On the Computation of Necessary and Sufficient Explanations

Adnan Darwiche, Chunxi Ji

Computer Science Department
University of California, Los Angeles
darwiche@cs.ucla.edu, jich@cs.ucla.edu

Abstract

The *complete reason* behind a decision is a Boolean formula that characterizes why the decision was made. This recently introduced notion has a number of applications, which include generating explanations, detecting decision bias and evaluating counterfactual queries. Prime *implicants* of the complete reason are known as *sufficient reasons* for the decision and they correspond to what is known as PI explanations and abductive explanations. In this paper, we refer to the prime *implicates* of a complete reason as *necessary reasons* for the decision. We justify this terminology semantically and show that necessary reasons correspond to what is known as contrastive explanations. We also study the computation of complete reasons for multi-class decision trees and graphs with nominal and numeric features for which we derive efficient, closed-form complete reasons. We further investigate the computation of shortest necessary and sufficient reasons for a broad class of complete reasons, which include the derived closed forms and the complete reasons for Sentential Decision Diagrams (SDDs). We provide an algorithm which can enumerate their shortest necessary reasons in output polynomial time. Enumerating shortest sufficient reasons for this class of complete reasons is hard even for a single reason. For this problem, we provide an algorithm that appears to be quite efficient as we show empirically.

Introduction

Reasoning about the behavior of AI systems has been receiving significant attention recently, particularly the decisions made by machine learning classifiers. Some methods operate directly on classifiers, e.g., (Ribeiro, Singh, and Guestrin 2016, 2018) while others operator on symbolic encodings of their input-output behavior, e.g., (Narodytska et al. 2018; Ignatiev, Narodytska, and Marques-Silva 2019a) which may be compiled into tractable circuits (Chan and Darwiche 2003; Shih, Choi, and Darwiche 2018b, 2019; Shi et al. 2020; Audemard, Koriche, and Marquis 2020; Huang et al. 2021a). When explaining decisions, the notion of a *sufficient reason* has been well investigated. This is a minimal subset of an instance that is sufficient to trigger the decision and can therefore be used to explain why it was made. Sufficient reasons were introduced in (Shih, Choi, and Darwiche 2018b) under the name of *PI explanations* and later referred to as *abductive explanations* (Ignatiev, Narodytska, and Marques-Silva 2019a).[1] Two related notions we discuss later are *contrastive explanations* as formalized in (Ignatiev et al. 2020) and *counterfactual explanations* as formalized in (Audemard, Koriche, and Marquis 2020).[2]

(Darwiche and Hirth 2020) introduced the *complete reason* for a decision as a Boolean formula that characterizes why a decision was made, and showed how it can be used to gather insights about the decision. This includes generating explanations, determining decision bias and evaluating counterfactual queries. For example, it was shown that sufficient reasons correspond to the *prime implicants* of the complete reason. Hence, if one has access to the complete reason behind a decision, then one can abstract the computation of sufficient reasons away from the classifier and its encoding or compilation. Consider a classifier for admitting applicants to an academic program based five Boolean features (Darwiche and Hirth 2020): passing the entrance exam (E), being a first time applicant (F), having good grades (G), having work experience (W) and coming from a rich hometown (R). The positive instances of this classifier are specified by the following Boolean formula: $\Delta = (e \vee g) \wedge (e \vee r) \wedge (e \vee w) \wedge (f \vee r) \wedge (\bar{f} \vee g \vee w)$. Luna ($\delta$) passed the entrance exam, has good grades and work experience, comes from a rich hometown but is not a first time applicant ($\delta = e, \bar{f}, g, r, w$). The classifier will admit Luna. The complete reason for this decision is: $\Gamma = (e \vee g) \wedge (e \vee w) \wedge (r) \wedge (\bar{f} \vee g \vee w)$. There are four prime implicants of Γ: $\{e, g, r\}, \{e, r, w\}, \{e, \bar{f}, r\}$ and $\{g, r, w\}$. Each is a minimal subset of instance δ which is sufficient to trigger the admit decision. Even though the

[1] See, e.g., (Choi, Xue, and Darwiche 2012; Ribeiro, Singh, and Guestrin 2018; Wang, Khosravi, and den Broeck 2021) for some approaches that can be viewed as approximating sufficient reasons and (Ignatiev, Narodytska, and Marques-Silva 2019b) for a study of the quality of some of these approximations.

[2] There is an extensive body of work in philosophy, social science and AI that discusses contrastive explanations and counterfactual explanations; see, e.g., (Garfinkel 1982; Lewis 1986; Temple 1988; Lipton 1990; Wachter, Mittelstadt, and Russell 2017; van der Waa et al. 2018; Miller 2019; Mittelstadt, Russell, and Wachter 2019; Goyal et al. 2019; Verma, Dickerson, and Hines 2020; Mothilal, Sharma, and Tan 2020). While the definitions of these notions are sometimes variations or refinements on one another, they are not always compatible.

number of sufficient reasons may be exponential, the complete reason can be compact and computed in linear time if the classifier is represented using a suitable form (Darwiche and Hirth 2020). Further insights can be obtained about a decision by analyzing its complete reason. For example, the decision on Luna is *biased* as it would be different if she did not come from a rich hometown. In that case, she would be denied admission *because* she does not come from a rich hometown and is not a first time applicant as this would be the only sufficient reason for rejection. These conclusions can be derived by operating directly, and efficiently, on the complete reason as shown in (Darwiche and Hirth 2020).

More recently, (Darwiche and Marquis 2021) introduced the notion of *universal literal quantification* to Boolean logic and used it to formulate complete reasons. According to this formulation, we can obtain the above complete reason Γ by computing $\forall e, \bar{f}, g, r, w \cdot \Delta$, to be explained later. We will base our treatment on this formulation while operating in a discrete instead of a Boolean setting. The conclusion section in (Darwiche and Marquis 2021) proposed a generalization of universal literal quantification to discrete variables but without further discussion. We will adopt this definition, study it further and exploit it to derive efficient, closed-form complete reasons for multi-class decision trees and graphs with nominal (discrete) and numeric (continuous) features. We will show that the obtained complete reasons belong to a particular logical form that arise when explaining the decisions of a broader class of classifiers. We will further show that the *prime implicates* of complete reasons correspond to contrastive explanations, which will provide further insights into the semantics and utility of these explanations. We will refer to these prime implicates as *necessary reasons* for the decision and semantically justify this terminology. We will then propose an output polynomial algorithm for computing the shortest necessary reasons of the identified class of complete reasons. We will finally show that computing shortest sufficient reasons is hard for this class of complete reasons and propose an algorithm for computing them which appears to be quite efficient based on an empirical evaluation. Proofs of all results can be found in (Darwiche and Ji 2022).

Syntax and Semantics of Discrete Formulas

We start by defining the syntax and semantics of discrete formulas which we use to capture classifiers with discrete features. The treatment in this section is largely classical and provides obvious generalizations of what is known on Boolean logic. But we spell it out so we can provide a formal treatment of our upcoming results, especially that we sometimes depart from what may be customary.

For a discrete variable X with values $x_1, \ldots, x_n$, we will call $X = x_i$ a *state* for variable X. A *discrete formula* is defined over a set of discrete variables as follows. Every state or constant ($\top$, $\bot$) is a discrete formula. If α and β are discrete formulas, then $\neg\alpha$, $\alpha \vee \beta$ and $\alpha \wedge \beta$ are discrete formulas. A *positive literal* is a state $X = x_i$ typically denoted by x_i. A *negative literal* is a negated state $\neg(X = x_i)$, typically denoted by $\bar{x}_i$. A negative literal will also be called a state if the variable has only two values. A *clause* is a disjunction of literals with at most one literal per variable. A *term* is a conjunction of literals with at most one literal per variable. A *CNF* is a conjunction of clauses. A *DNF* is a disjunction of terms. An *NNF* is defined as follows. Constants and literals are NNFs. If α and β are NNFs, then $\alpha \vee \beta$ and $\alpha \wedge \beta$ are NNFs (hence, conjunctions and disjunctions cannot be negated). An NNF is $\vee$*-decomposable* iff for each disjunction $\bigvee_i \alpha_i$ in the NNF, the disjuncts α_i do not share variables. An NNF is $\wedge$*-decomposable* iff for each conjunction $\bigwedge_i \alpha_i$ in the NNF, the conjuncts α_i do not share variables. An NNF is *positive* iff it contains only positive literals. Any NNF can be made positive by replacing negative literals $\bar{x}_i$ with $\bigvee_{j \neq i} x_j$. An NNF is *monotone* iff it is positive and does not contain distinct states x_i and x_j for any variable X.

A *positive term* contains only positive literals (i.e., states). The *conditioning* of discrete formula Δ on positive term γ is denoted $\Delta|\gamma$ and obtained as follows. For each state $x_i \in \gamma$, replace the occurrences of x_i with $\top$ and the occurrences of x_j, $j \neq i$, with $\bot$. The formula $\Delta|x_i$ does not mention variable X. An *instance* is a positive term which contains precisely one state for each variable. If we condition a discrete formula on an instance, we get a Boolean formula that does not mention any variables (evaluates to true or false).

The semantics of discrete formulas is symmetric to the semantics of Boolean formulas, except that the notion of a *world* (truth assignment) is now defined as a function that maps each discrete variable to one of its states (a world corresponds to an instance). A world ω *satisfies* a discrete formula α, written $\omega \models \alpha$, precisely when $\alpha|\omega$ evaluates to true. In this case, we say that world ω is a *model* of formula α. Notions such as satisfiability, validity, implication and equivalence can now be defined for discrete formulas as in Boolean logic. For example, formula α implies formula β, written $\alpha \models \beta$, iff every model of α is a model of β. We next define the notions of implicants and implicates. An *implicant* of a discrete formula Δ is a term δ such that $\delta \models \Delta$. The implicant is *prime* iff no other implicant $\delta^\star$ is such that $\delta^\star \subset \delta$. An *implicate* is a clause δ such that $\Delta \models \delta$. The implicate is *prime* iff no other implicate $\delta^\star$ is such that $\delta^\star \subset \delta$.

Our treatment will represent a classifier with discrete features and multiple classes $c_1, \ldots, c_n$ by a set of mutually exclusive and exhaustive discrete formulas $\Delta^1, \ldots, \Delta^n$, where the models of formula Δ^i capture the instances in class c_i. That is, instance δ is in class c_i iff $\delta \models \Delta^i$. We refer to each Δ^i as a *class formula.* When $\delta \models \Delta^i$, we say that instance δ is *decided positively* by Δ^i. The complete reason for this decision will then be the formula $\forall\delta \cdot \Delta^i$. The next section will explain what $\forall\delta$ is and how to compute it efficiently. In the upcoming discussion, we may use the engineering notation for Boolean operators when convenient, writing $x_1y_2 + x_2z_3$, for example, instead of $(x_1 \wedge y_2) \vee (x_2 \wedge z_3)$.

Quantifying States of Discrete Variables

(Darwiche and Marquis 2021) introduced universal literal quantification for Boolean logic and suggested the following generalization to discrete variables without further study.

Definition 1. *For formula Δ and variable X with states $x_1, \ldots, x_n$, the universal quantification of state x_i from Δ is defined as follows:* $\forall x_i \cdot \Delta = (\Delta|x_i) \wedge \bigwedge_{j \neq i} (x_i \vee \Delta|x_j)$.

Quantification is commutative so we can equivalently write $\forall x \cdot (\forall y \cdot \Delta)$, $\forall y \cdot (\forall x \cdot \Delta)$ or $\forall \{x, y\} \cdot \Delta$. We will study Definition 1 and exploit it for computing complete reasons.

Definition 2. *If instance δ is decided positively by class formula Δ, then $\forall \delta \cdot \Delta$ is the 'complete reason' for the decision.*

The next three results parallel Boolean ones in (Darwiche and Marquis 2021). They are followed by two novel results.

Proposition 1. *We have $\forall x_i \cdot \top = \top$ and $\forall x_i \cdot \bot = \bot$; $\forall x_i \cdot x_i = x_i$ and $\forall x_i \cdot \bar{x}_i = \bot$; $\forall x_i \cdot x_j = \bot$ and $\forall x_i \cdot \bar{x}_j = x_i$ when $j \neq i$; $\forall x_i \cdot y_j = y_j$ and $\forall x_i \cdot \bar{y}_j = \bar{y}_j$ when $X \neq Y$.*

The next result shows when $\forall x_i$ can be distributed.

Proposition 2. *For discrete formulas α, β and state x_i of variable X, we have $\forall x_i \cdot (\alpha \wedge \beta) = (\forall x_i \cdot \alpha) \wedge (\forall x_i \cdot \beta)$. Moreover, if variable X does not occur in both α and β, then $\forall x_i \cdot (\alpha \vee \beta) = (\forall x_i \cdot \alpha) \vee (\forall x_i \cdot \beta)$.*

Given Propositions 1 and 2, we can universally quantify states out of $\vee$-decomposable NNFs in linear time while preserving $\vee$-decomposability in the resulting NNF.

Proposition 3. *Let Δ be a $\vee$-decomposable NNF and γ be a set of states. Then $\forall \gamma \cdot \Delta$ can be obtained from Δ as follows. For each state $x_i \in \gamma$, replace the occurrences of literals $\bar{x}_i$, x_j and $\bar{x}_j$, $j \neq i$, in Δ with $\bot$, $\bot$ and x_i, respectively.*

Consider the class formula $\Delta = \bar{x}_1(x_2 + \bar{y}_1)(\bar{y}_1 + z_1)$ over ternary variables X, Y, Z and instance $\delta = x_2, y_2, z_1$ which is decided positively by Δ. The complete reason for this decision is $\forall \delta \cdot \Delta$. Since Δ is $\vee$-decomposable, Proposition 3 gives $\forall x_2, y_2, z_1 \cdot \Delta = (x_2)(x_2 + y_2)(y_2 + z_1) = x_2(y_2 + z_1)$. Hence, this instance was decided positively because it has characteristic x_2 and one of the characteristics y_2 and z_1.

We next identify conditions that allow the distribution of $\forall x_i$ over disjuncts that share variables.

Proposition 4. *Consider positive NNFs α, β and state x_i of variable X. If x_i does not occur in α, β, or x_j does not occur in α, β for all $j \neq i$, then $\forall x_i \cdot (\alpha \vee \beta) = (\forall x_i \cdot \alpha) \vee (\forall x_i \cdot \beta)$.*

For a Boolean variable X with states x and $\bar{x}$, Proposition 4 says that we can distribute $\forall x$ over disjuncts α and β even if they mention literal $\bar{x}$ (but do not mention x). This is a novel result compared to (Darwiche and Marquis 2021).

Next is another novel condition that licenses the distribution of $\forall x_i$ over disjuncts, which we use to derive closed forms for the complete reasons of decision trees and graphs.

Proposition 5. *Let α be an NNF, S be a set of states for variable X and $\beta = \bigvee_{x_k \in S} x_k$. If variable X occurs in α only in disjunctions of the form $\bigvee_{x_k \in S'} x_k$ where $S' \supseteq S$ are states of variable X, then $\forall x_i \cdot (\alpha \vee \beta) = (\forall x_i \cdot \alpha) \vee (\forall x_i \cdot \beta)$.*

Consider variables X $(x_1, \ldots, x_4)$ and Y (y_1, y_2) and the formulas $\alpha = y_1(x_1 + x_2 + x_4)$ and $\beta = x_1 + x_2$. We can invoke Proposition 5 to distribute $\forall x_1$ using $S = \{x_1, x_2\}$ and $S' = \{x_1, x_2, x_4\}$. Hence, $\forall x_1 \cdot (\alpha \vee \beta) = \forall x_1 \cdot (y_1(x_1 + x_2 + x_4)) \vee \forall x_1 \cdot (x_1 + x_2) = y_1 x_1 + x_1 = x_1$. Propositions 2 and 4 do not license this distribution of $\forall x_i$ though.

The Complete Reasons for Decision Graphs

We next provide closed forms for the complete reasons of decision graphs, which subsume decision trees, in the form of monotone, $\vee$-decomposable NNFs. This will later facilitate the computation of their prime implicants and implicates (sufficient and necessary reasons). We first treat multi-class decision graphs with nominal features and then treat decision graphs with numeric features; see Figures 1 and 3.

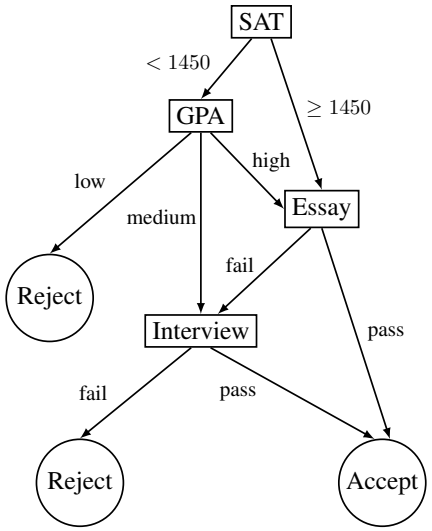

Figure 1: A classifier in the form of a decision graph.

Each leaf node in a decision graph is labeled with some class c. Moreover, each internal node T in the graph has outgoing edges $\xrightarrow{X,\, S_1} T_1, \ldots, \xrightarrow{X,\, S_n} T_n$, $n \geq 2$. We say in this case that node T *tests* variable X. The children of node T are $T_1, \ldots, T_n$ and $S_1, \ldots, S_n$ is a partition of *some* states of variable X. A decision graph will be represented by its root node. Hence, each node in the graph represents a smaller decision graph. We allow variables to be tested more than once on a path from the root to a leaf but under the following condition, which we call the *weak test-once property*. Consider path $\ldots, T \xrightarrow{X,\, S_j} T_j, \ldots, T' \xrightarrow{X,\, R_k} T_k, \ldots$ from the root to leaf and suppose that nodes T and T' test variable X. If no nodes between T and T' on the path test variable X, then $\{R_k\}_k$ must be a partition of states S_j. Moreover, if T is the first node that tests X on the path, then $\{S_j\}_j$ must be a partition of *all* states for X. For binary variables, the weak test-once property reduces to the standard test-once property: A variable can be tested at most once on any path from the root to a leaf. The weak test-once property is critical for treating numeric features. As we show later, one can easily discretize continuous variables based on the thresholds used at decision nodes, which leads to decision graphs that satisfy the weaker test-once property but not the standard one.

A decision graph classifies an instance δ as follows. Suppose $\delta[X]$ is the state of variable X in instance δ. We start at the graph root and repeat the following. When we are at node T that has outgoing edges $\xrightarrow{X,\, S_1} T_1, \ldots, \xrightarrow{X,\, S_n} T_n$, we follow the (unique) edge $\xrightarrow{X,\, S_i} T_i$ which satisfies $\delta[X] \in S_i$. This process leads us to a unique leaf node. The label c of this leaf node is then the class assigned to instance δ by the decision graph (that is, instance δ belongs to class c).

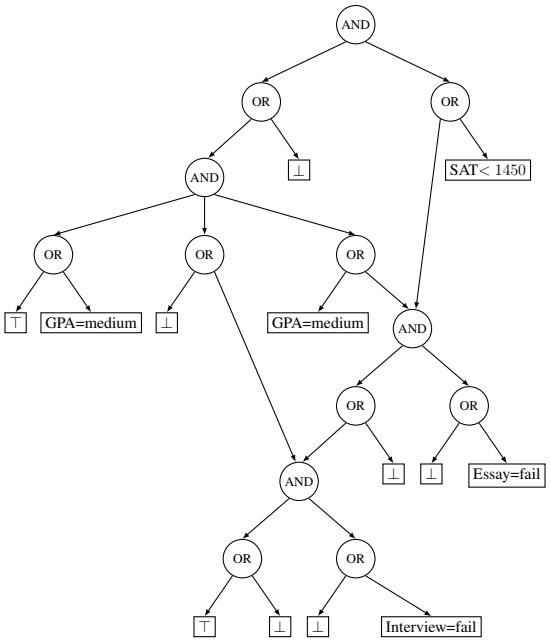

Figure 2: A complete reason constructed by Proposition 7 for the decision graph in Figure 1 and instance SAT < 1450, GPA=medium, Essay=fail, Interview=fail. This complete reason is in the form of a monotone, $\vee$-decomposable NNF.

We next provide a closed-form NNF that captures the instances belonging to some class c in a decision graph.

Definition 3. *The NNF for a decision graph T and class c is denoted $\Delta^c[T]$ and defined inductively as follows:*

$$\Delta^c[T] = \begin{cases} \top & \text{if } T \text{ has class } c \\ \bot & \text{if } T \text{ has class } c' \neq c \\ \bigwedge_j (\Delta^c[T_j] \vee \bigvee_{x_i \notin S_j} x_i) & \text{if } T \text{ has edges } \xrightarrow{X,\, S_j} T_j \end{cases}$$

Proposition 6. *For decision graph T, class c and instance δ, we have $\delta \models \Delta^c[T]$ iff T assigns class c to instance δ.*

This NNF is positive and can be constructed in linear time but is not $\vee$-decomposable: The disjuncts in $\bigvee_{x_j \notin S_i} x_j$ share variables and variable X will appear in $\Delta^c[T_j]$ if tested again in graph T_j. Yet, this NNF is tractable for universal quantification as revealed in the proof of the next result, which provides closed-form complete reasons for decision graphs.

Proposition 7. *Let T be a decision graph, δ be an instance in class c and $\delta[X]$ be the state of variable X in instance δ. The complete reason $\forall \delta \cdot \Delta^c[T]$ is given by the NNF:*

$$\Gamma^c[T] = \begin{cases} \top & \text{if } T \text{ has class } c \\ \bot & \text{if } T \text{ has class } c' \neq c \\ \bigwedge_j (\Gamma^c[T_j] \vee \ell_j) & \text{if } T \text{ has edges } \xrightarrow{X,\, S_j} T_j \end{cases} \quad (1)$$

where $\ell_j = \delta[X]$ if $\delta[X] \in S_k$ for some $k \neq j$, else $\ell_j = \bot$.

Consider the decision graph in Figure 1 and an applicant who scored < 1450 on the SAT, had a medium GPA and did not pass their essay or interview. This applicant is rejected by the classifier and the complete reason for the decision, as constructed by Proposition 7, is shown in Figure 2.

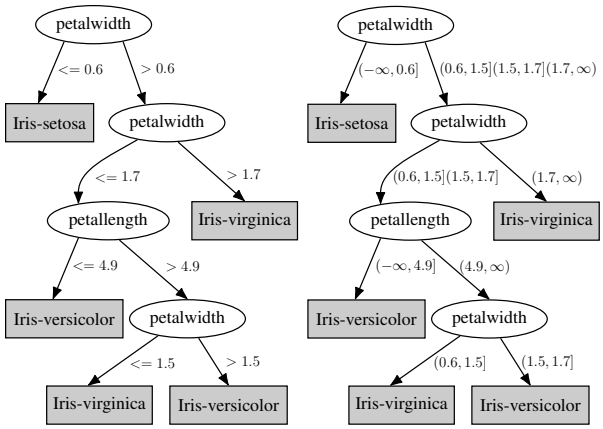

Figure 3: A decision tree with continuous variables learned using weka (left) and its discretization (right).

Proposition 8. *Let T be a decision graph and δ be an instance in class c. The complete reason $\forall \delta \cdot \Delta^c[T]$ in Equation 1 is an NNF that is monotone and $\vee$-decomposable.*

Even though we are working with decision graphs that include discrete variables and multiple classes, we get complete reasons in the form of monotone NNFs, which are effectively Boolean NNFs. This will simplify the computation of necessary and sufficient reasons in later sections, as it allows us to avoid certain complications that can arise when binarizing discrete variables; see, e.g., (Choi et al. 2020).

Numeric Features

Suppose we have a continuous variable X that is being tested at node T in a decision graph. The test will have the form $X \leq t_i$, where t_i is a threshold in $(-\infty, \infty)$. Node T will then have two outgoing edges, one is followed when $X \leq t_i$ (high edge) and the other is followed when $X > t_i$ (low edge); see Figure 3. Suppose now that $t_1, \ldots, t_n$ is the set of all thresholds for variable X in the decision graph and assume that these thresholds are in increasing order. We can then treat variable X as a discrete variable with the following $n + 1$ states: $(-\infty, t_1], (t_1, t_2], \ldots, (t_{n-1}, t_n], (t_n, \infty)$. If variable X is being tested first at node T, we label the high edge of node T with states $(-\infty, t_1], (t_1, t_2], \ldots, (t_{i-1}, t_i]$ and its low edge with states $(t_i, t_{i+1}], \ldots, (t_{n-1}, t_n], (t_n, \infty)$. Consider Figure 3 (left). Variable "petalwidth" (W) has three thresholds $0.6, 1.5, 1.7$, leading to four discrete states $S_W = (-\infty, 0.6]$, $(0.6, 1.5]$, $(1.5, 1.7]$, $(1.7, \infty)$. Variable "petallength" (L) has one threshold 4.9, leading to two discrete states $S_L = (-\infty, 4.9], (4.9, \infty)$. Variable W is tested three times in the decision tree. The first test ($W \leq 0.6$) splits states S_W into $S_1 = (-\infty, 0.6]$ for the high edge and $S_2 = (0.6, 1.5], (1.5, 1.7], (1.7, \infty)$ for the low edge. The second test ($W \leq 1.7$) splits S_2 into $S_{21} = (0.6, 1.5], (1.5, 1.7]$ for the high edge and $S_{22} = (1.7, \infty)$ for the low edge. The third and final test ($W \leq 1.5$) splits states S_{21} into $(0.6, 1.5]$ and $(1.5, 1.7]$. The resulting decision tree with discrete variables *does not* satisfy the test-once property but does satisfy

the weak test-once property as shown in Figure 3(right).

Consider now instance $\delta_1 : W = 0.8, L = 5.3$ which is classified as "Iris-virginica" by the decision tree with continuous variables (T_1). We can view this instance as the discrete instance $\delta_2 : W = (0.6, 1.5], L = (4.9, \infty)$ since $0.8 \in (0.6, 1.5]$ and $5.3 \in (4.9, \infty)$. The decision tree with discrete variables (T_2) will also classify instance δ_2 as "Iris-virginica." A continuous instance and its corresponding discrete instance will be classified identically by decision trees T_1 and T_2 because T_1 cannot discriminate continuous values that belong to the same interval. Finally, to generate the complete reason for instance δ_1, we compute $\forall \delta_2 \cdot \Delta^c[T_2]$ using Proposition 7 where c is class "Iris-virginica."

Further Extensions

The closed-form complete reason in Proposition 7 applies directly to *Free Binary Decision Diagrams (FBDDs)* (Gergov and Meinel 1994) and *Ordered Binary Decision Diagrams (OBDDs)* (Bryant 1986) as they are special cases of decision graphs. FBDDs use binary variables and binary classes ($\top$ and $\bot$). OBDDs are a subset of FBDDs which test variables in the same order along any path from the root to a leaf. We can similarly obtain closed forms for the complete reasons of *Sentential Decision Diagrams (SDDs)* (Darwiche 2011), which test on formulas (sentences) instead of variables. This is possible since given an SDD for Δ we can obtain an SDD for $\neg\Delta$ in linear time. An SDD Δ is an $\wedge$-decomposable NNF that represents instances for class $\top$. The SDD for $\neg\Delta$ is also an $\wedge$-decomposable NNF but represents instances for class $\bot$. If we negate Δ and $\neg\Delta$ using deMorgan's law, we obtain $\vee$-decomposable NNFs for classes $\bot$ and $\top$, respectively. This allows us to obtain a closed-form, monotone, $\vee$-decomposable complete reason for any instance using universal quantification. (Darwiche and Marquis 2021) showed that an SDD can be universally quantified in linear time. Earlier, (Darwiche and Hirth 2020) showed that Decision-DNNFs (Huang and Darwiche 2007) can be universally quantified in linear time as well.[3] Decision-DNNFs cannot be negated efficiently so they do not permit closed-form complete reasons unless we have Decision-DNNFs for classes $\top$ and $\bot$. While decision tree classifiers are normally learned from data, classifiers such as OBDDs and SDDs are compiled from other classifiers like Bayesian/neural networks and random forests; see, e.g., (Shih, Choi, and Darwiche 2019; Shi et al. 2020; Choi et al. 2020). The relative succinctness of these representations of classifiers has been well studied. FBDDs are a subset of Decision-DNNFs and there is a quasipolynomial simulation of Decision-DNNFs by equivalent FBDDs (Beame et al. 2013). SDDs and FBDDs are not comparable (Beame and Liew 2015; Bollig and Buttkus 2019) so SDDs and Decision-DNNFs are not comparable either. SDDs are exponentially more succinct than OBDDs (Bova 2016).

[3](Darwiche and Hirth 2020) introduced two linear-time operations on Decision-DNNFs: *consensus* and *filtering*. These operations implement universal literal quantification as shown in (Darwiche and Marquis 2021). Decision-DNNFs are $\wedge$-decomposable NNFs in which disjunctions have the form $(x \wedge \alpha) \vee (\bar{x} \wedge \beta)$.

Necessary and Sufficient Reasons

As mentioned earlier, the prime implicants of a complete reason can be interpreted as *sufficient reasons* for the decision. We next show that the prime implicates of a complete reason can be interpreted as *necessary reasons* for the decision and correspond to contrastive explanations (Ignatiev et al. 2020). We first provide further insights into complete reasons which will help in justifying this interpretation.

Definition 4. *Instances δ_1 and δ_2 are 'congruent' iff $\delta_1 \cap \delta_2 \models \Delta$ for some class formula Δ. We also say in this case that the decisions on instances δ_1 and δ_2 are congruent.*

If instances δ_1 and δ_2 are congruent, they must belong to the same class since $\delta_1 \models \Delta$ and $\delta_2 \models \Delta$ so they are decided similarly. Moreover, their common characteristics $\delta_1 \cap \delta_2$ are sufficient to justify the decision. That is, the decisions on them are equal and have a common justification.

Proposition 9. *Let $\forall \delta \cdot \Delta$ be the complete reason for the decision on instance δ. Then instance $\delta^\star$ is congruent to instance δ iff $\delta^\star \models \forall \delta \cdot \Delta$.*

Hence, the complete reason $\forall \delta \cdot \Delta$ captures all, and only, instances that are congruent to instance δ. The complete reason does not capture all instances that are decided similarly to δ since some of these instances may be decided that way for a different reason (the decisions are not congruent).

Consider the class formula $\Delta = \bar{x}_1(x_2 + \bar{y}_1)(\bar{y}_1 + z_1)$ over ternary variables X, Y and Z. The instance $\delta = x_2 y_2 z_1$ is decided positively by this formula ($\delta \models \Delta$) and the complete reason for this decision is $\forall x_2, y_2, z_1 \cdot \Delta = x_2(y_2 + z_1)$. There are four other instances that satisfy this complete reason, $x_2 y_2 z_2$, $x_2 y_2 z_3$, $x_2 y_1 z_1$ and $x_2 y_3 z_1$. All are decided positively by Δ and the states each share with instance δ justify the decision. Instance $x_3 y_2 z_1$ is also decided positively by Δ but for a different reason: the states $y_2 z_1$ it shares with instance δ do not justify the decision, $y_2 z_1 \not\models \Delta$. Hence, this instance is not captured by the complete reason for δ.

Implicants and Implicates as Reasons

We next review the interpretation of prime implicants as sufficient reasons and discuss the interpretation of prime implicates as necessary reasons for a decision. We will represent these notions by sets of literals, which are interpreted as conjunctions for prime implicants (terms) and as disjunctions for prime implicates (clauses).

Proposition 10. *The prime implicants and prime implicates of a complete reason $\forall \delta \cdot \Delta$ are subsets of instance δ.*

A prime implicant σ of the complete reason $\forall \delta \cdot \Delta$ can be viewed as a sufficient reason for the underlying decision as it is a minimal subset of instance δ that is guaranteed to sustain the decision, congruently. If we change any part of the instance but for σ, the decision will stick and for a common reason since the new and old instances are congruent. Consider the complete reason in Figure 2 which corresponds to a reject decision on the instance SAT < 1450, GPA=medium, Essay=fail, Interview=fail. There are two prime implicants for this complete reason: {SAT < 1450, GPA=medium, Interview=fail}

and {Essay=fail, Interview=fail}. Each of these prime implicants is a minimal subset of the instance that is sufficient to trigger the reject decision.

A prime implicate σ of the complete reason $\forall\delta \cdot \Delta$ can be viewed as a necessary reason for the underlying decision as it is a minimal subset of the instance that is essential for sustaining a congruent decision. If we change all states in σ, the decision on the new instance will be different or will be made for a different reason since the new and old instances will not be congruent (we provide a stronger semantics later). Consider again the complete reason in Figure 2 and the corresponding instance and reject decision. There are three prime implicates for this complete reason: {Interview=fail}, {SAT < 1450, Essay=fail} and {GPA=medium, Essay=fail}. Changing Interview to pass will change the decision. Changing SAT to ≥ 1450 and Essay to pass will also change the decision. Since GPA is a ternary variable, there are two ways to change its value. If we change GPA and Essay to high and pass, respectively, the decision will change. But if we change these features to low and pass, respectively, the decision will not change but the new instance (SAT < 1450, GPA=low, Essay=pass, Interview=fail) will not be congruent with the original instance (SAT < 1450, GPA=medium, Essay=fail, Interview=fail). That is, the common characteristics of these instances {SAT < 1450, Interview=fail} cannot on their own justify the reject decision.

For yet another example, consider again class formula $\Delta = \bar{x}_1(x_2 + \bar{y}_1)(\bar{y}_1 + z_1)$ over ternary variables X, Y and Z. The complete reason for positive instance $\delta = x_2y_2z_1$ is $\Gamma = \forall\delta \cdot \Delta = x_2(y_2 + z_1)$. The prime implicants of Γ are x_2y_2 and x_2z_1, which are the sufficient reasons for the decision. If we change instance δ while keeping one of these reasons intact, the decision sticks. The prime implicates of Γ are x_2 and $y_2 + z_1$, which are the necessary reasons for the decision. If we violate one of these reasons, the decision will be different or made for a different reason. Changing instance δ to $x_2y_1z_3$ violates the necessary reason $y_2 + z_1$, which leads to a negative decision. Changing the instance to $\delta^\star = x_3y_2z_1$ violates the necessary reason x_2. The decision remains positive though but for a different reason than why δ is positive. That is, the common characteristics $\delta \cap \delta^\star = y_2z_1$ do not justify the decision on these instances, $y_2z_1 \not\models \Delta$.

More on Necessity

We next show that necessary reasons correspond to *basic* contrastive explanations as formalized in (Ignatiev et al. 2020) using the following definition (modulo notation).

Definition 5. *Let δ be an instance decided positively by class formula Δ. A 'contrastive explanation' of this decision is a minimal subset γ of instance δ such that $\delta \setminus \gamma \not\models \Delta$.*

That is, it is possible to change the decision on instance δ by *only* changing the states in γ. Moreover, we must change *all* states in γ for the decision to change.

Proposition 11. *Let δ be an instance decided positively by class formula Δ. Then γ is a prime implicate of the complete reason $\forall\delta \cdot \Delta$ iff γ is a contrastive explanation.*

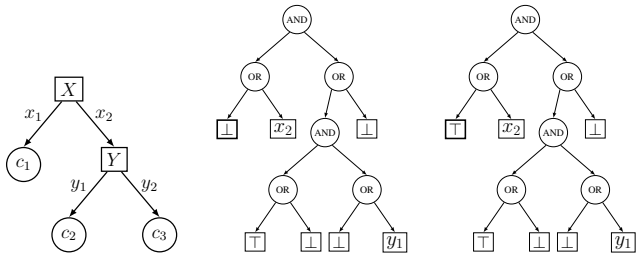

Figure 4: (a) decision tree (b) complete reason for "why x_2y_1 is c_2" (c) complete reason for "why x_2y_1 is not c_3."

This correspondence is perhaps not too surprising given the duality between abductive and contrastive explanations (Ignatiev et al. 2020) and the classical duality between prime implicants and prime implicates. However, it does provide further insights into contrastive explanations: changing the states of a contrastive explanation leads to a non-congruent decision. It also provides further insights on the necessity of prime implicates: while violating a necessary reason will only lead to an instance that is not congruent (decided differently or for a different reason), there must exist at least one violation of each necessary reason which is guaranteed to change the decision. This follows directly from Definition 5. If the variables of a necessary reason are all binary, there is only one way to violate the reason (by negating each variable in the reason). In this case, violating the necessary reason is guaranteed to change the decision.

For an example, let us revisit the complete reason in Figure 2 and the corresponding instance and reject decision. This decision has three necessary reasons: {Interview=fail}, {SAT < 1450, Essay=fail} and {GPA=medium, Essay=fail}. There is only one way to violate each of the first two reasons, and each violation leads to reversing the decision as we saw earlier. There are two ways to violate the third necessary reason. One of these violations (GPA=high, Essay=pass) reverses the decision but the other violation (GPA=low, Essay=pass) keeps the reject decision intact (but for a different reason).

In summary, a necessary reason (contrastive explanation) identifies a minimal subset of the instance which is guaranteed to change the decision if that subset is altered *properly.* The minimality condition ensures that we must alter every variable in a necessary reason to change the decision, but it does not specify how to alter it (except for binary variables). We will revisit this distinction when we discuss counterfactual explanations (Audemard, Koriche, and Marquis 2020).

Targeting a Particular Class

Beyond basic contrastive explanations, (Ignatiev et al. 2020) discussed *targeted* contrastive explanations which aim to change the instance class from c to some class $c^\star$; see (Lipton 1990; Miller 2019). This notion is particularly relevant to multi-class classifiers as it reduces to basic contrastive explanations when the classifier has only two classes. Targeted contrastive explanations can be obtained using the complete reason for *why* the instance was classified as *not* $c^\star$ (that is,

a class other than $c^\star$). This complete reason can be obtained using a slight modification of Equation 1 where we modify the first two conditions as follows:

$$\Gamma^c[T] = \begin{cases} \top & \text{if } T \text{ has class } c' \neq c^\star \\ \bot & \text{if } T \text{ has class } c^\star \\ \bigwedge_j (\Gamma^c[T_j] \vee \ell_j) & \text{if } T \text{ has edges } \xrightarrow{X,\, S_j} T_j \end{cases} \quad (2)$$

The prime implicates for this complete reason (i.e., necessary reasons) will then identify minimal subsets of the instance that lead to the targeted class $c^\star$, if altered properly.

Consider the decision tree in Figure 4(a) which has two binary features X and Y and three classes c_1, c_2, c_3. The instance x_2y_1 is classified as c_2. The complete reason for this decision, as computed by Equation 1, is shown in Figure 4(b) and has two necessary reasons x_2 and y_1. If we violate the first necessary reason ($x_2 \rightarrow x_1$), the class changes to c_1. If we violate the second necessary reason ($y_1 \rightarrow y_2$), the class changes to c_3. Suppose now we wish to change the class c_2 of this instance particularly to c_3. The complete reason for "why not c_3," as computed by Equation 2, is shown in Figure 4(c) and has only one necessary reason, y_1. Violating this necessary reason is guaranteed to change the class to c_3.

(Audemard, Koriche, and Marquis 2020) discussed the complexity of computing the related notion of *counterfactual explanations* which are defined as follows. Given an instance δ in class c, find an instance $\delta^\star$ in a different class $c^\star$ that is as close as possible to instance δ with respect to the hamming distance. In other words, instance $\delta^\star$ must maximize the number of characteristics it shares with instance δ. Consider now the characteristics γ of instance δ that do not appear in instance $\delta^\star$ ($\gamma = \delta \setminus \delta^\star$). Changing these characteristics to $\gamma^\star = \delta^\star \setminus \delta$ will change the class from c to $c^\star$. Hence, characteristics γ are a *length-minimal* subset of instance δ which, if changed properly, will guarantee a change from class c to class $c^\star$. Every characteristic of γ must be changed to ensure this class change, otherwise $\delta^\star$ would not be a counterfactual explanation. Moreover, when the features are binary, there is only one way to change the characteristics γ so the class will change from c to $c^\star$; that is, by flipping every characteristic in γ to yield $\gamma^\star$. In this case, counterfactual explanations are in one-to-one correspondence with the shortest necessary reasons which we discuss next.

Computing Shortest Explanations

A complete reason may have too many prime implicants and implicates. We will therefore provide algorithms for computing the *shortest* implicants and implicates (which must be prime) for monotone, $\vee$-decomposable NNFs. As discussed earlier, we can in linear time obtain complete reasons in this form for decision graphs and SDDs, which include FBDDs, OBDDs and decision trees as special cases.

Shortest Necessary Reasons. This will be an output polynomial algorithm that is based on three (conceptual) passes on the complete reason which we describe next.

Definition 6. *The 'implicate minimum length (*IML*)' of a valid formula is ∞. For non-valid formulas, it is the minimum length attained by any implicate of the formula.*

The first pass computes the implicate minimum length.

Algorithm 1: Shortest Necessary Reasons (SNRs)

Input: monotone and $\vee$-decomposable NNF Δ with no constants
Output: all shortest implicates of NNF Δ

1: **function** SNR(Δ) ▷ CACHE initialized to NIL
2: **if** CACHE(Δ) $\neq$ NIL **then return** CACHE(Δ)
3: **else if** $\Delta = x_i$ **then** $snr \leftarrow \{\{x_i\}\}$
4: **else if** $\Delta = \alpha_1 \vee \ldots \vee \alpha_n$ **then**
5: $snr \leftarrow \text{SNR}(\alpha_1) \times \ldots \times \text{SNR}(\alpha_n)$
6: **else** $\Delta = \alpha_1 \wedge \ldots \wedge \alpha_n$
7: $snr \leftarrow \bigcup_{\text{IML}(\alpha_i)=\text{IML}(\Delta)} \text{SNR}(\alpha_i)$
8: CACHE(Δ) $\leftarrow snr$
9: **return** snr

Algorithm 2: Shortest Sufficient Reasons (SSRs)

Input: monotone and $\vee$-decomposable NNF Δ with no constants
Output: all shortest implicants of NNF Δ

1: **function** SSR(Δ) ▷ CACHE initialized to NIL
2: $k \leftarrow 0$
3: **repeat**
4: $ssr = \text{IMP}(\Delta, k, \{\})$
5: $k \leftarrow k + 1$
6: **until** $ssr \neq \{\}$
7: **return** ssr
8: **function** IMP(Δ, k, σ) ▷ computes k-implicants for $\Delta|\sigma$
9: **if** CACHE(Δ, k, σ) $\neq$ NIL **then return** CACHE(Δ, k, σ)
10: $\Sigma \leftarrow \{\}$
11: **if** $\Delta = x$ **then**
12: **if** $x \in \sigma$ **then** $\Sigma \leftarrow \{\{\}\}$ ▷ $\Delta|\sigma$ valid
13: **else if** $k \geq 1$ **then** $\Sigma \leftarrow \{\{x\}\}$
14: **else if** $\Delta = \alpha_1 \vee \ldots \vee \alpha_n$ **then**
15: **for** $\Sigma_i \leftarrow \text{IMP}(\alpha_i, k, \sigma)$ **do**
16: **if** $\Sigma_i \neq \{\{\}\}$ **then** $\Sigma \leftarrow \Sigma \cup \Sigma_i$ ▷ $\alpha_i|\sigma$ not valid
17: **else** $\Sigma \leftarrow \{\{\}\}$; **break** ▷ $\Delta|\sigma$ valid
18: **else** $\Delta = \alpha_1 \wedge \ldots \wedge \alpha_n$
19: **for** $\sigma_1 \in \text{IMP}(\alpha_1, k, \sigma)$ **do**
20: **for** $\sigma_2 \in \text{IMP}(\alpha_2 \wedge \ldots \wedge \alpha_n, k - |\sigma_1|, \sigma \cup \sigma_1)$ **do**
21: $\Sigma \leftarrow \text{REMOVE\_SUBSUMED}(\Sigma \cup \{\sigma_1 \cup \sigma_2\})$
22: CACHE(Δ, k, σ) $\leftarrow \Sigma$
23: **return** Σ

Proposition 12. *The* IML *of a monotone, $\vee$-decomposable NNF is computed as follows:* $\text{IML}(\top) = \infty$, $\text{IML}(\bot) = 0$, $\text{IML}(x_i) = 1$, $\text{IML}(\alpha \vee \beta) = \text{IML}(\alpha) + \text{IML}(\beta)$ *and* $\text{IML}(\alpha \wedge \beta) = \min(\text{IML}(\alpha), \text{IML}(\beta))$.

The second pass prunes the NNF using the IML of nodes.

Proposition 13. *Let* $\text{PRUNE}(\Delta)$ *be the NNF obtained from monotone, $\vee$-decomposable NNF Δ by dropping α_i from conjunctions $\alpha = \alpha_1 \wedge \ldots \wedge \alpha_n$ if* $\text{IML}(\alpha_i) > \text{IML}(\alpha)$. *Then* $\text{PRUNE}(\Delta)$ *is a monotone, $\vee$-decomposable NNF and its prime implicates are the shortest implicates of Δ.*

The third pass computes the prime implicates of NNF $\text{PRUNE}(\Delta)$ in output polynomial time. Algorithm 1 implements the second and third passes assuming the first, linear-time pass has been performed. It represents an implicate by a set of literals and uses the Cartesian product operation on sets of implicates: $S_1 \times S_2 = \{\sigma_1 \cup \sigma_2 \mid \sigma_1 \in S_1, \sigma_2 \in S_2\}$. Algorithm 1 applies the second pass implicitly by excluding conjuncts on Line 7. This is the standard procedure for com-

puting the prime implicates of a monotone NNF, but with no subsumption checking which is critical for its complexity. In the standard procedure, one must ensure that the implicates computed on Lines 5 and 7 are reduced: no implicate σ_1 subsumes another σ_2 ($\sigma_1 \subseteq \sigma_2$).[4] Since NNF PRUNE($\Delta$) is $\vee$-decomposable, the disjuncts $\alpha_1, \ldots, \alpha_n$ on Line 5 do not share variables. Hence, if every SNR(α_i) is reduced, their Cartesian product is reduced. Moreover, due to pruning in the second pass, the implicates SNR(α_i) computed on Line 7 all have the same length so no subsumption is possible.

Proposition 14. *Let Δ be a monotone, $\vee$-decomposable NNF with M shortest implicates, N nodes and E edges. The time complexity of* SNR(Δ) *in Algorithm 1 is $O(M \cdot E)$ and its space complexity is $O(M \cdot N)$.*

We obtain a tighter complexity if we apply Algorithm 1 to the closed-form complete reasons of decision trees given by Proposition 7, due to the following bound on the number of prime implicates (a superset of shortest implicates).

Proposition 15. *For a decision tree, the complete reason for an instance in class c has $\leq L$ prime implicates, where L is the number of leaves in the tree labeled with a class $c' \neq c$.*

The complete reason for a decision tree T has $O(|T|)$ nodes and edges, where $|T|$ is the decision tree size (see Proposition 7). The time and space complexity of Algorithm 1 is then $O(|T| \cdot L)$ for decision trees.

(Huang et al. 2021b) showed that the number of contrastive explanations is linear in the decision tree size. Proposition 15 tightens this result by providing a more specific bound. For a decision T with binary variables and binary classes, (Audemard et al. 2021) showed that the set of all contrastive explanations can be computed in time polynomial in $|T| + n$, where n is the number of variables. Algorithm 1 comes with a tighter complexity for the computation of shortest contrastive explanations for decision trees and applies to multi-class decision trees with discrete features. Another related complexity result is that counterfactual explanations, as discussed earlier, can be enumerated with polynomial delay if the classifier satisfies some conditions as stated in (Audemard, Koriche, and Marquis 2020).

We finally observe that if an NNF is monotone and $\wedge$-decomposable, then one can develop a dual of Algorithm 1 for computing the shortest prime implicants of the NNF.

Shortest Sufficient Reasons We next present Algorithm 2 for computing the shortest implicants of monotone, $\vee$-decomposable NNFs which is a hard task. For decision trees, the problem of deciding whether there exists a sufficient reason of length $\leq k$ is NP-complete (Barceló et al. 2020). Since decision trees have closed-form complete reasons that are monotone and $\vee$-decomposable, computing the shortest implicants for this class of NNFs is hard. (Audemard et al. 2021) showed that the number of shortest sufficient reasons for decision trees can be exponential and provided an incremental algorithm for computing the shortest sufficient reasons for decision trees with binary variables and binary classes, based on a reduction to the PARTIAL MAXSAT problem. Algorithm 2 has a broader scope, does not require a reduction and is based on two key techniques.

The first technique is to compute all unsubsumed implicants of length $\leq k$, called k-implicants, starting with $k = 0$. If no implicants are found, k is incremented and the process is repeated. The second technique relates to computing the k-implicants of a conjunction $\alpha \wedge \beta$. If we have the k-implicants S for α and the k-implicants R for β, we can compute the Cartesian product $S \times R$ and keep unsubsumed implicants of length $\leq k$. Algorithm 2 does something more refined. It first computes the k-implicants S for α. For each implicant $\sigma \in S$, it then computes and accumulates the k'-implicants for $\beta|\sigma$ where $k' = k - |\sigma|$. These techniques control the number of generated k-implicants at each NNF node (smaller k leads to fewer k-implicants). Our implemented caching scheme on Lines 9 & 22 exploits the following properties. If the k-implicants for $\Delta|\sigma$ are $\{\{\}\}$, then these are also its j-implicants for all j. Further, if we cached the k-implicants for $\Delta|\sigma$, then we can use them to retrieve its j-implicants for any $j \leq k$ by selecting implicants of length $\leq j$. We empirically evaluate Algorithms 1 & 2 next.

Empirical Evaluation. Table 1 depicts an empirical evaluation on decision trees learned from OpenML datasets (Vanschoren et al. 2013) and binary decision graphs compiled from Bayesian network classifiers (Shih, Choi, and Darwiche 2018a, 2019). The decision trees were learned by WEKA (Frank et al. 2010) using python-weka-wrapper3 available at pypi.org. We used WEKA's J48 classifier with default settings, which learns pruned C4.5 decision trees with numeric and nominal features (Quinlan 1993). Each dataset was split using WEKA into training (85%) and testing (15%) data. We aimed for OpenML datasets with more than 100 features since many smaller datasets we tried were very easy, but we kept a few smaller ones since they are commonly reported on (adult, compas, spambase). Some of the learned decision trees had significantly fewer variables than the corresponding datasets (e.g., gisette has 5000 features but the learned decision tree has 111). The decision graphs we used are the reportedly largest ones compiled by (Shih, Choi, and Darwiche 2018a, 2019). For each decision tree, we computed reasons for decisions on 1000 instances sampled from testing data (or all testing data if smaller than 1000). We tried random instances but they were much easier. For each decision graph, we computed complete reasons for 1000 random instances (there is no corresponding data). The total number of instances for the fifteen benchmarks was 13963. We did not report the time for computing a complete reason as this is a closed form with linear size (Equation 1).

We compared four algorithms: SNR (Algorithm 1), NR (standard algorithm for computing prime implicates of a monotone NNF but with no subsumption checking at $\vee$-nodes since the input NNF is $\vee$-decomposable),[5] SSR (Algo-

[4] One can compute the prime implicates of a monotone NNF by simply converting it to a CNF and then removing subsumed clauses. Similarly, one can compute the prime implicants of a monotone NNF by converting it to a DNF and removing subsumed terms. See, e.g., (Crama and Hammer 2011).

[5] More precisely, NR is Algorithm 1 with two exceptions. First, the NNF is not pruned on Line 7 so the union is over all α_i. Second, subsumption checking is applied after Line 7 to ensure that all computed implicates are subset-minimal.

benchmark		decision tree/graph properties						SR		SSR		NR		SNR	
name	examples	nodes	num	nom	classes	card	acc	count	time	count	time	count	time	count	time
adult	48842	726	6	7	2	24	86.0	2.8	0.0005	1.4	0.0007	5.6	0.0004	2.9	0.0003
compas	5278	55	5	3	2	8	71.2	1.7	0.0002	1.1	0.0003	2.9	0.0002	1.8	0.0001
fash-mnist	70000	6681	734	0	10	29	80.8	1851.6	$3.7925^{\mathbf{657}}$	7.5	$0.0348^{\mathbf{8}}$	104.8	0.0123	12.4	0.0008
gisette	7000	231	111	0	2	3	93.9	3288.4	$6.2361^{\mathbf{486}}$	95.5	0.0545	31.8	0.0013	7.4	0.0004
isolet	7797	645	201	0	26	7	83.9	8.9	0.0008	1.5	0.0026	17.7	0.0006	10.1	0.0003
la1s.wc	3204	457	201	0	6	3	73.2	461.1	$1.8227^{\mathbf{36}}$	7.7	0.0368	42.1	0.0038	19.3	0.0004
mnist-784	70000	4365	477	0	10	18	88.3	1833.3	$4.6191^{\mathbf{622}}$	5.5	$0.0235^{\mathbf{1}}$	103.9	0.0099	11.5	0.0008
nomao	34465	932	61	25	2	17	95.3	1640.8	$4.9576^{\mathbf{156}}$	2.4	0.0180	38.6	0.0018	4.3	0.0005
ohscal.wc	11162	1761	582	0	10	6	70.9	787.1	$2.0714^{\mathbf{225}}$	86.2	$0.3018^{\mathbf{21}}$	59.9	0.0090	23.7	0.0006
spambase	4601	189	37	0	2	8	91.5	35.8	0.0029	4.6	0.0060	16.9	0.0007	4.1	0.0003
andes	—	5454	—	24	2	2	—	78.8	0.1660	2.6	0.0295	58.0	0.0300	4.5	0.0055
emdec6g30	—	4154	—	30	2	2	—	11.6	0.0362	1.8	0.0445	13.6	0.0184	2.9	0.0048
math-skills	—	3693629	—	46	2	2	—	9.6	$0.2895^{\mathbf{26}}$	7.6	0.0261	9.9	$0.4271^{\mathbf{18}}$	3.4	0.0290
mooring	—	14468	—	22	2	2	—	145.4	$2.3991^{\mathbf{14}}$	20.9	$3.9951^{\mathbf{7}}$	216.3	$0.7678^{\mathbf{1}}$	16.2	0.0189
tcc4e38	—	22508	—	38	2	2	—	14.4	0.1370	2.6	0.0155	10.4	0.0632	2.0	0.0103

Table 1: Evaluating Algorithms 1 & 2. Times in secs. First ten entries are decision trees. Last five entries are decision graphs.

rithm 2) and SR (dual of NR). Each instance had a timeout of 60 seconds. In Table 1, nodes, num, nom, classes; card and acc stand for number of nodes, numeric features, nominal features, classes; maximum cardinality of variables and accuracy. Count and time are averages over instances that both SR/SSR (NR/SNR) finished. The bolded exponent of time is the number of instances that timed out (not reported if zero). The supplementary material contains further statistics such as stdev, mean and max. We used a Python implementation on a dual Intel(R) Xeon E5-2670 CPUs running at 2.60GHz and 256GB RAM. As revealed by Table 1, SSR is quite effective. Its average running time is normally in milliseconds, it timed out on only 37 instances and can be two orders of magnitude faster than SR which timed out on 2222 instances. SNR is also much faster than NR but the latter is also very effective on decision trees (see Proposition 15) but timed out on 19 decision graph instances. All algorithms are quite effective on the easier benchmarks.

Conclusion

We studied the computation of complete reasons for multi-class classifiers with nominal and numeric features. We derived closed forms for the complete reasons of decision trees and graphs in the form of monotone, $\vee$-decomposable NNFs and showed how similar forms can be derived for SDDs. We further established a correspondence between the prime implicates of complete reasons and contrastive explanations. We then presented an output polynomial algorithm for enumerating the shortest implicates (shortest necessary reasons) for complete reasons in the above form. We also presented a simple algorithm for enumerating the shortest implicants (shortest sufficient reasons) which appears to be effective based on an empirical evaluation over fifteen datasets.

Acknowledgements

This work has been partially supported by NSF grant #ISS-1910317 and ONR grant #N00014-18-1-2561.

References

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.-M.; and Marquis, P. 2021. On the Explanatory Power of Decision Trees. *CoRR*, https://arxiv.org/pdf/2108.05266.pdf.

Audemard, G.; Koriche, F.; and Marquis, P. 2020. On Tractable XAI Queries based on Compiled Representations. In *KR*, 838–849.

Barceló, P.; Monet, M.; Pérez, J.; and Subercaseaux, B. 2020. Model Interpretability through the lens of Computational Complexity. In *NeurIPS*.

Beame, P.; Li, J.; Roy, S.; and Suciu, D. 2013. Lower Bounds for Exact Model Counting and Applications in Probabilistic Databases. In *UAI*. AUAI Press.

Beame, P.; and Liew, V. 2015. New Limits for Knowledge Compilation and Applications to Exact Model Counting. In *UAI*, 131–140. AUAI Press.

Bollig, B.; and Buttkus, M. 2019. On the Relative Succinctness of Sentential Decision Diagrams. *Theory Comput. Syst.*, 63(6): 1250–1277.

Bova, S. 2016. SDDs Are Exponentially More Succinct than OBDDs. In *AAAI*, 929–935. AAAI Press.

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Computers*, 35(8): 677–691.

Chan, H.; and Darwiche, A. 2003. Reasoning about Bayesian Network Classifiers. In *UAI*, 107–115. Morgan Kaufmann.

Choi, A.; Shih, A.; Goyanka, A.; and Darwiche, A. 2020. On Symbolically Encoding the Behavior of Random Forests. *CoRR*, abs/2007.01493.

Choi, A.; Xue, Y.; and Darwiche, A. 2012. Same-decision probability: A confidence measure for threshold-based decisions. *Int. J. Approx. Reason.*, 53(9): 1415–1428.

Crama, Y.; and Hammer, P. L. 2011. *Boolean Functions - Theory, Algorithms, and Applications*, volume 142 of *Ency-*

clopedia of mathematics and its applications. Cambridge University Press.

Darwiche, A. 2011. SDD: A New Canonical Representation of Propositional Knowledge Bases. In *IJCAI*, 819–826. IJCAI/AAAI.

Darwiche, A.; and Hirth, A. 2020. On the Reasons Behind Decisions. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, 712–720. IOS Press.

Darwiche, A.; and Ji, C. 2022. On the Computation of Necessary and Sufficient Explanations. *CoRR*, abs/abs/2203.10451.

Darwiche, A.; and Marquis, P. 2021. On Quantifying Literals in Boolean Logic and Its Applications to Explainable AI. *J. Artif. Intell. Res.*, 72: 285–328.

Frank, E.; Hall, M. A.; Holmes, G.; Kirkby, R.; Pfahringer, B.; Witten, I. H.; and Trigg, L. 2010. Weka-A Machine Learning Workbench for Data Mining. In *Data Mining and Knowledge Discovery Handbook*, 1269–1277. Springer.

Garfinkel, A. 1982. Forms of Explanation: Rethinking the Questions in Social Theory. *British Journal for the Philosophy of Science*, 33(4): 438–441.

Gergov, J.; and Meinel, C. 1994. Efficient Boolean Manipulation With OBDD's can be Extended to FBDD's. *IEEE Trans. Computers*, 43(10): 1197–1209.

Goyal, Y.; Wu, Z.; Ernst, J.; Batra, D.; Parikh, D.; and Lee, S. 2019. Counterfactual Visual Explanations. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 2376–2384. PMLR.

Huang, J.; and Darwiche, A. 2007. The Language of Search. *J. Artif. Intell. Res.*, 29: 191–219.

Huang, X.; Izza, Y.; Ignatiev, A.; Cooper, M. C.; Asher, N.; and Marques-Silva, J. 2021a. Efficient Explanations for Knowledge Compilation Languages. *CoRR*, abs/2107.01654.

Huang, X.; Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2021b. On Efficiently Explaining Graph-Based Classifiers. *CoRR*, abs/2106.01350.

Ignatiev, A.; Narodytska, N.; Asher, N.; and Marques-Silva, J. 2020. From Contrastive to Abductive Explanations and Back Again. In *AI*IA*, volume 12414 of *Lecture Notes in Computer Science*, 335–355. Springer.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019a. Abduction-Based Explanations for Machine Learning Models. In *Proceedings of the Thirty-Third Conference on Artificial Intelligence (AAAI)*, 1511–1519.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019b. On Validating, Repairing and Refining Heuristic ML Explanations. *CoRR*, abs/1907.02509.

Lewis, D. 1986. Causal Explanation. In Lewis, D., ed., *Philosophical Papers Vol. Ii*, 214–240. Oxford University Press.

Lipton, P. 1990. Contrastive Explanation. *Royal Institute of Philosophy Supplement*, 27: 247–266.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267: 1–38.

Mittelstadt, B.; Russell, C.; and Wachter, S. 2019. Explaining Explanations in AI. *Proceedings of the Conference on Fairness, Accountability, and Transparency*.

Mothilal, R. K.; Sharma, A.; and Tan, C. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*.

Narodytska, N.; Kasiviswanathan, S. P.; Ryzhyk, L.; Sagiv, M.; and Walsh, T. 2018. Verifying Properties of Binarized Deep Neural Networks. In *Proc. of AAAI'18*, 6615–6624.

Quinlan, R. 1993. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *KDD*, 1135–1144. ACM.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *AAAI*, 1527–1535. AAAI Press.

Shi, W.; Shih, A.; Darwiche, A.; and Choi, A. 2020. On Tractable Representations of Binary Neural Networks. In *KR*, 882–892.

Shih, A.; Choi, A.; and Darwiche, A. 2018a. Formal Verification of Bayesian Network Classifiers. In *PGM*, volume 72 of *Proceedings of Machine Learning Research*, 427–438. PMLR.

Shih, A.; Choi, A.; and Darwiche, A. 2018b. A Symbolic Approach to Explaining Bayesian Network Classifiers. In *IJCAI*, 5103–5111. ijcai.org.

Shih, A.; Choi, A.; and Darwiche, A. 2019. Compiling Bayesian Network Classifiers into Decision Graphs. In *AAAI*, 7966–7974. AAAI Press.

Temple, D. 1988. The contrast theory of why-questions. *Philosophy of Science*, 55(1): 141–151.

van der Waa, J.; Robeer, M.; van Diggelen, J.; Brinkhuis, M.; and Neerincx, M. 2018. Contrastive explanations with local foil trees. *arXiv preprint arXiv:1806.07470*.

Vanschoren, J.; van Rijn, J. N.; Bischl, B.; and Torgo, L. 2013. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15(2): 49–60.

Verma, S.; Dickerson, J.; and Hines, K. 2020. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*.

Wachter, S.; Mittelstadt, B. D.; and Russell, C. 2017. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *CoRR*, abs/1711.00399.

Wang, E.; Khosravi, P.; and den Broeck, G. V. 2021. Probabilistic Sufficient Explanations. In *IJCAI*, 3082–3088. ijcai.org.