Journal of Logic, Language and Information (2023) 32:63–88
https://doi.org/10.1007/s10849-022-09377-8

On the (Complete) Reasons Behind Decisions

Adnan Darwiche[1] · **Auguste Hirth**[1]

Accepted: 1 July 2022 / Published online: 18 August 2022

Abstract
Recent work has shown that the input-output behavior of some common machine learning classifiers can be captured in symbolic form, allowing one to reason about the behavior of these classifiers using symbolic techniques. This includes explaining decisions, measuring robustness, and proving formal properties of machine learning classifiers by reasoning about the corresponding symbolic classifiers. In this work, we present a theory for unveiling the *reasons* behind the decisions made by Boolean classifiers and study some of its theoretical and practical implications. At the core of our theory is the notion of a *complete reason,* which can be viewed as a necessary and sufficient condition for why a decision was made. We show how the complete reason can be used for computing notions such as sufficient reasons (also known as PI-explanations and abductive explanations), how it can be used for determining decision and classifier bias and how it can be used for evaluating counterfactual statements such as "a decision will stick even if …because … ." We present a linear-time algorithm for computing the complete reasoning behind a decision, assuming the classifier is represented by a Boolean circuit of appropriate form. We then show how the computed complete reason can be used to answer many queries about a decision in linear or polynomial time. We finally conclude with a case study that illustrates the various notions and techniques we introduced.

Keywords Explainable AI · Boolean classifiers · Tractable circuits

This work is an extended version of Darwiche and Hirth (2020) and has been partially supported by NSF grant #ISS-1910317, ONR grant #N00014-18-1-2561, DARPA XAI grant #N66001-17-2-4032 and a gift from JP Morgan.

✉ Adnan Darwiche
darwiche@cs.ucla.edu

Auguste Hirth
ahirth@g.ucla.edu

[1] Computer Science Department, University of California, Los Angeles, USA

1 Introduction

Consider Fig. 1 which depicts how most machine learning systems are constructed today. We have a labeled dataset that is used to learn a classifier, which is commonly a neural network, a Bayesian network or a random forest. These classifiers are effectively functions that map instances to decisions. For example, an instance could be a loan application and the decision is whether to approve or decline the loan. There is now considerable interest in reasoning about the behavior of such systems. Explaining decisions is at the forefront of current interests: Why did you decline Maya's application? Quantifying the robustness of these decisions is also attracting a lot of attention: Would reversing the decision on Maya require many changes to her application? In some domains, one expects the learned classifiers to satisfy certain properties, like monotonicity, and there is again an interest in proving such properties formally. For example, can we guarantee that a loan applicant will be approved when the only difference they have with another approved applicant is their higher income? These interests, however, are challenged by the numeric nature of machine learning classifiers and the fact that these systems are often model-free, e.g., neural networks, so they appear as black boxes that are hard to analyze.

Even though these machine learning classifiers are learned from data and are numeric in nature, they often implement discrete decision functions. One can therefore extract these functions and represent them symbolically. The outcome of this process is normally a logical formula or a Boolean circuit that precisely captures the input-output behavior of the learned classifier, which can then be used to reason about its behavior, symbolically. This includes explaining decisions, measuring robustness and formally proving properties. For a concrete example, consider Fig. 2 which depicts one of the simplest machine learning systems: a Naive Bayes classifier. We have a class variable P and three features B, U and S. Given an instance (patient) and their test results b, u and s, this classifier renders a decision by computing the posterior probability $Pr(p|b, u, s)$ and then checking whether it passes a given threshold T. If it does, we declare a positive decision; otherwise, a negative decision. While this classifier is numeric and its decisions are based on probabilistic reasoning, it does induce a discrete decision function. In fact, the function is Boolean in this case as it maps the Boolean variables B, U and S, which correspond to test results, into a binary decision (yes or no). This observation was originally made in Chan and Darwiche (2003), which proposed the compilation of Naïve Bayes classifiers into symbolic decision graphs as shown in Fig. 2. The compilation process guarantees that for every instance, the decision made by the (probabilistic) Naïve Bayes classifier is identical to the one made by the (symbolic) decision graph. This compilation algorithm was recently extended to Bayesian network classifiers with tree structures (Shih et al. 2018) and later to Bayesian network classifiers with arbitrary structures (Shih et al. 2019a).[1] Certain classes of neural networks can also be compiled into, or reasoned about using, decision diagrams as shown in Shih et al. (2019b), Shi et al. (2020). While Bayesian and neural networks are numeric in nature, random forests are not (at least the ones with majority voting). Hence, one can easily encode their input-output behavior using

[1] See http://reasoning.cs.ucla.edu/xai/ for related software.

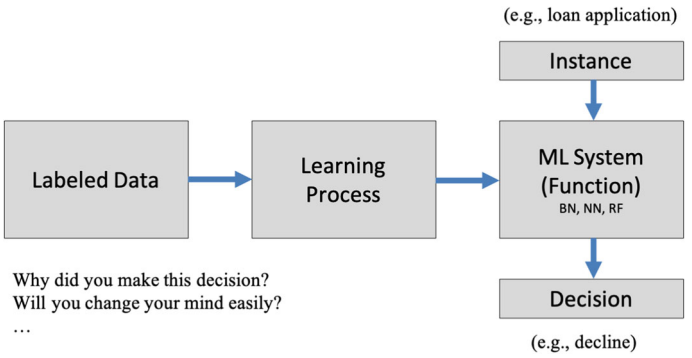

Fig. 1 Reasoning about machine learning classifiers

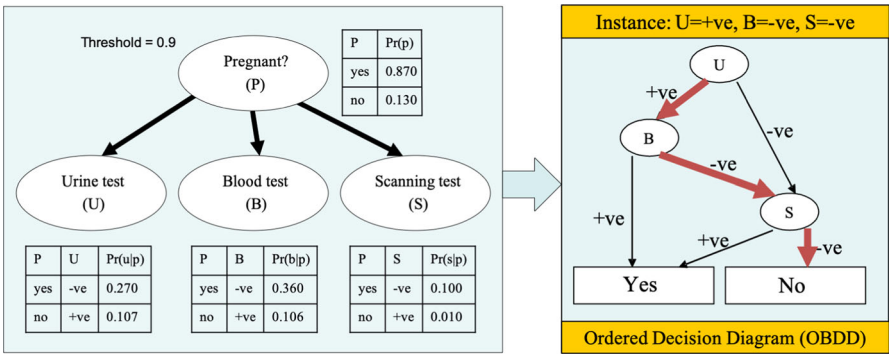

Fig. 2 Compiling a Naïve Bayes classifier into a symbolic decision graph. To classify an instance using the decision graph, we start at the root node and repeat the following. If the feature we are at is positive, we follow the left edge, otherwise we follow the left edge. We finally reach a leaf node, which determines the class of the given instance. The figure shows the path followed from the root to a leaf (no) for the instance $(\Leftarrow B, -ve), (\Leftarrow U, +ve)$ and $(\Leftarrow S, -ve)$

Boolean formulas. Since a random forest is an ensemble of decision trees, we first encode each decision tree into a Boolean formula. This is straightforward even in the presence of continuous variables as the learning algorithm discretizes variables by identifying a set of thresholds for each variable.[2] We then combine these formulas using a majority formula or circuit; see, e.g., (Audemard et al. 2020; Choi et al. 2020).

This methodology for reasoning about the behavior of machine learning classifiers has three dimension: (1) the kind of machine learning classifiers are we reasoning about, (2) the symbolic representation we use to encode their input-output behavior, and (3) the class of queries we are interested in and how to compute them efficiently. We will not concern ourselves with the first dimension in this paper as we will assume that the input-output behavior has already been encoded symbolically. Hence, our discussion will be orthogonal to where the symbolic representation came from. As to the second dimension, one can encode input-output behavior using standard logical

[2] Each path in the decision tree can be represented by a conjunction of variable values (term). The instances of a particular class can then be represented by a disjunction of corresponding terms (DNF).

formulas, which is the approach we shall pursue. While logical formulas are sufficient for our treatment as far as semantics is involved, we will use a particular class of logical representations for computational reasons: tractable Boolean circuits (Darwiche and Marquis 2002).[3] As for the third dimension, we will focus on developing a theory for reasoning about the decisions made by classifiers: What are the reasons behind them? How can they counterfactually change? And are they biased?

In the proposed theory, a *classifier* is a Boolean function. Its variables are called *features,* a particular input is called an *instance,* and the function output on some instance is called a *decision.* If the function outputs 1 on an instance, the instance and decision are said to be *positive;* otherwise, they are *negative.* Our main goal is to *explain* the decisions made by Boolean classifiers on specific instances by way of providing various insights into what caused these decisions. For some examples, consider Fig. 3 which depicts two classifiers ($\mathscr{C}_1$ and $\mathscr{C}_2$) for college admission, represented as Ordered Binary Decision Diagrams (OBDDs) (Bryant 1986) (in which variables are binary and ordered similarly on any path from the root to a leaf).[4] Consider also Susan who passed the entrance exam, is a first-time applicant, has no work experience and a high GPA. Susan will be admitted by classifier $\mathscr{C}_1$. She also comes from a rich hometown and will be admitted by classifier $\mathscr{C}_2$. We can say that Susan was admitted by classifier $\mathscr{C}_1$ *because* she passed the entrance exam and has a high GPA. We can also say that *one reason why* classifier $\mathscr{C}_2$ admitted Susan is that she passed the entrance exam and has a high GPA (there are other reasons in this case). Moreover, we can say that classifier $\mathscr{C}_2$ *would still* admit Susan *even if* she did not have a high GPA *because* she passed the entrance exam and comes from a rich hometown. Finally, we can say that classifier $\mathscr{C}_2$ can make biased decisions: ones that are based on *protected* features. For example, it will make different decisions on two applicants who have the same characteristics except that one comes from a rich hometown and the other does not. We will also show that one can sometimes prove classifier bias by inspecting the reasons behind one of its unbiased decisions. We will give formal definitions and semantics for the statements exemplified above and show how to evaluate them algorithmically and efficiently. As far as semantics, the main tool we will employ is Boolean logic and particularly the classical notion of prime implicants (Crama and Hammer 2011; Quine 1952; McCluskey 1956; Quine 1959). On the computational side, we will exploit tractable Boolean circuits as mentioned earlier (Darwiche and Marquis 2002), while providing some new fundamental results that further extend the reach of these circuits to computing explanations. At the core of our theory is the notion of *complete reason* behind a decision, which can be viewed as a necessary and sufficient condition for why the decision was made. Most of what we shall discuss will be based on complete reasons, both semantically and computationally.

[3] See Darwiche (2020) for a recent survey/tutorial on tractable Boolean circuits and their applications in AI. An alternative set of approaches abstract the machine learning classifier into symbolic form and reason about its behavior using SAT-based or SMT-based techniques; see, e.g., (Katz et al. 2017; Leofante et al. 2018; Narodytska et al. 2018; Ignatiev et al. 2019a).

[4] OBDDs are one example of tractable Boolean circuits. They can be unfolded into Boolean circuits in linear time (Darwiche and Marquis 2002). Moreover, they allow some hard queries to be answered in polynomial time Bryant (1986). Ordered decision diagrams do not have to be binary. For example, the algorithm of Chan and Darwiche (2003) compiled naïve Bayes classifiers into ordered decision diagrams with discrete variables.

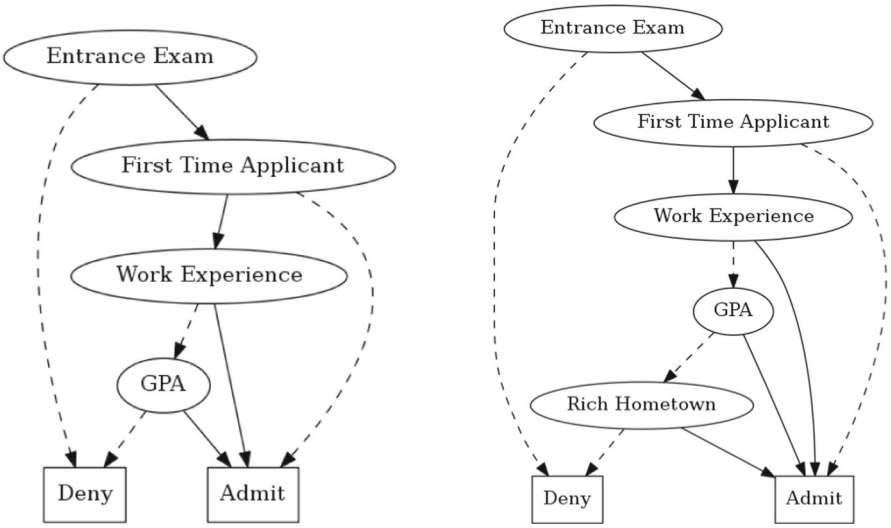

Fig. 3 Two OBDD classifiers: $\mathscr{C}_1$ (left) and $\mathscr{C}_2$ (right). Solid edges represent true values of a variable. Dotted edges represent false values

This paper is structured as follows. We start in Sect. 2 by reviewing some Boolean logic preliminaries including prime implicants. We then introduce the notion of complete reason and related notions such as sufficient reasons and necessary characteristics in Sects. 3–5. Counterfactual statements about decisions are discussed in Sect. 6, followed by a discussion of decision and classifier bias in Sect. 7. We dedicate Sect. 8 to algorithms that compute the introduced notions and then illustrate them in Sect. 9 using a case study. We finally close with some concluding remarks in Sect. 10.

2 Boolean Logic Preliminaries

A *literal* is a Boolean variable X (positive literal) or its negation $\neg X$ (negative literal). A *term* is a consistent conjunction of literals (e.g., $A \wedge \neg B \wedge C$). A *Disjunctive Normal Form (CNF)* is a disjunction of terms (e.g., $(A \wedge \neg B) \vee (B \wedge C) \vee (\neg A \wedge C \wedge D)$). An *instance* is a term that includes precisely one literal for each Boolean variable. Term τ_i *subsumes* term τ_j iff $\tau_j \models \tau_i$, where $\models$ denotes logical entailment. For example, term $E \wedge \neg F$ subsumes term $E \wedge \neg F \wedge G$. We treat a term as the *set* of its literals so we may write $\tau_i \subseteq \tau_j$ to also mean that τ_i subsumes τ_j. We will often refer to a literal as a *characteristic* and to a term τ as a *property* (of an instance that contains the term). We use $\overline{\tau}$ to denote the property resulting from negating every characteristic in property τ. We sometimes use a comma (,) instead of a conjunction ($\wedge$) when describing properties and instances (e.g., $E, \neg F$ instead of $E \wedge \neg F$).

We represent a classifier by a Boolean formula Δ whose models (i.e., satisfying assignments) correspond to positive instances. The negation of the formula characterizes negative instances. Classifiers $\mathscr{C}_1$ and $\mathscr{C}_2$ in Fig. 3 are represented by the following formulas:

$$\Delta_1 = E \wedge (\neg F \vee G \vee W)$$
$$\Delta_2 = E \wedge (\neg F \vee G \vee W \vee R)$$

We use $\Delta(\alpha)$ to denote the decision (0 or 1) of classifier Δ on instance α. That is, $\Delta(\alpha) = 1$ iff $\alpha \models \Delta$ and $\Delta(\alpha) = 0$ iff $\alpha \models \neg\Delta$. We also define $\Delta_\alpha = \Delta$ if the decision is positive and $\Delta_\alpha = \neg\Delta$ if the decision is negative. This notation is critical and will be used frequently later. By definition, for any two instances α and β, we have $\Delta_\alpha = \Delta_\beta$ iff $\alpha \models \Delta_\alpha$ and $\Delta(\alpha) = \Delta(\beta)$. Again, we use this observation frequently later.

An *implicant* τ of Boolean formula Δ is a term that satisfies Δ, $\tau \models \Delta$. A *prime implicant* is an implicant that is not subsumed by any other implicant. For example, $E \wedge \neg F \wedge G$ is an implicant of Δ_1 but is not prime since it is subsumed by another implicant $E \wedge \neg F$, which happens to be prime. Classifier $\mathscr{C}_1$ has the following prime implicants:

$$\Delta_1 : (E \wedge \neg F)\ \ (E \wedge G)\ \ (E \wedge W)$$
$$\neg\Delta_1 : (\neg E)\ \ (F \wedge \neg G \wedge \neg W)$$

Classifier $\mathscr{C}_2$ has the following prime implicants:

$$\Delta_2 : (E \wedge \neg F)\ \ (E \wedge G)\ \ (E \wedge W)\ \ (E \wedge R)$$
$$\neg\Delta_2 : (\neg E)\ \ (F \wedge \neg G \wedge \neg W \wedge \neg R)$$

The set of prime implicants for a Boolean formula can be quite large, which motivated the notion of a *prime implicant cover* (Quine 1952; McCluskey 1956; Quine 1959). A set of terms $\tau_1, \ldots, \tau_n$ is prime implicant cover for Boolean formula Δ if each term τ_i is a prime implicant of Δ and $\tau_1 \vee \ldots \vee \tau_n$ is equivalent to Δ. A cover may not include all prime implicants, with the missing ones called *redundant*. While covers can be useful computationally, they may not always be appropriate for explaining classifiers as they may lead to incomplete explanations (more on this later).

We will make use of the *conditioning* operation on Boolean formulas. To condition formula Δ on term τ, denoted $\Delta|\tau$, is to replace every literal l in Δ with constant 1 (true) if $l \in \tau$, and to replace it with constant 0 (false) if $\neg l \in \tau$. For example, if $\alpha = (A \vee \neg B) \wedge (C \vee D)$ and $\tau = B, \neg C$, then $\alpha|\tau = (A \vee \neg 1) \wedge (0 \vee D) = A \wedge D$. We will also use the *existential quantification* operation: $\exists X \Delta = (\Delta|X) \vee (\Delta|\neg X)$.

In the next few sections, we introduce notions such as the sufficient and complete reasons behind a decision. We use these notions later to define decision and classifier bias in addition to giving semantics to counterfactual statements relating to decisions.

3 Sufficient Reasons

Prime implicants have been studied and utilized extensively in the AI and computer science literature.[5] However, their active utilization in explaining decisions is more

[5] One classical application of prime implicants in AI has been in the area of model-based diagnosis, where they have been used to formalize the notion of *kernel diagnoses* (de Kleer et al. 1992). A kernel diagnosis is

recent, e.g., (Shih et al. 2018; Ignatiev et al. 2019a, b; Lindner and Möllney 2019). This recent utilization introduced a key connection to properties of instances that we highlight next and exploit computationally later.

Definition 1 (**Sufficient Reason** (Shih et al. 2018)) A *sufficient reason* for decision $\Delta(\alpha)$ is a property of instance α that is also a prime implicant of Δ_α (recall $\Delta_\alpha = \Delta$ if the decision is positive and $\Delta_\alpha = \neg\Delta$ if the decision is negative).

A sufficient reason identifies characteristics of an instance that justify the decision: The decision will stick even if other characteristics of the instance were different. A sufficient reason is minimal: None of its strict subsets can justify the decision. A decision can have multiple sufficient reasons, sometimes a large number of them.[6] There is a key difference between prime implicants and sufficient reasons as the latter must be properties of the given instance. This has significant computational implications that we exploit in Sect. 8.

Sufficient reasons were introduced in Shih et al. (2018) under the name of *PI-explanations*. They were also referred to as *abductive explanations* in Ignatiev et al. (2019a).[7] The new name we adopt is motivated by further distinctions that we draw later and was also used in Lindner and Möllney (2019). We will sometimes say "a reason" to mean "a sufficient reason."

Greg passed the entrance exam, is not a first time applicant, does not have a high GPA but has work experience ($\alpha = E, \neg F, \neg G, W$). Classifier $\mathscr{C}_1$ admits Greg, a decision that can be explained using either of the following sufficient reasons:

- Passed the entrance exam and is not a first time applicant ($E, \neg F$).
- Passed the entrance exam and has work experience (E, W).

Since Greg passed the entrance exam and has applied before, he will be admitted even if his other characteristics were different. Similarly, since Greg passed the entrance exam and has work experience, he will be admitted even if his other characteristics were different.

Proposition 1 *Every decision has at least one sufficient reason.*

Proof Consider decision $\Delta(\alpha)$. We have $\alpha \models \Delta_\alpha$, which means Δ_α is consistent and must have at least one prime implicant (the empty term if Δ_α is valid). Moreover, at least one of these prime implicants must be a property of instance α since $\alpha \models \Delta_\alpha$ and since Δ_α is equivalent to the disjunction of its prime implicants. Hence, we have at least one sufficient reason for the decision. □

defined for a given device behavior and is a minimal term representing the health of some device components. Any system state that is compatible with a kernel diagnosis is feasible under the given system behavior. Moreover, the set of kernel diagnoses characterize all feasible system states under the given behavior.

[6] The popular Anchor system (Ribeiro et al. 2018) can be viewed as computing approximations of sufficient reasons. The quality of these approximations has been evaluated on some datasets and corresponding classifiers in Ignatiev et al. (2019c), where an approximation is called *optimistic* if it is a strict subset of a sufficient reason and *pessimistic* if it is a strict superset of a sufficient reason.

[7] In contrast to *contrastive explanations* which were formalized in Ignatiev et al. (2020) based on Miller (2019). Contrastive explanations can be thought of as answering "why not" queries in contrast to the "why" queries answered by abductive explanations. A minimal hitting set duality between abductive and contrastive explanations was also shown in Ignatiev et al. (2020).

A classifier may make the same decision on two instances but for different reasons (i.e., disjoint sufficient reasons). However, if two decisions on distinct instances share a reason, they must be equal.

Proposition 2 *If decisions $\Delta(\alpha)$ and $\Delta(\beta)$ share a sufficient reason, the decisions must be equal $\Delta(\alpha) = \Delta(\beta)$.*

Proof Suppose the decisions share sufficient reason τ. Then τ is property of both α and β and τ is a prime implicant of both Δ_α and Δ_β. Hence, $\Delta_\alpha = \Delta_\beta$ since τ is consistent and $\Delta(\alpha) = \Delta(\beta)$. □

We will see later that sufficient reasons can provide insights about a classifier that go well beyond explaining its decisions.

4 Complete Reasons

A sufficient reason identifies a minimal property of an instance that can trigger a decision. The *complete reason* behind a decision characterizes all properties of an instance that can trigger the decision.

Definition 2 (**Complete Reason**) The *complete reason* for a decision is the disjunction of all its sufficient reasons.

The complete reason for decision $\Delta(\alpha)$ captures *every* property of instance α, and *only* properties of instance α, that can trigger the decision. It precisely captures why this particular decision was made on instance α.

Theorem 1 *Let $\mathscr{R}$ be the complete reason for decision $\Delta(\alpha)$. If instance β does not satisfy $\mathscr{R}$ and $\Delta(\beta) = \Delta(\alpha)$, then no sufficient reason for decision $\Delta(\beta)$ can be a property of instance α.*

Proof Suppose $\beta \not\models \mathscr{R}$ and $\Delta(\beta) = \Delta(\alpha)$. Then $\Delta_\beta = \Delta_\alpha$. Let τ be a sufficient reason for decision $\Delta(\beta)$. Then τ is a property of instance β and a prime implicant of both Δ_β and Δ_α. If τ were a property of instance α, then τ is a sufficient reason for decision $\Delta(\alpha)$, $\tau \models \mathscr{R}$ and $\beta \models \tau \models \mathscr{R}$, a contradiction. Hence, τ cannot be a property of instance α. □

We will sometimes say "the reason" to mean "the complete reason." Recall that we also say "a reason" to mean "a sufficient reason." According to Theorem 1, if the same decision is made on instances α and β, and if instance β does not satisfy the complete reason for decision $\Delta(\alpha)$, then these decisions were made for different reasons.

Classifier $\mathscr{C}_1$ admits Greg ($\alpha = E, \neg F, \neg G, W$) for the reason $\mathscr{R} = E \wedge (\neg F \vee W)$. Greg was admitted because he passed the entrance exam and satisfied one of two additional requirements: he applied before and has work experience. Classifier $\mathscr{C}_1$ also admits Susan ($\beta = E, F, G, \neg W$). Susan does not satisfy the reason $\mathscr{R}$. There is one sufficient reason for admitting Susan: she passed the entrance exam and has a good GPA (E, G), which is not a property of Greg. Hence, classifier $\mathscr{C}_1$ admitted Greg and Susan for different reasons.

The complete reason behind a decision is unique up to logical equivalence and can be used to enumerate all of the decision's sufficient reasons.

Theorem 2 *Let $\mathscr{R}$ be the complete reason for decision $\Delta(\alpha)$. The prime implicants of $\mathscr{R}$ are the sufficient reasons for decision $\Delta(\alpha)$.*

Proof Let $\tau_1, \ldots, \tau_n$ be the sufficient reasons for decision $\Delta(\alpha)$ and hence $\mathscr{R} = \tau_1 \vee \ldots \vee \tau_n$. The key observation is that each term τ_i is a property of instance α. Hence, for every two terms τ_i and τ_j, if term τ_i contains some literal X then term τ_j cannot contain literal $\neg X$. The DNF $\tau_1 \vee \ldots \vee \tau_n$ is then closed under consensus.[8] Since no term τ_i subsumes another term τ_j, the DNF $\tau_1 \vee \ldots \vee \tau_n$ contains all prime implicants of $\mathscr{R}$. Hence, the prime implicants of complete reason $\mathscr{R}$ are precisely the sufficient reasons of decision $\Delta(\alpha)$. □

We will later use Theorem 2 to provide a new approach for enumerating sufficient reasons, compared to earlier approaches such as those reported in Shih et al. (2018), Ignatiev et al. (2019a).

We will close this section by further highlighting how the complete reason for a decision can be viewed as a necessary and sufficient condition for explaining the decision. Consider the complete reason $\mathscr{R}$ for decision $\Delta(\alpha)$ and recall that it characterizes all properties of instance α that can trigger the decision: $\mathscr{R} \equiv \bigvee_{\tau \models \Delta_\alpha} \tau$, where τ is a property of instance α. The reason $\mathscr{R}$ is then a logical condition that triggers the decision ($\mathscr{R} \models \Delta_\alpha$). If the complete reason is weakened into a condition $\mathscr{R}_w$ that continues to trigger the decision ($\mathscr{R} \models \mathscr{R}_w \models \Delta_\alpha$), then $\mathscr{R}_w$ will admit properties not satisfied by instance α. Moreover, if it is strengthened into a condition $\mathscr{R}_s$, then $\mathscr{R}_s$ is guaranteed to trigger the decision ($\mathscr{R}_s \models \mathscr{R} \models \Delta_\alpha$) but will not admit some properties of instance α that can trigger the decision. Hence, the complete reason $\mathscr{R}$ is a necessary and sufficient condition for explaining the decision on instance α.

5 Necessary Characteristics and Properties

The *necessary property* of a decision is a maximal property of an instance that is essential for explaining the decision on that instance.

Definition 3 (**Necessary Characteristics and Properties**) A *characteristic is necessary* for a decision if it appears in every sufficient reason for the decision. The *necessary property* for a decision is the set of all its necessary characteristics.

The necessary property is unique but could be empty (when the decision has no necessary characteristics). If an instance ceases to satisfy one necessary characteristic, the corresponding decision is guaranteed to change.

Proposition 3 *If instance β disagrees with instance α on only one characteristic necessary for decision $\Delta(\alpha)$, then $\Delta(\alpha) \neq \Delta(\beta)$.*

[8] The consensus rule infers the term $\delta_1 \wedge \delta_2$ from terms $X \wedge \delta_1$ and $\neg X \wedge \delta_2$. One can convert a DNF into its set of prime implicants by closing the DNF under consensus and then removing subsumed terms; see (Crama and Hammer 2011, Chapter 3).

Proof Suppose α and β are as premised. If $\Delta(\alpha) = \Delta(\beta)$ then $\Delta_\alpha = \Delta_\beta$ and $\tau = \alpha \cap \beta$ is an implicant of Δ_α by consensus on the flipped characteristic ρ. Moreover, τ does not contain characteristic ρ so it cannot be necessary, a contradiction. □

If an instance ceases to satisfy more than one necessary characteristic, the decision does not necessarily change. However, if the decision sticks then it would be for completely different reasons.

Theorem 3 *Let β be an instance that disagrees with instance α on at least one characteristic necessary for decision $\Delta(\alpha)$. Decisions $\Delta(\alpha)$ and $\Delta(\beta)$ must have disjoint sufficient reasons.*

Proof Let σ be the necessary characteristics of decision $\Delta(\alpha)$ that instances α and β disagree on. A sufficient reason τ of $\Delta(\alpha)$ cannot be a property of instance β since $\sigma \subseteq \tau$ and β contains $\overline{\sigma}$. Hence, τ cannot be a sufficient reason for decision $\Delta(\beta)$ and the two decisions must have disjoint sufficient reasons. □

Consider a classifier $\Delta = (X \wedge Y \wedge Z) \vee (\neg X \wedge \neg Y \wedge Z)$ and instance $\alpha = X, Y, Z$. The decision $\Delta(\alpha)$ is positive with X, Y, Z as the only sufficient reason. Hence, all three characteristics of α are necessary: Flipping any single characteristic of instance α will lead to a negative decision. However, flipping the two characteristics X and Y preserves the positive decision but leads to a new, single sufficient reason $\neg X, \neg Y, Z$.

The complete reason for a decision has enough information to compute its necessary characteristics and property.

Proposition 4 *A characteristic is necessary for a decision iff it is implied by the decision's complete reason.*

Proof Follows from Definition 3 and Theorem 2. □

6 Decision Counterfactuals

We mentioned Susan earlier who passed the entrance exam, is a first time applicant, has a high GPA but no work experience ($\alpha = E, F, G, \neg W$). Classifier $\mathscr{C}_1$ admits Susan *because* she passed the entrance exam and has a high GPA as this is the only sufficient reason for the decision. Greg was also admitted by this classifier. His application is similar to Susan's except that he applied before and has work experience ($\beta = E, \neg F, G, W$). The decision on Greg has multiple sufficient reasons so we cannot issue a "because" statement when explaining this decision.

Definition 4 (**Because**) Consider decision $\Delta(\alpha)$ and property τ of instance α. We say the *decision is made because* τ if τ is the only sufficient reason for the decision.

Proposition 5 *Consider decision $\Delta(\alpha)$ and property τ of instance α. The decision is made because τ iff τ is the decision's complete reason.*

Proof Follows from Definitions 1 and 2. □

One may be interested in statements that provide insights into a decision beyond the reasons behind it. For example, we may want to know how the classifier may have decided an instance if some of its characteristics were to be different. An example of this is the statement we mentioned in Sect. 1 with regards to classifier $\mathscr{C}_2$: Susan would still be admitted even if she did not have a high GPA because she comes from a rich hometown and passed the entrance exam. This statement exemplifies counterfactuals of the following form: The decision will stick even if $\overline{\rho}$ because τ, where ρ and τ are properties of the given instance. Recall that $\overline{\rho}$ is the property which results from flipping every characteristic in property ρ.

Definition 5 (**Even-If-Because**) Consider decision $\Delta(\alpha)$ and properties ρ and τ of instance α. We say *the decision sticks even if* $\overline{\rho}$ *because* τ if τ is the complete reason for decision $\Delta(\beta)$, where instance β is the result of replacing property ρ in instance α with property $\overline{\rho}$.

The following result justifies the above definition.

Theorem 4 *Suppose decision* $\Delta(\alpha)$ *sticks even if* $\overline{\rho}$ *because* τ*, and let instance* β *be the result of replacing property* ρ *in instance* α *with* $\overline{\rho}$*. Then* $\Delta(\beta) = \Delta(\alpha)$*. Moreover,* τ *is the only sufficient reason for decision* $\Delta(\beta)$ *and must be disjoint from* ρ*.*

Proof Suppose decision $\Delta(\alpha)$ sticks even if $\overline{\rho}$ because τ, and let β be the described instance. By Definition 5, τ is the complete reason for decision $\Delta(\beta)$. Since τ is a property, it must be the only sufficient reason for decision $\Delta(\beta)$ by Theorem 2. Hence, τ is a property of instance β and must therefore be disjoint from property ρ since flipping the characteristics of ρ in instance α left property τ intact. Since property τ justifies decision $\Delta(\beta)$, $\tau \models \Delta_\beta$, and since τ is also a property of instance α, $\alpha \models \tau$, we now have $\alpha \models \tau \models \Delta_\beta$ and therefore $\Delta(\beta) = \Delta(\alpha)$. □

Applicant Susan who we discussed earlier ($\alpha = E, F, G, \neg W, R$) is admitted by classifier $\mathscr{C}_2$. The decision will stick even if Susan had a low GPA ($\neg G$) because she comes from a rich hometown and passed the entrance exam (E, R). This statement is justified since E, R is the complete reason for decision $\Delta(\beta)$. Here, $\beta = E, F, \neg G, \neg W, R$ is the result of replacing characteristic G by $\neg G$ in instance α.

Jackie did not pass the entrance exam, is not a first time applicant, has a low GPA but has work experience ($\alpha = \neg E, \neg F, \neg G, W$). Jackie is denied admission by classifier $\mathscr{C}_1$. The decision will stick even if Jackie had a high GPA (G) because she did not pass the entrance exam ($\neg E$). This statement is justified since $\neg E$ is the complete reason for decision $\Delta(\beta)$, where $\beta = \neg E, \neg F, G, W$ is the result of replacing characteristic $\neg G$ by G in instance α.

7 Decision Bias and Classifier Bias

We will now discuss the dependence of decisions on certain features, with a particular application to detecting decision and classifier bias.

Intuitively, a decision is *biased* if it depends on some *protected features:* ones that should not be used when making the decision (e.g., gender, zip code, or ethnicity).[9] We formalize bias next while making a distinction between classifier bias and decision bias. A classifier is biased if it makes some biased decisions, yet some of the other decisions it makes may still be unbiased. While classifier bias can always be detected by examining its decision function, we will show that it can sometimes be detected by examining the complete reason behind one of its unbiased decisions.

Definition 6 (**Decision Bias**) Decision $\Delta(\alpha)$ is *biased* if $\Delta(\alpha) \neq \Delta(\beta)$ for some β that disagrees with α on only protected features.

Bias can be positive or negative. For example, an applicant may be admitted because they come from a rich hometown, or may be denied admission because they did not come from a rich hometown. The following result provides a necessary and sufficient condition for detecting decision bias.

Theorem 5 *A decision is biased iff each of its sufficient reasons contains at least one protected feature.*

Proof We will show both directions of the theorem next.

Suppose decision $\Delta(\alpha)$ is biased yet has a sufficient reason τ with no protected features. We will now show a contradiction. Since the decision is biased, there must exist an instance β that disagrees with instance α on only protected features and $\Delta(\alpha) \neq \Delta(\beta)$. Since τ is a property of α and β, we have $\alpha \models \tau \models \Delta_\alpha$ and $\beta \models \tau \models \Delta_\alpha$. Hence, $\Delta_\alpha = \Delta_\beta$ and $\Delta(\alpha) = \Delta(\beta)$, which is a contradiction.

Suppose every sufficient reason for decision $\Delta(\alpha)$ contains at least one protected feature. Let $\mathbf{X}$ be these protected features and let τ be the characteristics of instance α that do not involve features $\mathbf{X}$. Assume $\Delta(\alpha) = \Delta(\beta)$ for every instance β that agrees with instance α on characteristics τ (that is, β disagrees with α only on the protected features $\mathbf{X}$). Term τ must then be an implicant of Δ_α and a subset σ of τ must be a prime implicant of Δ_α (could be τ itself). Since τ is a property of instance α, decision $\Delta(\alpha)$ has sufficient reason σ that does not include a protected feature in $\mathbf{X}$, which is a contradiction. Hence, $\Delta(\alpha) \neq \Delta(\beta)$ for some instance β that disagrees with instance α on only protected features in $\mathbf{X}$, and decision $\Delta(\alpha)$ is biased. □

We emphasize that Theorem 5 does not require sufficient reasons to share protected features, only that each must contain at least one protected feature.

Consider classifier $\mathscr{C}_3$, which admits applicants who have a good GPA (G) as long as they pass the entrance exam (E), are male (M) or come from a rich hometown (R):

$$\Delta_3 = (G \wedge E) \vee (G \wedge M) \vee (G \wedge R). \tag{1}$$

Bob has a good GPA, did not pass the entrance exam and comes from a rich hometown ($\alpha = G, \neg E, M, R$). He is admitted with two sufficient reasons: G, M and G, R. The decision is biased since each sufficient reason contains a protected feature (M and R). This classifier will not admit Nancy who has similar characteristics but does not come

[9] A protected feature may have been unprotected during classifier design.

from a rich hometown: $\beta = G, \neg E, \neg M, \neg R$. It will also admit Scott who has the same characteristics as Nancy: $\gamma = G, \neg E, M, \neg R$.

Even though this classifier is biased, some of its decisions may be unbiased. If an applicant has a good GPA and passes the entrance exam (G, E), they will be admitted regardless of their protected characteristics. Moreover, if an applicant does not have a good GPA ($\neg G$), they will be denied admission regardless of their other characteristics, including protected ones.

Definition 7 (**Classifier Bias**) A classifier is *biased* if at least one of its decisions is biased.

We emphasize again that a biased classifier may still make some unbiased decisions. As we show next, one can sometimes infer classifier bias by inspecting the sufficient reasons behind one of its unbiased decisions.

Theorem 6 *A classifier is biased iff one of its decisions has a sufficient reason that includes at least one protected feature.*

Proof We will next show both directions of the theorem.

Suppose classifier Δ is biased. By Definition 7, some decision $\Delta(\alpha)$ is biased. By Theorem 5, every sufficient reason of decision $\Delta(\alpha)$ must contain at least one protected feature.

Suppose decision $\Delta(\alpha)$ has a sufficient reason τ that contains protected features $\mathbf{X} \neq \emptyset$. For any instance β such that $\beta \models \tau$, we must have $\Delta(\beta) = \Delta(\alpha)$. We now show that there is an instance $\beta \models \tau$ and instance γ that disagrees with β on only features $\mathbf{X}$ such that $\Delta(\beta) \neq \Delta(\gamma)$. Suppose the contrary: for all such β and γ, we have $\Delta(\beta) = \Delta(\gamma) = \Delta(\alpha)$. Then $\tau \setminus \rho$ is an implicant of Δ_α, where ρ are the protected characteristics in τ. This is impossible since τ is a prime implicant of Δ_α. Hence, $\Delta(\beta) \neq \Delta(\gamma)$ for some β and γ with the stated properties so the classifier is biased. □

If decision $\Delta(\alpha)$ has protected features in some but not all of its sufficient reasons, the decision is not biased according to Theorem 5. But classifier Δ is biased according to Theorem 6 as it will make a biased decision on some other instance $\beta \neq \alpha$.

Consider classifier $\mathscr{C}_3$ in (1) and Lisa who has a good GPA, passed the entrance exam and comes from a rich hometown ($G, E, \neg M, R$). The classifier will admit Lisa for two sufficient reasons: G, E and G, R. The decision is unbiased: any applicant who has similar unprotected characteristics will be admitted. However, since one of the sufficient reasons contains a protected feature, the classifier is biased as it can make a biased decision on a different applicant. The proof of Theorem 6 suggests that the classifier will make different decisions on two applicants with a good GPA who disagree only on whether they come from a rich hometown. Nancy ($G, \neg E, \neg M, \neg R$) and Heather ($G, \neg E, \neg M, R$) are such applicants.

The following theorem shows how one can detect decision bias using the complete reason behind the decision. We will use this theorem (and Theorem 8) when discussing algorithms in Sect. 8.

Theorem 7 *A decision is biased iff* $\exists(X_1, \ldots, X_n)\mathscr{R}$ *is not valid where* $X_1, \ldots, X_n$ *are all unprotected features and* $\mathscr{R}$ *is the complete reason behind the decision.*

Proof Let $\tau_1, \ldots, \tau_n$ be the decision's sufficient reasons and hence $\mathscr{R} = \tau_1 \vee \ldots \vee \tau_n$. Existentially quantifying variables X_i from a DNF is done by replacing their literals with 1. The result is valid iff some term τ_i contains only variables in $X_1, \ldots, X_n$. Hence, $\exists X_1, \ldots, X_n \mathscr{R}$ is not valid iff each term τ_i contains variables beyond X_i (i.e., each sufficient reason contains protected features). □

The following result shows how classifier bias can sometimes be detected based on the complete reason behind an unbiased decision.

Theorem 8 *A classifier is biased if $\mathscr{R}|X \not\equiv \mathscr{R}|\neg X$ where X is a protected feature and $\mathscr{R}$ is the complete reason for some decision.*

Proof Given Theorems 2 and 6, it is sufficient to show that $\mathscr{R}|X \not\equiv \mathscr{R}|\neg X$ iff feature X appears in some prime implicant of $\mathscr{R}$. Let $\tau_1, \ldots, \tau_n$ be the prime implicants of $\mathscr{R}$. Feature X appears either positively or negatively in these prime implicants since terms τ_i are all properties of the same instance. Suppose without loss of generality that feature X appears positively in terms τ_i (if any). Then $\mathscr{R}|X \equiv \bigvee_{X \notin \tau_i} \tau_i \vee \bigvee_{X \in \tau_i} \tau_i \setminus \{X\}$ and $\mathscr{R}|\neg X \equiv \bigvee_{X \notin \tau_i} \tau_i$. Hence $\mathscr{R}|X \not\equiv \mathscr{R}|\neg X$ iff $X \in \tau_i$ for some prime implicant τ_i. □

Theorem 8 follows from Theorems 2 and 6 and a known result: A Boolean function depends on a variable X iff X appears in one of its prime implicants. We included the full proof for completeness.

8 Computing Reasons and Related Queries

The enumeration of PI-explanations (sufficient reasons) was treated in Shih et al. (2018) by modifying the algorithm in Coudert and Madre (1993) for computing prime implicant covers; see also (Coudert et al. 1993; Minato 1993). The modified algorithm optimizes the original one by integrating the instance into the prime implicant enumeration process, but we are unaware of a complexity bound for the original algorithm or its modification. Moreover, since the algorithm is based on prime implicant covers, it is incomplete. Consider classifier $\Delta = (X \wedge Z) \vee (Y \wedge \neg Z)$, which has three prime implicants: $(X \wedge Z)$, $(Y \wedge \neg Z)$ and $(X \wedge Y)$. The last prime implicant is redundant and may not be generated when computing a cover. Instance $\alpha = X, Y, Z$ leads to a positive decision and two sufficient reasons: $(X \wedge Z)$ and $(X \wedge Y)$. An algorithm based on covers may miss the sufficient reason $(X \wedge Y)$ and is therefore incomplete. This can be problematic for queries that rely on examining all sufficient reasons, such as decision and classifier bias (Definitions 6 and 7).

We next propose a new approach based on computing the complete reason $\mathscr{R}$ for a decision (Definition 2), which characterizes all sufficient reasons, and then use it to compute multiple queries. For example, we can enumerate all sufficient reasons using the reason $\mathscr{R}$ (Theorem 2). We can also use it to compute necessary characteristics (Proposition 4) and to detect decision bias (Theorem 7). Even classifier bias can sometimes be inferred directly using the reason $\mathscr{R}$ (Theorem 8) among other queries.

Assuming the classifier is represented using a suitable tractable Boolean circuit, our approach will compute the complete reason for a decision in linear time regardless

of how many sufficient reasons it may have (could be exponential). Moreover, it will ensure that the computed complete reason is represented by a tractable circuit, allowing us to answer many queries in polytime.

8.1 Computing Complete Reasons

Our approach for computing complete reasons requires the classifier Δ and its negation $\neg\Delta$ to be represented as Decision-DNNF circuits, which we define next.

Definition 8 (**Decision-DNNF Circuit**) An *NNF circuit* is a Boolean circuit that has literals or constants as inputs and two type of gates: and-gates and or-gates. A *DNNF circuit* is an NNF circuit in which the subcircuits feeding into each and-gate share no variables.[10] A *Decision-DNNF circuit* is a DNNF circuit in which every or-gate has exactly two inputs of the form: $X \wedge \mu$ and $\neg X \wedge \nu$, where X is a variable.[11]

DNNF circuits were introduced in Darwiche (2001). Decision-DNNF circuits were identified in Huang and Darwiche (2005, 2007). OBDDs which we discussed earlier are a subset of Decision-DNNF circuits as one can convert an OBDD into a Decision-DNNF circuit in linear time. Figure 4 depicts an OBDD and its corresponding Decision-DNNF circuit. The circuit is obtained by mapping each OBDD node with variable X, high child μ and low child ν into the circuit fragment $(X \wedge \mu) \vee (\neg X \wedge \nu)$ (two and-gates and one or-gate). For more on Decision-DNNF circuits and OBDD, see (Bryant 1986; Darwiche and Marquis 2002; Huang and Darwiche 2007; Oztok and Darwiche 2014). One can obtain a Decision-DNNF circuit by compiling a Boolean formula in Conjunctive Normal Form (CNF) using systems such as C2D[12] (Darwiche 2004) and D4[13] (Lagniez and Marquis 2017). One can also compile an OBDD from any Boolean formula using systems such as CUDD.[14]

We compute the complete reason for a decision $\Delta(\alpha)$ by applying two operations to a Decision-DNNF circuit for Δ_α: *consensus* then *filtering*.

Definition 9 (**Consensus Circuit**) The *consensus circuit* of Decision-DNNF circuit Γ is denoted $\mathsf{consensus}(\Gamma)$ and obtained by adding input $\mu \wedge \nu$ to every or-gate with inputs $X \wedge \mu$ and $\neg X \wedge \nu$.

Figure 4 depicts a Decision-DNNF circuit and its consensus circuit (third from left). The consensus operation adds four and-gates denoted with double circles.

Proposition 6 *A Decision-DNNF circuit Γ has the same satisfying assignments as its consensus circuit* $\mathsf{consensus}(\Gamma)$.

Proof $(X \wedge \mu) \vee (\neg X \wedge \nu) \equiv (X \wedge \mu) \vee (\neg X \wedge \nu) \vee (\mu \wedge \nu)$. □

[10] This is called the *decomposability* property.

[11] This is called the *decision* property.

[12] http://reasoning.cs.ucla.edu/c2d/

[13] http://www.cril.univ-artois.fr/kc/d4.html

[14] http://vlsi.colorado.edu/personal/fabio/CUDD/

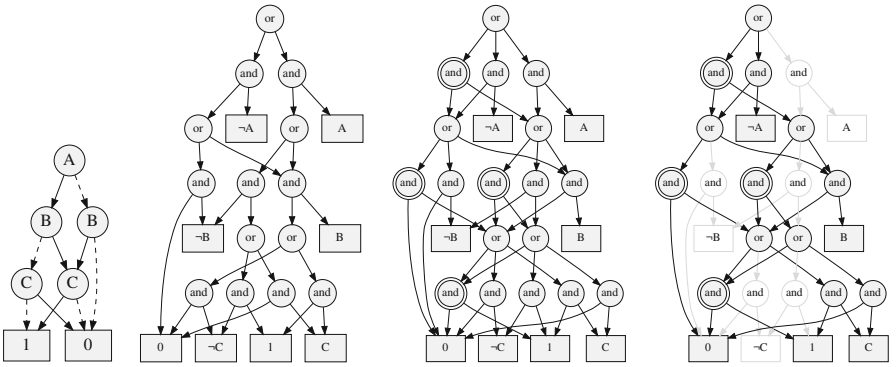

Fig. 4 From left to right: OBDD, Decision-DNNF circuit, consensus circuit, and the filtering of consensus circuit by instance $\neg A, B, C$

A consensus circuit can be obtained from a Decision-DNNF circuit in time linear. We next discuss the filtering of a consensus circuit, which leads to a tractable circuit.

Definition 10 (**Filtered Circuit**) The *filtering* of consensus circuit Γ by instance α, where $\Gamma(\alpha) = 1$, is denoted $\mathsf{filter}(\Gamma, \alpha)$ and obtained by replacing every literal $l \notin \alpha$ by constant 0.

Filtering is defined only on consensus circuits and requires an instance that satisfies the consensus circuit (we are only interested in such instances). Figure 4 depicts an example. The filtered circuit is on the far right of the figure, where grayed out nodes and edges can be dropped due to replacing literals by constant 0.

Filtering is also a linear time operation. Consensus preserves models (i.e., satisfying assignments of the circuit), but filtering drops some of them. We will characterize the models preserved by filtering after presenting two required results.

Let Γ be a circuit that results from filtering by instance α. The circuit is monotone in the following sense. If the common literals between instances α and β are a subset of the common literals between instances α and γ, then $\beta \models \Gamma$ only if $\gamma \models \Gamma$. For example, if $\alpha = X, Y, Z$, $\beta = \neg X, Y, \neg Z$ and $\gamma = \neg X, Y, Z$, then α and β agree on literals $\{Y\}$ while α and γ agree on literals $\{Y, Z\}$ so the condition is met in this case.

Theorem 9 *If circuit Γ results from filtering by instance α then every literal l that appears in Γ also appears in α. Moreover, $\Gamma(\gamma) \geq \Gamma(\beta)$ if $\gamma \cap \alpha \supseteq \beta \cap \alpha$.*

Proof Filtering removes every literal not in instance α. Hence, every literal in the filtered circuit Γ is in α, which implies the next result. Suppose that $\gamma \cap \alpha \supseteq \beta \cap \alpha$ and $\Gamma(\beta) = 1$. When evaluating circuit Γ at γ compared to β, the only literals that change values are $l_1 \in \gamma \setminus \beta$ and $l_2 \in \beta \setminus \gamma$. Literals l_1 change values from 0 to 1 and literals l_2 change values from 1 to 0. Changes to the values of l_1 cannot decrease the output of circuit Γ since it is an NNF circuit. Literals l_2 are not in α since $\gamma \cap \alpha \supseteq \beta \cap \alpha$ so do not appear in circuit Γ and changes to their values do not matter. Hence, $\Gamma(\gamma) = 1$. □

We also need the following result which identifies circuit models that are preserved by the filtering of a consensus circuit.

Proposition 7 *Consider a Decision-DNNF circuit Δ and instance α such that $\Delta(\alpha) = 1$. If τ is an implicant of Δ and $\alpha \models \tau$ then τ is also an implicant of* filter(consensus(Δ), α).

Proof Let $\Gamma = \mathsf{filter}(\mathsf{consensus}(\Delta), \alpha)$, $\mathscr{I}(\Delta) = \{\tau : \tau \models \Delta\}$ and $\mathscr{I}(\Delta, \alpha) = \{\tau : \tau \models \Delta \text{ and } \alpha \models \tau\}$. We need to show that $\mathscr{I}(\Delta, \alpha) \subseteq \mathscr{I}(\Gamma)$. That is, Γ preserves the implicants τ of Δ satisfied by α. The proof is by induction on the structure of Δ.

(Base Case) If Δ is a literal l or a constant, then $\Delta = \Gamma$ since consensus is not applicable and filtering will not replace literal l by constant 0 ($l \in \alpha$ since $\Delta(\alpha) = 1$). Hence, $\mathscr{I}(\Delta, \alpha) \subseteq \mathscr{I}(\Gamma)$.

(Inductive Step) If $\Delta = \Delta_1 \wedge \Delta_2$ then $\Gamma = \Gamma_1 \wedge \Gamma_2$ where $\Gamma_1 = \mathsf{filter}(\mathsf{consensus}(\Delta_1), \alpha)$ and $\Gamma_2 = \mathsf{filter}(\mathsf{consensus}(\Delta_2), \alpha)$. Since Δ_1 and Δ_2 share no variables (decomposability), $\mathscr{I}(\Delta) = \mathscr{I}(\Delta_1) \times \mathscr{I}(\Delta_2)$ (Cartesian product). Similarly, $\mathscr{I}(\Gamma) = \mathscr{I}(\Gamma_1) \times \mathscr{I}(\Gamma_2)$. By the induction hypothesis, $\mathscr{I}(\Delta_1, \alpha) \subseteq \mathscr{I}(\Gamma_1)$ and $\mathscr{I}(\Delta_2, \alpha) \subseteq \mathscr{I}(\Gamma_2)$. Hence,

$$\mathscr{I}(\Delta, \alpha) = \mathscr{I}(\Delta_1, \alpha) \times \mathscr{I}(\Delta_2, \alpha) \subseteq \mathscr{I}(\Gamma_1) \times \mathscr{I}(\Gamma_2) = \mathscr{I}(\Gamma).$$

(Inductive Step) If $\Delta = (l \wedge \Delta_1) \vee (\neg l \wedge \Delta_2)$ and literal $l \in \alpha$ then $\Gamma = (l \wedge \Gamma_1) \vee (\Gamma_1 \wedge \Gamma_2)$ where $\Gamma_1 = \mathsf{filter}(\mathsf{consensus}(\Delta_1), \alpha)$ and $\Gamma_2 = \mathsf{filter}(\mathsf{consensus}(\Delta_2), \alpha)$. Due to decomposability, l and $\neg l$ do not appear in Δ_1 or Δ_2. Hence, $\mathscr{I}(\Delta) = \mathscr{I}_1 \cup \mathscr{I}_2 \cup \mathscr{I}_c$ where

$$\begin{aligned}
\mathscr{I}_1 &= \{l, \tau : \tau \in \mathscr{I}(\Delta_1)\} \\
\mathscr{I}_2 &= \{\neg l, \tau : \tau \in \mathscr{I}(\Delta_2)\} \\
\mathscr{I}_c &= \mathscr{I}(\Delta_1 \wedge \Delta_2).
\end{aligned}$$

Since $\mathscr{I}_2 \cap \mathscr{I}(\Delta, \alpha) = \emptyset$ we have

$$\mathscr{I}(\Delta, \alpha) = \{l, \tau : \tau \in \mathscr{I}(\Delta_1, \alpha)\} \cup \mathscr{I}(\Delta_1 \wedge \Delta_2, \alpha).$$

Moreover, $\mathscr{I}(\Gamma) = \{l, \tau : \tau \in \mathscr{I}(\Gamma_1)\} \cup \mathscr{I}(\Gamma_1 \wedge \Gamma_2)$. By the induction hypothesis, $\mathscr{I}(\Delta_1, \alpha) \subseteq \mathscr{I}(\Gamma_1)$ and $\mathscr{I}(\Delta_2, \alpha) \subseteq \mathscr{I}(\Gamma_2)$, which gives $\{l, \tau : \tau \in \mathscr{I}(\Delta_1, \alpha)\} \subseteq \{l, \tau : \tau \in \mathscr{I}(\Gamma_1)\}$ and $\mathscr{I}(\Delta_1 \wedge \Delta_2, \alpha) \subseteq \mathscr{I}(\Gamma_1 \wedge \Gamma_2)$. Hence, $\mathscr{I}(\Delta, \alpha) \subseteq \mathscr{I}(\Gamma)$. □

The following fundamental result reveals the role of filtering a consensus circuit. It also reveals our linear-time procedure for computing the complete reason behind a decision as a (tractable) circuit that compactly characterizes all sufficient reasons.

Theorem 10 *Consider a Decision-DNNF circuit Δ and instance α such that $\Delta(\alpha) = 1$. Term τ is a prime implicant of Δ and $\alpha \models \tau$ (that is, τ is a sufficient reason for decision $\Delta(\alpha)$) iff τ is a prime implicant of* filter(consensus(Δ), α).

Proof Let $\Gamma = \mathsf{filter}(\mathsf{consensus}(\Delta), \alpha)$. Observe that $\Gamma \models \Delta$ since $\mathsf{consensus}(\Delta) \equiv \Delta$ and since Γ is the result of replacing some inputs of $\mathsf{consensus}(\Delta)$ with constant 0.

Suppose τ is a prime implicant of circuit Δ and $\alpha \models \tau$. Then τ is an implicant of circuit Γ by Proposition 7, $\tau \models \Gamma$. If τ is not a prime implicant of Γ, we must have some term $\rho \subset \tau$ such that $\rho \models \Gamma$. Therefore $\rho \models \Delta$ since $\Gamma \models \Delta$, which means that τ is not a prime implicant of Δ, a contradiction. Hence, τ is a prime implicant of Γ.

Suppose τ is a prime implicant of circuit Γ. Then τ is an implicant of Δ since $\Gamma \models \Delta$. We next show that τ is a prime implicant of Δ and $\alpha \models \tau$. Let β be an instance such that $\beta \models \tau$ and β disagrees with α on all variables outside τ. Then $\Gamma(\beta) = 1$ and $\alpha \cap \beta \subseteq \tau$. Every instance γ such that $\gamma \models \alpha \cap \beta$ must satisfy $\Gamma(\gamma) = 1$ since $\alpha \cap \gamma \supseteq \alpha \cap \beta$, leading to $\Gamma(\gamma) \geq \Gamma(\beta)$ by Theorem 9. Hence, $\alpha \cap \beta$ is an implicant of Γ. Since τ is a prime implicant of Γ, we must have $\alpha \cap \beta = \tau$ and hence $\alpha \models \tau$. Suppose now τ is not a prime implicant of Δ. Some term $\rho \subset \tau$ is then a prime implicant of Δ and $\alpha \models \rho$. By the first part of this theorem, ρ is a prime implicant of Γ, a contradiction. Therefore, τ is a prime implicant of Δ. □

This is our final definition in this section, which captures the computation of complete reasons using circuits.

Definition 11 (**Reason Circuit**) For classifier Δ, instance α and a Decision-DNNF circuit Γ for Δ_α, the circuit $\mathsf{filter}(\mathsf{consensus}(\Gamma), \alpha)$ is called a *reason circuit* and is denoted by $\mathsf{reason}(\Delta, \alpha)$.

The circuit $\mathsf{reason}(\Delta, \alpha)$ depends on the specific Decision-DNNF circuit Γ used to represent Δ_α but will always have the same models.

8.2 Tractability of Reason Circuits

We next show that reason circuits are tractable. Since we represent the complete reason for a decision as a reason circuit, many queries relating to the decision can then be answered efficiently.

Definition 12 (**Monotone**) An NNF circuit is *monotone* if every variable appears only positively or only negatively in the circuit.

Reason circuits are filtered circuits and hence monotone as shown by Theorem 9. The following theorem mirrors what is known about monotone Boolean formulas, but we include it for completeness.

Theorem 11 *The satisfiability of a monotone NNF circuit can be decided in linear time. A monotone NNF circuit can be negated and also conditioned in linear time to yield a monotone NNF circuit.*

Proof The satisfiability of a monotone NNF circuit can be decided using the following procedure. Constant 0 is not satisfiable. Constant 1 and literals are satisfiable. An or-gate is satisfiable iff any of its inputs is satisfiable. An and-gate is satisfiable iff all its inputs are satisfiable. All previous statements are always correct except the last one which depends on monotonicity. Consider a conjunction $\mu \wedge \nu$ and suppose every variable shared between the conjuncts appears either positively or negatively

Algorithm 1 PI(Δ, α)

input: Decision-DNNF circuit Δ and instance α such that $\Delta(\alpha) = 1$.
output: Prime implicants of circuit filter(consensus(Δ), α).

```
if cache(Δ) is set then
    return cache(Δ)
else if Δ is constant 0 then
    r = {}
else if Δ is constant 1 then
    r = {{}}
else if Δ = Δ1 ∧ Δ2 then
    r = cartesian_product(PI(Δ1, α), PI(Δ2, α))
else if Δ = (X ∧ Δ1) ∨ (¬X ∧ Δ2) then
    (ℓ, Γ) = (X, Δ1) if literal X in α else (¬X, Δ2)
    p = cartesian_product(PI(Δ1, α), PI(Δ2, α))
    q = {{ℓ} ∪ τ for τ ∈ PI(Γ, α)}
    r = p ∪ q
r = remove_subsumed(r)
cache(Δ) = r
return r
```

in both. Any model of μ can be combined with any model of ν to form a model for $\mu \wedge \nu$. Hence, the conjunction is satisfiable iff each of the conjuncts is satisfiable. Conditioning replaces literals by constants so it preserves monotonicity. To negate a monotone circuit, replace and-gates by or-gates, or-gates by and-gates and literals by their negations. Monotonicity is preserved. □

Given Theorem 11, the validity of a monotone NNF circuit can be decided in linear time (we check whether the negated circuit is unsatisfiable).[15] We can also conjoin the circuit with a literal in linear time to yield a monotone circuit since $\Delta \wedge l = (\Delta|l) \wedge l$.

Variables can be existentially quantified from a monotone circuit in linear time, with the resulting circuit remaining monotone. This is critical for efficiently detecting decision bias as shown by Theorem 7.

Theorem 12 *Replacing every literal of variable X with constant 1 in a monotone NNF circuit Γ yields a monotone NNF circuit equivalent to $\exists X\, \Gamma$.*

Proof If variable X appears only positively in circuit Γ then $\Gamma|\neg X \models \Gamma|X$ and $\exists X\ \Gamma = (\Gamma|X) \vee (\Gamma|\neg X) = \Gamma|X$. If variable X appears only negatively in Γ then $\Gamma|X \models \Gamma|\neg X$ and $\exists X\ \Gamma = (\Gamma|X) \vee (\Gamma|\neg X) = \Gamma|\neg X$. Variable X can therefore be existentially quantified by replacing its literals with constant 1. □

8.3 Computing Queries

We can now discuss algorithms. To compute the sufficient reasons for a decision $\Delta(\alpha)$: get a Decision-DNNF circuit for Δ_α, transform it into a consensus circuit, filter it by instance α and finally compute the prime implicants of filtered circuit. Algorithm 1

[15] Validity can be checked more directly as follows. Constant 1 is valid. Constant 0 and literals are not valid. An and-gate is valid iff all its inputs are valid. An or-gate is valid iff any of its inputs is valid. The previous statements are always correct except the last one which requires monotonicity.

does this in place, that is without explicitly constructing the consensus or filtered circuits. It assumes a positive decision (otherwise we pass $\neg\Delta$).

Algorithm 1 uses subroutine cartesian_product which conjoins two DNFs by computing the Cartesian product of their terms. It also uses remove_subsumed to remove subsumed terms from a DNF.

Theorem 13 *Consider a Decision-DNNF Δ and instance α. If $\Delta(\alpha) = 1$ then a call $\mathsf{PI}(\Delta, \alpha)$ to Algorithm 1 returns the prime implicants of circuit $\mathsf{filter}(\mathsf{consensus}(\Delta), \alpha)$.*

Proof Consensus and filtering are applied implicitly on Lines 10-11. Filtered circuit are monotone. We compute the prime implicants of a monotone circuit by converting it into DNF and removing subsumed terms (Crama and Hammer 2011, Chapter 3). This is precisely what Algorithm 1 does. □

Consider now a decision $\Delta(\alpha)$ and its complete reason $\mathscr{R} = \mathsf{reason}(\Delta, \alpha)$, which is a monotone NNF circuit. Let n be the size of circuit $\mathscr{R}$ and m be the number of features. We next show how to compute various queries using circuit $\mathscr{R}$.

Sufficient Reasons. By Theorems 2 and 13, the call $\mathsf{PI}(\Delta_\alpha, \alpha)$ to Algorithm 1 will return all sufficient reasons for decision $\Delta(\alpha)$, assuming Δ_α is a Decision-DNNF circuit. The number of sufficient reasons can be exponential, but we can actually answer many questions about them without enumerating them directly as shown below.

Necessary Property. By Proposition 4, characteristic (literal) l is necessary for decision $\Delta(\alpha)$ iff $\mathscr{R} \models l$. This is equivalent to $\mathscr{R}|\neg l$ being unsatisfiable, which can be decided in $O(n)$ time given Theorem 11. The necessary property (all necessary characteristics) can then be computed in $O(n \cdot m)$ time.

Because Statements. To decide whether decision $\Delta(\alpha)$ was made "because τ" we check whether property τ is the complete reason for the decision (Definition 4): $\tau \models \mathscr{R}$ and $\mathscr{R} \models \tau$. We have $\tau \models \mathscr{R}$ iff $\neg\mathscr{R}|\tau$ is unsatisfiable. Moreover, $\mathscr{R} \models \tau$ iff $\mathscr{R}|\neg l$ is unsatisfiable for every literal l in τ. All of this can be done in $O(n \cdot |\tau|)$ time.

Even if, Because Statements. To decide whether decision $\Delta(\alpha)$ would stick "even if $\overline{\rho}$ because τ" we replace property ρ with $\overline{\rho}$ in instance α to yield instance β (Definition 5). We then compute the complete reason for decision $\Delta(\beta)$ and check whether it is equivalent to τ. All of this can be done $O(n \cdot |\tau|)$ time.

Decision Bias. To decide whether decision $\Delta(\alpha)$ is biased we existentially quantify all unprotected features from circuit $\mathscr{R}$ and then check the validity of the result (Theorem 7). All of this can be done in $O(n)$ time given Theorems 11 and 12.

9 A Case Study

We now consider a more refined admission classifier to illustrate the notions and concepts we introduced more comprehensively.

This classifier highly values passing the entrance exam and being a first time applicant. However, it also gives significant leeway to students from a rich hometown. In fact, being from a rich hometown unlocks the only path to acceptance for those who failed the entrance exam. The classifier is depicted as an OBDD in Fig. 5. It corresponds to the following Boolean formula, which is not monotone (the previous

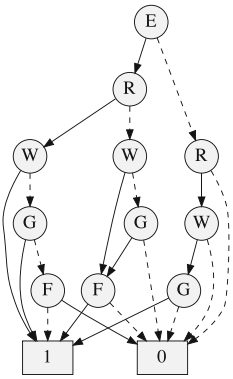

Fig. 5 Admission classifier

Applicant	Scott	Robin	April
Entrance Exam	✓	✓	✓
First Time Applicant	✗	✓	✓
Good GPA	✓	✓	✓
Work Experience	✓	✓	✓
Rich Hometown	✓	✓	✗
Decision	1	1	1

Fig. 6 Applicants and characteristics

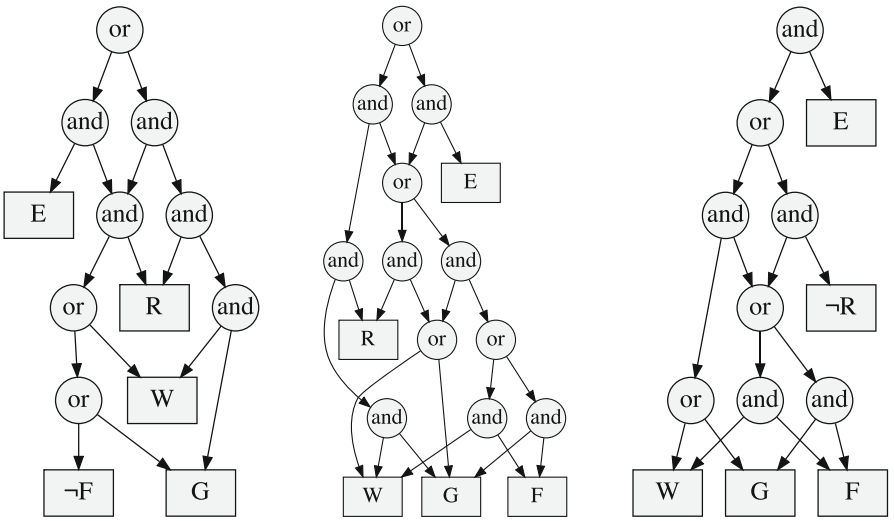

Fig. 7 From left to right: Reason circuit for the decision on applicants Scott, Robin and April (Fig. 6)

classifiers we considered were all monotone):

$$\Delta = [E \wedge [(F \wedge (G \vee W)) \vee (\neg F \wedge R)]] \vee [G \wedge R \wedge W].$$

The classifier has the following prime implicants, some are not essential (all prime implicants of a monotone formula are essential):

$$(E, F, W)(E, F, G)(G, R, W)(E, \neg F, R)(E, R, W)(E, G, R).$$

We will consider applicants Scott, Robin and April in Fig. 6, where feature R is protected (whether the applicant comes from a rich hometown). The complete reasons for the decisions on these applicants are shown in Fig. 7. These are reason circuits produced as suggested by Definition 11, except that we simplified the circuits by propagating and removing constant values (a reason circuit is satisfiable as it must be satisfied by the instance underlying the decision).

The decision on applicant Scott is *biased.* To check this, we can existentially quantify unprotected features E, F, G, W from the reason circuit in Fig. 7 and then check its validity (Theorem 7). Existential quantification is done by replacing the literals $E, \neg F, G, W$ in the circuit with constant 1. The resulting circuit is not valid. We can also confirm decision bias by considering the sufficient reasons for this decision, which all contain the protected feature R (Theorem 5):

$$(E, G, R)\ (E, R, W)\ (E, R, \neg F)\ (G, R, W)$$

If we flip the protected characteristic R to $\neg R$, the decision will flip with the complete reason being $\neg F, \neg R$ so Scott would be denied admission *because* he is not a first time applicant and does not come from a rich hometown (Definition 4).

The decision on Robin is *not biased.* If we existentially quantify unprotected features E, F, G, W from the reason circuit (by replacing their literals with constant 1), the circuit becomes valid. We can confirm this using the decision's sufficient reasons:

$$(E, F, G)\ (E, F, W)\ (E, G, R)\ (E, R, W)\ (G, R, W)$$

Two of these sufficient reasons do not contain the protected feature so the decision cannot be biased (Theorem 5). The decision will be the same on any applicant with the same characteristics as Robin except for the protected feature R. However, since some of the sufficient reasons contain a protected feature, the classifier must be biased (Theorem 6): It will make a biased decision on some other applicant. This illustrates how classifier bias can be inferred from the complete reason behind one of its *unbiased* decisions. This method is not complete though: the classifier may still be biased even if no protected feature appears in a sufficient reason for one of its decisions.

The decision on April is *not biased* even though the protected feature R appears in the reason circuit (the circuit is valid if we existentially quantify all features but R). Moreover, E, F are all the necessary characteristics for this decision (i.e., the necessary property). Flipping either of these characteristics will flip the decision. Recall that violating the necessary property may either flip the decision or change the reason behind it (Theorem 3) but flipping only one necessary characteristic is guaranteed to flip the decision (Proposition 3).

The decision on April would stick *even if* she were not to have work experience ($\neg W$) *because* she passed the entrance exam (E), has a good GPA (G) and is a first

time applicant (F). April would be denied admission if she were to also violate one of these characteristics (Definition 5 and Proposition 3).

We close this section by an important remark. Even though most of the notions we defined are based on prime implicants, our proposed theory does not necessarily require the computation of prime implicants which can be prohibitive. Reason circuits characterize all relevant prime implicants and can be obtained in linear time from Decision-DNNF circuits. Reason circuits are also monotone, allowing one to answer many queries about the embedded prime implicants in polytime. This is a major contribution of this work.

10 Concluding Remarks

We introduced a theory for reasoning about the decisions of Boolean classifiers, which is based on the fundamental notion of complete reasons. We presented applications of the theory to explaining decisions, evaluating counterfactual statements about decisions and identifying decision and classifier bias. We showed that if classifiers are represented by Decision-DNNFs, which are a superset of OBDDs, then the complete reason for a decision can be computed in linear time and in the form of a tractable Boolean circuit that we called a reason circuit. We then presented linear-time and polytime algorithms for computing most of the introduced notions based on reason circuits. More recently, the notion of a complete reason was formulated using quantified Boolean logic and shown to be also computable efficiently when classifiers are represented by CNFs or SDDs (Darwiche and Marquis 2021). An SDD is a decision diagram that branches on formulas (sentences) instead of variables (SDD stands for *Sentential Decision Diagram*) (Darwiche 2011). SDDs are also a superset of OBDDs but they are not comparable to Decision-DNNFs in terms of succinctness (Bollig and Buttkus 2019; Beame and Liew 2015; Beame et al. 2013).

There has been a significant interest recently in the computation and complexity of explanation queries, particularly sufficient reasons. This included investigations into the computation of *shortest* sufficient reasons which are length-minimal instead of subset-minimal. For Naïve Bayes (and linear) classifiers, it was shown that one sufficient reason can be generated in log-linear time, and all sufficient reasons can be generated with polynomial delay (Marques-Silva et al. 2020). For decision trees, the complexity of generating one sufficient reason was shown to be in polynomial time (Izza et al. 2020). Later works showed the same complexity for decision graphs (Huang et al. 2021b) and some classes of tractable circuits (Audemard et al. 2020; Huang et al. 2021a). The generation of sufficient reasons for decision trees was also studied in Audemard et al. (2021b), including the generation of shortest sufficient reasons which was shown to be hard even for a single reason. The generation of shortest sufficient reasons was also studied in a broader context that includes decision graphs and SDDs (Darwiche and Ji 2022). More general studies of complexity were also conducted in Audemard et al. (2020), Huang et al. (2021a), where classifiers where categorized based on the tractable circuits that represent them (Huang et al. 2021a) or the kinds of processing they permit in polynomial time (Audemard et al. 2020). The complexity of robustness queries and shortest sufficient reasons was studied in Barceló

et al. (2020) for Boolean classifiers which correspond to decision graphs and neural networks with ReLU activation functions. A comprehensive study of complexity was presented recently in Audemard et al. (2021a) for a large set of explanation queries and classes of Boolean classifiers.

References

Audemard, G., Bellart, S., Bounia, L., Koriche, F., Lagniez, J. M., & Marquis, P. (2021) On the explanatory power of decision trees. CoRR arXiv:2108.05266

Audemard, G., Bellart, S., Bounia, L., Koriche, F., Lagniez, J., & Marquis, P. (2021). On the computational intelligibility of Boolean classifiers. CoRR arXiv:abs/2104.06172

Audemard, G., Koriche, F., & Marquis, P. (2020). On tractable XAI queries based on compiled representations. In *KR* (pp. 838–849).

Barceló, P., Monet, M., Pérez, J., & Subercaseaux, B. (2020). Model interpretability through the lens of computational complexity. In *NeurIPS*.

Beame, P., & Liew, V. (2015). New limits for knowledge compilation and applications to exact model counting. In *UAI* (pp. 131–140). AUAI Press.

Beame, P., Li, J., Roy, S., & Suciu, D. (2013). Lower bounds for exact model counting and applications in probabilistic databases. In *UAI*. AUAI Press.

Bollig, B., & Buttkus, M. (2019). On the relative succinctness of sentential decision diagrams. *Theory of Computing Systems, 63*(6), 1250–1277.

Bryant, R. E. (1986). Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers, 35*(8), 677–691.

Chan, H., & Darwiche, A. (2003). Reasoning about Bayesian network classifiers. In *UAI* (pp. 107–115). Morgan Kaufmann.

Choi, A., Shih, A., Goyanka, A., & Darwiche, A. (2020). On symbolically encoding the behavior of random forests. CoRR arXiv:abs/2007.01493

Coudert, O., & Madre, J. C. (1993). Fault tree analysis: 10^{20} prime implicants and beyond. In *Proceedings of the annual reliability and maintainability symposium*.

Coudert, O., Madre, J. C., Fraisse, H., & Touati, H. (1993). Implicit prime cover computation: An overview. In *Proceedings of the 4th SASIMI workshop*.

Crama, Y., & Hammer, P. L. (2011). Boolean functions-theory, algorithms, and applications. In *Encyclopedia of mathematics and its applications* (vol. 142). Cambridge University Press.

Darwiche, A. (2004). New advances in compiling CNF into decomposable negation normal form. In *ECAI* (pp. 328–332). IOS Press.

Darwiche, A. (2011). SDD: A new canonical representation of propositional knowledge bases. In *IJCAI* (pp. 819–826). IJCAI/AAAI.

Darwiche, A. (2020). Three modern roles for logic in AI. In *PODS* (pp. 229–243). ACM

Darwiche, A., & Hirth, A. (2020). On the reasons behind decisions. In *ECAI, frontiers in artificial intelligence and applications* (vol. 325, pp. 712–720). IOS Press.

Darwiche, A., & Ji, C. (2022). On the computation of necessary and sufficient explanations. In *AAAI*. AAAI Press.

Darwiche, A. (2001). Decomposable negation normal form. *Journal of the ACM, 48*(4), 608–647.

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research, 17,* 229–264.
Darwiche, A., & Marquis, P. (2021). On literal quantification in Boolean logic and its applications to explainable AI. *Journal of Artificial Intelligence Research, 72,* 285–328.
de Kleer, J., Mackworth, A. K., & Reiter, R. (1992). Characterizing diagnoses and systems. *Artificial Intelligence, 56*(2–3), 197–222.
Huang, J., & Darwiche, A. (2005). DPLL with a trace: From SAT to knowledge compilation. In *IJCAI* (pp. 156–162). Professional Book Center.
Huang, X., Izza, Y., Ignatiev, A., & Marques-Silva, J. (2021). On efficiently explaining graph-based classifiers. CoRR arXiv:abs/2106.01350.
Huang, X., Izza, Y., Ignatiev, A., Cooper, M.C., Asher, N., & Marques-Silva, J. (2021). Efficient explanations for knowledge compilation languages. CoRR arXiv:abs/2107.01654.
Huang, J., & Darwiche, A. (2007). The language of search. *Journal of Artificial Intelligence Research, 29,* 191–219.
Ignatiev, A., Narodytska, N., & Marques-Silva, J. (2019). Abduction-based explanations for machine learning models. In *Proceedings of the thirty-three conference on artificial intelligence (AAAI)* (pp. 1511–1519).
Ignatiev, A., Narodytska, N., & Marques-Silva, J. (2019). On validating, repairing and refining heuristic ML explanations. CoRR arXiv:abs/1907.02509.
Ignatiev, A., Narodytska, N., Asher, N., & Marques-Silva, J. (2020). From contrastive to abductive explanations and back again. In *AI*IA, Lecture Notes in Computer Science*, (vol. 12414, pp. 335–355). Springer.
Ignatiev, A., Narodytska, N., Marques-Silva, J. (2019). On relating explanations and adversarial examples. In *Advances in neural information processing systems* (vol. 32, pp. 15883–15893). Curran Associates, Inc. URL http://papers.nips.cc/paper/9717-on-relating-explanations-and-adversarial-examples.pdf.
Izza, Y., Ignatiev, A., & Marques-Silva, J. (2020). On explaining decision trees. CoRR arXiv:abs/2010.11034.
Katz, G., Barrett, C. W., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. *Computer Aided Verification CAV, 5,* 97–117.
Lagniez, J., & Marquis, P. (2017). An improved Decision-DNNF compiler. In *IJCAI* (pp. 667–673). ijcai.org.
Leofante, F., Narodytska, N., Pulina, L., & Tacchella, A. (2018). Automated verification of neural networks: Advances, challenges and perspectives. CoRR arXiv:abs/1805.09938.
Lindner, F., & Möllney, K. (2019). Extracting reasons for moral judgments under various ethical principles. In C. Benzmüller & H. Stuckenschmidt (Eds.), *KI 2019: advances in artificial intelligence* (pp. 216–229). Cham: Springer International Publishing.
Marques-Silva, J., Gerspacher, T., Cooper, M. C., Ignatiev, A., & Narodytska, N. (2020). Explaining naive Bayes and other linear classifiers with polynomial time and delay. In *NeurIPS*.
McCluskey, E. J. (1956). Minimization of Boolean functions. *The Bell System Technical Journal, 35*(6), 1417–1444. https://doi.org/10.1002/j.1538-7305.1956.tb03835.x.
Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence, 267,* 1–38.
Minato, S. (1993). Fast generation of prime-irredundant covers from binary decision diagrams. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 76*(6), 967–973.
Narodytska, N., Kasiviswanathan, S. P., Ryzhyk, L., Sagiv, M., & Walsh, T. (2018). Verifying properties of binarized deep neural networks. In *Proceedings of the thirty-second AAAI conference on artificial intelligence (AAAI)*.
Oztok, U., & Darwiche, A. (2014). On compiling CNF into Decision-DNNF. In *CP, Lecture Notes in Computer Science* (vol. 8656, pp. 42–57). Springer.
Quine, W. V. (1952). The problem of simplifying truth functions. *The American Mathematical Monthly, 59*(8), 521–531.
Quine, W. V. (1959). On cores and prime implicants of truth functions. *The American Mathematical Monthly, 66*(9), 755–760.
Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *AAAI* (pp. 1527–1535). AAAI Press.
Shi, W., Shih, A., Darwiche, A., & Choi, A. (2020). On tractable representations of binary neural networks. In *KR* (pp. 882–892).

Shih, A., Choi, A., & Darwiche, A. (2018). A symbolic approach to explaining Bayesian network classifiers. In *IJCAI* (pp. 5103–5111). ijcai.org.
Shih, A., Choi, A., & Darwiche, A. (2019). Compiling Bayesian network classifiers into decision graphs. In *AAAI* (pp. 7966–7974). AAAI Press.
Shih, A., Darwiche, A., & Choi, A. (2019). Verifying binarized neural networks by angluin-style learning. In *SAT, Lecture Notes in Computer Science* (vol. 11628, pp. 354–370). Springer.