Proofs

Proof of Proposition 1

Proof. Let T be the complete binary tree of depth k, formed by $n = 2^k - 1$ internal nodes and 2^k leaves. We assume a breadth-first ordering of internal nodes, such that the root is labeled by x_1, the nodes of depth 1 are labeled by x_2 and x_3, and so on. Each internal node at depth $k-1$ from the root of T has two children, one of it is a 0-leaf and the other one is a 1-leaf. For an arbitrary instance $\boldsymbol{x} \in \{0,1\}^n$ and any complete subtree T' of T of depth d, let $s(\boldsymbol{x}, T')$ denote the set of sufficient reasons of $\boldsymbol{x}$ given T', and let $\sigma(\boldsymbol{x}, d) = |s(\boldsymbol{x}, T')|$ denote the number of those sufficient reasons. We show by induction on d that:

$$\sigma(\boldsymbol{x}, 1) = 1 \tag{1}$$

$$\sigma(\boldsymbol{x}, d+1) = \sigma(\boldsymbol{x}, d)(\sigma(\boldsymbol{x}, d) + 1) \tag{2}$$

For the base case (1), any complete subtree T' of T of depth $d = 1$ has a single internal node, say x_i, with two leaves labeled by 0 and 1, respectively. Therefore, the unique sufficient reason for $\boldsymbol{x}$ given T' is either x_i or $\overline{x}_i$, and hence, $\sigma(\boldsymbol{x}, 1) = 1$. Now, consider any complete subtree T' of T of depth $d+1$ rooted at a node x_i. Let $T'_l(x_i)$ and $T'_r(x_i)$ denote the subtrees of depth d, respectively rooted at the left child of x_i and the right child of x_i. Suppose without loss of generality that the unique path leading to $T'(\boldsymbol{x}) = 1$ includes the left child of x_i (i.e. $T'_l(\boldsymbol{x}) = 1$). By construction,

$$\begin{aligned} s(\boldsymbol{x}, T') = \{t_l \wedge t_r : t_l \in s(\boldsymbol{x}, T'_l), t_r \in s(\boldsymbol{x}, T'_r)\} \\ \cup \{l_i \wedge t_l : t_l \in s(\boldsymbol{x}, T'_l)\} \end{aligned}$$

where $l_i = \overline{x}_i$ if $x_i = 0$ in $\boldsymbol{x}$, and $l_i = x_i$ otherwise. Since by induction hypothesis $s(\boldsymbol{x}, T'_l) = s(\boldsymbol{x}, T'_r) = \sigma(\boldsymbol{x}, d)$, it follows that $\sigma(\boldsymbol{x}, d+1) = \sigma(\boldsymbol{x}, d)^2 + \sigma(\boldsymbol{x}, d)$. Finally, since the doubly exponential sequence[2] given by $a(1) = 1$ and $a(d+1) = a(d)^2 + a(d)$ satisfies $a(d) = \lfloor c^{2^{d-1}} \rfloor$, where $c \sim 1.59791$, it follows that $\sigma(\boldsymbol{x}, k) \geq \lfloor (3/2)^{2^{k-1}} \rfloor$. Using $2^{k-1} = (n+1)/2$, we get the desired result. □

Proof of Proposition 2

Proof. One first need the following lemma that gives a recursive characterization of the set of sufficient reasons for an instance given a Boolean classifier:

Lemma 1. *For any Boolean function $f \in \mathcal{F}_n$ and any instance $\boldsymbol{x} \in \{0,1\}^n$, the following inductive characterization of $sr(\boldsymbol{x}, f)$ holds:*

$$\begin{array}{ll} sr(\boldsymbol{x}, 1) & = \{1\} \\ sr(\boldsymbol{x}, 0) & = \{\} \\ sr(\boldsymbol{x}, f) & = sr(\boldsymbol{x}, (f \mid \ell) \wedge (f \mid \overline{\ell})) \cup \{\ell \wedge t_\ell : t_\ell \in sr(\boldsymbol{x}, f \mid \ell) \text{ s.t. } t_\ell \not\models f \mid \overline{\ell}\} \\ & \quad \textit{where } Var(\ell) \subseteq Var(f) \textit{ and } t_{\boldsymbol{x}} \models \ell \end{array}$$

and

$$sr(\boldsymbol{x}, (f \mid \ell) \wedge (f \mid \overline{\ell})) = max(\{t_\ell \wedge t_{\overline{\ell}} : t_\ell \in sr(\boldsymbol{x}, f \mid \ell), t_{\overline{\ell}} \in sr(\boldsymbol{x}, f \mid \overline{\ell})\}, \models).$$

Proof. Let us recall first the following inductive characterization of $pi(f)$, the set of prime implicants of $f \in \mathcal{F}_n$, based on the Shannon decomposition of f over any of its variables x (see e.g., [2]):

$$\begin{array}{ll} pi(1) & = \{1\} \\ pi(0) & = \{\} \\ pi(f) & = pi((f \mid \overline{x}) \wedge (f \mid x)) \\ & \cup \{\overline{x} \wedge t_{\overline{x}} : t_{\overline{x}} \in pi(f \mid \overline{x}) \text{ s.t. } \nexists t \in pi((f \mid \overline{x}) \wedge (f \mid x)), t_{\overline{x}} \models t\} \\ & \cup \{x \wedge t_x : t_x \in pi(f \mid x) \text{ s.t. } \nexists t \in pi((f \mid \overline{x}) \wedge (f \mid x)), t_x \models t\} \\ & \text{where } x \in Var(f) \end{array}$$

and

$$pi((f \mid \overline{x}) \wedge (f \mid x)) = max(\{t_{\overline{x}} \wedge t_x : t_{\overline{x}} \in pi(f \mid \overline{x}), t_x \in pi(f \mid x)\}, \models).$$

[2] See `https://oeis.org/A007018`.

For the base cases $sr(\boldsymbol{x}, 1) = \{1\}$ and $sr(\boldsymbol{x}, 0) = \{\}$, the result is obvious. For the general case, taking $x \in Var(\ell)$, we have:

$$\begin{array}{ll} sr(\boldsymbol{x}, f) & = \{t \in pi(f) : t_{\boldsymbol{x}} \models t\} \\ & = \{t \in pi((f \mid \overline{x}) \wedge (f \mid x)) : t_{\boldsymbol{x}} \models t\} \\ & \cup \{t \in \{\overline{x} \wedge t_{\overline{x}} : t_{\overline{x}} \in pi(f \mid \overline{x}) \text{ s.t. } \nexists t' \in pi((f \mid \overline{x}) \wedge (f \mid x)), t_{\overline{x}} \models t'\}, \text{ and } t_{\boldsymbol{x}} \models t\} \\ & \cup \{t \in \{x \wedge t_x : t_x \in pi(f \mid x) \text{ s.t. } \nexists t' \in pi((f \mid \overline{x}) \wedge (f \mid x)), t_x \models t'\}, \text{ and } t_{\boldsymbol{x}} \models t\} \\ & = sr(\boldsymbol{x}, (f \mid \overline{x}) \wedge (f \mid x)) \\ & \cup \{t \in \{\overline{x} \wedge t_{\overline{x}} : t_{\overline{x}} \in pi(f \mid \overline{x}) \text{ s.t. } \nexists t' \in pi((f \mid \overline{x}) \wedge (f \mid x)), t_{\overline{x}} \models t'\}, \text{ and } t_{\boldsymbol{x}} \models t\} \\ & \cup \{t \in \{x \wedge t_x : t_x \in pi(f \mid x) \text{ s.t. } \nexists t' \in pi((f \mid \overline{x}) \wedge (f \mid x)), t_x \models t'\}, \text{ and } t_{\boldsymbol{x}} \models t\} \end{array}$$

Now, since $\boldsymbol{x}$ is an instance, whatever ℓ, it cannot be the case that $t_{\boldsymbol{x}} \models \ell$ and $t_{\boldsymbol{x}} \models \overline{\ell}$. Suppose that $\ell = x$ (the case $\ell = \overline{x}$ is similar). In this situation, no element of $\{\overline{x} \wedge t_{\overline{x}} : t_{\overline{x}} \in pi(f \mid \overline{x}) \text{ s.t. } \nexists t \in pi((f \mid \overline{x}) \wedge (f \mid x)), t_{\overline{x}} \models t\}, \text{ and } t_{\boldsymbol{x}} \models t\}$ can belong to $sr(\boldsymbol{x}, f)$. As a consequence, we get that:

$$\begin{array}{ll} sr(\boldsymbol{x}, f) & = sr(\boldsymbol{x}, (f \mid \overline{x}) \wedge (f \mid x)) \\ & \cup \{t \in \{\ell \wedge t_\ell : t_\ell \in pi(f \mid \ell) \text{ s.t. } \nexists t' \in pi((f \mid \overline{\ell}) \wedge (f \mid \ell)), t_\ell \models t'\}, \text{ and } t_{\boldsymbol{x}} \models t\} \\ & \text{where } Var(\ell) \subseteq Var(f) \text{ and } t_{\boldsymbol{x}} \models \ell \end{array}$$

If $t = \ell \wedge t_\ell$ is such that $t_{\boldsymbol{x}} \models t$ holds, then we have $t_{\boldsymbol{x}} \models t_\ell$. Hence, we have:

$$\begin{array}{ll} sr(\boldsymbol{x}, f) & = sr(\boldsymbol{x}, (f \mid \overline{x}) \wedge (f \mid x)) \\ & \cup \{\ell \wedge t_\ell : t_\ell \in sr(\boldsymbol{x}, f \mid \ell) \text{ s.t. } \nexists t' \in pi((f \mid \overline{\ell}) \wedge (f \mid \ell)), t_\ell \models t'\} \\ & \text{where } Var(\ell) \subseteq Var(f) \text{ and } t_{\boldsymbol{x}} \models \ell \end{array}$$

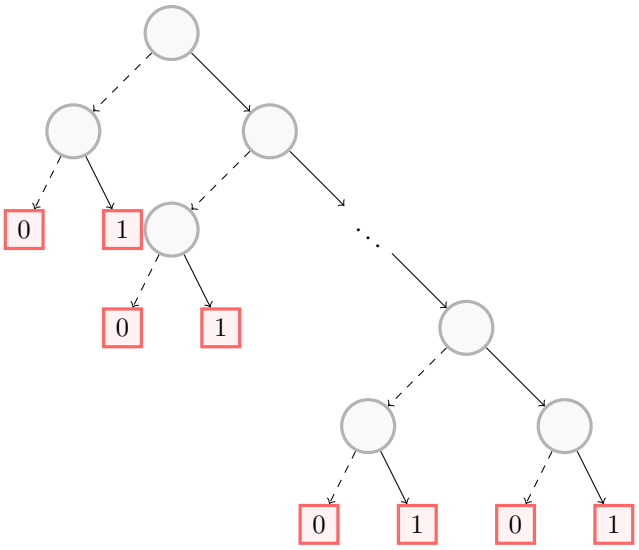
Consider now the condition $\exists t' \in pi((f \mid \overline{\ell}) \wedge (f \mid \ell)), t_\ell \models t'$ and suppose that it is satisfied. Since $pi((f \mid \overline{\ell}) \wedge (f \mid \ell)) = max(\{t'_{\overline{\ell}} \wedge t'_\ell : t'_{\overline{\ell}} \in pi(f \mid \overline{\ell}), t'_\ell \in pi(f \mid \ell)\}, \models)$, there exist $t'_{\overline{\ell}} \in pi(f \mid \overline{\ell})$ and $t'_\ell \in pi(f \mid \ell)$ such that $t' = t'_{\overline{\ell}} \wedge t'_\ell$. Thus, we have $t_\ell \models t'_{\overline{\ell}} \wedge t'_\ell$, and in particular $t_\ell \models t'_\ell$ holds. But since t_ℓ and t'_ℓ are prime implicants of $f \mid \ell$, this implies that $t_\ell \equiv t'_\ell$ holds. Furthermore, from $t_\ell \models t'_{\overline{\ell}} \wedge t'_\ell$ we get that $t_\ell \models t'_{\overline{\ell}}$. In addition, a prime implicant $t'_{\overline{\ell}}$ of $f \mid \overline{\ell}$ such that $t_\ell \models t'_{\overline{\ell}}$ exists if and only if $t_\ell \models f \mid \overline{\ell}$. Altogether, the condition $\exists t' \in pi((f \mid \overline{\ell}) \wedge (f \mid \ell)), t_\ell \models t'$ is equivalent to $t_\ell \models f \mid \overline{\ell}$. Thus, we get that:

$$\begin{array}{ll} sr(\boldsymbol{x}, f) & = sr(\boldsymbol{x}, (f \mid \ell) \wedge (f \mid \overline{\ell})) \\ & \cup \{\ell \wedge t_\ell : t_\ell \in sr(\boldsymbol{x}, f \mid \ell) \text{ s.t. } t_\ell \not\models f \mid \overline{\ell}\} \\ & \text{where } Var(\ell) \subseteq Var(f) \text{ and } t_{\boldsymbol{x}} \models \ell \end{array}$$

Finally, if $t \in max(\{t_{\overline{x}} \wedge t_x : t_{\overline{x}} \in pi(f \mid \overline{x}), t_x \in pi(f \mid x)\}, \models)$, then by construction t is such that there exist $t_{\overline{x}} \in pi(f \mid \overline{x})$ and $t_x \in pi(f \mid x)$ satisfying $t = t_{\overline{x}} \wedge t_x$. If $t_{\boldsymbol{x}} \models t$ holds, then $t_{\boldsymbol{x}} \models t_{\overline{x}}$ and $t_{\boldsymbol{x}} \models t_x$ hold. Hence $t_{\overline{x}} \in sr(\boldsymbol{x}, f \mid \overline{x})$ and $t_x \in sr(\boldsymbol{x}, f \mid x)$. Consequently, $t \in max(\{t_{\overline{x}} \wedge t_x \mid t_{\overline{x}} \in sr(\boldsymbol{x}, f \mid \overline{x}), t_x \in sr(\boldsymbol{x}, f \mid x)\}, \models)$. □

From the inductive characterization of $sr(\boldsymbol{x}, f)$ given by the previous proposition, we can easily derive a bottom-up algorithm allowing to derive $sr(\boldsymbol{x}, f)$ when f is represented by a decision tree.

Consider now a decision tree T of depth $k \geq 1$ having the following form:



T has $2k-1$ decision nodes and $2k$ leaves. Suppose that the variables associated with the decision nodes are in one-to-one correspondence with the decision nodes (i.e., they are all distinct). The number of variables occurring in T is thus $n = 2k-1$, therefore T has $2n+1$ nodes. Consider now the instance $\boldsymbol{x} \in \{0,1\}^n$ such that $x_i = 1$ for every $i \in [n]$. We are going to prove by induction on the depth k of such a tree T that $\boldsymbol{x}$ has 2^{k-1} minimal reasons given T, each of them containing k literals. The proof takes advantage of the recursive characterization of the set of all sufficient reasons for an instance given a decision tree, as made precise by Lemma 1.

- Base case $k = 1$. We have $n = 1$. T consists of a decision node labelled by the single variable of X_n, say x, a left child that is a 0-leaf and a right child that is a 1-leaf. T is equivalent to x and x is implied by $t_{\boldsymbol{x}}$. Hence, x is the unique sufficient reason for $\boldsymbol{x}$ given T, so it is also the unique minimal reason for $\boldsymbol{x}$ given T. As expected, the number of minimal reasons for $\boldsymbol{x}$ given T is equal to 2^{k-1}. The size of the unique minimal reason is $k = 1$.
- Inductive step $k > 1$. Let x be the variable of X_n labelling the root node of T. By construction, the left child T_l of T is equivalent to a single variable, say x_l, that is the unique minimal reason for $\boldsymbol{x}$ given T_l. The right child T_r of T has the same form as T, but with depth $k-1$. By induction hypothesis, we know that $\boldsymbol{x}$ has 2^{k-2} minimal reasons given T_r, each of them containing $k-1$ literals. As shown by Lemma 1, provided that the variables labelling the decision nodes are pairwise distinct, the minimal reasons for $\boldsymbol{x}$ given T are obtained by extending every minimal reason for $\boldsymbol{x}$ given T_r with x_l and by extending every minimal reason for $\boldsymbol{x}$ given T_r with x. Accordingly, $\boldsymbol{x}$ has $2 \times (2^{k-2}) = 2^{k-1}$ minimal reasons given T and each of them contains $k-1+1 = k$ literals.

Finally, since $n = 2k-1$, we have $k = \frac{n+1}{2}$ and the number of minimal reasons for $\boldsymbol{x}$ given T is equal to $2^{k-1} = 2^{\sqrt{n-1}}$. □

Proof of Proposition 3

Proof. The algorithms to compute $Nec_s(\boldsymbol{x}, T)$, $Rel_s(\boldsymbol{x}, T)$, and $Irr_s(\boldsymbol{x}, T)$ are as follows: first compute $\texttt{CNF}(T)$ and then remove from this set of clauses every literal that does not belong to $t_{\boldsymbol{x}}$. This can be done in $\mathcal{O}(n|T|)$ time. By construction, the resulting CNF formula f is monotone: every literal in it occurs with the same polarity as the one it has in $t_{\boldsymbol{x}}$. Furthermore, the size of f cannot exceed the size of $\texttt{CNF}(T)$, thus the size of T.

Since f is a monotone CNF formula, its prime implicates can be computed by removing from f every clause that is a strict superset of another clause of f.This can be achieved in quadratic time in the size of f, thus in the size of T. Let g be the resulting formula in prime implicates form and equivalent to f. g is equivalent to the complete reason for $\boldsymbol{x}$ given T. Since it is in prime implicates form, g is

Lit-dependent on every literal occurring in it (i.e., g is Lit-simplified, see Proposition 8 in [23] for details), hence so is the complete reason for $\boldsymbol{x}$ given T.

This means that for every literal ℓ occurring in g, there exists a sufficient reason for $\boldsymbol{x}$ given T that contains ℓ, so that $Rel_s(\boldsymbol{x}, T)$ is the set of literals occurring in g and $Irr_s(\boldsymbol{x}, T)$ is the complement of $Rel_s(\boldsymbol{x}, T)$ in the set of all literals over X_n. Finally, since by definition the literals of $Nec_s(\boldsymbol{x}, T)$ must belong to every sufficient reason for $\boldsymbol{x}$ given T, they are given by the unit clauses that belong to g. □

Proof of Proposition 4

Proof. We call MINIMAL REASON the problem that asks, given $T \in \mathtt{DT}_n$, $\boldsymbol{x} \in \{0,1\}^n$ with $T(x) = 1$ and $k \in \mathbb{N}$, whether there is an implicant t of T of size at most k that covers $\boldsymbol{x}$.

Our objective is to prove that MINIMAL REASON is NP-hard. To this end, let us first recall that a *vertex cover* of an undirected graph $G = (X, E)$ is a subset $V \subseteq X$ of vertices such that $\{y, z\} \cap V \neq \varnothing$ for every edge $e = \{y, z\}$ in E. In the MIN VERTEX COVER problem, we are given a graph G together with an integer $k \in \mathbb{N}$, and the task is to find a vertex cover V of G of size at most k. MIN VERTEX COVER is a well-known NP-hard problem [20], and we now show that it can be reduced in polynomial time to MINIMAL REASON.

Suppose that we are given a graph $G = (X, E)$ and assume, without loss of generality, that G does not include isolated vertices. For any $y \in X$, let $E_y = \{e \in E : y \in e\}$ denote the set of edges in G that are adjacent to y, and let $N_y = \{z \in X : \{y, z\} \in E\}$ denote the set of neighbors of y in G. By $G \setminus y$, we denote the deletion of y from G, obtained by removing y and its adjacent edges, i.e., $G \setminus y = (X \setminus \{y\}, E \setminus E_y)$. We associate with G a decision tree $T(G)$ over $X_n = X$ using the following recursive algorithm. If G is the empty graph (i.e. $E = \varnothing$), then return the decision tree rooted at a 1-leaf. Otherwise, pick a node $y \in X$ and generate a decision tree $T(G)$ such that:

(1) the root is labeled by y;

(2) the left child is the decision tree encoding the monomial $\bigwedge N_y$;

(3) the right child is the decision tree $T(G')$ returned by calling the algorithm on $G' = G \setminus y$.

By construction, $T(G)$ is a complete backtrack search tree of the formula $\mathtt{CNF}(E) = \bigwedge\{(y \vee z) : \{y, z\} \in E\}$, which implies that $T(G)$ and $\mathtt{CNF}(E)$ are logically equivalent. Furthermore, $T(G)$ is a comb-shaped tree since recursion only on the rightmost branch. In particular, the algorithm runs in $\mathcal{O}(n|E|)$ time, since step (1) takes $\mathcal{O}(1)$ time, step (2) takes $\mathcal{O}(n)$ time, and step (3) is called at most $|E|$ times.

Now, with an instance $\boldsymbol{P}_1 = (G, k)$ of MIN VERTEX COVER, we associate the instance $\boldsymbol{P}_2 = (T(G), \boldsymbol{x}, k)$ of MINIMAL REASON, where $\boldsymbol{x} = (1, \cdots, 1)$. Based on the above algorithm, $\boldsymbol{P}_2$ can be constructed in time polynomial in the size of $\boldsymbol{P}_1$.

Let V be a solution of $\boldsymbol{P}_1$. Since V is a vertex cover of G, the term $t_V = \bigwedge V$ is an implicant of the formula $\mathtt{CNF}(E)$. Since $t_V \subseteq t_{\boldsymbol{x}}$ and $|t_V| \leq k$, it follows from the fact that $\mathtt{CNF}(E)$ and $T(G)$ are logically equivalent that t_V is a solution of $\boldsymbol{P}_2$.

Conversely, let t be a solution of $\boldsymbol{P}_2$. Since t is an implicant of $T(G)$, it follows that t is an implicant of $\mathtt{CNF}(E)$. This together with the fact that $t \subseteq t_{\boldsymbol{x}}$ implies that the subset of vertices $V \subseteq X_n$, satisfying $\bigwedge V = t$, is a vertex cover of G. Since $|V| \leq k$, it is therefore a solution of $\boldsymbol{P}_1$.

□

Proof of Proposition 5

Proof. Let $\boldsymbol{x}^*$ be any solution of $(C_{\text{soft}}, C_{\text{hard}})$. Observe that the set of all hard clauses $c \cap t_{\boldsymbol{x}}$ (where c is a clause of $\mathtt{CNF}(T)$) corresponds to a *monotone* CNF formula. Therefore, in order to satisfy such a clause $c \cap t_{\boldsymbol{x}}$, $\boldsymbol{x}^*$ must set a literal ℓ of $t_{\boldsymbol{x}}$ to 1. Thus, $\boldsymbol{x}^*$ satisfies all the hard clauses of C_{hard} if and only if the term consisting of the literals that are shared by $t_{\boldsymbol{x}} = \bigwedge_{i=1}^{n} \ell_i$ and $t_{\boldsymbol{x}^*}$ is an implicant of T and is implied by $\boldsymbol{x}$.

The soft clauses of C_{soft} are used to select among the assignments that satisfy all the hard clauses, the ones that correspond to minimal sufficient reasons. Soft clauses are given by literals ℓ_i, which are precisely the complementary literals to those occurring in $t_{\boldsymbol{x}}$. Having a soft clause ℓ_i violated by $\boldsymbol{x}^*$ means that the literal $\overline{\ell}$ of $t_{\boldsymbol{x}}$ is necessary to get an implicant of T given the assignment of the other variables in $\boldsymbol{x}^*$. Whenever a soft clause ℓ_i is violated by $\boldsymbol{x}^*$ a penalty of 1 incurs. This ensures that the term consisting of the literals that are shared by $t_{\boldsymbol{x}} = \bigwedge_{i=1}^{n} \ell_i$ and $t_{\boldsymbol{x}^*}$ is a minimal sufficient reason for $\boldsymbol{x}$ given T. □

Proof of Proposition 6

Proof. By definition, the sufficient reasons t for $\boldsymbol{x}$ given f are the prime implicants of f that covers $\boldsymbol{x}$. Thus, they are precisely the prime implicants of the (conjunctively-interpreted) set of clauses $\{c \cap t_{\boldsymbol{x}} : c \in \texttt{CNF}(f)\}$ where $\texttt{CNF}(f)$ is any CNF formula equivalent to f. Furthermore, the complete reason for $\boldsymbol{x}$ given f (equivalent to the disjunction of all the sufficient reasons for $\boldsymbol{x}$ given f [8]) is a monotone Boolean function because every sufficient reason covers $\boldsymbol{x}$ which assigns in a unique way every variable from X_n. The prime implicates of such a monotone function are precisely the minimal hitting sets of the prime implicants of the function. Because of the minimal hitting set duality between sufficient reasons and contrastive explanations for $\boldsymbol{x}$ given f [16], the contrastive explanations for $\boldsymbol{x}$ given f are thus the sets of literals corresponding to the prime implicates of $\{c \cap t_{\boldsymbol{x}} : c \in \texttt{CNF}(f)\}$. Now, since the (conjunctively-interpreted) set of clauses $\{c \cap t_{\boldsymbol{x}} : c \in \texttt{CNF}(f)\}$ is equivalent to the complete reason for $\boldsymbol{x}$ given f, it is a monotone function, and as a consequence, its prime implicates are its minimal elements w.r.t. $\subseteq$. This comes from the correctness of any resolution-based algorithm for generating prime implicates (see e.g., [26]). Finally, when f is a decision tree T, $\{c \cap t_{\boldsymbol{x}} : c \in \texttt{CNF}(T)\}$ can be computed in time polynomial in $n + |T|$ because $\texttt{CNF}(T)$ can be computed in time linear in $|T|$. Using an extra quadratic time in the size of this set $\{c \cap t_{\boldsymbol{x}} : c \in \texttt{CNF}(T)\}$, its minimal elements w.r.t. $\subseteq$ can be selected. The resulting set is by construction the set of all the contrastive explanations for $\boldsymbol{x}$ given T, and this set has been computed in time polynomial in $n + |T|$. □