Traceability and Accountability by Construction*

Julius Wenzel[2], Maximilian A. Köhl[1], Sarah Sterz[1], Hanwei Zhang[1], Andreas Schmidt[1], Christof Fetzer[2], and Holger Hermanns[1]

[1] Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
{koehl,sterz,andreas.schmidt,hermanns}@cs.uni-saarland.de
hanwei.zhang@uni-saarland.de
[2] Technische Universität Dresden, Dresden, Germany
{julius.wenzel,christof.fetzer}@tu-dresden.de

Abstract. As artificial intelligence (AI) systems influence ever more high-stake decision-making processes, such as university applicant screening or medical diagnoses, ensuring the trustworthiness of these systems and their decisions is crucial. This paper presents a significant step towards achieving trustworthy AI decisions by introducing a novel framework for enhancing traceability and accountability by construction. Our approach encompasses the entire decision-making pathway—from the raw datasets used to train the AI system, through the algorithms and programs employed, to the involved parties and the final decisions made. At the core of our methodology is the *Decision Bill of Materials* (DBOM), which meticulously documents all elements contributing to a decision while ensuring accountability and traceability through cryptographic signatures. Furthermore, we leverage results from logic programming to enable systematic reasoning about the processes and decision documented in a DBOM. This allows us to verify that the system meets specific certification standards and that individual decisions can be qualified as trustworthy. This framework not only advances the construction of reliable AI systems but also aligns technological developments with ethical imperatives and regulatory expectations.

1 Introduction

Establishing trust in AI systems is paramount for their acceptance and integration into societal frameworks [34, 36–38]. This is especially pronounced whenever those systems put fundamental rights at risk or have a significant impact on people's health or safety. Examples of such high-stake decision-making applications include university applicant screening [16, 68], credit scoring [49], visa decisions [46], predictive policing [33], court decisions (as with COMPAS [6, 21, 26, 39]), selection of job applicants [47, 48], or tenant screening [60].

* This work was partially supported by the DFG under the project TRR 248 (CPEC, see https://perspicuous-computing.science, project ID 389792660) and by VolkswagenStiftung as part of Grant AZ 98514 – EIS.

Trustworthy AI necessitates robust, transparent methodologies that ensure compliance with both ethical imperatives and regulatory standards [34,36,42,55]. This paper proposes a novel framework designed to inherently enhance the traceability and accountability of AI decisions. We suggest an approach for tracking certain key interactions of trusted parties with the system (such as certifications, validations, or audits), for making inferences from these, and for mapping out the relevant processes contributing to a decision in a structured manner. By encompassing the full decision-making pathway—from the raw datasets used for training the AI system, to the algorithms and programs employed, to the parties involved and the final decisions made—our approach advocates for a systematic, verifiable, and holistic process with automatable checks.

Central to our approach is the implementation of a *Decision Bill of Materials* (DBOM)—a comprehensive record that documents all elements and processes contributing to an AI's decision-making pathway and an individual decision. The concept of a *Bill of Materials* (BOM) originates from manufacturing and engineering, where it is used to document all components, parts, and raw materials required to build a certain product. In the context of AI decisions, we adapt this concept to form the DBOM detailing and documenting how a decision came about. The idea to use a BOM to document elements of an AI system is not new. With CycloneDX's *Machine Learning Bill of Materials* (ML-BOM) [23] and The Linux Foundation's SPDX AI profile [30] there already exist well-established standards for documenting software components, models, and datasets associated with an AI system. Our proposal extends beyond those existing approaches as it concerns individual decisions and not just the system itself.

This documentation strategy enhances transparency and facilitates audits, allowing stakeholders to verify that all aspects of the AI system and its decision adhere to stipulated requirements. By incorporating cryptographic signatures, the DBOM assures accountability and transparency for each element contributing to a decision-making process. Additionally, through the application of concepts from logic programming, we introduce a method to systematically and rigorously reason about DBOMs. This analysis helps to ensure that the AI system aligns with predefined certification standards and facilitates the independent verification of each decision as potentially trustworthy.

By embedding these mechanisms at the core of the AI development and decision-making process, we aim to foster AI systems that are not only effective and efficient but also ensure traceability and accountability by construction, thereby aligning technological advancements with ethical imperatives and legal expectations. A relevant legal framework might, for example, be the AI Act of the European Union [27]. Our approach could help to shape and instantiate parts of the required documentation procedures (e.g., Art. 11, 12, 18), the fulfillment of transparency obligations (e.g., Art. 13), as well as the tracking of mandatory certificates (Art. 44). Our approach could also inform the design of harmonized standards as envisioned by the EU (Art. 40).

Contributions. This paper contributes a methodology for ensuring accountability and for assessing a system's trustworthiness. We discuss the concrete instantia-

tion of this methodology by exploiting existing techniques and propose a concrete specification for DBOMs based on CycloneDX and JSON Schema.

Structure. In Sec. 2, we introduce a concrete running example revolving around a university admission system. In Sec. 3, we present the central building blocks of DBOMs and how existing techniques such as shielding and monitoring fit into the picture. In Sec. 4, we develop an approach for the automated verification of certain aspects of a DBOM based on logic programming. In Sec. 5, we discuss our approach and give an outlook on future work.

2 Example: University Admission

To illustrate the concepts introduced in this paper, we present an example of an AI system used for desk-rejecting applications at a university. The core objectives in this process are manifold: on the one hand, it should be ensured that the system is accurate in its decisions, meaning that it does not desk-reject promising candidates and that it does not erroneously propose strictly underqualified applicants for acceptance. On the other hand, it has to be ensured that fundamental risks are guarded, that every applicant can trust that they have been treated fairly throughout the selection process, and that the system is transparent to the university officials who govern the selection and ultimately take responsibility for it.

2.1 An Ideal Process

In our example, an automated decision-making system is implemented at a university to assess applications and decide which ones are desk-rejected. The system is designed to streamline the admission process by filtering out applications that do not meet the university's admission criteria, and therefore reduce the workload on university employees. Ideally, this system would be designed to ensure transparency and accountability at every step.

First of all, the development of such a system requires the selection of training data. To this end, historical data from previous applications to the university is collected. This data includes both accepted and rejected applications, annotated with comprehensive feedback from admissions officers. Special care must be taken to ensure that the data represents a diverse applicant pool in terms of demographics, socio-economic backgrounds, and geographic locations. If the training data does not reflect the diversity of future applicants, the model might discriminate against minorities which were not represented in the training data. Having an equal representation inside each category is not necessary, however no group that exists in reality should be left out entirely.

With the data selected, a machine learning model can be trained. A variety of features are extracted from the applications, such as GPA, test scores, and the richness of the personal statement. The model is trained using a supervised learning algorithm, where the outcomes (accepted or rejected) serve as labels.

Throughout the training process, the model's performance must be continuously evaluated using a validation set, ensuring that it generalizes well beyond just the training data. Before deployment, the model must undergo thorough auditing to check for biases and fairness, as university officials are aware that historical data is prone to the preservation of existing bias. Adjustments must be made to the training process to mitigate any discovered biases.

Upon successful auditing, the system is finally deployed. As applications come in, they are automatically processed by the AI system. Each application receives a score based on the learned weights of various features like academic achievement and extracurricular involvement. Applications falling below a certain threshold score are recommended for desk rejection. Thereby, the system provides a recommendation to either proceed with a detailed review or desk-reject the application. The system's recommendations must be logged for future audits. While the system is in operation, a fairness monitor [13] must be used to check each individual decision for individual unfairness, as well as all decisions made over a certain period of time for group unfairness.

A human oversight person must be in the position to inspect the AI's decisions at all times and override any outputs. This enables taking into account additional contextual information that the AI might not have sufficiently accounted for. The human oversight person will especially review all cases that were flagged by the fairness monitor but can also pay special attention to other cases as they see fit. During routine inspections, when a specific issue is reported, or an applicant wants to contest the system's decision, the systems outputs are scrutinized to ensure compliance with fairness and performance standards. For example, if an applicant appeals their rejection, claiming potential bias, the specific case along with similar cases handled by the system are reviewed. The model's decision-making process is analyzed to identify whether any features disproportionately influenced the outcome. To this end, a comprehensive record of the case is required. Depending on the findings, the model may be retrained, or the decision logic adjusted to prevent future occurrences of the issue.

2.2 Requirements

To summarize, the ideal process sketched above embodies the following requirements for the development of the system and its operation.

- **R1 Data Selection:** Training data must be carefully selected to ensure that the data represents a diverse applicant pool in terms of demographics, socio-economic backgrounds, and geographic locations.
- **R2 Feature Selection:** Features for the decision-making process must be selected such that they minimize the chance of any biases and are appropriate for the task at hand. In our example, only features that are known to correlate with academic success should be chosen. Note that features may need to include sensitive information like ethnicity in order to account for any statistical biases in other data.

R3 Training Algorithm: The algorithm used for training the system and making decisions must be appropriate for the task at hand. In our example, the algorithm has to support a great variety of different criteria that often cannot be applied to all of the traning data. For instance, not every candidate will have profiency in Mandarin, but it might still be part of a feature to detect candidates worth accepting.

R4 Validation Set: The validation set must be chosen such that it ensures that the system generalizes well beyond just the training data. An overfitting must be avoided, so that the trained model is able to treat data sets that differ a lot from what was shown during training.

R5 Auditing: Before deploying the system, it must undergo thorough auditing to check for any biases and unfairness within the system.

R6 Logging: Every decision must be logged together with all important information to enable its investigation and ensure transparency.

R7 Fairness Monitor: At runtime, a fairness monitor must be used to detect unfairness that may have made it through the audit.

R8 Human Oversight: A human oversight process must be put in place to review flagged decisions and handle appeals.

Notably, these requirements do not only concern the AI system itself but also cover the process used to build it as well as aspects of its deployment and operation. To ensure that these requirements are met and all involved entities can be held accountable for them, we envision the implementation of a DBOM. In case of this example, the DBOM would record the sources and characteristics of the training data, it would document the features, algorithms, and validation set which has been used for training, it would document the auditing process and the parties involved in it, and it would document the operational measures taken to ensure that each decision is fair and accurate.

3 The DBOM Approach

A typical DBOM can be expected to include information about the origins and characteristics of the used training data, about the feature extraction methods and training procedures, about validation and auditing processes, and about operational monitoring and oversight mechanisms. By systematically documenting each component and their interactions, the DBOM ensures transparency, traceability, and accountability. It aims to provide a clear audit trail that stakeholders can review to understand how a decision has been made and verify that the system operates according to ethical standards and regulatory requirements. As such, we envision DBOMs to become essential for building trust in AI systems, as they allow for thorough inspections and validations of decisions.

Fig. 1 depicts the architecture we are proposing in this paper. Here, a DBOM encodes facts about a decision and the involved processes, systems, and parties. As an example, it may state that the data used to train an AI system has been checked for biases by an accredited institution and that this AI system has

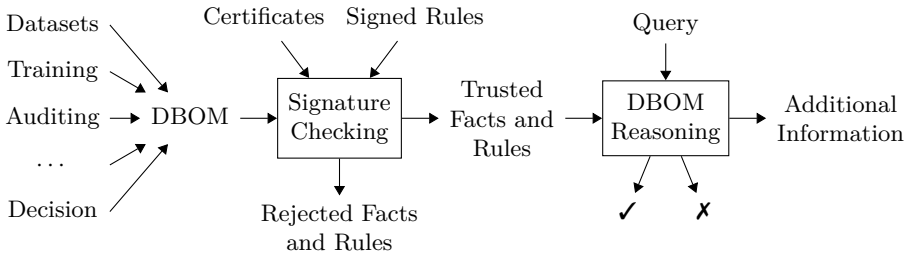

Fig. 1. Proposed architecture for ensuring traceability and accountability.

indeed been used for the decision. To ensure accountability and integrity, these facts are secured with cryptographic signatures issued by trusted parties who can be held responsible for their correctness. The trusted parties form a *Trust Domain*, which may vary between specific use cases. In the example mentioned just before, the Trust Domain includes the accredited institution and a cloud provider, such as AWS or Azure. The former may sign facts concerning the AI's training process, while the latter may sign facts confirming that a specific AI running on the cloud provider's infrastructure has produced the decision.

Given a set of trusted certificates, a stakeholder can then check the signatures of the facts within the DBOM to make sure that the facts have indeed been emitted by the respective parties and that those parties are also trusted to emit those facts. While this allows a stakeholder to ascertain characteristics of the AI system and the decision, the facts alone are insufficient to determine whether the decision adheres to ethical and regulatory norms applicable in the specific context. For instance, as discussed before, the AI system may need to be trained on data that has been checked for absence of biases but may also need to contain a fairness monitor [13] to continuously check for unfairness during operation. This is where reasoning about DBOMs becomes necessary.

To systematically and rigorously reason about a DBOM, trusted rules need to be provided in addition to the facts. These rules can originate from legal requirements or be directly provided by the stakeholders themselves. Again, rules should also be signed to make sure that they indeed originate from trustworthy entities. Finally, the trusted facts and rules are fed to some reasoning engine together with one or more queries. Queries are criteria a person or other stakeholder affected by an AI decision may want to check. A query may for instance ask whether the decision-making process adheres to ethical or regulatory requirements. The reasoning engine may then either affirm or reject the query and can provide additional information, for instance, concerning the facts and parties involved in affirming a query.

In the following, we concretely instantiate this idea by detailing considerations for its practical implementation.

3.1 Predicates and Assertions

At the core of the DBOM approach are *facts* that represent certain statements about an AI system, its operation, and its development.

Definition 1. *Given a set $\mathcal{P}$ of* predicates *and a set C of* constants*, a* fact *is a tuple $\langle P, c_1, \ldots, c_k \rangle$ where $P \in \mathcal{P}$ is a predicate and $c_i \in C$ are constants. We also use the notation $P(c_1, \ldots, c_k)$ to denote facts.*

Definition 2. *Given a set $\mathcal{P}$ of* predicates *and a set C of* constants*, an* assertion *ϕ is a finite set $\left\{P_i(c_{\langle i,1\rangle}, \ldots, c_{\langle i,k_i\rangle})\right\}_{i=1}^{n}$ of facts.*

Concrete Example. Image that the German TÜV, as a possible accredited institution, asserts (i) that they audited the training data D according to the regulations in place, e.g., for biases and other problems, and (ii) that they checked the program P used for training an AI system. Formally, this may correspond to the following assertion:

$$\left\{ \ \texttt{Audited}(\mathsf{T\ddot{U}V}, \mathsf{D}),\ \texttt{Checked}(\mathsf{T\ddot{U}V}, \mathsf{P}) \ \right\}$$

Note that the authority or entity responsible for the assertion occurs in the individual facts. This allows us to track which facts have been produced by which entity. Now, based on the program P and data D, a cloud provider such as AWS may assert that a neural network NN resulted from training with the program P on the data D. Formally, this corresponds to the assertion:

$$\left\{ \ \texttt{Trained}(\mathsf{AWS}, \mathsf{D}, \mathsf{P}, \mathsf{NN}) \ \right\}$$

In practice, elements of the process such as the training data, the training program, or the resulting neural network would be represented as cryptographic hashes linking the constants to the real elements in a way that is impossible to forge.

In addition to those assertions about the system itself, the DBOM for a particular decision, for instance, a desk rejection, may also state operational facts. For instance, the system making the decisions may run in the Azure cloud. Hence, Azure may assert that a particular decision made, like rejecting the applicant, has been the result of running the system S:

$$\left\{ \ \texttt{Produces}(\mathsf{Azure}, \mathsf{S}, \mathsf{RejectA}) \ \right\}$$

To link the system to the neural network trained on the audited data, the TÜV may again have asserted that the system S is based on the neural network NN. In addition, the TÜV may also have checked that the system S contains the required fairness monitor:

$$\left\{ \ \texttt{HasMonitor}(\mathsf{T\ddot{U}V}, \mathsf{S}),\ \ \texttt{BasedOn}(\mathsf{T\ddot{U}V}, \mathsf{S}, \mathsf{NN}) \ \right\}$$

All these assertions together make up the DBOM and encode various statements about the decision-making process and the parties and components that have been involved in it.

3.2 Cryptographic Signatures

As facts, rules, and queries risk to be corrupted during transmission or tampered with by an adversary, we use cryptographic signatures and certificates to establish trust in them. At their core, cryptographic signatures use algorithms and a signing key to generate a unique bitstring of fixed length, known as a *signature*, for a message or document.

The main objectives of cryptographic signatures are *integrity protection* and *non-repudiation*. Integrity protection ensures that an adversary can not alter a message without causing the signature verification to fail. Non-repudiation ensures that the signer, identified by a public key, is bound to the signature and cannot disown the signed content. The signer's public key is derived from the signing key and is needed for a valid signature verification. In combination, when provided with a cryptographically valid signature, stakeholders can have confidence that the signed data has not been tampered with and that the identified signer is indeed responsible for the content.

Note that we want to sign information that has not necessarily only one canonical textual representation. For example, the bitstrings "$P \wedge Q$" and "P AND Q" may express the same thing but their signatures will differ. Therefore, we need to establish one or more formats for such information, that is then used for both signature and verification.

As part of the signature verification process, *certificates* allow us to decide if a signature has been issued by an authorized entity. The set of certificates used form our Trust Domain. Certificates must contain a subject, a validity period and a scope or namespace. The subject is usually a public key identifying the signer and can be used for signature verification. The validity period enforces a regular key rotation and helps us against signers that became malicious or whose private keys leaked. The scope/namespace is a set of fact/rule identifiers/names that restrict the power of the signer. Restricting the signer's competences follows the principle of last privilege and protects against malicious actors. Without namespaces, a malicious actor could very easily abuse its power by gaining a small amount of trust and then signing completely different formulas. Although namespaces do not prevent against the risk of power abuse completely, it forces the issuer of a certificate to reflect on the exact extend of the trust expressed.

3.3 Signature Verification

Existing BOM formats like CycloneDx allow their users to sign elements of the BOM individually, therefore we suppose that every assertion inside a DBOM is signed individually. Each signature should at least consist of the signature algorithm, the public key of the signer, and the signature value. Consequently, each signed assertion from a BOM has the form of a tuple:

$$\langle \phi, \sigma \rangle$$

where ϕ is the assertion itself and σ is the signature consisting of an algorithm $\mathcal{A}$, a public key K^p derived from a secret key K^s and a signature value:

$$\sigma = \langle \mathcal{A}, K^p, \mathsf{sign}(\mathcal{A}, \phi, K^s) \rangle$$

When verifying the signature, the following conditions must be met:

- The public key K^p must be part of the Trust Domain, i.e., the verifying entity needs a certificate `Cert` it can trust.
- The verification time must be in the validity of the certificate. This moment in time does not have to be the moment the verification happens. The verifying entity might also verify the signature for a moment in the past or the future.
- The assertion must be inside the certificates namespace. That means that the predicate names occuring within the assertion have to be elements of the namespace of the certificate.

If taking t as verification time, this can be written as:

$$\begin{gathered} \langle \phi, \sigma \rangle \\ \wedge\ \exists \texttt{Cert}(K^p, [S,T], \mathsf{Ns}) \in \mathsf{TD} \\ \wedge\ \mathsf{verifySignature}(\phi, \sigma, K^p) \wedge\ \ t \in [S,T] \\ \wedge\ \forall \texttt{P} \in \phi(P \in \mathsf{Ns}) \\ \Rightarrow \phi \end{gathered}$$

3.4 DBOM Encoding

For the implementation of signed DBOMs, we can reuse existing BOM structures. Based on CycloneDx, we propose a standard for a possible structure for DBOMs consisting of a JSON schema.

The idea is to include signed facts into a CycloneDx BOM. Therefore, we add a new entry to the root of the BOM object and call it `decisionAssertions`. The following listing shows how the facts we've seen so far would look like inside this entry. As we can see, facts do not need to be signed individually. Instead, they can be bundled, which reduces the signature effort. Each fact is an object with only one property carrying the predicate name. The value of this property is an object containing the predicate's arguments.

To make our format more flexible, we propose to separate the definitions of possible predicates for facts and the definition of the general DBOM structure. The latter definitions form an outer schema, defining the signed assertions and facts as well as their signature format. In order to reach a good compatibility with CycloneDx, we also use the JSON Signature Format (JSF) for signatures. The former definitions form an inner schema, defining possible predicates, their arity, and the meaning of their arguments. Its content is up to everyone who implements a DBOM. DBOMs using different inner schemas will all validate against the outer schema. The inner schema will also be used to generate a

```
[
  {
    "facts": [
      {
        "audited": {
          "auditor": "TUEV Ost",
          "database": "31d94e3975..."
        }
      },
      {
        "checked": {
          "auditor": "TUEV Ost",
          "program": "md5:7e202fe0b6..."
        }
      }
    ],
    "signature": {
      "algorithm": "Ed25519",
      "value": "BR495M3IPZ..."
    }
  },
  {
    "facts": [
      {
        "trained": {
          "database": "31d94e3975...",
          "program": "md5:7e202fe0b6...",
          "neuralNetwork": {
            "name": "TensorStick",
            "version": "2.5.10",
            "digest": "e27e5b3f0..."
          }
        }
      }
    ],
    "signature": {
      "algorithm": "PS512",
      "value": "L4K9SJP3OR1..."
    }
  }
]
```

Listing 1.1. Example of a DBOM

parser for DBOMs. We think of a possible parser fine-tuning using JSON schema comments, the rest follows the inner schema structure (which might need a meta schema).

3.5 Exploiting Existing Techniques

In order to ensure that AI systems remain trustworthy throughout their lifecycle, it is essential to employ a combination of techniques that can monitor and enforce compliance with established requirements both at deployment and during runtime. Furthermore, XAI techniques may be required to provide explanations as part of a DBOM. These explanations may then also be subject to certain requirements. We will give a brief overview over the various techniques found in the literature that may be used to ensure the reliability of an AI or other system. The fact that these techniques have been used during development and during runtime as part of a system, would be recorded in a DBOM.

To truly ensure that the methodologies and techniques developed for verifying and enforcing the correctness and fairness of AI systems are effective, it is crucial to confirm that these verified or proven components are the ones actually being utilized in operation. This verification is particularly important as discrepancies between the tested or verified model and the actual operational system can undermine the system's integrity and trustworthiness. A DBOM records their usage and ascribes responsibility to trusted parties.

Runtime verification involves monitoring the system to check if it behaves as expected while it is operational. This approach uses formally defined properties or specifications that the system should adhere to, and checks log files or outputs in real-time to ensure compliance [4, 10, 11, 25, 40]. For our university admissions AI, runtime verification could involve continuously checking that the AI's decisions adhere to fairness constraints defined in the specifications [13].

Shielding involves creating a layer around the AI system that prevents it from making undesirable decisions, regardless of the input it receives. This method is particularly useful in operational environments where the system may encounter scenarios not previously accounted for during training [35]. For the admissions system, a shield could be designed to automatically detect and block any decisions that significantly deviate from expected fairness norms before they affect the application outcomes, thus providing a safety net against unforeseen biases. The basis for shielding can be a fairness monitor [13].

Runtime enforcement is similar to shielding. It can be seen as an extension of runtime verification where, instead of only detecting violations, the system actively intervenes to rectify them [14,28,43,52,53]. In the university admissions example, runtime enforcement mechanisms could adjust the weights assigned to certain applicant features on-the-fly to balance decision outcomes across different groups, ensuring continuous adherence to fairness standards.

Model checking is a systematic technique that checks whether a model of a system meets a given specification. This is typically done by exhaustively exploring all possible states of the system [9]. In the context of the AI application

selection system, model checking can be employed to verify that under all possible conditions, the system's decision-making process does not lead to outcomes that violate predefined fairness rules. It ensures that even in edge cases or under unusual conditions, the system behaves as expected. Notably, exhaustive model checking will likely be too expensive for real-world applications, in such cases, statistical model checking has proven effective [32].

Attribution is a prominent Explainable AI (XAI) method used to assign importance scores to inputs based on their influence on the output of an AI model. Various techniques exist to generate attributions, each with distinct methodologies. Gradient-based methods [2,8,61,63] use the gradient of a target class score with respect to the input to assess the impact of different regions of an image on the prediction. Particularly designed for images, CAM-based methods [18,58,73] replicate the attribution process without needing additional training or specific architectures. Occlusion-based methods [29,50,54,57] employ multiple candidate masks to measure their effects on the prediction and subsequently combine the results into a single saliency map. Learning-based methods [17, 24, 51, 74] incorporate an additional network or branch, which is trained on additional data and image-level labels to predict a saliency map for a given input image. Examples include the use of generators [17] or auto-encoders [24, 51]. Such techniques can be used to extract explanations for a decision and make them part of a DBOM.

Prototype method involves identifying and using representative examples from the dataset to explain the behavior and predictions of a machine learning model. These prototypes are selected to provide clear and interpretable references that illustrate the model's decisions. One basic intuition is to seek a set of representative samples with in a class or for more classes that provide interpretability [12]. Part-prototype [19,41,59] focuses on explaining model predictions by identifying and utilizing representative parts or segments of the input data. This method helps in breaking down complex inputs into understandable and meaningful components. As a DBOM is meant for stakeholders, care must be taken when including such information from the training set in it. In particular, sensitive and individualized information must not be leaked. Still, pointing out certain similarities to examples may facilitate the evaluation of a decision.

Counterfactual explaination [20,62,65] aims to elucidate the decision-making process of a machine learning model by presenting alternative scenarios. These explanations describe how the outcome would change if certain aspects of the input were altered. By highlighting these changes, counterfactual explanations help users understand the sensitivity and rationale behind a model's predictions. As attribution techniques, counterfactual explanations may be provided as part of a DBOM and could be checked to ensure compliance.

Verbal interpretability explains decisions or predictions of AI models using natural, human-understandable language. One branch of this involves methods based on decision sets or rule sets. These rule extraction techniques can utilize network-specific information [5, 22], such as the network structure or learned weights. Alternatively, they can treat the AI model as a black-box and derive explanations directly from the examples generated by the model [15,64]. Another

branch pertains to Visual Question Answering (VQA) [7, 44, 70], which involves generating textual explanations about objects appearing in images. With the advancement of large language models like ChatGPT [1], techniques such as Chain-of-Thought prompting [66, 69, 72], where complex tasks are broken down into smaller, manageable steps within the promptare also considered part of verbal interpretability. If a decision is based on text-generation or -processing, such techniques may be used to generate explanations for DBOMs.

Clearly, covering all relevant literature here is impossible and ultimately, what makes sense for a system and a DBOM depends on the use case. In any case, integrating at least some of the aforementioned approaches directly into AI decision making and documenting their proper usage via a DBOM promotes trustworthiness of AI systems. In the university admissions example, deploying these methods would involve setting up the system with embedded checks and balances that continuously assess and adjust the decision-making processes, and providing applications with reasonable and accurate explanations. This not only helps in maintaining fairness but also builds trust among stakeholders that the system is performing as intended and adhering to ethical guidelines.

In practice, for the university admissions example, the DBOM would include details such as versions of the fairness algorithms used, data sets for training, and specifics of the runtime enforcement and explanation mechanisms. Each component's cryptographic signature would verify its authenticity and compliance with the specified standards. If any component is updated or modified, its new version and signature must be documented in the updated BOM, ensuring a continuous chain of trust. This mechanism is essential to maintain transparency, foster trust among stakeholders, and ensure that the system adheres to ethical and legal standards. Such a practice not only enhances the reliability of the AI system but also bolsters its acceptance by providing a clear, traceable record of compliance and integrity.

4 Reasoning About DBOMs

Having verified that all facts have been emitted by entities which are trusted to emit the respective facts, we aim to analyze a DBOM to automatically check certain aspects of it. To this end, we reuse algorithms and results from logic programming.

4.1 Rules

AI regulation coming from legislators does not include details of how to implement the required features, but relate to industry standards, or even presuppose that those are developed within a limited time frame (as done in the European AI Act [27]). Indeed, industry standards and certification bodies such as the TÜV often lay out detailed criteria to fullfil. In general, it is natural that those criteria are presented in the form of conditionals of the form if-then, for instance: *If* a decisision produced by an AI system including a fairness monitor is based

on an approved training set, and the system is properly audited, *then* it can be qualified as trustworthy. This layout of facts and rules allows us to automate the validation process using logic programming.

In general, computationally solving problems in First-Order Logic (FOL) is not feasible due to undecidability and/or complexity. However, there are subsets such as Horn clauses where automated reasoning becomes possible [67]. The data we want to process for validation can easily be regarded as Horn clauses, where DBOM information corresponds to facts or ground terms and the validation criteria can be seen as definite Horn clauses. If this is not the case, e.g., for explanations that may take some other shape, we assume that facts can still be extracted from the provided information. Finally, we can use the assertions we want to validate as goal clauses, as will be explained below. The resulting system can be expressed in a logic programming language, and it can be validated automatically by a logic reasoning engine.

We propose to use an engine based on the Datalog programming language [31]. In contrast to other logic programming languages, Datalog is purely declarative and can be optimized for a performance comparable to non-logic tools [45,56,71].

We use the facts extracted from the DBOM as input for a Datalog reasoning engine. For the rules, we use a similar mechanism as described in Sec. 4.3. Our rules are signed the same way as our facts are, and we also use certificates for signature verification. However, there is a slight difference regarding namespaces: When checking the signature, we only want the right-hand side of the rule to be part of the namespace. For example, if we have a rule allowing to deduce the predicate `Certified` from other facts

$$\frac{\texttt{Audited}(x, d) \quad \texttt{Checked}(y, p) \quad \texttt{Trained}(z, d, p, n)}{\texttt{Certified}(n)}$$

we only need `Certified` to be element of the certificate's namespace.

We strongly suggest to sign rules and DBOMs separately, ideally by different participants. If (one of the) creator(s) of a DBOM wants to derive new facts from the existing ones, they can simply can add those facts to the DBOM instead. To store rules, we therefore propose a different format, described in Sec. 4.3. The reasoning engine recieves a DBOM and one or multiple signed rulesets.

Having verified the DBOM and the set(s) of rules cryptographically, this gives us a set of trusted facts F and a set of trusted rules R. Both together form what is often called a *logic database*, because of its similarities to relational databases. This database is used to be queried with the desired assertions regarding the AI decision. In most logic programming languages like Datalog, queries (also called *goals*), are represented as negated predicates:

$$\neg\texttt{Trustable}(\mathsf{Admitted}(\mathsf{JohnDoe}))$$

This means that we actually want `Trustable` to be true. Datalog then tries to prove that when including a query q, the set $\{F \cup R \cup \{q\}\}$ is inconsistent and leads to a contradiction. It does so by trying to find a fact q or a rule that leads to q. In case of the rule, all its conditions are then recursively checked

like q. At each step, the reasoning engine can keep track and provide us with a *trace* afterwards. This trace contains all steps that were necessary to prove q. It contains the used facts and rules and can be used for debugging and for deeper understanding how a decision has been made.

4.2 Reasoning Example

Using the facts from the university admission example, we can now reason about the decision process. In this case, assume that a rejected applicant would like to know whether their rejection is *trustable* according to the rules in place. That is, the query is:

$$\texttt{Trustable}(\mathsf{RejectA})$$

Recall the previously discussed rule rule saying that a neural network is certified if it has been trained on audited data with a program that has been checked:

$$\frac{\texttt{Audited}(x, d) \quad \texttt{Checked}(y, p) \quad \texttt{Trained}(z, d, p, n)}{\texttt{Certified}(n)}$$

In addition, we assume that the following rules exist:

$$\frac{\texttt{HasMonitor}(x, s) \quad \texttt{BasedOn}(y, s, n) \quad \texttt{Certified}(n)}{\texttt{Certified}(s)}$$

$$\frac{\texttt{Produces}(x, s, o) \quad \texttt{Certified}(s)}{\texttt{Trustable}(o)}$$

So, to be trustable, a decision must have been produced by a certified system, and to be certified, a system must contain a fairness monitor and must be based on a neural network which has been certified. Using the facts from the BOM, we first establish that the neural network NN has been certified:

$$\frac{\texttt{Audited}(\mathsf{T\ddot{U}V}, \mathsf{D}) \quad \texttt{Checked}(\mathsf{T\ddot{U}V}, \mathsf{P}) \quad \texttt{Trained}(\mathsf{AWS}, \mathsf{D}, \mathsf{P}, \mathsf{NN})}{\texttt{Certified}(\mathsf{NN})}$$

As the predicates have been signed by the respective authorities, we can trace exactly which entities where collectively responsible for establishing the fact that the NN is certified.

Now, the system S is certified if it contains a fairness monitor and is based on a NN which has been certified. We establish this fact as follows:

$$\frac{\texttt{HasMonitor}(\mathsf{T\ddot{U}V}, \mathsf{S}) \quad \texttt{BasedOn}(\mathsf{T\ddot{U}V}, \mathsf{S}, \mathsf{NN}) \quad \texttt{Certified}(\mathsf{NN})}{\texttt{Certified}(\mathsf{S})}$$

Finally, we establish that the decision is trustable:

$$\frac{\texttt{Produces}(\mathsf{Azure}, \mathsf{S}, \mathsf{RejectA}) \quad \texttt{Certified}(\mathsf{S})}{\texttt{Trustable}(\mathsf{RejectA})}$$

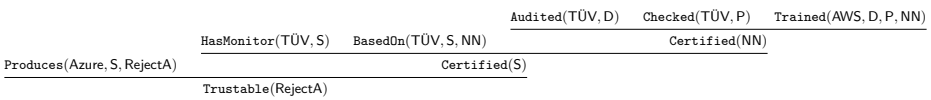

Fig. 2. Reasoning example.

Combining the individual derivations, we obtain the tree shown in Fig. 2.

So, the decision is indeed trustable based on the provided facts and rules. The inference tree allows us to exactly trace which parties have been involved in the decision and are accountable for which parts of the system used to reach the decision. For instance, if it turns out that the system running was not actually the system S, then Azure would be accountable, and if it turns out that there was not actually a fairness monitor, then TÜV would be accountable. For the enduser, e.g., a rejected applicant, this information could then be visualized such that they are able to understand it. If the number of facts and rules used for an inference tree becomes bigger and bigger, the comprehensibility of a visualization will suffer. In this case, we would need a mechanism similar to *signature-based proof condensation*, which is already used for the visualization of proofs in description logic [3]. The idea is to automatically hide the sub-proofs for concepts that are well-known for the user. In our example, if NN had a global reputation for its fairness, the display of the inference tree would stop at `Certified(`NN, with an option to "unfold" it, if the rejected applicant remains sceptical.

4.3 Rules Encoding

As for facts, we also propose a standard for encoding rules. Our DBOM allows us to bind facts or assertions and signatures together. For the signatures of rules, we plan to use a similar format. However, we do not really need to define the argument types for the rules' predicates; they are resulting from the fact's definitions. Consequently, we do not need an inner and an outer schema.

The example in Lst. 1.2 shows how the first rule from Sec. 4.2 would look like when signed in the proposed format. For the rules, one might also use fixed constants as arguments for our predicates instead of variables. In this case, the `variable` has to be replaced with `value`

5 Conclusion and future work

To ensure traceability and accountability of AI systems and their decisions, we proposed an approach centered around a Decision Bill of Materials (DBOM). A DBOM documents all elements contributing to a decision and the parties that have been involved in it. By equipping a DBOM with cryptographic signatures, accountability for the stated facts is ensured. This not only can ensure that any decision of the AI system is rooted in verified and audited components that

```
{
  "rules": [
    {
      "head": {
        "name": "certified",
        "arguments": [
          { "variable": "neuralNetwork" }
        ]
      },
      "body": [
        {
          "name": "audited",
          "arguments": [
            { "variable": "auditor1" },
            { "variable": "database" }
          ]
        },
        {
          "name": "checked",
          "arguments": [
            { "variable": "auditor2" },
            { "variable": "program" }
          ]
        },
        {
          "name": "trained",
          "arguments": [
            { "variable": "provider" },
            { "variable": "database" },
            { "variable": "program" },
            { "variable": "neuralNetwork" }
          ]
        }
      ]
    }
  ],
  "signature": {
    "algorithm": "PS512",
    "value": "P2TJK3WROL..."
  }
}
```

Listing 1.2. Encoding of the first rule from sec. 4.2

have been used as intended, but also allows for tracking and auditing additional operational and runtime aspects. As discussed, a host of existing techniques can be used in conjunction with DBOMs, either by merely recording their usage or by producing additional information such as explanations.

As next steps, we plan an implementation of our proposed requirement checker that is able to process DBOMs and rulesets. In this context, we are also thinking about accepting existing certificate formats such as X.509 as *Cert* predicates. Another potential add-on would be the automatic generation of parts of the DBOM. As an example, the *Trained* predicate might be generated automatically using a signing mechanism provided by the cloud operator. This would allow to quickly adapt the evolving parts of the decision making process, whereas fundamental facts and rules will remain subject to manual certification.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Adebayo, J., Gilmer, J., Goodfellow, I.J., Kim, B.: Local explanation methods for deep neural networks lack sensitivity to parameter values. ICLR Workshop (2018)
3. Alrabbaa, C., Baader, F., Borgwardt, S., Dachselt, R., Koopmann, P., Méndez, J.: Evonne: Interactive proof visualization for description logics (system description). In: Blanchette, J., Kovács, L., Pattinson, D. (eds.) Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13385, pp. 271–280. Springer (2022). https://doi.org/10.1007/978-3-031-10769-6_16, https://doi.org/10.1007/978-3-031-10769-6_16
4. Alur, R., Fisman, D., Raghothaman, M.: Regular programming for quantitative properties of data streams. In: Thiemann, P. (ed.) Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9632, pp. 15–40. Springer (2016). https://doi.org/10.1007/978-3-662-49498-1_2, https://doi.org/10.1007/978-3-662-49498-1_2
5. Andrews, R., Diederich, J., Tickle, A.B.: Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-based systems **8**(6), 373–389 (1995)
6. Angwin, J., Larson, J., Mattu, S., Kirchner, L.: Machine Bias (2016), https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing
7. Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C.L., Parikh, D.: Vqa: Visual question answering. In: Proceedings of the IEEE international conference on computer vision. pp. 2425–2433 (2015)
8. Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Müller, K.: How to explain individual classification decisions. J. MLR (2010)
9. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
10. Barringer, H., Goldberg, A., Havelund, K., Sen, K.: Rule-based runtime verification. In: Steffen, B., Levi, G. (eds.) Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Venice, Italy, January 11-13, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2937, pp. 44–57. Springer (2004). https://doi.org/10.1007/978-3-540-24622-0_5, https://doi.org/10.1007/978-3-540-24622-0_5
11. Bauer, A., Leucker, M., Schallhart, C.: Monitoring of real-time properties. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4337, pp. 260–272. Springer (2006). https://doi.org/10.1007/11944836_25, https://doi.org/10.1007/11944836_25
12. Bien, J., Tibshirani, R.: Prototype selection for interpretable classification. Ann. Appl. Statist (2011)
13. Biewer, S., Baum, K., Sterz, S., Hermanns, H., Hetmank, S., Langer, M., Lauber-Rönsberg, A., Lehr, F.: Software doping analysis for human oversight. Formal Methods in System Design pp. 1–50 (2024)

14. Bloem, R., Könighofer, B., Könighofer, R., Wang, C.: Shield synthesis: - runtime enforcement for reactive systems. In: Baier, C., Tinelli, C. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9035, pp. 533–548. Springer (2015). https://doi.org/10.1007/978-3-662-46681-0_51, https://doi.org/10.1007/978-3-662-46681-0_51
15. Boz, O.: Extracting decision trees from trained neural networks. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 456–461 (2002)
16. Burke, L.: The Death and Life of an Admissions Algorithm (2020), https://www.insidehighered.com/admissions/article/2020/12/14/u-texas-will-stop-using-controversial-algorithm-evaluate-phd
17. Chang, C., Creager, E., Goldenberg, A., Duvenaud, D.: Explaining image classifiers by counterfactual generation. ICLR (2019)
18. Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In: WACV (2018)
19. Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., Su, J.K.: This looks like that: deep learning for interpretable image recognition. Advances in neural information processing systems **32** (2019)
20. Chou, Y.L., Moreira, C., Bruza, P., Ouyang, C., Jorge, J.: Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications. Information Fusion **81**, 59–83 (2022)
21. Chouldechova, A.: Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. Big Data **5**(2), 153–163 (2017). https://doi.org/10.1089/big.2016.0047, https://doi.org/10.1089/big.2016.0047
22. Craven, M.W., Shavlik, J.W.: Using sampling and queries to extract rules from trained neural networks. In: Machine learning proceedings 1994, pp. 37–45. Elsevier (1994)
23. CycloneDX: Machine learning bill of materials (ml-bom), https://web.archive.org/web/20240303083218/https://cyclonedx.org/capabilities/mlbom/
24. Dabkowski, P., Gal, Y.: Real time image saliency for black box classifiers. NIPS (2017)
25. 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
26. Dressel, J., Farid, H.: The accuracy, fairness, and limits of predicting recidivism. Science advances **4**(1), eaao5580 (2018)
27. European Parliament and Council of the EU: Regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence and amending regulations (ec) no 300/2008, (eu) no 167/2013, (eu) no 168/2013, (eu) 2018/858, (eu) 2018/1139 and (eu) 2019/2144 and directives 2014/90/eu, (eu) 2016/797 and (eu) 2020/1828 (artificial intelligence act) (2024), https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L_202401689

28. Falcone, Y., Mounier, L., Fernandez, J., Richier, J.: Runtime enforcement monitors: composition, synthesis, and enforcement abilities. Formal Methods Syst. Des. **38**(3), 223–262 (2011). https://doi.org/10.1007/S10703-011-0114-4, https://doi.org/10.1007/s10703-011-0114-4
29. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: ICCV (2017)
30. Foundation, T.L.: Spdx ai, https://web.archive.org/web/20240405072139/https://spdx.dev/learn/areas-of-interest/ai/
31. Gallaire, H., Minker, J., Nicolas, J.M.: Logic and databases: A deductive approach. ACM Computing Surveys (CSUR) **16**(2), 153–185 (1984)
32. Gros, T.P., Hermanns, H., Hoffmann, J., Klauck, M., Steinmetz, M.: Deep statistical model checking. In: Gotsman, A., Sokolova, A. (eds.) Formal Techniques for Distributed Objects, Components, and Systems - 40th IFIP WG 6.1 International Conference, FORTE 2020, Held as Part of the 15th International Federated Conference on Distributed Computing Techniques, DisCoTec 2020, Valletta, Malta, June 15-19, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12136, pp. 96–114. Springer (2020). https://doi.org/10.1007/978-3-030-50086-3_6, https://doi.org/10.1007/978-3-030-50086-3_6
33. Heaven, W.D.: Predictive policing algorithms are racist. They need to be dismantled. (2020), https://www.technologyreview.com/2020/07/17/1005396/predictive-policing-algorithms-racist-dismantled-machine-learning-bias-criminal-justice/
34. Jacovi, A., Marasović, A., Miller, T., Goldberg, Y.: Formalizing trust in artificial intelligence: Prerequisites, causes and goals of human trust in ai. In: Proceedings of the 2021 ACM conference on fairness, accountability, and transparency. pp. 624–635 (2021)
35. Jansen, N., Könighofer, B., Junges, S., Serban, A., Bloem, R.: Safe reinforcement learning using probabilistic shields (invited paper). In: Konnov, I., Kovács, L. (eds.) 31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference). LIPIcs, vol. 171, pp. 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPICS.CONCUR.2020.3, https://doi.org/10.4230/LIPIcs.CONCUR.2020.3
36. Kästner, L., Langer, M., Lazar, V., Schomäcker, A., Speith, T., Sterz, S.: On the relation of trust and explainability: Why to engineer for trustworthiness. In: 2021 IEEE 29th international requirements engineering conference workshops (REW). pp. 169–175. IEEE (2021)
37. Kaur, D., Uslu, S., Rittichier, K.J., Durresi, A.: Trustworthy artificial intelligence: a review. ACM computing surveys (CSUR) **55**(2), 1–38 (2022)
38. Langer, M., Oster, D., Speith, T., Hermanns, H., Kästner, L., Schmidt, E., Sesing, A., Baum, K.: What do we want from explainable artificial intelligence (xai)? a stakeholder perspective on xai and a conceptual model guiding interdisciplinary xai research. Artificial Intelligence **296**, 103473 (2021). https://doi.org/https://doi.org/10.1016/j.artint.2021.103473, https://www.sciencedirect.com/science/article/pii/S0004370221000242
39. Larson, J., Mattu, S., Kirchner, L., Angwin, J.: How We Analyzed the COMPAS Recidivism Algorithm (2016), https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm
40. Leucker, M., Schallhart, C.: A brief account of runtime verification. J. Log. Algebraic Methods Program. **78**(5), 293–303 (2009). https://doi.org/10.1016/j.jlap.2008.08.004, https://doi.org/10.1016/j.jlap.2008.08.004

41. Li, O., Liu, H., Chen, C., Rudin, C.: Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
42. Liang, W., Tadesse, G.A., Ho, D., Fei-Fei, L., Zaharia, M., Zhang, C., Zou, J.: Advances, challenges and opportunities in creating data for trustworthy ai. Nature Machine Intelligence **4**(8), 669–677 (2022)
43. Ligatti, J., Bauer, L., Walker, D.: Run-time enforcement of nonsafety policies. ACM Trans. Inf. Syst. Secur. **12**(3), 19:1–19:41 (2009). https://doi.org/10.1145/1455526.1455532, https://doi.org/10.1145/1455526.1455532
44. Lu, J., Yang, J., Batra, D., Parikh, D.: Hierarchical question-image co-attention for visual question answering. Advances in neural information processing systems **29** (2016)
45. Maier, D., Tekle, K.T., Kifer, M., Warren, D.S.: Datalog: concepts, history, and outlook, p. 3100. ACM (Sep 2018). https://doi.org/10.1145/3191315.3191317, http://dx.doi.org/10.1145/3191315.3191317
46. Meurrens, S.: The Increasing Role of AI in Visa Processing (2021), https://canadianimmigrant.ca/immigrate/immigration-law/the-increasing-role-of-ai-in-visa-processing
47. O'Neil, C.: How algorithms rule our working lives (2016), https://www.theguardian.com/science/2016/sep/01/how-algorithms-rule-our-working-lives, Online; accessed: 2023-06-23
48. Orcale: AI in human resources: The time is now (2019), https://www.oracle.com/a/ocom/docs/applications/hcm/oracle-ai-in-hr-wp.pdf
49. Organisation for Economic Co-operation and Development (OECD): Artificial intelligence, machine learning and big data in finance: Opportunities, challenges and implications for policy makers. Tech. rep., OECD, [Paris] : (2021), https://www.oecd.org/finance/financial-markets/Artificial-intelligence-machine-learning-big-data-in-finance.pdf
50. Petsiuk, V., Das, A., Saenko, K.: Rise: Randomized input sampling for explanation of black-box models. BMVC (2018)
51. Phang, J., Park, J., Geras, K.J.: Investigating and simplifying masking-based saliency methods for model interpretability. arXiv preprint arXiv:2010.09750 (2020)
52. Pinisetty, S., Preoteasa, V., Tripakis, S., Jéron, T., Falcone, Y., Marchand, H.: Predictive runtime enforcement. In: Ossowski, S. (ed.) Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016. pp. 1628–1633. ACM (2016). https://doi.org/10.1145/2851613.2851827, https://doi.org/10.1145/2851613.2851827
53. Pinisetty, S., Roop, P.S., Smyth, S., Allen, N., Tripakis, S., von Hanxleden, R.: Runtime enforcement of cyber-physical systems. ACM Trans. Embed. Comput. Syst. **16**(5s), 178:1–178:25 (2017). https://doi.org/10.1145/3126500, https://doi.org/10.1145/3126500
54. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?": Explaining the predictions of any classifier. In: SIGKDD. KDD '16 (2016)
55. Schlicker, N., Langer, M.: Towards warranted trust: A model on the relation between actual and perceived system trustworthiness. In: Proceedings of Mensch und Computer 2021. pp. 325–329 (2021)
56. Scholz, B., Jordan, H., Suboti, P., Westmann, T.: On fast large-scale program analysis in datalog. In: Proceedings of the 25th International Conference on Compiler Construction. CGO 16, ACM (Mar 2016). https://doi.org/10.1145/2892208.2892226, http://dx.doi.org/10.1145/2892208.2892226

57. Schulz, K., Sixt, L., Tombari, F., Landgraf, T.: Restricting the flow: Information bottlenecks for attribution. arXiv preprint arXiv:2001.00396 (2020)
58. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: CVPR (2017)
59. Sicre, R., Zhang, H., Dejasmin, J., Daaloul, C., Ayache, S., Artières, T.: Dp-net: Learning discriminative parts for image recognition. In: 2023 IEEE International Conference on Image Processing (ICIP). pp. 1230–1234. IEEE (2023)
60. Smith, E., Vogell, H.: How Your Shadow Credit Score Could Decide Whether You Get an Apartment (2021), https://www.propublica.org/article/how-your-shadow-credit-score-could-decide-whether-you-get-an-apartment, Online; accessed: 2023-06-23
61. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: The all convolutional net. ICLR (2015)
62. Stepin, I., Alonso, J.M., Catala, A., Pereira-Fariña, M.: A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. IEEE Access **9**, 11974–12001 (2021)
63. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: ICML (2017)
64. Tickle, A.B., Andrews, R., Golea, M., Diederich, J.: The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. IEEE Transactions on Neural Networks **9**(6), 1057–1068 (1998)
65. Verma, S., Dickerson, J., Hines, K.: Counterfactual explanations for machine learning: A review. arXiv preprint arXiv:2010.10596 **2**, 1 (2020)
66. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)
67. Warren, D.H.: Applied logic: its use and implementation as a programming tool (1978)
68. Waters, A., Miikkulainen, R.: Grade: Machine learning support for graduate admissions. AI Magazine **35**(1), 64 (Mar 2014). https://doi.org/10.1609/aimag.v35i1.2504, https://ojs.aaai.org/index.php/aimagazine/article/view/2504
69. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems **35**, 24824–24837 (2022)
70. Wu, C., Liu, J., Wang, X., Dong, X.: Chain of reasoning for visual question answering. Advances in Neural Information Processing Systems **31** (2018)
71. Yang, M., Shkapsky, A., Zaniolo, C.: Scaling up the performance of more powerful datalog systems on multicore machines. The VLDB Journal **26**(2), 229248 (Dec 2016). https://doi.org/10.1007/s00778-016-0448-z, http://dx.doi.org/10.1007/s00778-016-0448-z
72. Yu, Z., He, L., Wu, Z., Dai, X., Chen, J.: Towards better chain-of-thought prompting strategies: A survey. arXiv preprint arXiv:2310.04959 (2023)
73. Zhang, H., Torres, F., Sicre, R., Avrithis, Y., Ayache, S.: Opti-cam: Optimizing saliency maps for interpretability. arXiv preprint arXiv:2301.07002 (2023)
74. Zolna, K., Geras, K.J., Cho, K.: Classifier-agnostic saliency map extraction. CVIU **196**, 102969 (2020)