Tractable Explaining of Multivariate Decision Trees

Clément Carbonnel[1] , **Martin C. Cooper**[2] , **João Marques-Silva**[3]
[1]LIRMM, CNRS, University of Montpellier, France
[2]IRIT, University of Toulouse, France
[3]IRIT, CNRS, Toulouse, France
clement.carbonnel@lirmm.fr, {cooper, joao.marques-silva}@irit.fr

Abstract

We study multivariate decision trees (MDTs), in particular, classes of MDTs determined by the language of relations that can be used to split feature space. An abductive explanation (AXp) of the classification of a particular instance, viewed as a set of feature-value assignments, is a minimal subset of the instance which is sufficient to lead to the same decision. We investigate when finding a single AXp is tractable. We identify tractable languages for real, integer and boolean features. Indeed, in the case of boolean languages, we provide a P/NP-hard dichotomy.

1 Background

Decision trees (DTs) are a classical family of ML models. There is considerable interest in their multivariate extension (MDTs) in which feature-space is split according to conditions on several features rather than on a single feature (Brodley and Utgoff 1995; Zhu et al. 2020; Cañete-Sifuentes, Monroy, and Medina-Pérez 2021). For example, in oblique DTs these conditions are linear inequalities (Heath, Kasif, and Salzberg 1993; Murthy, Kasif, and Salzberg 1994; Barros et al. 2014; Wickramarachchi et al. 2016; Carreira-Perpiñán and Tavallali 2018). In this paper we study families of MDTs, parameterized by the language of possible multivariate conditions, from the point of view of the tractability of explaining decisions.

A multivariate condition can be seen as a *constraint* which can be decomposed into its *scope* (a list ℓ of features) and its *relation* of arity $|\ell|$. This allows us to study multivariate decision trees according to the language of possible constraint relations.

Definition 1. *A* multivariate decision tree *is a decision tree in which the condition tested at a node is a constraint on any number of features. An $\mathcal{L}$-DT is a multivariate decision tree in which the constraint relations belong to the language $\mathcal{L}$.*

A multivariate DT may be exponentially smaller than a DT. Consider the case of a parity function κ on n boolean features: trivially an $\mathcal{L}$-DT of depth one can capture this function provided $\kappa \in \mathcal{L}$, whereas a classical DT would necessarily be of exponential size.

Tractable constraint languages have been investigated in the context of the Constraint Satisfaction Problem (CSP). A CSP instance consists of a set of n variables, each with its domain, together with a set of constraints, where each constraint is defined by its scope (a list of variables) and the relation that must hold on the variables in this scope. The decision version of the CSP consists in determining whether there exists some assignment to all n variables in the cartesian product of the domains that satisfies all the constraints. Given a language $\mathcal{L}$ of relations, CSP($\mathcal{L}$) is the subproblem of the decision version of the CSP in which all relations belong to the language $\mathcal{L}$. The languages $\mathcal{L}$ we consider are, as is classical in CSPs, arbitrary sets of relations that can apply to any variables/features.

As we will see later, testing whether a subset of the feature assignments comprising the instance is sufficient to explain the decision involves solving a constraint satisfaction problem consisting of the conditions along each path to a leaf corresponding to a different decision. In classical DT's these conditions are unary and the resulting CSP is trivial, but for multivariate conditions, the resulting CSP is, in general, NP-hard. We will see the close relationship between tractability of explaining $\mathcal{L}$-DTs and the tractability of CSP($\mathcal{L}$). However, there is an important difference. In an MDT, for each edge corresponding to the satisfaction of a relation R there is an alternate edge corresponding to its complement relation $\neg R$. It follows that in the context of MDTs, it is important to study languages *closed under complement*: languages $\mathcal{L}$ such that $R \in \mathcal{L} \Rightarrow \neg R \in \mathcal{L}$. There is large body of work on the characterisation of languages $\mathcal{L}$ for which CSP($\mathcal{L}$) $\in$ P, culminating in a dichotomy theorem in the finite-domain

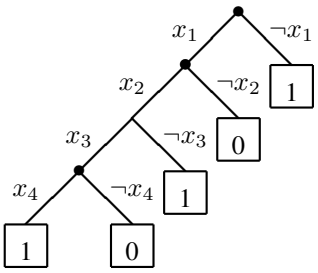

Figure 1: A decision tree corresponding to the classifier $\kappa(\mathbf{x}) = \neg x_1 \vee (x_2 \wedge (\neg x_3 \vee x_4))$.

case (Bulatov 2017; Zhuk 2020). This result implies a similar dichotomy for finite languages closed under complement, but the dichotomy criterion does not provide an *explicit* description of the tractable cases.

Although DTs are sometimes considered to be inherently interpretable, it has recently been shown that DT paths can exhibit significant redundancy, both in theory and in practice, when considered as explanations of decisions (Izza, Ignatiev, and Marques-Silva 2022). In this paper, we therefore study the notion of abductive explanation (AXp) (Shih, Choi, and Darwiche 2018; Ignatiev, Narodytska, and Marques-Silva 2019) which can provide a more succinct explanation of a particular decision than the (M)DT path corresponding to the decision (Izza, Ignatiev, and Marques-Silva 2022).

Definition 2. *Let κ be a classifier and $\mathbf{v}$ a feature-vector. A* weak AXp *(weak abductive explanation) of the decision $\kappa(\mathbf{v}) = c$ is a subset S of the features such that any assignment $\mathbf{y}$ that agrees with $\mathbf{v}$ on the features in S satisfies $\kappa(\mathbf{y}) = c$. An* AXp *of a decision is a subset-minimal weak AXp.*

Example 1. *Consider the classifier $\kappa(\mathbf{x}) = \neg x_1 \vee (x_2 \wedge (\neg x_3 \vee x_4))$ where $x_1, x_2, x_3, x_4 \in \{0, 1\}$ are boolean features. κ can be represented by the decision tree in Figure 1. An abductive explanation (AXp) for the decision $\kappa(1, 1, 1, 1) = 1$ is $\{x_2, x_4\}$ since any feature-vector $\mathbf{y}$ with $y_2 = y_4 = 1$ satisfies $\kappa(\mathbf{y}) = 1$ (but neither $y_2 = 1$ nor $y_4 = 1$ alone is sufficient to guarantee $\kappa(\mathbf{y}) = 1$). This AXp is half the length of the path in the DT of Figure 1 corresponding to this decision (i.e. the leftmost path).*

The need to apply formal reasoning to explainable artificial intelligence (XAI), and in particular to decisions taken by ML models, has been pointed out by many researchers (Guidotti et al. 2019; Miller 2019; Marques-Silva and Ignatiev 2022; Amgoud and Ben-Naim 2022). The computational complexity of finding abductive explanations is an active field of research in the application of formal reasoning to explaining decisions taken by classifiers (Audemard et al. 2022; Barceló et al. 2020; Cooper and Marques-Silva 2023; Huang et al. 2022; Ignatiev, Narodytska, and Marques-Silva 2019; Wäldchen et al. 2021). Izza et al (Izza, Ignatiev, and Marques-Silva 2022) showed that finding an AXp of a decision taken by a DT is in P. (This corresponds to the case in which all constraints are *unary* , i.e. of the form $x_i \in S$ for some subset S of the domain of x_i). In this paper we explore the tractability of this problem for MDTs parameterised by the constraint language $\mathcal{L}$. We show that, in general, this problem is NP-hard, but that there are nonetheless many interesting tractable cases.

Let WAXPDT($\mathcal{L}$) denote the problem of deciding whether a set of features is a weak AXp for a given decision taken by a $\mathcal{L}$-DT, where $\mathcal{L}$ is a language of constraint relations. As we show in Section 2, whenever WAXPDT($\mathcal{L}$) $\in$ P, there is a polynomial-time algorithm to find an AXp: starting with the set of all features, for each feature test whether deleting the feature still leaves a weak AXp (Chen and Toda 1995; Cooper and Marques-Silva 2023).

When $\mathcal{L}$ is the set of unary constraints, then an $\mathcal{L}$-DT can be viewed as a classical DT. In this case, WAXPDT($\mathcal{L}$) is known to be tractable (Izza, Ignatiev, and Marques-Silva 2022). After identifying, in Section 2, several languages $\mathcal{L}$ for which WAXPDT($\mathcal{L}$) is tractable, in Sections 3 and 4, we describe a dichotomy theorem in the case of boolean languages. In Section 5 we consider a different type of abductive explanation and show that the dichotomy theorem for boolean languages also holds for this type of explanation.

2 Tractable explaining of decision tree decisions

We begin by recalling a simple algorithm to find minimal subsets satisfying a monotone property (Chen and Toda 1995). We say that a property $\mathcal{H}$ is *monotone* if for all sets $S \subseteq T$, $\mathcal{H}(S) \Rightarrow \mathcal{H}(T)$.

Lemma 1. *Given an initial finite set S_0 and a monotone property $\mathcal{H}$ that can be tested in polynomial time, a minimal subset S of S_0 satisfying $\mathcal{H}$ can be found in polynomial time.*

Proof. The following so-called 'deletion' algorithm finds a minimal $S \subseteq S_0$ by testing $|S_0|$ times the property $\mathcal{H}$.

for each element $e \in S_0$:
    if $\mathcal{H}(S \setminus \{e\})$ then $S \leftarrow S \setminus \{e\}$

□

The following corollary follows from the fact that being a weak AXp is a monotone property and that the set of all features is trivially a weak AXp (and hence can be used as the initial set S_0 in the deletion algorithm).

Corollary 1. *For any family of classifiers, finding a single AXp is polytime if testing whether a subset of features is a weak AXp is in P.*

We assume that an MDT in $\mathcal{L}$-DT is represented as a binary tree in which each leaf node is labelled by a class and each internal node is linked to its two child-nodes by edges labelled respectively by a relation $R \in \mathcal{L}$ and its complement $\neg R \in \mathcal{L}$. The assumption of an explicit representation of $\neg R$ avoids technical issues related to the possible large disparity between the sizes of the explicit representation of $\neg R$ and its implicit representation as the complement of R. In the following proposition, we do not impose a fixed representation of relations (as a table of tuples or as a formula) but we do assume the same representation of relations in CSP($\mathcal{L}$) and in MDTs in $\mathcal{L}$-DT.

Given an MDT, we use the notation path(α) to represent the set of conditions satisfied on the path from the root to a leaf α. Let $Asst$ represent all unary constraints consisting of assignments, i.e. $x_i = u$ for some feature x_i and some constant u. We can view a feature-vector $\mathbf{v}$ as a set of literals (i.e. variable-value assignments). For a fixed feature-vector $\mathbf{v}$, it will be convenient to interpret a set X of features as a partial assignment, i.e. the set of literals corresponding to the subset of $\mathbf{v}$ on these variables.

Proposition 1. *Let $\mathcal{L}$ be a language such that $\mathcal{L}$ is closed under complement. Suppose that $\mathcal{L} \cup Asst \subseteq \mathcal{C}$ where CSP($\mathcal{C}$) $\in$ P. Then* WAXPDT($\mathcal{L}$) $\in$ *P and an AXp of any decision taken by an $\mathcal{L}$-DT can be found in polynomial time.*

Proof. Let κ be the classifier defined by an $\mathcal{L}$-DT and consider a decision $\kappa(\mathbf{v}) = c$ to be explained. By Corollary 1, we only need to show that we can test that a set X is a Weak AXp in polynomial time. Testing whether X is a Weak AXp can be achieved by testing whether for all leaves α corresponding to a decision different to c, X (considered as a partial assignment) is incompatible with the set of constraints path(α). The constraints of path(α) are in $\mathcal{L}$. Furthermore, the partial assignment X can be viewed as a set of constraints in $Asst$, so this test of incompatibility is a CSP with constraints in $\mathcal{L} \cup Asst$, and hence, by the hypotheses $\mathcal{L} \cup Asst \subseteq \mathcal{C}$ and CSP($\mathcal{C}$) $\in$ P, is solvable in polynomial time. □

In all the following examples, $\mathcal{L}$ is closed under complement, $\mathcal{L} \cup Asst \subseteq \mathcal{C}$ and CSP($\mathcal{C}$) $\in$ P, and so Proposition 1 applies.

Boolean domains We begin with examples in which features are boolean. Two well-known boolean languages $\mathcal{C}$ for which CSP($\mathcal{C}$) is tractable are conjunctions of Horn clauses and conjunctions of 2-clauses.

Example 2. *Let $\mathcal{L}$ be the class of Horn clauses and their negations. The complement (negation) of a Horn clause is a conjunction of unary clauses and unary clauses are trivially Horn. $\mathcal{C}$ is the class of conjunctions of Horn clauses, and hence CSP($\mathcal{C}$) $\in$ P since it corresponds to HORNSAT.*

Note that, in general, the complement of a *conjunction* of Horn clauses is not the conjunction of Horn clauses. In Section 4.1 we identify the maximal generalisation of the class in Example 2. It consists of a specific form of conjunctions of Horn clauses.

Example 3. *Let $\mathcal{L}$ be the class of 2-conjunctions of 2-clauses (i.e. the conjunction of at most two clauses each of which contains at most two literals) together with the complements of such constraints. The complement of a 2-conjunction of 2-clauses is also the conjunction of 2-clauses, since $\neg((a \vee b) \wedge (c \vee d)) \equiv (\neg a \vee \neg c) \wedge (\neg a \vee \neg d) \wedge (\neg b \vee \neg c) \wedge (\neg b \vee \neg d)$. $\mathcal{L} \cup Asst \subseteq \mathcal{C}$ where $\mathcal{C}$ is the set of conjunctions of 2-clauses. CSP($\mathcal{C}$) $\in$ P by tractability of 2SAT.*

In general, the complement of an arbitrary conjunction of 2-clauses is not the conjunction of 2-clauses. We identify the maximal generalisation of this example in Section 4.3.

Finite domains We now consider finite feature-domains of arbitrary size. Define a *two-fan* constraint to be a constraint of the form $x_i = a \vee x_j = b$, where a, b are constants.

Example 4. *Let $\mathcal{L}$ be the class of two-fan constraints and their complements, together with all unary constraints $x_i \in S$ where S is any subset of the domain of x_i. The complement of the two-fan $x_i = a \vee x_j = b$ is the constraint $x_i \neq a \wedge x_j \neq b$ which is the conjunction of two unary constraints. Let maj : $D^3 \to D$ be the function defined by maj$(a, b, c) = b$ if $b = c$ and maj$(a, b, c) = a$ if $b \neq c$. It returns the majority value among its arguments, if it exists, and its first argument otherwise. A binary relation R is maj-closed if $(a_1, a_2), (b_1, b_2), (c_1, c_2) \in R \Rightarrow$ (maj(a_1, b_1, c_1),maj$(a_2, b_2, c_2)) \in R$, and all unary constraints are maj-closed. All two-fan constraints and conjunctions of unary constraints are maj-closed. It is well known that CSP($\mathcal{C}$) $\in$ P where $\mathcal{C}$ is the set of maj-closed relations (Cooper, Cohen, and Jeavons 1994; Jeavons, Cohen, and Gyssens 1995).*

Now suppose that all domains are finite and totally ordered. Define a *generalised interval constraint* (GIC) to be a constraint of the form $x_i \leq a \vee x_j \geq b$, where a, b are constants.

Example 5. *Let $\mathcal{L}$ be the set of GIC's and their complements, together with all unary constraints $x_i \in S$ where S is any subset of the domain of x. The complement of the GIC $x_i \leq a \vee x_j \geq b$ is the constraint $x_i > a \wedge x_j < b$, which is the conjunction of unary constraints. A binary relation R is said to be* max-closed *if $(a_1, a_2), (b_1, b_2) \in R \Rightarrow (\max(a_1, b_1), \max(a_2, b_2)) \in R$, and all unary constraints are max-closed (Jeavons and Cooper 1995). It is easy to check that GIC's and their complements are max-closed. Let $\mathcal{C}$ be the class of conjunctions of max-closed constraints of arity at most two. Then $\mathcal{L} \cup Asst \subset \mathcal{C}$ and CSP($\mathcal{C}$) $\in$ P since instances in this class are solved by arc consistency (Jeavons and Cooper 1995).*

Infinite domains We now consider infinite domains, firstly integer domains and then real domains.

Example 6. *A unit two variable per inequality (UTVPI) constraint is of the form $ax_i + bx_j \leq d$ where x_i and x_j are integer variables, the coefficients $a, b \in \{-1, 0, 1\}$ and the bound d is an integer constant. The negation of such a constraint is $-ax_i - bx_j \leq -(d+1)$ and is hence also an UTVPI constraint. A unary assignment $x_i = d$ is equivalent to $x_i \leq d \wedge -x_i \leq -d$, a conjunction of UTVPI constraints. Let $\mathcal{L}$ be the set of UTVPI constraints and $\mathcal{C}$ the class of constraints consisting of conjunctions of UTVPI constraints. Then $\mathcal{L} \cup Asst \subset \mathcal{C}$ and it is known that CSP($\mathcal{C}$) $\in$ P (Lahiri and Musuvathi 2005).*

Example 7. *Let $\mathcal{L}$ be the class of linear inequalities ($\leq$ or $<$) over the reals. The complement of a linear inequality is again a linear inequality and assignments $x_i = u$ can be viewed as two linear inequalities ($x_i \leq u$ and $-x_i \leq -u$). $\mathcal{C}$ is the set of systems of linear inequalities over $\mathbb{R}$. Hence $\mathcal{L} \cup Asst \subset \mathcal{C}$ and it is well known that CSP($\mathcal{C}$) $\in$ P.*

Since an oblique decision tree is an MDT in which all conditions are linear inequalities, we can deduce that there is a polynomial-time algorithm to find an AXp of a decision taken by an oblique decision tree. The dual of an abductive explanation is a contrastive explanation, a minimal set of features that if changed changes the output of the classifier. It has been observed that an optimal contrastive explanation, known as a counterfactual explanation or adversarial example, can be found for oblique decision trees in polynomial time for a linear error function, by reduction to Linear Programming (Carreira-Perpiñán and Hada 2021).

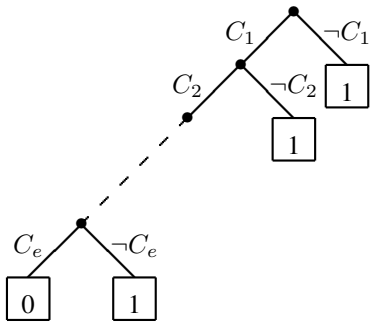

Figure 2: A decision tree T_I which has a non-empty AXp if and only if the constraints $C_1, \ldots, C_e$ are simultaneously satisfiable.

3 Tractable boolean languages: the algebraic approach

Recall that we are interested in languages $\mathcal{L}$ closed under taking complements, i.e. $R \in \mathcal{L} \Rightarrow \neg R \in \mathcal{L}$.

We first study the characterisation of tractable languages $\mathcal{L}$ for wAXpDT($\mathcal{L}$) from an abstract algebraic point of view, before looking for a detailed characterisation.

Let $f : D^k \to D$ be a function. A relation R has f as a *polymorphism* (we say that R is closed under f) if $\forall t_1, \ldots, t_k \in R$, the tuple $f(t_1, \ldots, t_k)$ obtained by applying f componentwise to the k vectors $t_1, \ldots, t_k$ belongs to R (Jeavons, Cohen, and Gyssens 1997). We say that a language $\mathcal{L}$ has the polymorphism f if all relations in $\mathcal{L}$ are closed under f.

In the following, let max (min) be the binary function which returns the maximum (minimum) of its two arguments. Let maj : $\{0,1\}^3 \to \{0,1\}$ be the ternary majority function (already introduced in Example 4) defined by

$$\text{maj}(x, y, z) = \begin{cases} y & \text{if } y = z \\ x & \text{otherwise} \end{cases}$$

Let miny : $\{0,1\}^3 \to \{0,1\}$ be the ternary minority function defined by

$$\text{miny}(x, y, z) = \begin{cases} z & \text{if } x = y \\ \neg z & \text{otherwise} \end{cases}$$

It returns the minority value if the three values x, y, z are not all equal.

Theorem 1. *Let $\mathcal{L}$ be a finite boolean language closed under taking complements. Then, assuming P$\neq$NP,* wAXpDT*($\mathcal{L}$) $\in$ P iff $\mathcal{L}$ has either max, min, maj or miny as a polymorphism.*

Proof. $\Leftarrow$: Suppose that $\mathcal{L}$ has either max, min, maj or miny as a polymorphism. It is well known that this implies that CSP($\mathcal{L}$) $\in$ P (Jeavons, Cohen, and

Gyssens 1997). Furthermore, all unary constraints have these four polymorphisms. Thus, we also have CSP($\mathcal{L} \cup Asst$) $\in$ P, and hence by Proposition 1, wAXpDT($\mathcal{L}$) $\in$ P.

$\Rightarrow$: We first give a polynomial reduction from CSP($\mathcal{L}$) to wAXpDT($\mathcal{L}$). Let I be an instance of CSP($\mathcal{L}$) consisting of constraints $C_1, \ldots, C_e$. We build a DT T_I, shown in Figure 2, as a sequence of tests corresponding to these constraints. C_1 is the test at the root of T_I, and each C_i ($i = 2, \ldots, e$) is the test at the positive child of C_{i-1} (i.e. the node attained after a positive response to the test C_{i-1}). The positive child of C_e is a leaf node labelled 0. All negative children of all nodes of T_I are leaf nodes labelled 1. Let κ be the function defined by the DT T_I. Now consider any decision $\kappa(\mathbf{v}) = 1$. The empty set is a weak AXp of this decision iff it is impossible to simultaneously satisfy the constraints $C_1, \ldots, C_e$, since the only leaf node labelled 0 can only be reached if all these constraints are satisfied. Thus deciding whether $\emptyset$ is a weak AXp amounts to solving $I \in$ CSP($\mathcal{L}$).

Thus, assuming P$\neq$NP, wAXpDT($\mathcal{L}$) $\in$ P only if $\mathcal{L}$ is a sublanguage of one of Schaefer's tractable boolean constraint languages (Schaefer 1978). By Schaefer's theorem, assuming P$\neq$NP, CSP($\mathcal{L}$) $\in$ P iff $\mathcal{L}$ has (at least) one of the six polymorphisms 0, 1, max, min, maj or miny. A relation R is a-closed, where $a \in \{0, 1\}$, iff the tuple $(a, \ldots, a)$ (of length the arity of R) belongs to R. So it is clear that R and $\neg R$ cannot both be a-closed. Thus there is no non-empty language $\mathcal{L}$ closed under complement which is a-closed. Thus $\mathcal{L}$ has either max, min, maj or miny as a polymorphism. The empty language $\mathcal{L} = \emptyset$ trivially has all polymorphisms. □

Theorem 1 shows that there is a complexity dichotomy. In the next section we provide a more explicit characterisation of the tractable boolean languages.

4 Characterisation of tractable boolean languages

We now study tractable boolean languages closed under taking complements, in order to gain a better insight into the tractable classes identified in Theorem 1. Let $\mathcal{L}_f$ be the language of boolean relations having the polymorphism f. It is well known (Jeavons, Cohen, and Gyssens 1995; Jeavons and Cooper 1995; Jeavons, Cohen, and Gyssens 1997) that

1. $\mathcal{L}_{\min}$ is the set of conjunctions of Horn clauses.
2. $\mathcal{L}_{\max}$ is the set of conjunctions of anti-Horn clauses.
3. $\mathcal{L}_{\text{miny}}$ is the set of conjunctions of affine constraints (i.e. linear equations).
4. $\mathcal{L}_{\text{maj}}$ is the set of conjunctions of 2-clauses.

In all four cases, $\mathcal{L}_f$ is not closed under complement and so we require extra work to identify the (unique) maximal sublanguage closed under complement.

4.1 Horn and anti-Horn

We start with the language $\mathcal{L}_{\min}$. By the discussion above we need to characterise the maximal sublanguage of $\mathcal{L}_{\min}$ closed under complement, or equivalently the Horn formulas whose negation is expressible by a Horn formula. We will prove that these formulas are exactly those in which the sets of negative literals appearing in clauses are totally ordered with respect to set inclusion. We call such formulas *star-nested*.

Definition 3. *A Horn formula ψ is* star-nested *if and only if there exist sets of literals L and $\emptyset = S_0 \subset S_1 \subset S_2 \subset \ldots \subset S_q$ such that*

- *all literals in L are positive, and*
- *all literals in S_q are negative, and*
- *every clause C in ψ is of the form $C = \bigvee_{s \in S_i} s$ or $C = l \vee \left(\bigvee_{s \in S_i} s\right)$ with $l \in L$.*

To clarify the definition, we point out that each set S_i may occur more than once in the formula (in clauses with different positive literals l). In particular, star-nested Horn formulas may contain any number of unit clauses with positive literals (which correspond to the set $S_0 = \emptyset$). Clearly, since the sets S_i are nested, a star-nested formula with no redundant clauses contains at most one clause consisting of only negative literals and at most one clause for each positive literal $l \in L$.

Proposition 2. *Let ψ be a star-nested Horn formula. Then, $\neg\psi$ is equivalent to a star-nested Horn formula.*

Proof. We proceed by induction on the number of sets S_i. For $q = 0$, we have $\neg\psi = \bigvee_{l \in L} \neg l$ and hence $\neg\psi$ is a star-nested Horn formula. Now, let $q > 0$ and ψ be a star-nested Horn formula with sets $L, S_0, \ldots, S_q$. Suppose that the claim is true for all formulas with strictly fewer sets. If we denote by L_0 the subset of literals in L that appear in unit clauses of ψ, then ψ can be rewritten as

$$\psi = \left(\bigwedge_{l \in L_0} l\right) \wedge \left((\bigvee_{s \in S_1} s) \vee \phi\right)$$

where ϕ is Horn and star-nested with sets $L \setminus L_0, S_1 \setminus S_1, S_2 \setminus S_1, \ldots, S_q \setminus S_1$. In particular, ϕ is star-nested with one fewer set than ψ. By induction, $\neg\phi$ can be assumed to be Horn and star-nested with sets $L', S'_0, \ldots, S'_p$. Then, we have

$$\neg\psi = \left(\bigvee_{l \in L_0} \neg l\right) \vee \left((\bigwedge_{s \in S_1} \neg s) \wedge \neg\phi\right)$$

and hence $\neg\psi$ is star-nested with sets $S''_0 = \emptyset, S''_1 = S'_0 \cup \{\neg l \mid l \in L_0\}, \ldots, S''_{p+1} = S'_p \cup \{\neg l \mid l \in L_0\}$, and $L'' = L' \cup \{\neg s \mid s \in S_1\}$. □

Proposition 3. *Let R be a boolean relation such that min is a polymorphism of both R and $\neg R$. Then $R(x_1, \ldots, x_r) \equiv \psi(x_1, \ldots, x_r)$, where ψ is a star-nested Horn formula.*

Proof. We proceed by induction on the arity r of R. The claim is true for $r = 1$ since R is either empty, complete, or equivalent to a unit clause; in all cases it is expressible by a star-nested Horn formula. Let $r > 1$ and suppose that the claim is true for all relations whose arity is strictly smaller than r. Let R be a relation of arity r such that min is a polymorphism of both R and $\neg R$. We assume without loss of generality that the all-zeroes tuple of length r belongs to R. (If this is not the case, then $\neg R$ contains this tuple and we prove the claim on $\neg R$ instead.) If R is complete then we are done. Otherwise, its negation $\neg R = \{t_1, \ldots, t_n\}$ is not empty. Since $\neg R$ has the polymorphism min (which we can assume to be of any arity), we have $t = \min(t_1, \ldots, t_n) \in \neg R$. Note that each t_i is a tuple, so here the operation min is applied componentwise to the set of tuples $t_1, \ldots, t_n$. The tuple $(0, \ldots, 0)$ does not belong to $\neg R$, so the set $P = \{i \leq r \mid t[i] = 1\}$ is not empty. We assume without loss of generality that $P = \{1, \ldots, c\}$. Since $t_j[i] = 1$ for all $j \in \{1, \ldots, n\}$ and $i \in P$, there exists a relation Q such that $\neg R(x_1, \ldots, x_r) \equiv x_1 \wedge \ldots \wedge x_c \wedge Q(x_{c+1}, \ldots, x_r)$. Both Q and $\neg Q$ have the polymorphism min (because Q is a projection of $\neg R$ and $\neg Q$ is a projection of a conjunction of R with unit clauses; the polymorphism min is invariant under these transformations) and the arity of Q is strictly smaller than r. By induction, there exists a star-nested Horn formula ψ such that $\neg Q(x_{c+1}, \ldots, x_r) \equiv \psi(x_{c+1}, \ldots, x_r)$. Then, we have

$$\begin{aligned} &R(x_1, \ldots, x_r) \\ &\quad \equiv \neg(x_1 \wedge \ldots \wedge x_c \wedge Q(x_{c+1}, \ldots, x_r)) \\ &\quad \equiv \neg x_1 \vee \ldots \vee \neg x_c \vee \neg Q(x_{c+1}, \ldots, x_r) \\ &\quad \equiv \neg x_1 \vee \ldots \vee \neg x_c \vee \psi(x_{c+1}, \ldots, x_r) \end{aligned}$$

and hence R is equivalent to a star-nested Horn formula by distributivity of $\vee$ over $\wedge$. □

Theorem 2. *Let $\mathcal{L}$ be a boolean constraint language. The following are equivalent:*

(i) $\mathcal{L}$ has the polymorphism min and is closed under taking complements

(ii) Each relation in $\mathcal{L}$ is equivalent to a star-nested Horn formula

Proof. Follows from Proposition 2 and Proposition 3. □

We also note that, given in input the list of tuples of a relation R, star-nested formulas for R and its complement $\neg R$ can be constructed in polynomial time if they exist. The algorithm is given by the recursive constructions used in the proofs of Proposition 2 and Proposition 3.

An anti-Horn formula is *star-nested* if replacing each literal by its negation yields a star-nested Horn formula. The following directly follows from the arguments above, with only slight adaptations.

Theorem 3. *Let $\mathcal{L}$ be a boolean constraint language. The following are equivalent:*

(i) $\mathcal{L}$ has the polymorphism max and is closed under taking complements

(ii) Each relation in $\mathcal{L}$ is equivalent to a star-nested anti-Horn formula

4.2 Affine

We now turn our attention to the case of $\mathcal{L}_{\text{miny}}$, which is straightforward.

Theorem 4. *Let $\mathcal{L}$ be a boolean constraint language. The following are equivalent:*

(i) $\mathcal{L}$ has the polymorphism miny and is closed under taking complements

(ii) Each relation in $\mathcal{L}$ is equivalent to a linear equation over GF(2), the finite field of two elements.

Proof. The fact that any language satisfying (ii) is closed under taking complements is trivial, as the complement of the equation $a_1x_1 + \ldots + a_rx_r = b$ is $a_1x_1 + \ldots + a_rx_r = 1 - b$. In addition, relations equivalent to linear equations over GF(2) have the minority polymorphism (Jeavons, Cohen, and Gyssens 1995). This establishes (ii) ⇒ (i).

Now, let R be a relation of arity r such that both R and $\neg R$ have the minority polymorphism. If R is either empty or complete then it is expressible as a linear equation ($0 = 1$ or $0 = 0$, respectively). Otherwise, both R and $\neg R$ correspond to the solution sets of systems of linear equations over GF(2) that are not degenerate (i.e. at least one equation has a nonzero coefficient). Since any nondegenerate linear equation over GF(2) over r variables has exactly 2^{r-1} solutions, we have $|R| = |\neg R| = 2^{r-1}$ and only one equation will remain in both systems after discarding all redundant equations. This establishes (i) ⇒ (ii) and concludes the proof. □

4.3 Conjunctions of 2-clauses

As mentioned above, over boolean domains a relation has the polymorphism maj if and only if it is a conjunctions of 2-clauses (clauses containing up to two literals). Thus, to complete the study of tractable cases identified in Theorem 1, we now characterise those formulas Φ such that both Φ and $\neg\Phi$ are expressible as conjunctions of 2-clauses.

A *2-clause* is a clause consisting of at most two literals and a *2-term* is a term consisting of at most two literals. The following lemma follows immediately from De Morgans' theorem.

Lemma 2. *A boolean formula Φ such that $\neg\Phi$ is expressible as conjunction of 2-clauses is expressible as a disjunction of 2-terms.*

Lemma 3. *Suppose that a boolean formula Φ is such that Φ is expressible as conjunctions of 2-clauses and also as a disjunction of 2-terms. Suppose, furthermore, that $\Phi \equiv (a \vee b) \wedge \Phi_1$ and $\Phi \equiv (c \wedge d) \vee \Phi_2$. Then there is a non-empty intersection between the two sets of literals $\{a, b\}$ and $\{c, d\}$.*

Proof. With the assignments $a = b = 0$ and $c = d = 1$ we have a contradiction. This can only be avoided if the sets of literals $\{a, b\}$ and $\{c, d\}$ intersect. □

Lemma 4. *Suppose that a boolean formula Φ is such that Φ is expressible as a conjunction of 2-clauses and also as a disjunction of 2-terms of the form $\Phi = a \vee \Phi_1$, where a is a literal. Then Φ is of one of the three forms (1) a, (2) $a \vee b$, or (3) $(a \vee b) \wedge (a \vee c)$.*

Proof. Suppose that $\Phi \equiv (b \vee c) \wedge \Phi_2$. Setting $a = 1$ and $b = c = 0$ leads to a contradiction, so to render this impossible we must have $a = b$ or $a = c$. Since this is true for any conjunct, when Φ is expressed as a conjunction of 2-clauses, we can deduce that $\Phi \equiv \bigwedge_{i=1}^{m} (a \vee b_i)$ for some literals $b_1, \ldots, b_m$. Since Φ is also expressible as a disjunction of 2-terms, we only need to consider the cases in which $m \leq 2$. When we include the case $\Phi = a$ we have the three cases (1) a, (2) $a \vee b$, (3) $(a \vee b) \wedge (a \vee c)$. □

We give without proof the analogous lemma obtained by exchanging conjunction and disjunction.

Lemma 5. *Suppose that a boolean formula Φ is such that Φ is expressible as a disjunction of 2-terms and also as a conjunction of 2-clauses of the form $\Phi = a \wedge \Phi_1$, where a is a literal. Then Φ is of one of the three forms (1) a, (2) $a \wedge b$, or (3) $(a \wedge b) \vee (a \wedge c)$.*

Observe that case (3) in Lemma 5 when written as a conjunction of 2-clauses is $a \wedge (b \vee c)$.

A *binary term* is a 2-term that contains exactly two distinct literals.

Lemma 6. *Suppose that a boolean formula $\Phi \neq \bot$ is such that Φ is expressible as a conjunction of 2-clauses and also as a disjunction of binary terms of the form $\Phi = (a \wedge c) \vee (b \wedge d) \vee \Phi_1$, where a, b, c, d are distinct literals. Then Φ is of one of the three forms (1) $(a \vee b) \wedge (c \vee d)$, (2) $(a \vee b) \wedge (b \vee c) \wedge (c \vee d)$, or (3) $(a \vee b) \wedge (b \vee c) \wedge (a \vee d) \wedge (c \vee d)$ for distinct literals a, b, c, d.*

Proof. Applying Lemma 3 twice, we know that all conjuncts, when Φ is expressed as a conjunction of 2-clauses, must contain one of a, c and one of b, d. Since a, b, c, d are distinct literals, we can deduce that the only possible 2-clauses are $(a \vee b)$, $(b \vee c)$, $(a \vee d)$ and $(c \vee d)$. Eliminating symmetrically equivalent cases, by exhaustive search, we easily obtain only three distinct cases, namely Φ is of one of the three forms (1) $(a \vee b) \wedge (c \vee d)$, (2) $(a \vee b) \wedge (b \vee c) \wedge (c \vee d)$, or (3) $(a \vee b) \wedge (b \vee c) \wedge (a \vee d) \wedge (c \vee d)$. □

Observe that although a, b, c, d are distinct literals, the variables are not necessarily distinct. For example, if $d = \neg a$ then case (1) becomes $(a \vee b) \wedge (\neg a \vee c)$.

Lemma 7. *Suppose that a boolean formula Φ, expressible as a non-empty conjunction of 2-clauses, is also expressible as a non-empty disjunction of binary terms in which each pair of terms share a literal. Then either Φ is of the form $\Phi = a \wedge \Phi_1$, where a is a literal, or Φ is of the form $(a \vee b) \wedge (b \vee c) \wedge (a \vee c)$.*

Proof. If Φ can be expressed as a disjunction of 2-terms with only one term or two terms (which share a literal), then Φ is of the form $\Phi = a \wedge \Phi_1$, for some literal a. If Φ can be expressed as a disjunction of three distinct binary terms (where each pair of terms shares a literal), then Φ is of the form $(a \vee b) \wedge (b \vee c) \wedge (a \vee c)$. There is no set of four distinct binary terms which satisfy the property that each pair shares a literal. □

We now obtain the following characterisation theorem.

Proposition 4. *Let Φ be a boolean formula such that both Φ and $\neg\Phi$ are expressible as non-empty conjunctions of 2-clauses. Then Φ has one of the following forms (in which a, b, c, d are distinct literals):*

(1) a,
(2) $a \vee b$,
(3) $a \wedge b$,
(4) $a \wedge (b \vee c)$,
(5) $(a \vee b) \wedge (a \vee c)$,
(6) $(a \vee b) \wedge (c \vee d)$,
(7) $(a \vee b) \wedge (b \vee c) \wedge (c \vee d)$,

(8) $(a \vee b) \wedge (b \vee c) \wedge (a \vee c)$,

(9) $(a \vee b) \wedge (b \vee c) \wedge (a \vee d) \wedge (c \vee d)$.

Proof. By Lemma 2, we are interested in Φ that can be expressed as a conjunction of 2-clauses and a disjunction of 2-terms. If Φ, when written as a disjunction of 2-terms, has a unary term (i.e. Φ can be written in the form $a \vee \Phi_1$), then Lemma 4 applies (cases (1), (2), (5)). If Φ can be expressed as a disjunction of binary terms, two of which share no literals, then Lemma 6 applies (cases (6), (7), (9)). If Φ can be expressed as a disjunction of binary terms, each pair of which share a literal, then Lemma 7 applies (case (8)). In the subcase of Lemma 7 in which Φ can be written in the form $a \wedge \Phi_1$, Lemma 5 applies (cases (1), (3), (4)). □

The following corollary is simply a more succinct rewriting of Proposition 4.

Corollary 2. *If Φ is a boolean formula such that both Φ and $\neg\Phi$ are expressible as non-empty conjunctions of 2-clauses, then Φ has one of the three following forms (in which the four literals are not necessarily distinct):*

(i) $(a \vee b) \wedge (c \vee d)$,

(ii) $(a \vee b) \wedge (b \vee c) \wedge (c \vee d)$,

(iii) $(a \vee b) \wedge (b \vee c) \wedge (a \vee d) \wedge (c \vee d)$.

Proof. We can obtain the nine cases listed in Proposition 4 as follows: (1) set $a = b = c$ in (iii), (2) set $a = c$ and $b = d$ in (iii), (3) set $a = b$ and $c = d$ in (iii), (4) set $a = d$ in (iii), (5) set $a = c$ in (iii), (6) is case (i) (7) is case (ii), (8) set $a = d$ in (ii), (9) is case (iii). □

It is straightforward to verify that the converse to Corollary 2 holds, that is, any formula Φ satisfying at least one of items (i), (ii) or (iii) is such that both Φ and $\neg\Phi$ are expressible as conjunctions of 2-clauses. In the following, we use the name *square 2CNF* for formulas that are expressible as both conjunctions of 2-clauses and disjunctions of 2-terms (characterised in Proposition 4 and Corollary 2). The name reflects the fact these formulas are the subformulas of the square given by item (iii) of Corollary 2 (seeing literals a,b,c,d as vertices and clauses as edges).

It is worth observing that square 2CNF formulas include all binary relations. For example, the relation $a \neq b$ can be obtained by setting $c = \neg b$ and $d = \neg a$ in $(a \vee b) \wedge (b \vee c) \wedge (c \vee d)$.

Theorem 5. *Let $\mathcal{L}$ be a boolean constraint language. The following are equivalent:*

(i) $\mathcal{L}$ has the polymorphism maj and is closed under taking complements

(ii) Each relation in $\mathcal{L}$ is equivalent to a square 2CNF, i.e. either empty, complete, or expressible in one of the three following forms (in which the four literals are not necessarily distinct): (i) $(a \vee b) \wedge (c \vee d)$, *(ii)* $(a \vee b) \wedge (b \vee c) \wedge (c \vee d)$, *(iii)* $(a \vee b) \wedge (b \vee c) \wedge (a \vee d) \wedge (c \vee d)$.

4.4 The dichotomy for boolean languages

Bringing together what we have learnt in this section, we have the following theorem.

Theorem 6. *Let $\mathcal{L}$ be a finite boolean language closed under taking complements. Then, assuming P≠NP,* wAXpDT*($\mathcal{L}$) ∈ P iff at least one of the conditions holds:*

1. *Each relation in $\mathcal{L}$ is equivalent to a star-nested Horn formula*
2. *Each relation in $\mathcal{L}$ is equivalent to a star-nested anti-Horn formula*
3. *Each relation in $\mathcal{L}$ is equivalent to a linear equation over GF(2)*
4. *Each relation in $\mathcal{L}$ is equivalent to a square 2CNF formula.*

The requirement that $\mathcal{L}$ is finite in Theorem 6 arises from technicalities related to the representation of infinite languages. Indeed, certain degenerate representations for the relations of an infinite language $\mathcal{L}$ may be problematic from an algorithmic perspective. For example, the promise that the relations of $\mathcal{L}$ are equivalent to star-nested Horn formulas might not be sufficient to ensure tractability (or even membership in NP) if they are encoded in a way that makes even the most elementary relational operations NP-hard. However, this theorem is still true for infinite languages if one makes the mild assumptions that (i) relations equivalent to linear equations are always represented as such, and (ii) the representation used for relations equivalent to star-nested Horn/anti-Horn formulas allows for checking in polynomial time whether a given assignment extends to a tuple.

Example 8. *Consider the language $\mathcal{L}$ of Example 3, which consists of all 2-conjunctions of 2-clauses. Now, extend $\mathcal{L}$ with pseudo-boolean constraints $a + b + c \geq 2$ for any literals a, b, c, where summation is over $\mathbb{Z}$. This larger language $\mathcal{L}'$ is closed under taking complements (the complement of $a + b + c \geq 2$ is $\neg a + \neg b + \neg c \geq 2$), and all constraints in $\mathcal{L}'$ can be expressed as square 2CNF formulas because $a+b+c \geq 2 \equiv (a \vee b) \wedge (b \vee c) \wedge (c \vee a)$. Therefore, by Theorem 6 we have* wAXpDT*($\mathcal{L}'$) ∈ P. However, no quaternary pseudo-boolean constraint $a + b + c + d \geq k$ with $1 \leq k < 4$ can be expressed as a square 2CNF formula. In fact, adding any such constraint to $\mathcal{L}$ would cause the corresponding* wAXpDT *problem to become NP-complete by*

Theorem 6 as the resulting language would violate each of the four tractability conditions.

5 Path-based explanations

Izza et al. (Izza, Ignatiev, and Marques-Silva 2022) introduced the notion of path-based explanations for decision trees: a path-based explanation is a subset of the conditions on a path to a leaf. Since they study DT's in which conditions are arbitrary unary constraints of the form $x_i \in S$, this is also the basic building block of path explanations. Such explanations are potentially more useful to the user than an AXp which is composed of literals of the form $x_i = u$. We generalize the notion of path-based explanations to MDT's, before showing that the P/NP-hard dichotomy for boolean languages also holds for this alternative notion of explanation. Recall that we use path(α) to represent the set of conditions satisfied on the path from the root to a leaf α of a MDT.

Definition 4. *Let κ be a classifier calculated by an MDT, $\mathbf{v}$ a feature-vector, and α the leaf of the MDT attained when calculating $\kappa(\mathbf{v})$. A* weak APXp *(weak abductive path explanation) of the decision $\kappa(\mathbf{x}) = c$ is a subset P of the conditions path(α) such that any assignment $\mathbf{y}$ that satisfies the conditions P also satisfies $\kappa(\mathbf{y}) = c$. An* APXp *(abductive path explanation) of a decision is a subset-minimal Weak APXp.*

Let wAPXpDT($\mathcal{L}$) denote the problem of deciding whether a set of constraints is a weak APXp for a given decision taken by an $\mathcal{L}$-DT. We can deduce from Lemma 1 that finding an APXp of a decision taken by an $\mathcal{L}$-DT is polynomial-time if wAPXpDT($\mathcal{L}$) $\in$ P. We omit the proof of the following theorem since its proof is almost identical to the proof of Theorem 1.

Theorem 7. *Let $\mathcal{L}$ be a finite boolean language closed under taking complements. Then, assuming P$\neq$NP,* wAPXpDT*($\mathcal{L}$) $\in$ P iff $\mathcal{L}$ has either max, min, maj or miny as a polymorphism.*

Corollary 3. *Let $\mathcal{L}$ be a finite boolean language closed under taking complements. Then* wAPXpDT*($\mathcal{L}$) $\in$ P iff* wAXpDT*($\mathcal{L}$) $\in$ P*

It follows that we have the same tractable-explainability dichotomy for boolean languages for path-based explanations (APXp's) as for instance-based explanations (AXp's) (Theorem 6).

6 Conclusion

We have shown the close link between classes of multivariate decision trees for which decisions can be explained in polynomial time and tractable constraint languages closed under complement. We have shown that tractable explainability applies to existing and well-studied classes of MDTs, such as oblique DTs, but also to novel classes of MDTs. Such novel classes provide generalisations of classical DTs in that branching is possible not only on the value of a single variable but also according to specific (non-linear) conditions on two or more variables.

Interesting open questions concern the evaluation of the practical utility (Cañete-Sifuentes, Monroy, and Medina-Pérez 2021; Li, Dong, and Kothari 2005) and the theoretical computational power of such generalised DTs. There is a rich history of the study of MDTs with linear conditions as a computational model, such as bounds on the depth of such decision trees to test the equality of two sets (Reingold 1972). An avenue of future work is a similar theoretical study of the computational power of MDTs with generalised interval constraints, two-fan constraints, UTVPI constraints, star-nested Horn constraints (studied in Section 4.1), or square 2CNF formulas (studied in Section 4.3) to determine whether there is a substantial gain in depth or size when compared with classical DTs.

Our P/NP-hard dichotomy for boolean languages closed under complement is an interesting theoretical result which may find applications in other domains. This dichotomy for boolean languages can also be seen as a foundation on which to build a characterisation of tractable finite-domain languages closed under complement.

An independent question is the so-called recognition problem: given an arbitrary multivariate DT, determine whether the set of constraints it uses is a sublanguage of one of the tractable languages we have identified. It is reasonable to assume that this problem would be solved off-line, if at all.

Acknowledgments

This work was supported by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future – PIA3" under grant agreement no. ANR-19-PI3A-0004.

References

Amgoud, L., and Ben-Naim, J. 2022. Axiomatic foundations of explainability. In Raedt, L. D., ed., *IJCAI*, 636–642. ijcai.org.

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.; and Marquis, P. 2022. On the explanatory power of boolean decision trees. *Data Knowl. Eng.* 142:102088.

Barceló, P.; Monet, M.; Pérez, J.; and Subercaseaux, B. 2020. Model interpretability through the lens of computational complexity. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *NeurIPS*.

Barros, R. C.; Jaskowiak, P. A.; Cerri, R.; and de Leon Ferreira de Carvalho, A. C. P. 2014. A framework for bottom-up induction of oblique decision trees. *Neurocomputing* 135:3–12.

Brodley, C. E., and Utgoff, P. E. 1995. Multivariate decision trees. *Mach. Learn.* 19(1):45–77.

Bulatov, A. A. 2017. A dichotomy theorem for nonuniform csps. In Umans, C., ed., *FOCS*, 319–330. IEEE Computer Society.

Cañete-Sifuentes, L.; Monroy, R.; and Medina-Pérez, M. A. 2021. A review and experimental comparison of multivariate decision trees. *IEEE Access* 9:110451–110479.

Carreira-Perpiñán, M. Á., and Hada, S. S. 2021. Counterfactual explanations for oblique decision trees: Exact, efficient algorithms. In *AAAI*, 6903–6911. AAAI Press.

Carreira-Perpiñán, M. Á., and Tavallali, P. 2018. Alternating optimization of decision trees, with application to learning sparse oblique trees. In Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *NeurIPS*, 1219–1229.

Chen, Z., and Toda, S. 1995. The complexity of selecting maximal solutions. *Inf. Comput.* 119(2):231–239.

Cooper, M. C., and Marques-Silva, J. 2023. Tractability of explaining classifier decisions. *Artif. Intell.* 316.

Cooper, M. C.; Cohen, D. A.; and Jeavons, P. 1994. Characterising tractable constraints. *Artif. Intell.* 65(2):347–361.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2019. A survey of methods for explaining black box models. *ACM Comput. Surv.* 51(5):93:1–93:42.

Heath, D. G.; Kasif, S.; and Salzberg, S. 1993. Induction of oblique decision trees. In Bajcsy, R., ed., *IJCAI*, 1002–1007. Morgan Kaufmann.

Huang, X.; Izza, Y.; Ignatiev, A.; Cooper, M. C.; Asher, N.; and Marques-Silva, J. 2022. Tractable explanations for d-dnnf classifiers. In *AAAI*, 5719–5728. AAAI Press.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019. Abduction-based explanations for machine learning models. In *AAAI*, 1511–1519. AAAI Press.

Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2022. On tackling explanation redundancy in decision trees. *J. Artif. Intell. Res.* 75:261–321.

Jeavons, P., and Cooper, M. C. 1995. Tractable constraints on ordered domains. *Artif. Intell.* 79(2):327–339.

Jeavons, P.; Cohen, D. A.; and Gyssens, M. 1995. A unifying framework for tractable constraints. In Montanari, U., and Rossi, F., eds., *Principles and Practice of Constraint Programming - CP'95*, volume 976 of *LNCS*, 276–291. Springer.

Jeavons, P.; Cohen, D. A.; and Gyssens, M. 1997. Closure properties of constraints. *J. ACM* 44(4):527–548.

Lahiri, S. K., and Musuvathi, M. 2005. An efficient decision procedure for UTVPI constraints. In Gramlich, B., ed., *Frontiers of Combining Systems, 5th International Workshop*, volume 3717 of *Lecture Notes in Computer Science*, 168–183. Springer.

Li, Y.; Dong, M.; and Kothari, R. 2005. Classifiability-based omnivariate decision trees. *IEEE Trans. Neural Networks* 16(6):1547–1560.

Marques-Silva, J., and Ignatiev, A. 2022. Delivering trustworthy AI through formal XAI. In *AAAI*, 12342–12350. AAAI Press.

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

Murthy, S. K.; Kasif, S.; and Salzberg, S. 1994. A system for induction of oblique decision trees. *J. Artif. Intell. Res.* 2:1–32.

Reingold, E. M. 1972. On the optimality of some set algorithms. *J. ACM* 19(4):649–659.

Schaefer, T. J. 1978. The complexity of satisfiability problems. In Lipton, R. J.; Burkhard, W. A.; Savitch, W. J.; Friedman, E. P.; and Aho, A. V., eds., *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, 216–226. ACM.

Shih, A.; Choi, A.; and Darwiche, A. 2018. A symbolic approach to explaining bayesian network classifiers. In Lang, J., ed., *IJCAI*, 5103–5111. ijcai.org.

Wäldchen, S.; MacDonald, J.; Hauch, S.; and Kutyniok, G. 2021. The computational complexity of understanding binary classifier decisions. *J. Artif. Intell. Res.* 70:351–387.

Wickramarachchi, D. C.; Robertson, B. L.; Reale, M.; Price, C. J.; and Brown, J. 2016. HHCART: an oblique decision tree. *Comput. Stat. Data Anal.* 96:12–23.

Zhu, H.; Murali, P.; Phan, D. T.; Nguyen, L. M.; and Kalagnanam, J. 2020. A scalable MIP-based method for learning optimal multivariate decision trees. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *NeurIPS*.

Zhuk, D. 2020. A proof of the CSP dichotomy conjecture. *J. ACM* 67(5):30:1–30:78.