ECAI 2020
G.D. Giacomo et al. (Eds.)
© 2020 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA200158

On the Reasons Behind Decisions

Adnan Darwiche[1] and **Auguste Hirth**[2]

Abstract. Recent work has shown that some common machine learning classifiers can be compiled into Boolean circuits that have the same input-output behavior. We present a theory for unveiling the *reasons* behind the decisions made by Boolean classifiers and study some of its theoretical and practical implications. We define notions such as sufficient, necessary and complete reasons behind decisions, in addition to classifier and decision bias. We show how these notions can be used to evaluate counterfactual statements such as "a decision will stick even if . . . because . . . ." We present efficient algorithms for computing these notions, which are based on new advances on tractable Boolean circuits, and illustrate them using a case study.

1 Introduction

Recent work has shown that some common machine learning classifiers can be compiled into Boolean circuits that make the same decisions. This includes Bayesian network classifiers with discrete features [2, 31] and some types of neural networks [32, 3]. Proposals were also extended to explain and verify these numeric classifiers by operating on their compiled circuits [30, 29]. We extend this previous work by proposing a theory for reasoning about the decisions made by classifiers and discus its theoretical and practical implications.

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.* Figure 1 depicts a classifier ($\mathcal{C}_1$) for college admission, represented as an Ordered Binary Decision Diagram (OBDD) [1]. This OBDD was compiled from the Bayesian network (BN) classifier in Figure 2 using the algorithm in [31]. The OBDD is guaranteed to make the same decision as the BN classifier on every instance (same input-output behavior).

Our main goal is to *explain* the decisions made by a classifier on specific instances by way of providing various insights into what caused these decisions. Consider 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 $\mathcal{C}_1$ depicted in Figure 1. She also comes from a rich hometown and will be admitted by classifier $\mathcal{C}_2$ depicted in the same figure. We can say that Susan was admitted by classifier $\mathcal{C}_1$ *because* she passed the entrance exam and has a high GPA. We can also say that *one reason why* classifier $\mathcal{C}_2$ 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 $\mathcal{C}_2$ *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 $\mathcal{C}_2$ is biased as it 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.

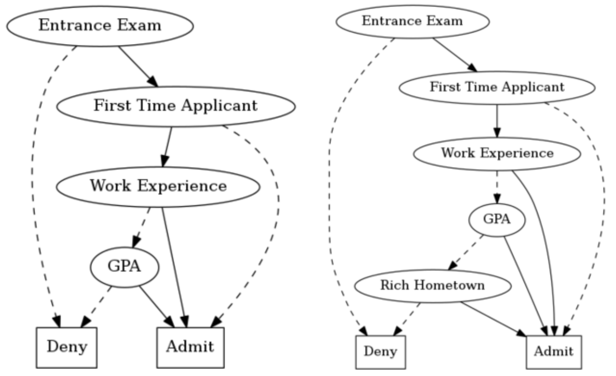

Figure 1. OBDD $\mathcal{C}_1$ (left) and OBDD $\mathcal{C}_2$ (right). To classify an instance, we start at the root OBDD node and repeat the following. If the feature we are at is positive, we follow the solid edge, otherwise the dotted edge.

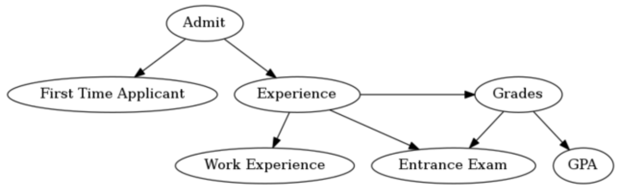

Figure 2. The structure of a Bayesian network classifier.

We will give formal definitions and justifications for the statements exemplified above and show how to compute them algorithmically. As far as semantics, the main tool we will employ is the classical notion of prime implicants [6, 25, 21, 26]. On the computational side, we will exploit tractable Boolean circuits [9] while providing some new fundamental results that further extend the reach of these circuits to computing explanations.

This paper is structured as follows. We review prime implicants in Section 2. We follow by introducing the notions of sufficient, necessary and complete reasons in Sections 3–5. Counterfactual statements about decisions are discussed in Section 6, followed by a discussion of decision and classifier bias in Section 7. We dedicate Section 8 to algorithms that compute the introduced notions while illustrating them using a case study in Section 9. We finally close with some concluding remarks in Section 10.

[1] University of California, Los Angeles, email: darwiche@cs.ucla.edu
[2] University of California, Los Angeles, email: ahirth@cs.ucla.edu

2 Classifiers, Decisions and Prime Implicants

We represent a classifier by a propositional formula Δ whose models (i.e., satisfying assignments) correspond to positive instances. The negation of the formula characterizes negative instances. Classifiers $\mathcal{C}_1$ and $\mathcal{C}_2$ of Figure 1 are represented by the following formulas:

$$\begin{aligned}\Delta_1 &= E \wedge (\neg F \vee G \vee W)\\ \Delta_2 &= E \wedge (\neg F \vee G \vee W \vee R)\end{aligned}$$

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 we use it frequently later noting that $\alpha \models \Delta_\alpha$ and $\Delta(\alpha) = \Delta(\beta)$ iff $\Delta_\alpha = \Delta_\beta$.

A *literal* is a variable (positive literal) or its negation (negative literal). A *term* is a consistent conjunction of literals. Term τ_i *subsumes* term τ_j, written $\tau_j \models \tau_i$, iff τ_j includes the literals of τ_i. 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 sometimes refer to a literal as a **characteristic** and to a term τ as a **property** (of an instance). 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 $E \wedge \neg F$).

An *implicant* τ of propositional formula Δ is a term that satisfies Δ, written $\tau \models \Delta$. 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 $\mathcal{C}_1$ has the following prime implicants:

$$\begin{aligned}\Delta_1 &: (E \wedge \neg F)\ (E \wedge G)\ (E \wedge W)\\ \neg\Delta_1 &: (\neg E)\ (F \wedge \neg G \wedge \neg W)\end{aligned}$$

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

$$\begin{aligned}\Delta_2 &: (E \wedge \neg F)\ (E \wedge G)\ (E \wedge W)\ (E \wedge R)\\ \neg\Delta_2 &: (\neg E)\ (F \wedge \neg G \wedge \neg W \wedge \neg R)\end{aligned}$$

The set of prime implicants for a propositional formula can be quite large, which motivated the notion of a *prime implicant cover* [25, 21, 26]. A set of terms $\tau_1, \ldots, \tau_n$ is prime implicant cover for propositional 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 propositional formula. To condition formula Δ on literals τ, denoted $\Delta|\tau$, is to replace every literal l in Δ with 1 if $l \in \tau$ and with 0 if $\neg l \in \tau$. We will also use *existential quantification:* $\exists X \Delta = (\Delta|X) \vee (\Delta|\neg X)$.

In the next few sections, we introduce the notions of sufficient, complete and necessary 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.[3] However, their active utilization in explaining decisions is more recent, e.g., [30, 14, 15, 19], and introduced a key connection to properties of instances that we highlight next and exploit computationally later.

Definition 1 (Sufficient Reason [30]). *A sufficient reason for decision $\Delta(\alpha)$ is a property of instance α that is also a prime implicant of Δ_α (recall Δ_α is Δ if the decision is positive and $\neg\Delta$ otherwise).*

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 very large number of them.[4]

There is a key difference between prime implicants and sufficient reasons: the latter must be properties of the given instance. This has significant computational implications that we exploit in Section 8.

Sufficient reasons were introduced in [30] under the name of *PI-explanations.* The new name we adopt is motivated by further distinctions that we draw later and was also used in [19]. We will also 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 $\mathcal{C}_1$ 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. □

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.

[3] 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* [10]. A kernel diagnosis is 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.

[4] The LIME [27] and Anchor [28] systems can be viewed as computing approximations of sufficient reasons. The quality of these approximations has been evaluated on some datasets and corresponding classifiers in [16], 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.

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 the particular decision is made.

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

Proof. Suppose $\beta \not\models \mathcal{R}$ 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 $\Delta(\alpha)$, $\tau \models \mathcal{R}$ and $\beta \models \tau \models \mathcal{R}$, a contradiction. Hence, τ cannot be a property of instance α. □

We will sometimes say "the reason" to mean "the complete reason." Classifier $\mathcal{C}_1$ admits Greg ($\alpha = E, \neg F, \neg G, W$) for the reason $\mathcal{R} = E \wedge (\neg F \vee W)$. Greg was admitted because he passed the entrance exam and satisfied one of two additional requirements: he applied before and has work experience. Classifier $\mathcal{C}_1$ also admits Susan ($\beta = E, F, G, \neg W$). Susan does not satisfy the reason $\mathcal{R}$. 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. The classifier admitted Greg and Susan for different reasons.

The complete reason for a decision is unique up to logical equivalence and can be used to enumerate its sufficient reasons.[5]

Theorem 2. *Let $\mathcal{R}$ be the complete reason for decision $\Delta(\alpha)$. The prime implicants of $\mathcal{R}$ 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 $\mathcal{R} = \tau_1 \vee \ldots \vee \tau_n$. The key observation is that terms τ_i are properties of instance α. Hence, for every two terms τ_i and τ_j, term τ_i cannot contain some literal X while term τ_j containing literal $\neg X$. The DNF $\tau_1 \vee \ldots \vee \tau_n$ is then closed under consensus.[6] Since no term τ_i subsumes another term τ_j, the DNF $\tau_1 \vee \ldots \vee \tau_n$ contains all prime implicants of $\mathcal{R}$. Hence, the prime implicants of complete reason $\mathcal{R}$ are precisely the sufficient reasons of decision $\Delta(\alpha)$. □

5 Necessary Properties and Reasons

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 iff it appears in every sufficient reason for the decision. The necessary property for a decision is the set of all its necessary characteristics.*

[5] Pierre Marquis observed that the complete reason can be formulated using the notion of *literal forgetting* which is a more fine grained notion than *variable forgetting* (also known as existential quantification) [20, 18, 11].

[6] 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 [6, Chapter 3].

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 $\Delta(\alpha) \neq \Delta(\beta)$.*

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 necessary 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. □

We can now define the notion of necessary reason.

Definition 4 (Necessary Reason). *The necessary property of a decision is called the necessary reason for the decision iff it is the only sufficient reason for the decision.*

There may be no necessary reason for a decision as there may be no instance property that is both sufficient and necessary for triggering the decision. We next highlight how the complete reason for a decision, being a *condition* on an instance instead of a *property,* is always necessary and sufficient for explaining the decision.

Consider the complete reason $\mathcal{R}$ for decision $\Delta(\alpha)$ and recall that it characterizes all properties of instance α that can trigger the decision: $\mathcal{R} \equiv \bigvee_{\tau \models \Delta_\alpha} \tau$, where τ is a property of instance α. The reason $\mathcal{R}$ is then a logical condition that triggers the decision ($\mathcal{R} \models \Delta_\alpha$). If the complete reason is weakened into a condition $\mathcal{R}_w$ that continues to trigger the decision ($\mathcal{R} \models \mathcal{R}_w \models \Delta_\alpha$), then $\mathcal{R}_w$ will admit properties not satisfied by instance α. Moreover, if it is strengthened into a condition $\mathcal{R}_s$, then $\mathcal{R}_s$ will continue to trigger the decision ($\mathcal{R}_s \models \mathcal{R} \models \Delta_\alpha$) but will stop admitting some properties of instance α that can trigger the decision. Hence, the complete reason $\mathcal{R}$ is a necessary and sufficient condition (not necessarily a property) for explaining the decision on instance α.

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 $\mathcal{C}_1$ admits Susan *because* she passed the entrance exam and has a high GPA. Greg was also admitted by this classifier. His application is similar to Susan's except that he applied before and has work experience ($\beta = E, \neg F, G, W$). We cannot pinpoint a single property of Greg that triggered admission, so we cannot issue a "because" statement when explaining this decision.

Definition 5 (**Because**). *Consider decision $\Delta(\alpha)$ and let τ be a property of instance α. The decision is made "because τ" iff τ is the complete reason for the decision.*

Proposition 5. *A decision is made because τ iff τ is the necessary reason for the decision (i.e., the only sufficient reason).*

Proof. Follows from Definitions 1, 2 and 4. □

One may be interested in statements that provide insights into a decision beyond the reasons behind it. For example, how the classifier may have decided if some instance characteristics were different.

An example statement is the one we mentioned in Section 1: Susan would have been 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.

Definition 6 (**Even-If-Because**). *Consider decision $\Delta(\alpha)$ and let ρ and τ be properties of instance α. The decision sticks "even if $\overline{\rho}$ because τ" iff τ is the complete reason for the decision after changing property ρ of instance α to $\overline{\rho}$ (i.e., flipping all characteristics in ρ).*

Let β be the result of replacing property ρ of instance α by $\overline{\rho}$ and suppose that τ is the complete reason for decision $\Delta(\beta)$. Then τ is the only sufficient reason for decision $\Delta(\beta)$ by Definition 2. Hence $\beta \models \tau$ and properties ρ and τ must be disjoint. Moreover, $\alpha \models \tau \models \Delta_\beta$ so $\Delta_\alpha = \Delta_\beta$ and $\Delta(\alpha) = \Delta(\beta)$. Hence, the decision sticks "even if $\overline{\rho}$ because τ."

Applicant Susan discussed earlier ($\alpha = E, F, G, \neg W, R$) is admitted by classifier $\mathcal{C}_2$. 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 $\Delta(\beta)$ where $\beta = E, F, \neg G, \neg W, R$ 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 $\mathcal{C}_1$. 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 a *protected feature:* one that should not be used when making the decision (e.g., gender, zip code, or ethnicity).[7] We formalize bias next while making a distinction between classifier bias and decision bias: A classifier may be biased in that it could make biased decisions, but the particular decisions it already made may have been 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.

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

Definition 7 (**Decision Bias**). *Decision $\Delta(\alpha)$ is biased iff $\Delta(\alpha) \neq \Delta(\beta)$ 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 4. *A decision is biased iff each of its sufficient reasons contains at least one protected feature.*

Proof. 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 of decision $\Delta(\alpha)$ contains at least one protected feature. Let $\mathbf{X}$ be these protected features and 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 features in $\mathbf{X}$). 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. □

Theorem 4 does not require sufficient reasons to share a protected feature, only that each must contain at least one protected feature.

Consider classifier $\mathcal{C}_3$, 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):

$$\Delta_3 = (G \wedge E) \vee (G \wedge M) \vee (G \wedge R). \quad (1)$$

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. This classifier will not admit Nancy who has similar characteristics but does not come 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 8 (**Classifier Bias**). *A classifier is biased iff at least one of its decisions is biased.*

A classifier may be biased, but some of its decisions may be unbiased. Moreover, one can sometimes infer classifier bias by inspecting the sufficient reasons behind one of its unbiased decisions.

Theorem 5. *A classifier is biased iff one of its decisions has a sufficient reason that includes a protected feature.*

Proof. Suppose classifier Δ is biased. By Definition 8, some decision $\Delta(\alpha)$ is biased. By Theorem 4, every sufficient reason of decision $\Delta(\alpha)$ must contain at least one protected feature.

Suppose some 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 is true: 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 and 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 4. But classifier Δ is biased according to Theorem 5 as we can *prove* that it will make a biased decision on some other instance $\beta \neq \alpha$.

Consider classifier $\mathcal{C}_3$ 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 5 suggests that the classifier will make different decisions on two applicants with a good GPA that 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 use this theorem (and Theorem 7) when discussing algorithms in Section 8.

Theorem 6. *A decision is biased iff $\exists(X_1, \ldots, X_n)\mathcal{R}$ is not valid where $X_1, \ldots, X_n$ are all unprotected features and $\mathcal{R}$ is the complete reason behind the decision.*

Proof. Let $\tau_1, \ldots, \tau_n$ be the decision's sufficient reasons and hence $\mathcal{R} = \tau_1 \vee \ldots \vee \tau_n$. 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, $\exists X_1, \ldots, X_n\mathcal{R}$ 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 7. *A classifier is biased if $\mathcal{R}|X \not\equiv \mathcal{R}|\neg X$ where X is a protected feature and $\mathcal{R}$ is the complete reason for some decision.*

Proof. Given Theorems 2 and 5, it is sufficient to show that $\mathcal{R}|X \not\equiv \mathcal{R}|\neg X$ iff feature X appears in some prime implicant of $\mathcal{R}$. Let $\tau_1, \ldots, \tau_n$ be the prime implicants of $\mathcal{R}$. 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 $\mathcal{R}|X \equiv \bigvee_{X \notin \tau_i} \tau_i \vee \bigvee_{X \in \tau_i} \tau_i \setminus \{X\}$ and $\mathcal{R}|\neg X \equiv \bigvee_{X \notin \tau_i} \tau_i$. Hence $\mathcal{R}|X \not\equiv \mathcal{R}|\neg X$ iff $X \in \tau_i$ for some prime implicant τ_i. □

Theorem 7 follows from Theorems 2 and 5 and a known result: A Boolean function depends on a variable X iff X appears in one of its prime implicants. We include the full proof for completeness.

8 Computing Reasons and Related Queries

The enumeration of PI-explanations (sufficient reasons) was treated in [30] by modifying the algorithm in [4] for computing prime implicant covers; see also [5, 22]. 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 7 and 8).

We next propose a new approach based on computing the complete reason $\mathcal{R}$ 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 $\mathcal{R}$ (Theorem 2). We can also use it to compute the necessary reason for a decision (Proposition 4) and to detect decision bias (Theorem 6). Even classifier bias can sometimes be inferred directly using the reason $\mathcal{R}$ (Theorem 7) among other queries.

Assuming the classifier is represented using a suitable tractable circuit (e.g., OBDD), 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 is based on Decision-DNNF circuits, obtained using compilers such as C2D[8] [8], MINI_C2D[9] [23, 24] and D4[10] [17].

Definition 9 (Decision-NNF Circuit). *A DNNF circuit has literals or constants as inputs and two type of gates: and-gates and or-gates, where the subcircuits feeding into each and-gate share no variables. It is called a Decision-DNNF circuit if every or-gate has exactly two inputs of the form: $X \wedge \mu$ and $\neg X \wedge \nu$, where X is a variable.*

DNNF circuits were introduced in [7]. Decision-DNNF circuits were identified in [12, 13] and include Ordered Binary Decision Diagrams (OBDDs) [1, 13]. Figure 3 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 DNNF circuits and OBDD, see [9, 23].

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

Definition 10 (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 3 depicts a Decision-DNNF circuit and its consensus circuit (third from left). The consensus operation adds four and-gates denoted with double circles.

[8] `http://reasoning.cs.ucla.edu/c2d/`
[9] `http://reasoning.cs.ucla.edu/minic2d/`
[10] `http://www.cril.univ-artois.fr/kc/d4.html`

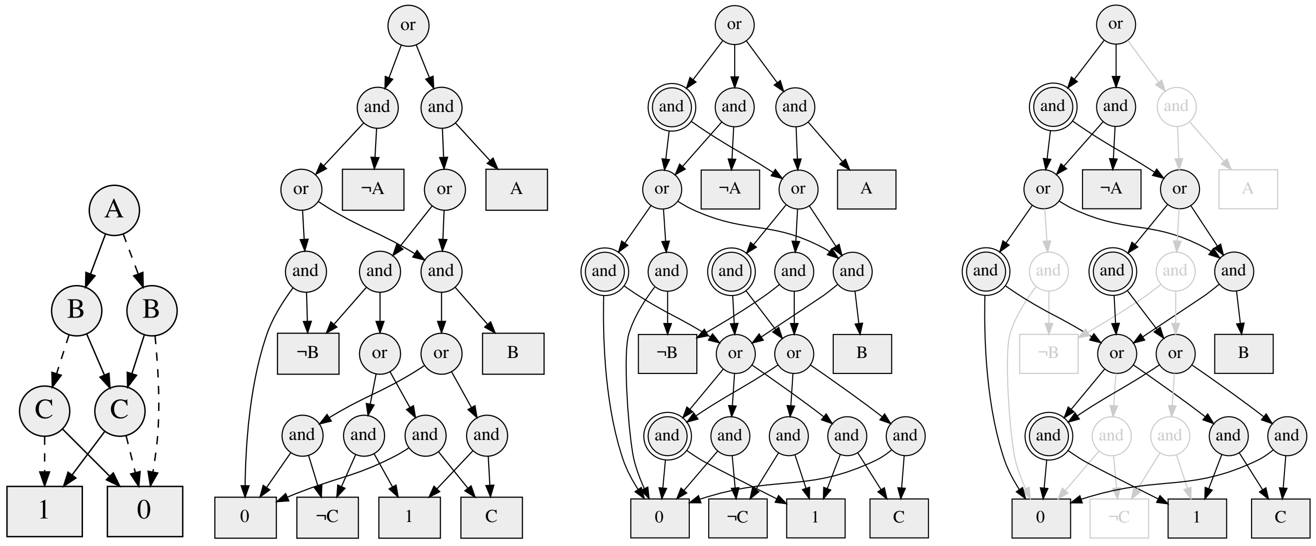

Figure 3. From left to right: OBDD, Decision-DNNF circuit, consensus circuit, and the filtering of consensus circuit by instance $\neg A, B, C$.

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

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

A consensus circuit can be obtained in time linear in the size of Decision-DNNF circuit, but is not a DNNF circuit. We next discuss the filtering of a consensus circuit, which leads to a tractable circuit.

Definition 11 (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 only defined on consensus circuits and requires an instance that satisfies the circuit. Figure 3 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, 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 instance γ agrees with instance α no less than instance β does, then $\beta \models \Gamma$ implies $\gamma \models \Gamma$. For example, if $\alpha = X, Y, Z$, $\beta = \neg X, Y, \neg Z$ and $\gamma = \neg X, Y, Z$.

Theorem 8. *If circuit Γ results from filtering by instance α then every literal l in Γ appears in α, and $\Gamma(\gamma) \geq \Gamma(\beta)$ 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 filtering due to having applied consensus.

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* $\mathsf{filter}(\mathsf{consensus}(\Delta), \alpha)$.

Proof. Let $\Gamma = \mathsf{filter}(\mathsf{consensus}(\Delta), \alpha)$, $\mathcal{I}(\Delta) = \{\tau : \tau \models \Delta\}$ and $\mathcal{I}(\Delta, \alpha) = \{\tau : \tau \models \Delta \text{ and } \alpha \models \tau\}$. We need to show that $\mathcal{I}(\Delta, \alpha) \subseteq \mathcal{I}(\Gamma)$. That is, Γ preserves the implicants τ of Δ that are 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, $\mathcal{I}(\Delta, \alpha) \subseteq \mathcal{I}(\Gamma)$.

(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 do not share variables (decomposability), $\mathcal{I}(\Delta) = \mathcal{I}(\Delta_1) \times \mathcal{I}(\Delta_2)$ (Cartesian product). Similarly, $\mathcal{I}(\Gamma) = \mathcal{I}(\Gamma_1) \times \mathcal{I}(\Gamma_2)$. By the induction hypothesis, $\mathcal{I}(\Delta_1, \alpha) \subseteq \mathcal{I}(\Gamma_1)$ and $\mathcal{I}(\Delta_2, \alpha) \subseteq \mathcal{I}(\Gamma_2)$. Hence,

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

(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, $\mathcal{I}(\Delta) = \mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_c$ where

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

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

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

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

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 9. *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* $\mathsf{filter}(\mathsf{consensus}(\Delta), \alpha)$.

Proof. Let $\Gamma = \mathsf{filter}(\mathsf{consensus}(\Delta), \alpha)$ and observe that $\Gamma \models \Delta$ since $\mathsf{consensus}(\Delta) \equiv \Delta$ and Γ is the result of replacing some inputs of NNF circuit $\mathsf{consensus}(\Delta)$ 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 8. 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. □

Definition 12 (Reason Circuit). *For classifier Δ, instance α and a Decision-DNNF circuit Γ for Δ_α, circuit* $\mathsf{filter}(\mathsf{consensus}(\Gamma), \alpha)$ *is called a "reason circuit" and denoted* $\mathsf{reason}(\Delta, \alpha)$.

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 13 (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 8. The following theorem mirrors what is known on monotone propositional formula, but we include it for completeness.

Theorem 10. *The satisfiability of a monotone NNF circuit can be decided in linear time. A monotone NNF circuit can be negated and 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 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. □

Algorithm 1 $\mathsf{PI}(\Delta, \alpha)$

```
input: Decision-DNNF circuit $\Delta$, instance $\alpha$ (assumes $\Delta(\alpha) = 1$).
output: Prime implicants of circuit $\mathsf{filter}(\mathsf{consensus}(\Delta), \alpha)$.
main:
1:  if $\mathsf{cache}(\Delta)$ is set then
2:      return $\mathsf{cache}(\Delta)$
3:  else if $\Delta$ is constant 0 then
4:      $r = \{\}$
5:  else if $\Delta$ is constant 1 then
6:      $r = \{\{\}\}$
7:  else if $\Delta = \Delta_1 \wedge \Delta_2$ then
8:      $r = \mathsf{cartesian\_product}(\mathsf{PI}(\Delta_1, \alpha), \mathsf{PI}(\Delta_2, \alpha))$
9:  else if $\Delta = (X \wedge \Delta_1) \vee (\neg X \wedge \Delta_2)$ then
10:     $(\ell, \Gamma) = (X, \Delta_1)$ if literal $X$ in $\alpha$ else $(\neg X, \Delta_2)$
11:     $p = \mathsf{cartesian\_product}(\mathsf{PI}(\Delta_1, \alpha), \mathsf{PI}(\Delta_2, \alpha))$
12:     $q = \{\{\ell\} \cup \tau$ for $\tau \in \mathsf{PI}(\Gamma, \alpha)\}$
13:     $r = p \cup q$
14: $r = \mathsf{remove\_subsumed}(r)$
15: $\mathsf{cache}(\Delta) = r$
16: return $r$
```

Given Theorem 10, the validity of a monotone NNF circuit can be decided in linear time (we check whether the negated circuit is unsatisfiable).[11] 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 6.

Theorem 11. *Replacing every literal of variable X with constant 1 in monotone NNF circuit Γ yields a 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 does this in place, that is without explicitly constructing the consensus or filtered circuit. It assumes a positive decision (otherwise we pass $\neg\Delta$).

Algorithm 1 uses subroutine $\mathsf{cartesian\_product}$ which conjoins two DNFs by computing the Cartesian product of their terms. It also uses $\mathsf{remove\_subsumed}$ to remove subsumed terms from a DNF.

Theorem 12. *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 [6, Chapter 3]. This is precisely what Algorithm 1 does. □

[11] 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.

Consider a decision $\Delta(\alpha)$ and its complete reason $\mathcal{R} = \mathsf{reason}(\Delta, \alpha)$ as a monotone NNF circuit obtained by consensus then filtering. Let n be the size of circuit $\mathcal{R}$ and m be the number of features. We next show how to compute various queries using circuit $\mathcal{R}$.

Sufficient Reasons. By Theorems 2 and 12, 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 $\mathcal{R} \models l$. This is equivalent to $\mathcal{R}|\neg l$ being unsatisfiable, which can be decided in $O(n)$ time given Theorem 10. The necessary property (all necessary characteristics) can then be computed in $O(n \cdot m)$ time.

Necessary Reason. To compute the necessary reason (if any) we compute the necessary property and check whether it satisfies the complete reason. This can be done 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 5): $\tau \models \mathcal{R}$ and $\mathcal{R} \models \tau$. We have $\tau \models \mathcal{R}$ iff $(\neg\mathcal{R})|\tau$ is unsatisfiable. Moreover, $\mathcal{R} \models \tau$ iff $\mathcal{R}|\neg l$ 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 6). 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 $\mathcal{R}$ and then check the validity of the result (Theorem 6). All of this can be done in $O(n)$ time given Theorems 10 and 11.

9 Another Admissions Classifier

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 Figure 4. It corresponds to the following Boolean formula, which is not monotone (the previous 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 Figure 5, 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 Figure 6. These are reason circuits produced as suggested by Definition 12, 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).

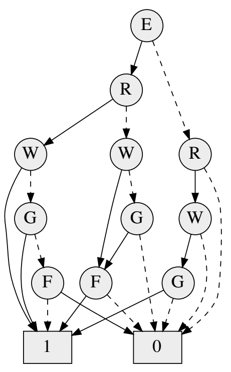

Figure 4. Admission classifier.

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

Figure 5. Applicants and their characteristics.

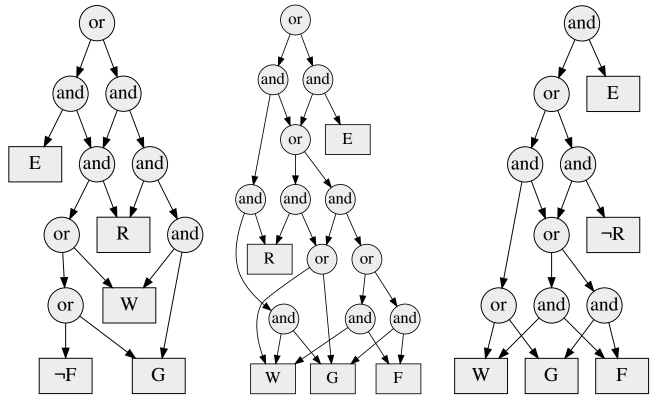

Figure 6. From left to right: Reason circuit for the decision on applicants Scott, Robin and April (Figure 5).

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 Figure 6 and then check its validity (Theorem 6). 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 4):

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

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 5).

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 by examining 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 4). 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 5): 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 6 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 Conclusion

We introduced a theory for reasoning about the decisions of Boolean classifiers, which is based on the notions of sufficient, necessary and complete reasons. We presented applications of the theory to explaining decisions, evaluating counterfactual statements about decisions and identifying decision bias and classifier bias. We also presented polytime and linear-time algorithms for computing most of the introduced notions based on the new and tractable class of reason circuits.

ACKNOWLEDGEMENTS

We wish to thank Arthur Choi and Jason Shen for providing valuable feedback. This work 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. The views in this paper do not necessarily represent those of sponsors.

REFERENCES

[1] Randal E. Bryant, ‘Graph-based algorithms for boolean function manipulation’, *IEEE Trans. Computers*, **35**(8), 677–691, (1986).

[2] Hei Chan and Adnan Darwiche, ‘Reasoning about bayesian network classifiers’, in *UAI*, pp. 107–115. Morgan Kaufmann, (2003).

[3] Arthur Choi, Weijia Shi, Andy Shih, and Adnan Darwiche, ‘Compiling neural networks into tractable Boolean circuits’, in *AAAI Spring Symposium on Verification of Neural Networks (VNN)*, (2019).

[4] Olivier Coudert and Jean Christophe Madre, ‘Fault tree analysis: 10^{20} prime implicants and beyond’, in *Proc. of the Annual Reliability and Maintainability Symposium*, (1993).

[5] Olivier Coudert, Jean Christophe Madre, Henri Fraisse, and Herve Touati, ‘Implicit prime cover computation: An overview’, in *Proceedings of the 4th SASIMI Workshop*, (1993).

[6] Yves Crama and Peter L. Hammer, *Boolean Functions - Theory, Algorithms, and Applications*, volume 142 of *Encyclopedia of mathematics and its applications*, Cambridge University Press, 2011.

[7] Adnan Darwiche, ‘Decomposable negation normal form’, *J. ACM*, **48**(4), 608–647, (2001).

[8] Adnan Darwiche, ‘New advances in compiling CNF into decomposable negation normal form’, in *ECAI*, pp. 328–332. IOS Press, (2004).

[9] Adnan Darwiche and Pierre Marquis, ‘A knowledge compilation map’, *J. Artif. Intell. Res.*, **17**, 229–264, (2002).

[10] Johan de Kleer, Alan K. Mackworth, and Raymond Reiter, ‘Characterizing diagnoses and systems’, *Artif. Intell.*, **56**(2-3), 197–222, (1992).

[11] Andreas Herzig, Jérôme Lang, and Pierre Marquis, ‘Propositional update operators based on formula/literal dependence’, *ACM Trans. Comput. Log.*, **14**(3), 24:1–24:31, (2013).

[12] Jinbo Huang and Adnan Darwiche, ‘DPLL with a trace: From SAT to knowledge compilation’, in *IJCAI*, pp. 156–162. Professional Book Center, (2005).

[13] Jinbo Huang and Adnan Darwiche, ‘The language of search’, *J. Artif. Intell. Res.*, **29**, 191–219, (2007).

[14] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva, ‘Abduction-based explanations for machine learning models’, in *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1511–1519, (2019).

[15] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva, ‘On relating explanations and adversarial examples’, in *Advances in Neural Information Processing Systems 32*, 15883–15893, Curran Associates, Inc., (2019).

[16] Alexey Ignatiev, Nina Narodytska, and João Marques-Silva, ‘On validating, repairing and refining heuristic ML explanations’, *CoRR*, **abs/1907.02509**, (2019).

[17] Jean-Marie Lagniez and Pierre Marquis, ‘An improved decision-dnnf compiler’, in *IJCAI*, pp. 667–673. ijcai.org, (2017).

[18] Jérôme Lang, Paolo Liberatore, and Pierre Marquis, ‘Propositional independence: Formula-variable independence and forgetting’, *J. Artif. Intell. Res.*, **18**, 391–443, (2003).

[19] Felix Lindner and Katrin Möllney, ‘Extracting reasons for moral judgments under various ethical principles’, in *KI 2019: Advances in Artificial Intelligence*, eds., Christoph Benzmüller and Heiner Stuckenschmidt, pp. 216–229, Cham, (2019). Springer International Publishing.

[20] Pierre Marquis, *Consequence finding algorithms*, volume 5 of *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, chapter 2, 41–145, Kluwer Academic Publisher, 2000. Moral S. et Kohlas J. (eds.), Gabbay D. et Smets Ph. (series eds.).

[21] E. J. McCluskey, ‘Minimization of boolean functions’, *The Bell System Technical Journal*, **35**(6), 1417–1444, (Nov 1956).

[22] Shin-ichi Minato, ‘Fast generation of prime-irredundant covers from binary decision diagrams’, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **76**(6), 967–973, (1993).

[23] Umut Oztok and Adnan Darwiche, ‘On compiling CNF into decision-dnnf’, in *CP*, volume 8656 of *Lecture Notes in Computer Science*, pp. 42–57. Springer, (2014).

[24] Umut Oztok and Adnan Darwiche, ‘An exhaustive DPLL algorithm for model counting’, *J. Artif. Intell. Res.*, **62**, 1–32, (2018).

[25] W. V. Quine, ‘The problem of simplifying truth functions’, *The American Mathematical Monthly*, **59**(8), 521–531, (1952).

[26] W. V. Quine, ‘On cores and prime implicants of truth functions’, *The American Mathematical Monthly*, **66**(9), 755–760, (1959).

[27] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin, ‘”why should I trust you?”: Explaining the predictions of any classifier’, in *KDD*, pp. 1135–1144. ACM, (2016).

[28] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin, ‘Anchors: High-precision model-agnostic explanations’, in *AAAI*, pp. 1527–1535. AAAI Press, (2018).

[29] Andy Shih, Arthur Choi, and Adnan Darwiche, ‘Formal verification of bayesian network classifiers’, in *PGM*, volume 72 of *Proceedings of Machine Learning Research*, pp. 427–438. PMLR, (2018).

[30] Andy Shih, Arthur Choi, and Adnan Darwiche, ‘A symbolic approach to explaining bayesian network classifiers’, in *IJCAI*, pp. 5103–5111. ijcai.org, (2018).

[31] Andy Shih, Arthur Choi, and Adnan Darwiche, ‘Compiling bayesian network classifiers into decision graphs’, in *AAAI*, pp. 7966–7974. AAAI Press, (2019).

[32] Andy Shih, Adnan Darwiche, and Arthur Choi, ‘Verifying binarized neural networks by angluin-style learning’, in *SAT*, volume 11628 of *Lecture Notes in Computer Science*, pp. 354–370. Springer, (2019).