate the more samples become available. We utilize the fact that in our applications, the single system execution describes a number of independent trials pertaining to the specification we care about, e.g., a single execution of the COMPAS tool for assessing the recidivism risk of defendants describes a large number of independent risk screenings. Hence, we propose a principled way to extract multiple samples from the observed system execution. At its core lies the definition of the probability space $(\Omega_{\omega}, \mathcal{E}_{\omega}, \mathbb{P}_{\omega})$ associated with the data streams $\omega \in \mathbb{W}$. The sample space Ω_{ω} is constructed as the set of all possible sequences of dependent events, which we identify through a

dependence relation $\delta \subseteq \mathbb{E}_\omega^2$. This predefined δ is an equivalence relation over the stream events $\mathbb{E}_\omega$ whose equivalence classes define the possible sets of events that form mutually independent trials. Elements of the sample space Ω_ω are ordered subsets of such dependent events: $\Omega_\omega := \{E_0 \dots E_n \in E^n \mid \forall\, 0 \leq i \leq j \leq n.\, t(E_i) \leq t(E_j) \wedge \delta(E_i, E_j)\}$ and we take $\mathcal{E}_\omega$ simply as the powerset of Ω_ω, while $\mathbb{P}_\omega$ is unknown to us.

Example 2. Consider the COMPAS recidivism risk assessment tool described in Section 1.1 and the corresponding data streams illustrated in Figure 1. We assume that the outcomes of individual screenings do not affect each other, and hence define the dependence relation such that two events are dependent if they refer to the same defendant (identified through the stream `id`), i.e., $\delta := \{(E_0, E_1) \mid E_0(\texttt{id}) = E_1(\texttt{id})\}$. Consequently, the data streams ω illustrated in Figure 1 describe the following samples $s_{0,1,2} \in \Omega_\omega$.

$$
\begin{aligned}
s_0 &= (0.0, \texttt{SCREEN}, \texttt{0}, \texttt{A}, \texttt{HIGH}) \dots \\
s_1 &= (0.0, \texttt{SCREEN}, \texttt{1}, \texttt{B}, \texttt{LOW})(1.0, \texttt{RECIDIVISM}, \texttt{1}, \texttt{-}, \texttt{-}) \dots \\
s_2 &= (1.0, \texttt{SCREEN}, \texttt{2}, \texttt{B}, \texttt{HIGH}) \dots
\end{aligned}
$$

Hence, our dependence relation δ partitions the data streams of the system into independent sequences of stream events, that naturally grow the more events are produced by the system. Note that the first components in the stream events with the values 0.0 and 1.0 encode the dates, i.e., `2013-01-02` and `2013-01-03`, via the *StreamMap* as outlined in Definition 4.

3.2 Defining Indicator Variables

Having defined our probability space through a dependence relation δ, the next step is to define Bernoulli random variables $X : \Omega_\omega \to \{0, 1\}$ that serve as indicator variables for the events relevant to algorithmic fairness.

Example 3. For instance, we may want to specify equalized odds (Definition 3) for the COMPAS risk assessment tool from Section 1.1. We may naturally define the prediction $\hat{Y}$ for a defendant associated with the sample ω as $\hat{Y}(\omega) := \exists i.\, \omega(\texttt{event})(i) = \texttt{SCREEN} \wedge \omega(\texttt{risk})(i) = \texttt{HIGH}$, and similarly, the true outcome is defined as $Y(\omega) := \exists i.\, \omega(\texttt{event})(i) = \texttt{RECIDIVISM}$. Here, we quantify over the time stamps i. Membership to, e.g., group `A` is captured by $G_{\texttt{A}}(\omega) := \exists i.\, \omega(\texttt{event})(i) = \texttt{SCREEN} \wedge \omega(\texttt{group})(i) = \texttt{A}$. It is also possible to define a sanity check as an additional variable that we condition on. For example, we may only consider recidivism events that happen less than two years after a screening event, as this is the specific time horizon that the COMPAS tool is targeting [5,37]. We can achieve this by utilizing the real-time information of the stream events with the variable $Y_{<2y} := \exists i, j.\, \omega(\texttt{event})(i) = \texttt{SCREEN} \wedge \omega(\texttt{event})(j) = \texttt{RECIDIVISM} \wedge \omega(j) - \omega(i) < 730.0$. Hence the FPR part of a specification of equalized odds with $\epsilon = 0.1$ is:

$$
\varphi := \left| \mathbb{P}(\hat{Y} = 1 \mid G_{\texttt{A}} = 1, Y_{<2y} = 1) - \mathbb{P}(\hat{Y} = 1 \mid G_{\texttt{A}} = 0, Y_{<2y} = 1) \right| \leq 0.1 \ .
$$

3.3 Maximum A Posteriori Estimation

Since during monitoring we obtain samples sequentially, the first samples have an unproportionally large impact on the assessment of fairness at the start of monitoring, since the estimation of the conditional probabilities in a formula like φ only gets more robust over time. Hence, we use methods from Bayesian statistics to control the trigger behavior of the monitor at the start of an execution: *maximum a posteriori (MAP)* estimation [36] allows us to take a prior belief about the conditional probabilities that make up the fairness specifications into consideration, as well as a degree of confidence therein. Formally, for every conditional probability $\Theta = \mathbb{P}(A \mid B)$ in our specification we require a prior γ and a confidence κ. Then, the estimate $\hat{\Theta}$ is given as:

$$\hat{\Theta} = \frac{S_{A \cap B} + \gamma(\omega)\kappa}{S_B + \kappa} \ ,$$

where $S_{A \cap B}$ is the number of samples that satisfy A and B, while S_B is the number of samples satisfying B. The parameters γ, κ and ϵ suffice to achieve sufficient initial robustness of the monitor, which we demonstrate experimentally in Section 5. The longer the observed system execution gets and the more samples become available, the less influence these parameters have on the monitor verdict.

Dynamic Updating of the Prior Belief. While MAP is a standard method from statistics, we face unique challenges when dynamically analyzing data streams, since we only have limited knowledge about the monitored system. Certain background knowledge like how many free places and applicants emerge during the execution may change the prior belief we have about the conditional probabilities. For instance, we may know that a university always fills all seminars with students, but the chance of an individual student's application to be accepted of course still depends on the number of seminar places and the number of other students applying. To account for such dynamic updates to the prior belief, we consider the prior γ to be a function of the data streams ω, such that it may be defined, e.g., as the ratio of places and applying students.

4 Implementation in RTLola

This section describes the implementation of the fairness definitions from Section 2.2 in the stream-based specification language RTLola. In general, each fairness specification follows the same structure: First, we extract information on independent trials from the input data and store it in parameterized streams that directly correspond to the indicator variables that are relevant in a given fairness specification. These variables can use the full power of RTLola expressions such as stream aggregations and real-time properties. We then build accumulators that are used in estimating the conditional probabilities. Last, we define trigger conditions that indicate that the estimates violate the fairness specification.

4.1 Implementation of Equalized Odds for the COMPAS Tool

We illustrate this principle by discussing the implementation of equalized odds (cf. Example 3). The RTLola specification for this fairness property, in the context of the COMPAS system, is shown in Figure 3. The specification is defined over input data streams that encode the relevant events of the COMPAS system as described in Section 1.1: The `"SCREEN"` event includes the unique identifier of a defendant in the input stream `id`, their group attribute in the input stream `group` and the COMPAS score describing the predicted likelihood of that person re-offending in the input stream `score`. The COMPAS score is an integer value between 0 and 10 as in the original data set. We use the same classification of any score above 6 as high risk as used by ProPublica [5] in the original investigation. If the defendant re-offends, the second event `"RECIDIVISM"` is given to the monitor together with the identifier of the defendant. The timestamps of these events are implicitly included through RTLola.

Storing Independent Trials in Parameterized Streams. The specification uses three parameterized output streams to store the relevant information of independent trials, where the parameter `i` identifies the trial, e.g., an individual defendant. The streams are `days_per`, `has_re` and `tp_event`. Each of these streams has a lifecycle of exactly 730 days after screening the defendant. In RTLola, this lifecycle is represented with the `spawn`, starting the lifecycle with the first occurrence for each identifier, and the `close` declaration, ending the lifecycle when the associated condition is satisfied. The output stream `days_per` counts the number of days after the screening of the defendant and the output stream `has_re` maps a `"RECIDIVISM"` event to the defendant. Then, the output stream `tp_event` synchronizes all information about one defendant after 730 days. This realizes the extraction of independent trials as described formally in Section 3.1. For example, the indicator variable $Y_{<2y}$ is described with the second clause of the eval-when declaration of the stream using stream aggregation (line 18), i.e., `has_re(i).aggregate(over: 730d, using: ∃)`. This expression checks if the defendant re-offended during a timeframe of 730 days using stream aggregation and follows the definition from Example 3. The stream `tp_event` is additionally parametrized with the group and the score of the defendant from which we can derive the indicator variables $G_{\mathtt{A}}$ and C directly. After the indicator variables are computed and used by the accumulators as described in the following paragraph, we close these stream instances to free the underlying memory since their value is not required after the first use of the variable.

Accumulator Variables and MAP Estimation. The specification then stores the accumulated information for each group using stream parameterization, where this time the parameter `g` identifies the group associated with the stream. It uses the stream `abs_re` to count the number of defendants that re-offendend in a given group (line 22). Similarly, the stream `abs_hr_re` counts the number of re-offenders per group that were scored as high-risk by COMPAS (line 26). The parameterized stream `tp_ratio` then computes for each group the true-positive ratio $\mathbb{P}(\hat{Y} = 1 \mid G = g, Y = 1)$, i.e., the probability that a person was

```
input event : String
input id : Int64
input group : String
input score : Int64

/// Defendant Information
output days_per(i)
  spawn with id
  eval @Global(1d) with days_per(i).last(or: 0) + 1
  close when days_per(i) = 730
output has_re(i)
    spawn with id
    eval when id == i with event == "RECIDIVISM"
    close @Global(1d) when days_per(i).hold(or: 0) = 730
output tp_event(i, g, s)
  spawn with (id, group, score)
  eval @Global(1d)
    when days_per(i).hold(or: 0) = 730 ∧ has_re(i).aggregate(over:
        730d, using: ∃) with s > 6
  close @Global(1d) when days_per(i).hold(or: 0) = 730

/// TP Ratio
output abs_re(g) : UInt64
    spawn with group
    eval @Global(1d) with abs_re(g).last(or: 100) +
    tp_event.aggregate(over_instances: All(ii, ig, is => ig = g),
        using: count)
output abs_hr_re(g) : UInt64
  spawn with group
  eval @Global(1d) with abs_hr_re(g).last(or: 50) +
    tp_event.aggregate(over_instances: All(ii, ig, is => ig = g),
        using: sum)
output tp_ratio(g)
    spawn with group
    eval when abs_re(g) != 0
        with cast<UInt64, Float64>(abs_hr_re(g)) / cast<UInt64,
            Float64>(abs_re(g))

/// Equalized Odds: True Positive
trigger @1d tp_ratio.aggregate(over_instances: all, using:
    max).defaults(to: 0.0) - tp_ratio.aggregate(over_instances: all,
    using: min).defaults(to: 0.0) > 0.1
```

Fig. 3: RTLola specification computing and checking the differences of the true positive ratios between all groups, which makes up one half of the equalized-odds specification for the COMPAS data set.

assigned a high-risk score under the condition that this person has re-offended. To encode the MAP estimation from Section 3.3, we assign the `abs_re` and `abs_hr_re` streams different default values when accessing the previous value, which effectively initializes the streams with these default values at the first time point. Finally, the trigger (line 36) encodes a violation of the fairness definition using the following underlying formula:

$$max_{g \in G}\{\mathbb{P}(\hat{Y} = 1 \mid G = g, Y = 1)\} - min_{g \in G}\{\mathbb{P}(\hat{Y} = 1 \mid G = g, Y = 1)\} \leq \epsilon.$$

Here, G is the set of all groups. Hence, this formula takes the maximum difference between *any* two groups and compares it against the threshold ϵ. This suffices to infer a violation in all cases. Additionally, the exact values of the ratios can be read from the parameterized streams such as `tp_ratio`. The full specification for equalized odds extends this principle to the false positive ratio by defining parameterized streams `abs_not_re` to count the defendants per group that did not re-offend, `abs_hr_not_re` to count the number of these that were screened high-risk, and `fp_ratio` for the resulting ratio. Additionally, the trigger condition is extended to account for all pairs of parameters of the `fp_ratio` stream. The experimental results of running this specification on the COMPAS data from the original ProPublica investigation can be found in Section 5.2.

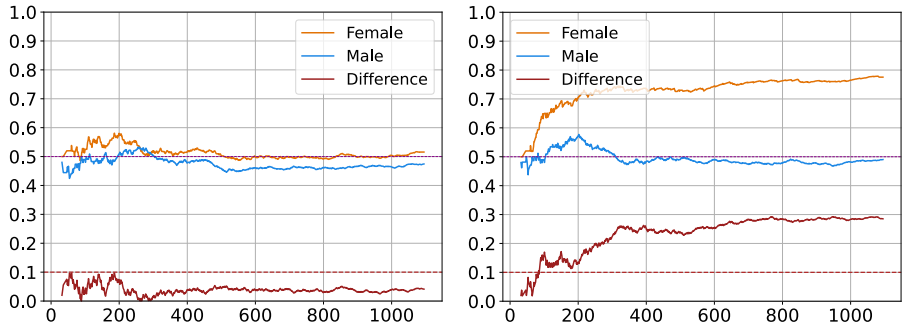
5 Case Studies

We specified all algorithmic fairness requirements defined in Section 2.2 with RTLola in a similar way as outlined for equalized odds in Section 4. In this section, we report on experiments with these fairness specifications in a variety of settings[2]. We first consider synthetically constructed data streams related to hiring and application scenarios that allow us to study the utility and efficiency of the approach under varying assumptions. Afterward, we consider data from the COMPAS recidivism risk assessment tool discussed in Section 1.1 to assess the utility of our tool in a real-world setting. The experiments were conducted with Ubuntu 24.04, a 4-core Intel i5 2.30GHz processor, as well as 8GB of memory.

5.1 Synthetic Scenarios

Our two synthetic scenarios deal with hiring done by a company and seminar assignments at a university. For the hiring scenario, we make the simplifying assumption that the company has no fixed limit on the number of employees it can hire. For the seminar assignment, we assume that each seminar has a fixed number of places. Both scenarios are synthesized from a generator script that allows us to specify and scale a number of interesting parameters such as the number of applicants and seminars, as well as the number of places per seminar. The input streams of both scenarios encode the individual applicants and events related to them, i.e., there is an event for an applicant with a specific

[2] Our artifact is available on Zenodo: `https://doi.org/10.5281/zenodo.14627198`.



(a) Fair algorithm. **(b)** Unfair algorithm.

Fig. 4: Demographic parity of hiring algorithms. The Female and Male lines correspond to the estimates $\hat{\Theta}_{F,M}$ for women and men, respectively. The Difference line shows the absolute difference between these ratios, while the dashed red line at 0.1 indicates the threshold parameter ϵ (cf. Definition 1).

id, *gender*, and *qualification*. For the seminar assignment system, also an input *seminar* to indicate which seminar the applicant applies to, such that we can also monitor conditional statistical parity in addition to demographic parity. Further, seminars have a predefined maximum number of places. Additionally, there is a separate input stream *accepted* that gives the IDs of accepted applicants. The generator allows to specify which decision algorithm should be used. We discuss these with the experimental results in the following.

Company Hiring. This scenario modeling a hiring system at a company consists of truly independent trials. We can adjust the probabilities $\Theta_F = \mathbb{P}(A = 1 \mid F = 1)$ and $\Theta_M = \mathbb{P}(A = 1 \mid M = 1)$ with which women and men get accepted, respectively. In this way, we can compare the monitoring outcome of a hiring system that is unfair by construction to a truly fair one. In the unfair system, we set the probability to be accepted for men to $\Theta_M = 0.5$ and for women to $\Theta_F = 0.2$, while both are 0.5 in the fair system. We then monitor for demographic parity (cf. Definition 1). The prior (cf. Section 3.3) is set to 0.5 with a confidence of 24 (a more detailed discussion on how to set these parameters follows in the next paragraph). Graphs for the estimated conditional probabilities $\hat{\Theta}_{F,M}$, their difference, and the trigger condition (based on $\epsilon = 0.1$) are depicted in Figure 4. As we can see, the fair algorithm stabilizes far below the trigger threshold. The diffuse behavior at the start, which usually would result in a number of false alarms, is held back by our MAP approach, such that no triggers are thrown. In contrast, after around time point 85, the unfair algorithm constantly raises triggers indicating unfairness, as the difference of the conditional probabilities stabilizes far above the threshold of 0.1, overpowering the prior belief. These results confirm that monitoring can adequately discern between unfair and fair systems after a reasonably small number of decisions has been made.

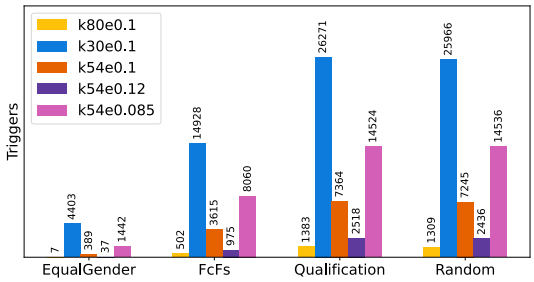
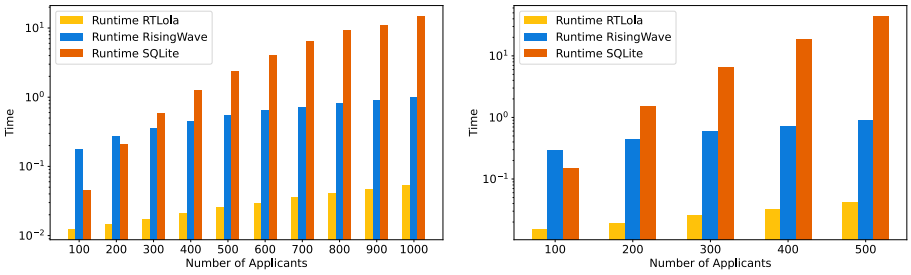


Fig. 5: Number of triggers thrown for different values of confidence κ (k) and threshold ϵ (e).

University Application. How to choose the right parameters? We now show that synthetic experiments can be an effective way to choose the confidence κ and threshold ϵ. We consider different decision-making algorithms for distributing places to applicants. This lets us explore how the different parameters influence the number of triggers on different algorithms. The first algorithm is *First Come First Served (FCFS)*, which accepts the first people applying regardless of other attributes. The second is *Randomize*, which picks randomly in the pool of applicants for a given seminar. The third algorithm, called *Qualification*, picks the most qualified people for each seminar. The last algorithm is *EqualGender*, which tries to ensure the same acceptance rates for all groups in the long run. Note that demographic parity does not take into account additional attributes such as the qualification, and hence the fairness of, e.g., the Qualification algorithm completely depends on the randomization of the qualification values. Similarly, the fairness of FCFS depends on the application times which are generated randomly. In Figure 5, we compare the number of thrown triggers for the different parameters on two thousand generated scenarios for every algorithm with a hundred applicants each. Our specification of demographic parity with parameters $\kappa = 54$ and $\epsilon = 0.085$ gets violated 1031 times over all the scenarios generated with the EqualGender algorithm (which have 200000 distinct events). A general trend that can be inferred from Figure 5 is that parameter values that are too low lead to a large amount of triggers. Finding the right parameters requires estimating how many applicants are expected, and selecting them to achieve a desired contrast between the different algorithms on the simulated scenarios. For instance, a confidence of 80 and threshold of 0.1 results in 187× as many triggers on the random algorithm than on the EqualGender algorithm, while a confidence of 30 and threshold of 0.1 results only in around 6× as many triggers on the random algorithm.

Runtime Comparison. It is a viable question to ask what advantages monitoring with a stream-based specification language has over a simple database implementation. Therefore, we have compared our approach with a naïve implementation using SQLite, and an advanced implementation using RisingWave [44], a state-of-the-art streaming database [24]. For the databases we first defined a SQL query that encodes the fairness specifications and returns a Boolean value, similar to an RTLola trigger. During execution we then iteratively update the database with new events. Crucially, the streaming database is optimized for such incremental computations and only updates the changed values in the query. This is a similar approach to the RTLola monitor, which also incrementally and effi-



(a) Demographic parity. **(b)** Conditional statistical parity.

Fig. 6: Runtime comparison between monitoring and database implementations. The bars report the average runtime over ten generated scenarios.

ciently updates its valuation upon encountering new events. We have generated seminar application scenarios with a varying number of applicants and report the average runtime of the three approaches in Figure 6. We stopped at 500 applicants in the case of conditional statistical parity because the SQLite approach already took more than 100 seconds. The results show that RTLola is faster than the database approaches in our scenarios. As a side result, we also see that the streaming database RisingWave outperforms the SQLite implementation on all but the smallest inputs, which is even more pronounced for conditional statistical parity. Monitoring with RTLola still significantly outperforms even the advanced streaming database approach. This runtime advantage gets particularly important for systems meant to be deployed at a large scale, such as the COMPAS recidivism risk assessment tool.

5.2 Monitoring Fairness of the COMPAS Tool

We revisit the motivating example from Section 1.1 to study the utility of our approach on real-world data from the recidivism risk prediction tool COMPAS. We use the same data set of COMPAS screenings between 2013 and 2014 in Broward County, Florida, which was also used by ProPublica [5] in their original investigation. We converted their original data into streams that are temporally ordered to simulate online monitoring of the COMPAS tool. We then executed our RTLola monitor with the equalized-odds specification as outlined in Section 4 for every combination of social groups. We used a confidence κ of 100, prior $\gamma(\omega) = 0.5$ and a threshold $\epsilon = 0.1$. In Figure 7, we illustrate the probability estimates and corresponding differences for African-American and European-American defendants. Note that the first two years are not shown, as a false positive result can only be definitely inferred after two years without recidivism, since this is the prediction horizon of the COMPAS tool as outlined in the COMPAS user guide [37]. As we can see, once the first two years have passed and the first definite outcomes can be inferred, unfairness can be established after less than a month, since the false positive rates of the groups quickly diverge. Since

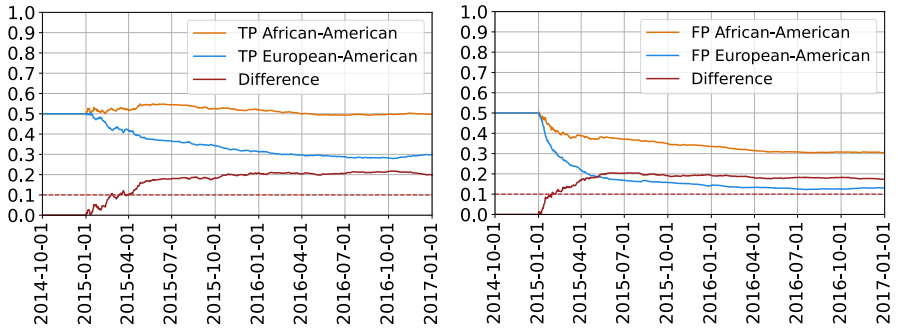


Fig. 7: True positive rates (TP) and false positive rates (FP) of African-American and European-American defendants while monitoring equalized odds on the COMPAS data set [5]. The dashed red line shows threshold ϵ.

such tools are deployed over a long time-horizon, these initial two years without verdict bear comparatively little weight. Moreover, the judgment is robust and stays far above the threshold afterward. This experiment with data from the COMPAS tool shows that stream-based monitoring can be a viable method to detect unfairness of prediction systems early, and hence reduce the number of unfair decisions and predictions.

6 Conclusion

We have studied the monitoring of algorithmic fairness with the stream-based specification language RTLola. This language not only allows us to encode the estimation of conditional probabilities inherent to algorithmic-fairness specifications but also the timing requirements common to real-world applications where these specifications are crucial. We have demonstrated this exemplarily with the COMPAS tool that is used to predict the recidivism risk of defendants. Moreover, we have contributed a benchmark generator for constructing synthetic scenarios related to job application and university admission scenarios and have used these scenarios for an extensive evaluation of our approach, which shows that it is able to detect the ground truth reliably and efficiently. In the future, we plan on leveraging RTLola's innate capabilities for reasoning about data and time to express even more complex algorithmic-fairness specifications dealing with, e.g., expected values of credit scores or response times.

Acknowledgments. This work was partially supported by the DFG in project 389792660 (TRR 248 – CPEC) and by the ERC Grant HYPER (No. 101055412). Funded by the European Union. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

1. Agarwal, A., Beygelzimer, A., Dudík, M., Langford, J., Wallach, H.M.: A reductions approach to fair classification. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 60–69. PMLR (2018), http://proceedings.mlr.press/v80/agarwal18a.html
2. Albarghouthi, A., D'Antoni, L., Drews, S.: Repairing decision-making programs under uncertainty. In: Majumdar, R., Kuncak, V. (eds.) Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10426, pp. 181–200. Springer (2017). https://doi.org/10.1007/978-3-319-63387-9_9, https://doi.org/10.1007/978-3-319-63387-9_9
3. Albarghouthi, A., D'Antoni, L., Drews, S., Nori, A.V.: Fairsquare: probabilistic verification of program fairness. Proc. ACM Program. Lang. **1**(OOPSLA), 80:1–80:30 (2017). https://doi.org/10.1145/3133904, https://doi.org/10.1145/3133904
4. Albarghouthi, A., Vinitsky, S.: Fairness-aware programming. In: danah boyd, Morgenstern, J.H. (eds.) Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019. pp. 211–219. ACM (2019). https://doi.org/10.1145/3287560.3287588, https://doi.org/10.1145/3287560.3287588
5. Angwin, J., Larson, J., Mattu, S., Kirchner, L.: Machine bias. there's software used across the country to predict future criminals. and it's biased against blacks. ProPublica (2016), https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing
6. Artificial intelligence act (regulation (EU) 2024/1689), official journal version of 13 june 2024, http://data.europa.eu/eli/reg/2024/1689/oj (Accessed: 28.01.2024)
7. Bastani, O., Zhang, X., Solar-Lezama, A.: Probabilistic verification of fairness properties via concentration. Proc. ACM Program. Lang. **3**(OOPSLA), 118:1–118:27 (2019). https://doi.org/10.1145/3360544, https://doi.org/10.1145/3360544
8. Baum, K., Biewer, S., Hermanns, H., Hetmank, S., Langer, M., Lauber-Rönsberg, A., Sterz, S.: Taming the AI monster: Monitoring of individual fairness for effective human oversight. In: Neele, T., Wijs, A. (eds.) Model Checking Software - 30th International Symposium, SPIN 2024, Luxembourg City, Luxembourg, April 8-9, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14624, pp. 3–25. Springer (2024). https://doi.org/10.1007/978-3-031-66149-5_1, https://doi.org/10.1007/978-3-031-66149-5_1
9. Baumeister, J., Finkbeiner, B., Kohn, F., Scheerer, F.: A tutorial on stream-based monitoring. In: Platzer, A., Rozier, K.Y., Pradella, M., Rossi, M. (eds.) Formal Methods - 26th International Symposium, FM 2024, Milan, Italy, September 9-13, 2024, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14934, pp. 624–648. Springer (2024). https://doi.org/10.1007/978-3-031-71177-0_33, https://doi.org/10.1007/978-3-031-71177-0_33
10. Baumeister, J., Finkbeiner, B., Kohn, F., Schirmer, S., Torens, C., Löhr, F., Manfredi, G.: Monitoring unmanned aircraft: Specification, integration, and lessons-learned. In: Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, Canada, July 22-27, 2024 (2024)

11. Baumeister, J., Finkbeiner, B., Schirmer, S., Schwenger, M., Torens, C.: Rtlola cleared for take-off: Monitoring autonomous aircraft. In: Lahiri, S.K., Wang, C. (eds.) Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12225, pp. 28–39. Springer (2020). https://doi.org/10.1007/978-3-030-53291-8_3, https://doi.org/10.1007/978-3-030-53291-8_3
12. Blyth, C.R.: On simpson's paradox and the sure-thing principle. Journal of the American Statistical Association **67**(338), 364–366 (1972). https://doi.org/10.1080/01621459.1972.10482387
13. Bostrom, N., Yudkowsky, E.: The ethics of artificial intelligence, p. 316–334. Cambridge University Press (2014)
14. Cano, F., Henzinger, T.A., Könighofer, B., Kueffner, K., Mallik, K.: Fairness shields: Safeguarding against biased decision makers. CoRR **abs/2412.11994** (2024). https://doi.org/10.48550/ARXIV.2412.11994, https://doi.org/10.48550/arXiv.2412.11994
15. Colorado senate bill 24-205, https://leg.colorado.gov/sites/default/files/2024a_205_signed.pdf (Accessed: 28.01.2024)
16. Convent, L., Hungerecker, S., Leucker, M., Scheffel, T., Schmitz, M., Thoma, D.: Tessla: Temporal stream-based specification language. In: Massoni, T., Mousavi, M.R. (eds.) Formal Methods: Foundations and Applications - 21st Brazilian Symposium, SBMF 2018, Salvador, Brazil, November 26-30, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11254, pp. 144–162. Springer (2018). https://doi.org/10.1007/978-3-030-03044-5_10, https://doi.org/10.1007/978-3-030-03044-5_10
17. Corbett-Davies, S., Pierson, E., Feller, A., Goel, S., Huq, A.: Algorithmic decision making and the cost of fairness. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017. pp. 797–806. ACM (2017). https://doi.org/10.1145/3097983.3098095, https://doi.org/10.1145/3097983.3098095
18. D'Angelo, B., Sankaranarayanan, S., Sánchez, C., Robinson, W., Finkbeiner, B., Sipma, H.B., Mehrotra, S., Manna, Z.: LOLA: runtime monitoring of synchronous systems. In: 12th International Symposium on Temporal Representation and Reasoning (TIME 2005), 23-25 June 2005, Burlington, Vermont, USA. pp. 166–174. IEEE Computer Society (2005). https://doi.org/10.1109/TIME.2005.26, https://doi.org/10.1109/TIME.2005.26
19. Dastin, J.: Amazon scraps secret ai recruiting tool that showed bias against women (2018), https://www.reuters.com/article/idUSKCN1MK0AG/ (Accessed: 19.04.2024)
20. 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
21. Faymonville, P., Finkbeiner, B., Schirmer, S., Torfah, H.: A stream-based specification language for network monitoring. In: Falcone, Y., Sánchez, C. (eds.) Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings. Lecture Notes in Computer Science, vol. 10012, pp. 152–168. Springer (2016). https://doi.org/10.1007/978-3-319-46982-9_10, https://doi.org/10.1007/978-3-319-46982-9_10

22. Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: Certifying and removing disparate impact. In: Cao, L., Zhang, C., Joachims, T., Webb, G.I., Margineantu, D.D., Williams, G. (eds.) Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015. pp. 259–268. ACM (2015). https://doi.org/10.1145/2783258.2783311, https://doi.org/10.1145/2783258.2783311
23. Finkbeiner, B., Kohn, F., Schledjewski, M.: Leveraging static analysis: An IDE for rtlola. In: André, É., Sun, J. (eds.) Automated Technology for Verification and Analysis - 21st International Symposium, ATVA 2023, Singapore, October 24-27, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14216, pp. 251–262. Springer (2023). https://doi.org/10.1007/978-3-031-45332-8_13, https://doi.org/10.1007/978-3-031-45332-8_13
24. Fragkoulis, M., Carbone, P., Kalavri, V., Katsifodimos, A.: A survey on the evolution of stream processing systems. VLDB J. **33**(2), 507–541 (2024). https://doi.org/10.1007/S00778-023-00819-8, https://doi.org/10.1007/s00778-023-00819-8
25. Gorostiaga, F., Sánchez, C.: Striver: Stream runtime verification for real-time event-streams. In: Colombo, C., Leucker, M. (eds.) Runtime Verification - 18th International Conference, RV 2018, Limassol, Cyprus, November 10-13, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11237, pp. 282–298. Springer (2018). https://doi.org/10.1007/978-3-030-03769-7_16, https://doi.org/10.1007/978-3-030-03769-7_16
26. Hardt, M., Price, E., Srebro, N.: Equality of opportunity in supervised learning. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 3315–3323 (2016), https://proceedings.neurips.cc/paper/2016/hash/9d2682367c3935defcb1f9e247a97c0d-Abstract.html
27. Henzinger, T.A., Karimi, M., Kueffner, K., Mallik, K.: Monitoring algorithmic fairness. In: Enea, C., Lal, A. (eds.) Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13965, pp. 358–382. Springer (2023). https://doi.org/10.1007/978-3-031-37703-7_17, https://doi.org/10.1007/978-3-031-37703-7_17
28. Henzinger, T.A., Karimi, M., Kueffner, K., Mallik, K.: Runtime monitoring of dynamic fairness properties. In: Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency, FAccT 2023, Chicago, IL, USA, June 12-15, 2023. pp. 604–614. ACM (2023). https://doi.org/10.1145/3593013.3594028, https://doi.org/10.1145/3593013.3594028
29. Henzinger, T.A., Kueffner, K., Mallik, K.: Monitoring algorithmic fairness under partial observations. In: Katsaros, P., Nenzi, L. (eds.) Runtime Verification - 23rd International Conference, RV 2023, Thessaloniki, Greece, October 3-6, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14245, pp. 291–311. Springer (2023). https://doi.org/10.1007/978-3-031-44267-4_15, https://doi.org/10.1007/978-3-031-44267-4_15
30. Kamiran, F., Calders, T.: Data preprocessing techniques for classification without discrimination. Knowl. Inf. Syst. **33**(1), 1–33 (2011). https://doi.org/10.1007/S10115-011-0463-8, https://doi.org/10.1007/s10115-011-0463-8
31. Kamishima, T., Akaho, S., Asoh, H., Sakuma, J.: Fairness-aware classifier with prejudice remover regularizer. In: Flach, P.A., Bie, T.D., Cristianini, N. (eds.)

Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II. Lecture Notes in Computer Science, vol. 7524, pp. 35–50. Springer (2012). https://doi.org/10.1007/978-3-642-33486-3_3, https://doi.org/10.1007/978-3-642-33486-3_3

32. Kusner, M.J., Loftus, J.R., Russell, C., Silva, R.: Counterfactual fairness. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., 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. pp. 4066–4076 (2017), https://proceedings.neurips.cc/paper/2017/hash/a486cd07e4ac3d270571622f4f316ec5-Abstract.html
33. Lin, Z., Jung, J., Goel, S., Skeem, J.: The limits of human predictions of recidivism. Science Advances **6**(7) (2020). https://doi.org/10.1126/sciadv.aaz0652
34. Matthias, A.: The responsibility gap: Ascribing responsibility for the actions of learning automata. Ethics and Information Technology **6**(3), 175–183 (2004). https://doi.org/10.1007/s10676-004-3422-1, https://doi.org/10.1007/s10676-004-3422-1
35. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A survey on bias and fairness in machine learning. ACM Comput. Surv. **54**(6), 115:1–115:35 (2022). https://doi.org/10.1145/3457607, https://doi.org/10.1145/3457607
36. Mitchell, T.M.: Machine learning, International Edition. McGraw-Hill Series in Computer Science, McGraw-Hill (1997), https://www.worldcat.org/oclc/61321007
37. Northpoint Inc. d/b/a equivant: Practitioner's guide to compas core, https://archive.epic.org/algorithmic-transparency/crim-justice/EPIC-16-06-23-WI-FOIA-201600805-COMPASPractionerGuide.pdf (Accessed: 11.10.2024)
38. Pessach, D., Shmueli, E.: Algorithmic Fairness, pp. 867–886. Springer International Publishing, Cham (2023). https://doi.org/10.1007/978-3-031-24628-9_37, https://doi.org/10.1007/978-3-031-24628-9_37
39. Ruoss, A., Balunovic, M., Fischer, M., Vechev, M.T.: Learning certified individually fair representations. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (2020), https://proceedings.neurips.cc/paper/2020/hash/55d491cf951b1b920900684d71419282-Abstract.html
40. Teuber, S., Beckert, B.: An information-flow perspective on algorithmic fairness. In: Wooldridge, M.J., Dy, J.G., Natarajan, S. (eds.) Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada. pp. 15337–15345. AAAI Press (2024). https://doi.org/10.1609/AAAI.V38I14.29458, https://doi.org/10.1609/aaai.v38i14.29458
41. Thomas, D., Ravi, K.: The potential for artificial intelligence in healthcare. Future Healthc Journal **6** (2019). https://doi.org/10.7861/futurehosp.6-2-94
42. Tramèr, F., Atlidakis, V., Geambasu, R., Hsu, D.J., Hubaux, J., Humbert, M., Juels, A., Lin, H.: Fairtest: Discovering unwarranted associations in data-driven applications. In: 2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017. pp. 401–416. IEEE (2017). https://doi.org/10.1109/EuroSP.2017.29, https://doi.org/10.1109/EuroSP.2017.29

43. Udeshi, S., Arora, P., Chattopadhyay, S.: Automated directed fairness testing. In: Huchard, M., Kästner, C., Fraser, G. (eds.) Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018. pp. 98–108. ACM (2018). https://doi.org/10.1145/3238147.3238165, https://doi.org/10.1145/3238147.3238165
44. Wang, Y., Liu, Z.: A sneak peek at risingwave: a cloud-native streaming database. In: Zhou, Y., Chrysanthis, P.K., Gulisano, V., Zacharatou, E.T. (eds.) 16th ACM International Conference on Distributed and Event-based Systems, DEBS 2022, Copenhagen, Denmark, June 27 - 30, 2022. pp. 190–193. ACM (2022). https://doi.org/10.1145/3524860.3543284, https://doi.org/10.1145/3524860.3543284