Journal of Artificial Intelligence Research 75 (2022) 261-321 Submitted 12/2021; published 09/2022

On Tackling Explanation Redundancy in Decision Trees

Yacine Izza yacine.izza@univ-toulouse.fr
University of Toulouse, Toulouse, France

Alexey Ignatiev alexey.ignatiev@monash.edu
Monash University, Melbourne, Australia

Joao Marques-Silva joao.marques-silva@irit.fr
IRIT, CNRS, Toulouse, France

Abstract

Decision trees (DTs) epitomize the ideal of interpretability of machine learning (ML) models. The interpretability of decision trees motivates explainability approaches by so-called intrinsic interpretability, and it is at the core of recent proposals for applying interpretable ML models in high-risk applications. The belief in DT interpretability is justified by the fact that explanations for DT predictions are generally expected to be succinct. Indeed, in the case of DTs, explanations correspond to DT paths. Since decision trees are ideally shallow, and so paths contain far fewer features than the total number of features, explanations in DTs are expected to be succinct, and hence interpretable. This paper offers both theoretical and experimental arguments demonstrating that, as long as interpretability of decision trees equates with succinctness of explanations, then decision trees ought not be deemed interpretable. The paper introduces logically rigorous path explanations and path explanation redundancy, and proves that there exist functions for which decision trees must exhibit paths with explanation redundancy that is arbitrarily larger than the actual path explanation. The paper also proves that only a very restricted class of functions can be represented with DTs that exhibit no explanation redundancy. In addition, the paper includes experimental results substantiating that path explanation redundancy is observed ubiquitously in decision trees, including those obtained using different tree learning algorithms, but also in a wide range of publicly available decision trees. The paper also proposes polynomial-time algorithms for eliminating path explanation redundancy, which in practice require negligible time to compute. Thus, these algorithms serve to indirectly attain irreducible, and so succinct, explanations for decision trees. Furthermore, the paper includes novel results related with duality and enumeration of explanations, based on using SAT solvers as witness-producing NP-oracles.

1. Introduction

The cognitive limits of human decision makers (Miller, 1956) substantiate why succinctness is one of the key requirements of explanations of machine learning (ML) models. Succinct explanations are generally accepted to be easier to understand by human decision makers, but are also easier to diagnose or debug. Decision trees (DTs) epitomize so-called interpretable machine learning models (Breiman, 2001; Rudin, 2019; Molnar, 2020), in part because paths in the tree (which are possibly short, and so potentially succinct) represent explanations of predictions.

Decision trees (DTs) find a wide range of practical uses[1]. Moreover, DTs are the most visible example of a collection of machine learning (ML) models that have recently been advocated as essential for high-risk applications (Rudin, 2019). Decision trees also represent explainability approaches based on intrinsic interpretability (Molnar, 2020)[2]. Given a decision tree, some input and the resulting prediction, the explanation associated with that prediction is the path in the decision tree consistent with the input. This simple observation justifies in part why decision trees have been deemed interpretable for at least two decades (Breiman, 2001), an observation that is widely taken for granted (Freitas, 2013; Bertsimas et al., 2019a; Molnar, 2020; Arrieta et al., 2020), that motivates many of the applications referenced above, and which explains the interest in learning optimal decision trees, especially in recent years[3], and notably when it is well-known that learning optimal (smallest) DTs is NP-hard (Hyafil and Rivest, 1976). It should be noted that earlier work encompasses different optimality criteria, some of which is tightly related with succinctness of explanations (e.g. as measured by average path length). In contrast with earlier work, this paper offers a different perspective. Concretely, the paper proves that paths in decision trees can be arbitrarily larger (on the number of features) than a logically rigorous explanation for a prediction. Furthermore, the experimental results, obtained on a wide range of datasets and also on publicly available DTs, demonstrate that DTs in practice naturally exhibit the same limitation, i.e. DTs almost invariably have paths that contain more literals than what a logically rigorous explanation requires. The experiments also demonstrate that redundancy of literals in DT paths exists *even* for optimal (and/or sparse) decision trees (Bertsimas and Dunn, 2017; Hu et al., 2019; Lin et al., 2020; Rudin et al., 2021). The main corollary of the paper's theoretical and experimental results is that succinctness of explanations cannot be ensured by the paths in decision trees, and must instead be computed with logically rigorous approaches. This corollary has significant practical consequences, in some high-risk applications, but also in situations that are safety-critical. For example, in a medical application (Valdes et al., 2016), an explanation that contains literals that are unnecessary, may prevent a physician from focusing on the symptoms that are actually crucial for correct diagnosis. In more general settings, non-succinct explanations may be beyond the grasp of human-decision makers (Miller, 1956), whereas (subset-minimal) succinct explanations may not.

1. From an ever-increasing range of practical uses, example references include (Wu et al., 2008; Wu and Kumar, 2009; Kotsiantis, 2013; Valdes et al., 2016; Bertsimas et al., 2018a,b, 2019c; Fletcher and Islam, 2019; Bertsimas et al., 2019a,b,d,e,f; Lundberg et al., 2020; Gennatas et al., 2020; Bertsimas et al., 2020c; Bertsimas and Wiberg, 2020; Bertsimas et al., 2020b; Ong et al., 2020; Orfanoudaki et al., 2020; Cho et al., 2020; Bertsimas et al., 2020a; Sosa-Hernández et al., 2021; Siers and Islam, 2021; Bertsimas and Stellato, 2021; El Hechi et al., 2021; Maurer et al., 2021; Bertsimas et al., 2021; Bandi and Bertsimas, 2021).
2. Interpretability is generally accepted to be a subjective concept, without a rigorous definition (Lipton, 2018). Similar to other works (Molnar, 2020), this paper relates interpretability with succinctness of the explanations provided.
3. Standard references include (Nijssen and Fromont, 2007; Bessiere et al., 2009; Nijssen and Fromont, 2010; Bertsimas and Dunn, 2017; Verwer and Zhang, 2017; Narodytska et al., 2018; Verwer and Zhang, 2019; Hu et al., 2019; Avellaneda, 2019, 2020; Verhaeghe et al., 2020a; Aglin et al., 2020a; Lin et al., 2020; Janota and Morgado, 2020; Hu et al., 2020; Verhaeghe et al., 2020b; Aglin et al., 2020b; Demirovic and Stuckey, 2021; Schidler and Szeider, 2021; Ordyniak and Szeider, 2021; Shati et al., 2021; Alos et al., 2021; Demirovic et al., 2022; McTavish et al., 2022)

Explanations, such as the ones informally sketched above, essentially represent an answer to a "**Why?**" question, i.e. *why* is the prediction the one obtained? Such explanations aim at succinctness by being subset-minimal (or irreducible). These explanations are referred to as PI-explanations or abductive explanations (AXp's) (Shih et al., 2018; Ignatiev et al., 2019a). A different class of explanations answer a "**Why not?**" question, i.e. *why didn't* one get a prediction different from the one obtained? Or what would be necessary to change to get a different prediction? This sort of explanations also aim at succinctness by being subset-minimal, and are referred to as contrastive explanations (CXp's) (Miller, 2019; Ignatiev et al., 2020b).

This paper shows that succinctness of explanations of paths in DTs can be achieved efficiently in practice. Concretely, the paper shows that logically rigorous explanations, i.e. both AXp's and CXp's, can be computed in polynomial time, and so in practice require negligible time to compute. Furthermore, the paper shows that, whereas AXp's can be arbitrarily smaller than a path in a DT, CXp's *cannot*. Concretely, the paper shows that, for any prediction, a contrastive explanation corresponds exactly to the conditions provided by one of the paths in the decision tree. Furthermore, the paper proposes path-specific variants of both AXp's and CXp's, as opposed to the instance-specific definitions studied in earlier work. Path-specific explanations relate with the conditions (i.e. the literals) on a given path, and so are instance-independent. In addition, the paper shows that these variants of AXp's and CXp's can also be computed in polynomial time.

Compared with earlier work (Izza et al., 2020; Huang et al., 2021b), this paper offers comprehensive evidence regarding the redundancy of path-based explanations in DTs. Concretely, the paper proves that i) size-minimal DTs can exhibit arbitrary explanation redundancy, ii) in practice explanation redundancy is often observed, iii) DTs without explanation redundancy correspond to a very specific class of classifiers represented as non-overlapping minimal disjunctive normal form formulas, iv) (provably) optimal sparse DTs also invariably exhibit path explanation redundancy, v) example DTs used in most textbooks and other representative references also exhibit explanation redundancy and, finally, vi) other types of explanations (concretely path explanations, which are investigated in this paper) reveal important properties in terms of redundancy of explanations. In addition, the paper builds on earlier work (Izza et al., 2020; Huang et al., 2021b) showing that tools claiming interpretable AI solutions (Bertsimas and Dunn, 2017; IAI, 2020) also exhibit path explanation redundancy, and that this occurs with other well-known decision tree learners. Therefore, our results serve to complement any state-of-the-art approach for learning DTs, allowing the computation of path explanations which are often shorter than DT paths. More importantly, our results can be used to provide much-needed succinct explanations in high-risk and safety-critical applications.

Main results. The paper's main results are organized as follows:

1. The paper formalizes in detail the computation of explanations in decision trees, such that decision trees are allowed to have both categorical and ordinal features, taking values from arbitrary domains, and such that an arbitrary number of classes is allowed;
2. The paper introduces explanation functions (as an extension of prime implicant explanations), and proposes conditions for monotonicity of the definition of abductive and

contrastive explanations, which in turn yields a generalized form of minimal hitting set duality between abductive and contrastive explanations;

3. The paper identifies nesting properties of abductive and contrastive explanations, which allows enumerating abductive explanations from a subset of the features;
4. The paper uses the two previous results to introduce path explanations and path explanation redundancy, where path explanations are distinguished from instance-specific explanations;
5. The paper proves that optimal decision trees can exhibit path explanation redundancy, and that DTs that do not exhibit path explanation redundancy must correspond to minimal generalized decision functions (GDF) (Huang et al., 2022) represented in disjunctive normal form (DNF). The class of functions that can be represented with such DNF GDFs is argued to be very unlikely to be obtained in practice;
6. The paper proposes algorithms for the computation of path explanations, as follows:
   (a) Three algorithms for computing abductive path explanations, two of which build on earlier work (Izza et al., 2020; Huang et al., 2021b), and a novel one that relates with reasoning about overconstrained Horn formulas;
   (b) One algorithm for computing all contrastive path explanations; and
   (c) One algorithm for enumerating abductive path explanations by starting from the hypergraph of contrastive path explanations.
7. The paper offers extensive experimental evidence, attesting to the significance of identifying and removing explanation redundancy from decision tree paths.

Organization. The paper is organized as follows. Section 2 introduces the notation and definitions used in the rest of the paper. Sections 3 and 4 detail the paper's theoretical foundations, namely duality results and path explanations for DTs. Path explanations are significant, because these allow relating abductive and contrastive explanations with the literals in the DT paths. Concretely, Section 3 proposes a generalization of abductive and contrastive explanations to explanation functions such that duality between explanations is respected. This section also reveals a nesting property of explanations, and introduces path explanations. Furthermore, the section shows how the two previous results apply in the case of path explanations for DTs. Moreover, Section 4 builds on path explanations to formalize *path explanation redundancy* (PXR) for DTs. First, this section proves that there exist minimum-size DTs that necessarily exhibit PXR. Second, the section relates DTs that do not exhibit path explanation redundancy with minimal generalized decision functions (Huang et al., 2021a, 2022). In addition, this section shows that optimal sparse DTs (Hu et al., 2019) exhibit PXR, and shows that PXR can represent in practice a much larger fraction of a path than the explanation itself. Section 5 proposes three polynomial-time algorithms for computing one abductive path explanation, including a novel and simple propositional Horn encoding. This section also covers the computation of contrastive path explanations, and the enumeration of path explanations. Section 6 presents experimental results that confirm the paper's main claims: i) PXR occurs naturally, and can be found in DTs used in a vast number of research and survey papers and textbooks published over the years; ii) PXR is ubiquitous in DTs learned with different tree learning algorithms, and that the time taken to compute explanations (be them abductive or contrastive) is always negligible; iii) PXR can represent a very significant percentage of the length of tree paths;

and iv) PXR exists even in trees that are optimal (and sparse) (Verwer and Zhang, 2019; Hu et al., 2019; Lin et al., 2020; Rudin et al., 2021). Section 7 overviews related work on computing explanations for DTs, and Section 8 concludes the paper.

2. Preliminaries

This section overviews the definitions and notation used in the remainder of the paper. Section 2.1 briefly summarizes the notation for functions used in the paper, emphasizing function parameterizations, which we will use to represent families of functions. Section 2.2 includes a brief overview of the logic foundations the paper builds upon. Section 2.3 introduces classification problems and the associated notation. Section 2.4 introduces decision trees and outlines a formalization that is vital for reasoning about DTs. Although DTs are among the best understood ML models, a rigorous formalization is required to reason about explanations. Afterwards, Section 2.5 overviews formal explainability. Finally, Section 2.6 summarizes the notation introduced in this section and used in the rest of the paper.

2.1 Function Representation

A function is well-known to be a mapping from one set to another. We will allow functions to be parameterized, thus in fact defining families of related functions, which depend on the choices of parameters. Furthermore, we will allow functions to be parameterized on an arbitrary (and not necessarily defined a priori) number of parameters. (Parameterization serves to represent families of functions, with arguments which are distinguished from the other arguments, e.g. selected features vs. points in feature space.) As an example, $f : D \to C$ maps a domain D into a codomain C. If $d \in D$, then $f(d)$ denotes the value of C that $d \in D$ is mapped to. $f(d; \pi_1, \pi_2)$ denotes that f is parameterized on some given parameters π_1 and π_2. Moreover, $f(d; \pi, \ldots)$ denotes that f is parameterized on π as well as on a number of additional but yet-undefined parameters. To keep the notation as simple as possible, we will reveal parameterizations only when relevant.

2.2 Logic Foundations

Definitions and notation standard in mathematical logic, concretely related with propositional logic and decidable fragments of first-order logic, will be used throughout the paper (Biere et al., 2021). Propositional formulas are defined over boolean variables taken from some set $X = \{x_1, x_2, \ldots, x_m\}$, where each boolean variable takes values from $\mathbb{B} = \{0, 1\}$. A literal is a variable x_i or its negation $\neg x_i$. A propositional formula is defined inductively using literals and the standard logic operators $\vee$ and $\wedge$ [4]: i) Literals are propositional formulas; ii) If φ_1 and φ_2 are propositional formulas, then $\varphi_1 \vee \varphi_2$ is a propositional formula; and iii) If φ_1 and φ_2 are propositional formulas, then $\varphi_1 \wedge \varphi_2$ is a propositional formula. A conjunctive normal form (CNF) formula φ is a conjunction of disjunctions of literals. A disjunction of literals is referred to as a *clause*. A disjunctive normal form (DNF) formula is a disjunction of conjunctions of literals. A conjunction of literals is referred to as

4. For simplicity, we restrict the set of allowed logic operators. The inductive definition of propositional formulas above could be extended to accommodate for universal and existential operators; it could also be extended to accommodate for other well-known logic operators, including $\neg$, $\to$ and $\leftrightarrow$, among others.

a *term*. A Horn formula is a CNF formula where each clause does not contain more than one non-negated literal. We will use quantification where necessary, with $\forall$ and $\exists$ having respectively the meaning of universal and existential quantification of variables over their domains.

A *truth assignment* represents a point $\mathbf{v} = (v_1, \ldots, v_m)$ of $\mathbb{B}^m = \{0,1\}^m$, where the value assigned to each x_i is associated with coordinate i. $\mathbf{v} \vDash \varphi$ is defined inductively on the structure of φ: i) $\mathbf{v} \vDash (\varphi_1 \vee \varphi_2)$ iff $\mathbf{v} \vDash \varphi_1$ or $\mathbf{v} \vDash \varphi_2$; ii) $\mathbf{v} \vDash (\varphi_1 \wedge \varphi_2)$ iff $\mathbf{v} \vDash \varphi_1$ and $\mathbf{v} \vDash \varphi_2$; iii) $\mathbf{v} \vDash \neg x_i$ iff $v_i = 0$; and iv) $\mathbf{v} \vDash x_i$ iff $v_i = 1$. If a truth assignment $\mathbf{v}$ is such that $\mathbf{v} \vDash \varphi$, then φ is *satisfied* by $\mathbf{v}$, and we say that $\mathbf{v}$ is a *model*; otherwise φ is *falsified* by $\mathbf{v}$, and we write $\mathbf{v} \nvDash \varphi$. A formula φ is *satisfiable* if there exists a truth assignment that satisfies φ; otherwise it is *unsatisfiable* (or *overconstrained*, or *inconsistent*). If π and κ are propositional formulas, then we write $(\pi \vDash \kappa)$ to denote that $\forall(\mathbf{x} \in \mathbb{B}^m).(\mathbf{x} \vDash \pi) \rightarrow (\mathbf{x} \vDash \kappa)$. Similarly, we write $\pi \nvDash \kappa$ to denote that $\exists(\mathbf{x} \in \mathbb{B}^m).(\mathbf{x} \vDash \pi) \wedge (\mathbf{x} \nvDash \kappa)$. A term π is a *prime implicant* of κ, if $\pi \vDash \kappa$ and for any term θ such that $\theta \vDash \pi \wedge \pi \nvDash \theta$, it does not hold that $\theta \vDash \kappa$. Similarly, a clause ψ is a *prime implicate* of κ, if $\kappa \vDash \psi$ and for any clause γ such that $\gamma \vDash \psi \wedge \psi \nvDash \gamma$, it does not hold that $\kappa \vDash \gamma$.

The definitions above can be extended to domains other than boolean domains, by allowing the variables to take values from domains that are not necessarily boolean, and by defining literals using appropriate relational operators (Biere et al., 2021). Well-known examples of relational operators include those in $\{\le, \ge, <, >, =, \in\}$, among others. We can also consider functions whose codomain is not necessarily boolean, and can also include those in logic formulas again using suitable relational operators. Concrete examples will be introduced later in this section when describing decision trees, but also when introducing formal explanations. Furthermore, in a number of situations, it is convenient to talk about formulas that consist of conjunctions of other formulas as *sets of constraints*, where each constraint can represent a clause, or a more complex (propositional) formula, thus allowing set notation to be used with conjunctions of constraints.

For an overconstrained formula, not all of its constraints can be satisfied simultaneously. In general, overconstrained formulas are split into a set of *hard* constraints (i.e. $\mathcal{H}$) and a set of *soft* (or *breakable*, or weighted, or costed) constraints (i.e. $\mathcal{B}$), In such settings, a number of computational problems can be defined for reasoning about the pairs $(\mathcal{H}, \mathcal{B})$, including: i) finding an assignment that maximizes the cost of satisfied soft constraints, i.e. the maximum satisfiability (MaxSAT) problem; ii) finding subset-maximal subsets of $\mathcal{B}$ which, together with $\mathcal{H}$ are satisfiable, i.e. finding a maximal satisfiable subset (MSS); iii) finding a subset-minimal set of clauses $\mathcal{C} \subseteq \mathcal{B}$ which, if removed from $\mathcal{B}$, cause $\mathcal{H} \cup (\mathcal{B} \setminus \mathcal{C})$ to be satisfiable, i.e. finding a minimal correction subset (MCS); and iv) finding a subset-minimal set of clauses $\mathcal{U} \subseteq \mathcal{B}$ which together with $\mathcal{H}$ are inconsistent, i.e. finding a minimal unsatisfiable subset (MUS). There is a comprehensive body of research on algorithms for reasoning about overconstrained formulas (Belov et al., 2012; Marques-Silva et al., 2013a,b; Mencía et al., 2015; Arif et al., 2015; Liffiton et al., 2016; Mencía et al., 2016; Marques-Silva et al., 2016, 2017; Marques-Silva and Mencía, 2020; Biere et al., 2021).

For some problems, we will use a SAT solver as an oracle. Although a SAT solver is used for solving a well-known NP-complete problem (Cook, 1971), it is also the case that a SAT solver ought not be equated with an NP oracle (Marques-Silva et al., 2017). This observation is justified by the fact that SAT solvers report satisfying assignments (or witnesses) for satisfiable formulas. Moreover, most SAT solvers also report *summaries* in the case of unsatisfiable formulas, where a summary is a subset of the clauses that is itself inconsistent. As a result, when using a SAT solver as a oracle, we are in fact considering a witness-producing (and most often summary-providing) NP-oracle.

2.3 Classification Problems

The paper considers classification problems, defined on a set of features $\mathcal{F} = \{1, \ldots, m\}$, where each feature i takes values from a domain $\mathcal{D}_i$, and $m = |\mathcal{F}|$ denotes the number of features. Each domain $\mathcal{D}_i$ may be categorical or ordinal. Ordinal domains can be integer or real-valued. The set of domains is represented by $\mathbb{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_m)$. The union of domains is $\mathbb{U} = \cup_{i \in \mathcal{F}} \mathcal{D}_i$. (For the sake of simplicity, several of the examples studied in this paper consider $\mathcal{D}_i = \{0, 1\}$ (i.e. binary features).) Feature space is defined by $\mathbb{F} = \mathcal{D}_1 \times \mathcal{D}_2 \times \ldots \times \mathcal{D}_m$. To refer to an arbitrary point in feature space we use the notation $\mathbf{x} = (x_1, \ldots, x_m)$, whereas to refer to a concrete (constant) point in feature space we use the notation $\mathbf{v} = (v_1, \ldots, v_m)$, with $v_i \in D_i$, $i = 1, \ldots, m$. Similarly to the case of domains, and for the sake of simplicity, most examples in the paper consider a binary classification problem, with two classes $|\mathcal{K}| = 2$, e.g. $\mathcal{K} = \{\mathbf{0}, \mathbf{1}\}$, $\mathcal{K} = \{\mathbf{N}, \mathbf{Y}\}$, or $\mathcal{K} = \{\ominus, \oplus\}$. However, the results in the paper apply to any decision (or classification) tree used as a classifier. A classifier computes a non-constant classification function κ that maps the feature space $\mathbb{F}$ into a set of classes, $\kappa : \mathbb{F} \to \mathcal{K}$. Furthermore, a *boolean classifier* is such that $\mathbb{F} = \{0, 1\}^m$ and $\mathcal{K} = \{0, 1\}$. An *instance* $\mathcal{I}$ (or example) denotes a pair $\mathcal{I} = (\mathbf{v}, c)$, where $\mathbf{v} \in \mathbb{F}$ and $c \in \mathcal{K}$, such that $\kappa(\mathbf{v}) = c$. To train a classifier (in our case we are interested in DTs), we start from a set of instances $\mathcal{I}_T = \{I_1, \ldots, I_n\}$. Algorithms for learning different families of classifiers can be found in standard references (Breiman et al., 1984; Quinlan, 1986, 1993; Ripley, 1996; Mitchell, 1997; Russell and Norvig, 2010; Flach, 2012; Zhou, 2012; Shalev-Shwartz and Ben-David, 2014; Alpaydin, 2014; Poole and Mackworth, 2017; Bramer, 2020; Zhou, 2021). There are also algorithms that learn optimal classifiers (e.g. decision trees), and some examples are referenced in Section 1.

In this paper, a *literal* represents a condition on the values of a feature. Depending on the value assigned to the feature, the literal can be satisfied or falsified. Throughout the paper, and for consistency of notation, literals will always be of the form $(x_i \in S_l)$, where $S_l \subseteq \mathcal{D}_i$. This literal is satisfied when feature i is assigned a value from set S_l; otherwise it is falsified. For simplicity of notation, when $|S_l| = 1$ and $S_l = \{v_i\}$, we may instead represent a literal by $(x_i = v_i)$. Moreover, the universe of literals is $\mathbb{L} = \{\mathcal{L} \,|\, \mathcal{L} = (x_i \in S_l), i \in \mathcal{F}, S_l \subseteq \mathcal{D}_i\}$.

A point $\mathbf{v} = (v_1, \ldots, v_m)$ in feature space ($\mathbf{v} \in \mathbb{F}$) can also be described by a set of m *literals*, each of the form $(x_i = v_i)$, i.e. $\{(x_i = v_i)|i = 1, \ldots, m\}$. Alternatively, literals may be represented using set notation, i.e. $\{(x_i \in \{v_i\}) \,|\, i = 1, \ldots, m\}$.

Finally, given the definitions above, the universe of *classification problems* is represented by the set $\mathbb{M} = \{\mathcal{M} \,|\, \mathcal{M} = (\mathcal{F}, \mathbb{D}, \mathbb{F}, \mathcal{K}, \kappa)\}$, where each tuple $\mathcal{M} = (\mathcal{F}, \mathbb{D}, \mathbb{F}, \mathcal{K}, \kappa)$ represents a concrete classification problem.

2.4 Decision Trees

A decision tree $\mathcal{T} = (V, E)$ is a directed acyclic graph having at most one path between every pair of nodes, with $V = \{1, \ldots, \mathfrak{V}\}$ and $E \subseteq V \times V$. Moreover, V is partitioned into a set of non-terminal nodes N and a set of terminal nodes T, i.e. $V = N \cup T$. When referring to the *size* of the decision tree, we will use $|\mathcal{T}|$. $\mathcal{T}$ has a root node, $\mathsf{root}(\mathcal{T}) \in V$ characterized by having no incoming edges, with the convention being that $\mathsf{root}(\mathcal{T}) = 1$. All other nodes have exactly one incoming edge. Each terminal node is associated with an element c of $\mathcal{K}$. Concretely, we assume a function ς mapping terminal nodes to one of the classes, $\varsigma : T \to \mathcal{K}$. For non-terminal nodes $\sigma : N \to 2^V$ maps each node r to the set of child nodes of r. The paper considers only univariate decision trees (i.e. each non-terminal node tests only a single feature). (Possible alternatives include multivariate decision trees (Brodley and Utgoff, 1995), but also non-grounded decision trees (Blockeel and Raedt, 1998); these are beyond the scope of this paper.) As a result, each non-terminal node is assigned a single feature. Specifically, we assume a function ϕ mapping non-terminal nodes to one of the features, $\phi : N \to \mathcal{F}$. As noted earlier, a variable x_i is used to denote values (from $\mathcal{D}_i$) that can be assigned to feature i. A feature i may be associated with multiple nodes connecting the tree's root node to some terminal node. Each edge $(r, s) \in E$, with $\phi(r) = i$, is associated with a literal, representing the values from $\mathcal{D}_i$ for which the edge is declared consistent. Concretely, $\varepsilon : E \to \mathbb{L}$ maps each edge (r, s) with a literal of the form $x_i \in S_l$, with $i = \phi(r)$ and $S_l \subseteq \mathcal{D}_i$. As noted earlier, literals will *always* be of the form $(x_i \in S_l)$, with $S_l \subsetneq \mathcal{D}_i$. (We could consider a larger set of relational operators for representing literals, e.g. $\{\notin, =, \neq, <, \leq, \geq, >\}$ among others. To simplify reasoning about decision trees, only the $\in$ relational operator will be used; the other relational operators can be translated to the $\in$ operator.) The definition of literals assumed in the paper allows an edge to be consistent with multiple values, and so the DTs considered in this paper effectively correspond to multi-edge decision trees (Appuswamy et al., 2011). This more generalized definition of literals allows modeling the DTs learned by well-known tree learning tools (Utgoff et al., 1997). Nevertheless, when denoting that an edge for a node labeled with feature i is consistent only with a single value v_i, we may simply label the edge with v_i or with $x_i = v_i$, for the sake of simplicity. For a given feature, two literals are inconsistent if these represent sets of values that do not intersect.

Example. The literals $(x_1 \in \{0\})$ and $(x_1 \in \{1\})$ are inconsistent, because $\{0\} \cap \{1\} = \emptyset$. In contrast, the literals $(x_1 \in \{1, 3\})$ and $(x_1 \in \{2, 3, 4\})$ are consistent, because $\{1, 3\} \cap \{2, 3, 4\} = \{3\} \neq \emptyset$.

A (complete) path R_k in a DT $\mathcal{T}$ represents a sequence of nodes $\langle r_1, r_2, \ldots, r_l \rangle$, with $r_1, r_2, \ldots, r_l \in V$, that connect the root node to one of the terminal nodes, and such that $(r_j, r_{j+1}) \in E$. Hence, $r_1 = \mathsf{root}(\mathcal{T}) = 1$ and $r_l \in T$. Furthermore, the number of paths in a DT $\mathcal{T}$ is $|T|$, i.e. the number of terminal nodes. Each path is assigned an identifier $R_k \in \mathcal{R}$, where $\mathcal{R}$ denotes the set of paths of $\mathcal{T}$. The sequence of nodes associated with path $R_k \in \mathcal{R}$, is $\mathsf{seq}(R_k) = \langle r_1, r_2, \ldots, r_l \rangle$. For simplicity, and with a mild abuse of notation, we also use R_k to represent the sequence of nodes associated with the identified R_k.

A DT (as any other classifier) computes a (non-constant) classification function $\kappa : \mathbb{F} \to \mathcal{K}$. When studying explanations, we will consider a concrete instance $(\mathbf{v}, c)$, with $c = \kappa(\mathbf{v})$, and distinguish two sets of paths, one corresponding to paths with prediction c and another corresponding to paths with a different prediction. Thus, the set $\mathcal{P} = \{P_1, \ldots, P_{k_1}\} \subseteq \mathcal{R}$ denotes the paths corresponding to a prediction of c. Moreover, the set $\mathcal{Q} = \{Q_1, \ldots, Q_{k_2}\} \subseteq \mathcal{R}$ denotes the paths corresponding to a prediction in $\mathcal{K} \setminus \{c\}$.

Furthermore, given the definition of $\mathcal{P}$ and $\mathcal{Q}$, it is the case that $\mathcal{R} = \mathcal{P} \cup \mathcal{Q}$. When referring to $R_k \in \mathcal{R}$, this may represent a path in $\mathcal{P}$ or a path in $\mathcal{Q}$.

Each path in a DT $\mathcal{T}$ is associated with a (consistent) conjunction of literals, denoting the values assigned to the features so as to reach the terminal node in the path. We will represent the set of literals of some tree path $R_k \in \mathcal{R}$ by $\Lambda(R_k)$. Likewise, the set of features in some tree path $R_k \in \mathcal{R}$ is represented by $\Phi(R_k)$. Moreover, each terminal node associated with a path is represented by $\tau(R_k)$, $\tau : \mathcal{R} \to T$. Each path in the tree *entails* (meaning that it is sufficient for) the prediction associated with the path's terminal node. Let $c \in \mathcal{K}$ denote the prediction associated with path R_k, i.e. $c = \varsigma(\tau(R_k))$. Then, it holds that,

$$\forall(\mathbf{x} \in \mathbb{F}).\left[\bigwedge_{(x_i \in S_l) \in \Lambda(R_k)} (x_i \in S_l)\right] \rightarrow (\kappa(\mathbf{x}) = c) \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_m)$, $c \in \mathcal{K}$, and each $S_l \subseteq \mathcal{D}_i$.

Example. For the example shown in Figure 1, it is the case that,

$$\forall((x_1, x_2, x_3) \in \{0,1\}^3).\,[(x_1 \in \{1\}) \land (x_2 \in \{1\}) \land (x_3 \in \{1\})] \rightarrow (\kappa(c) = \mathbf{1})$$

Furthermore, the classification function associated with this DT can be represented as follows, for $\mathbf{x} = (x_1, x_2, x_3)$:

$$\kappa(\mathbf{x}) = \begin{cases} 1 & \text{iff } [(x_1 \in \{1\}) \land (x_2 \in \{1\}) \land (x_3 \in \{1\})] \lor [(x_1 \in \{1\}) \land (x_2 \in \{0\})] \\ 0 & \text{iff } [(x_1 \in \{0\})] \lor [(x_1 \in \{1\}) \land (x_2 \in \{1\}) \land (x_3 \in \{0\})] \end{cases}$$

As discussed below, one underlying assumption is that any pair of paths in $\mathcal{R}$ must have at least one pair of inconsistent literals. Let (r_j, r_{j+1}) denote some edge in path $R_k \in \mathcal{R}$. Let i be the feature associated with r_j, and let $S_{ij} \subsetneq \mathcal{D}_i$ represent the set of the literal $(x_i \in S_{ij})$, that is associated with the edge (r_j, r_{j+1}). Given $\mathbf{v} \in \mathbb{F}$, the edge (r_j, r_{j+1}) is consistent with $\mathbf{v}$ if $v_i \in S_{ij}$; otherwise the edge is inconsistent. A predicate $\mathsf{consistent}(R_k, \mathbf{v})$ is associated with each path R_k and each point $\mathbf{v}$ in feature space, $\mathsf{consistent} : \mathcal{R} \times \mathbb{F} \to \{0, 1\}$ (or alternatively, $\mathsf{consistent} \subseteq \mathcal{R} \times \mathbb{F}$). The predicate $\mathsf{consistent}$ is defined as follows: given the path $R_k \in \mathcal{R}$ and the point in feature space $\mathbf{v} \in \mathbb{F}$, $\mathsf{consistent}$ takes value 1 if all edges of R_k are consistent given $\mathbf{v}$; otherwise $\mathsf{consistent}$ takes value 0.

The paper makes the following general assumption with respect to decision trees.

Assumption 1. For a DT $\mathcal{T}$, it holds that:

1. For each point $\mathbf{v}$ in feature space, there exists exactly one path consistent with $\mathbf{v}$.

$$\forall(\mathbf{x} \in \mathbb{F}).\,[\exists(R_k \in \mathcal{R}).\mathsf{consistent}(R_k, \mathbf{x}) \land \forall(R_l \in \mathcal{R} \setminus \{R_k\}).\neg\mathsf{consistent}(R_l, \mathbf{x})]$$

   i.e. each point in feature space must be consistent with at least one path, and no point in feature space can be consistent with more than one path.

2. For each tree path $R_k \in \mathcal{R}$, there exists at least one point in feature space that is consistent with the path:

$$\forall(R_k \in \mathcal{R}).\exists(\mathbf{x} \in \mathbb{F}).\mathsf{consistent}(R_k, \mathbf{x})$$

   i.e. there can be no logically inconsistent paths in a DT.

Unless stated otherwise, for the results presented in this paper it is presupposed that Assumption 1 holds[5].

The following additional definitions will be considered for DTs. First, let $\rho : \mathcal{F} \times \mathcal{R} \to 2^{\mathbb{U}}$ be such that $\rho(i, R_k)$ represents the set of values of feature i, taken from $\mathcal{D}_i$, that are consistent with path $R_k \in \mathcal{R}$. Clearly, $\rho(i, R_k)$ is computed by intersecting all the literals testing the value of feature i:

$$\rho(i, R_k) = \bigcap_{(x_i \in S_l) \in \Lambda(\mathrm{R_k})} S_l \tag{2}$$

Observe that $\rho(i, R_k)$ serves to aggregate literals that test the same feature into a single set of values, each of which is consistent with path R_k.

Example. For the example shown in Figure 1, with $Q_2 = \langle 1, 3, 5, 6 \rangle$, $\rho(2, P_2) = \{1\}$ and $\rho(3, P_2) = \{0\}$.

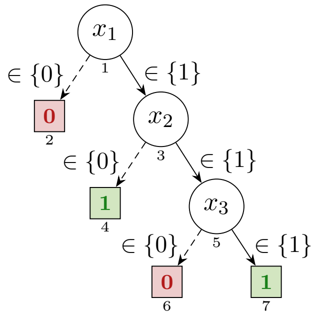
Moreover, let $\chi_I : \mathbb{F} \times \mathcal{R} \to 2^{\mathcal{F}}$ be such that $\chi_I(\mathbf{v}, Q_l)$ represents the subset of features i which takes a value (in $\mathbf{v}$) that is inconsistent with the values of i that are consistent with Q_l. Similarly, let $\chi_P : \mathcal{R} \times \mathcal{R} \to 2^{\mathcal{F}}$ be such that $\chi_P(P_k, Q_l)$ represents the subset of features i for which each value consistent with P_k is inconsistent with the consistent values of i that are consistent with Q_l.

Example. For the example shown in Figure 1, with $\mathbf{v} = (0, 0, 0, 0)$, $P_1 = \langle 1, 3, 4 \rangle$, and $Q_2 = \langle 1, 3, 5, 6 \rangle$, then $\chi_I(\mathbf{v}, P_1) = \{1\}$ and $\chi_P(P_1, Q_2) = \{2\}$.

Running examples. Throughout the paper, a number of decision trees will be used as running examples. These DTs are taken from existing references (Poole and Mackworth, 2017; Hu et al., 2019; Zhou, 2021; Rudin et al., 2021)[6]. For each of the running examples, and with the purpose of simplifying the analysis, original feature domains are mapped to symbolic (numbered) domains. Moreover, all examples of classification problems map to two classes, which we will represent either by $\{\mathbf{0}, \mathbf{1}\}$ or by $\{\mathbf{N}, \mathbf{Y}\}$. Clearly, these modifications do not change in any way the semantics of the original problems.

Example 1. [7] Figure 1 is adapted from (Poole and Mackworth, 2017). The original DT is learned from a given dataset (Poole and Mackworth, 2017) using a variant of ID3 (Quinlan, 1993). As can be observed, $N = \{1, 3, 5\}$ and $T = \{2, 4, 6, 7\}$. Given the instance $(\mathbf{v}, c) = ((1, 1, 1), 1)$, we set $P_1 = \langle 1, 3, 4 \rangle$, $P_2 = \langle 1, 3, 5, 7 \rangle$, $Q_1 = \langle 1, 2 \rangle$, $Q_2 = \langle 1, 3, 5, 6 \rangle$. Moreover, P_2 is the path consistent with the instance. For path P_2, we have $\Lambda(P_2) = \{(x_1 \in \{1\}), (x_2 \in \{1\}), (x_3 \in \{1\})\}$. Clearly, the literals associated with $\mathbf{v} = (1, 1, 1)$, i.e. $x_1 = 1$, $x_2 = 1$ and

5. Assumption 1 outlines what one might consider fairly reasonable conditions regarding the organization of decision trees, and indeed appears to capture the intuitive notion of what a decision tree should represent. However, and perhaps surprisingly, there are recent examples of tree learning tools that can learn DTs with logically inconsistent paths, e.g. (Valdes et al., 2016, Fig. 4) and (Hu et al., 2019, Fig. 6b). Fortunately, it is simple to devise linear-time algorithms, on the size of the DT (and for domains of constant size), for removing logically inconsistent paths. There are also well-known examples of DTs with points in feature space inconsistent with all the DT paths, i.e. DTs with *dead-ends* (Duda et al., 2001, Figure 8.1).
6. The choice of examples taken from published references is deliberate, and aims at illustrating the importance of computing path explanations for decision trees.
7. In this paper, examples that are referenced from the text or by other examples are numbered; the others are not.



(a) Decision tree

Feature	ID	Var.	Domain	Coded Domain
Length	1	x_1	{Long, Short}	{0, 1}
Thread	2	x_2	{New, Follow-Up}	{0, 1}
Author	3	x_3	{Uknown, Known}	{0, 1}

Classes	Coded Representation
{Skips, Reads}	{0, 1}

(b) Mapping of features and classes

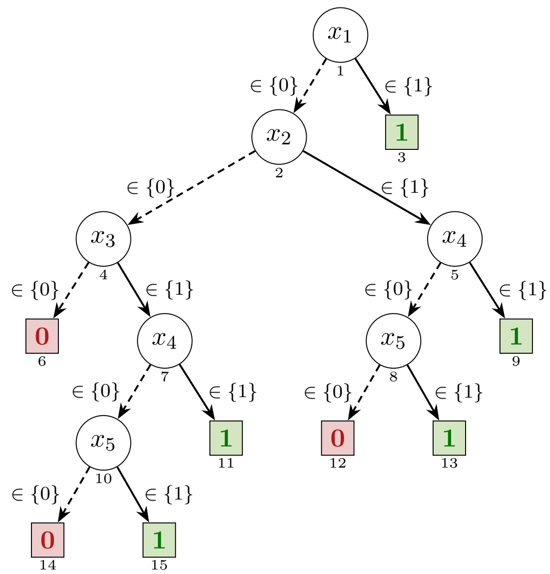
Figure 1. Decision tree, adapted from (Poole and Mackworth, 2017, Ch. 07, Fig. 7.4)

$x_3 = 1$, are consistent with $x_1 \in \{1\}$, $x_2 \in \{1\}$ and $x_3 \in \{1\}$, respectively. Additional results for this DT are summarized in Table 11 (see Page 307). ◁

Example 2. Figure 2 is adapted from (Hu et al., 2019). The original DT was produced with the tool OSDT (optimal sparse decision trees) (Hu et al., 2019). As can be observed, $N = \{1, 2, 4, 5, 7, 8, 10\}$ and $T = \{3, 6, 9, 11, 12, 13, 14, 15\}$. Given the instance $(\mathbf{v}, c) = ((0, 0, 1, 0, 1), 1)$, we set $P_1 = \langle 1, 2, 4, 7, 10, 15\rangle$, $P_2 = \langle 1, 2, 4, 7, 11\rangle$, $P_3 = \langle 1, 2, 5, 8, 13\rangle$, $P_4 = \langle 1, 2, 5, 9\rangle$, $P_5 = \langle 1, 3\rangle$, and then $Q_1 = \langle 1, 2, 4, 6\rangle$, $Q_2 = \langle 1, 2, 4, 7, 10, 14\rangle$, $Q_3 = \langle 1, 2, 5, 8, 12\rangle$. Moreover, P_1 is the path consistent with the instance. As can be observed, $\Lambda(P_1) = \{(x_1 \in \{0\}), (x_2 \in \{0\}), (x_3 \in \{1\}), (x_4 \in \{0\}), (x_5 \in \{1\})\}$, and the literals associated with $\mathbf{v} = (0, 0, 1, 0, 1)$ are $\{x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 1\}$, hence being pairwise consistent. Additional results for this DT are summarized in Table 12 (see Page 307). ◁

Example 3. Figure 3 is adapted from (Zhou, 2021), and illustrates the application of a standard tree learning algorithm, but where the features are categorical (and non-binary). In this case, and for completeness, we show features 5 and 6 (resp. sound and umbilicus), but these are not associated with any node in the DT. As can be observed, $N = \{1, 2, 3, 6, 11\}$, $T = \{4, 5, 7, 8, 9, 10, 12, 13\}$. Given the instance $(\mathbf{v}, c) = ((1, 2, 1, 2), \mathbf{Y})$, we set $P_1 = \langle 1, 2, 5\rangle$, $P_2 = \langle 1, 2, 6, 10\rangle$, $P_3 = \langle 1, 2, 6, 11, 12\rangle$, $P_4 = \{1, 3, 9\}$, and then $Q_1 = \langle 1, 2, 6, 11, 13\rangle$, $Q_2 = \langle 1, 2, 7\rangle$, $Q_3 = \langle 1, 3, 8\rangle$, $Q_4 = \langle 1, 4\rangle$. Moreover, P_3 is the path consistent with the instance. Additional results for this DT are summarized in Table 11 (see Page 307). ◁

Example 4. Figure 4 is adapted from (Rudin et al., 2021), and shows a DT for the recividism dataset (Angwin et al., 2016). Features are categorical or ordinal. (Feature Priors ranges from 0 to 38, and feature Age ranges from 18 to 96. The symbolic names $\text{MxP} = 38$, $\text{MnA} = 18$, $\text{MxA} = 96$ are shown in the DT.) A distinguishing feature of this running example is that one of the features (Priors) is tested more than once along some of the paths. As can be observed, $N = \{1, 3, 4, 6\}$, $T = \{2, 5, 7, 8, 9\}$. Given the instance $(\mathbf{v}, c) = ((2, 20, 0), \mathbf{Y})$, we set $P_1 = \langle 1, 2\rangle$, $P_2 = \langle 1, 3, 4, 6, 8\rangle$, $P_3 = \langle 1, 3, 4, 7\rangle$, and then $Q_1 = \langle 1, 3, 4, 6, 9\rangle$, $Q_2 = \langle 1, 3, 5\rangle$. Moreover, P_2 is the path consistent with the instance. Additional results for this DT are summarized in Table 12 (see Page 307). ◁



(a) Decision tree

Feature in (Hu et al., 2019)	Boolean feature
middle-middle=x	x_1
top-left=x	x_2
bottom-right=x	x_3
bottom-left=x	x_4
top-right=x	x_5

Definitions	
$\mathcal{F}$	$\{1, 2, 3, 4, 5\}$
$\mathcal{D}_1, \ldots, \mathcal{D}_5$	$\{0, 1\}$
$\mathcal{K}$	$\{0, 1\}$

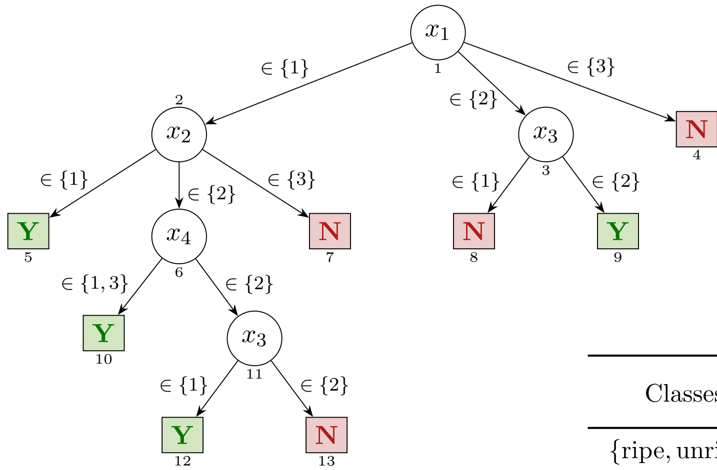
(b) Mapping of features

Figure 2. Decision tree, adapted from (Hu et al., 2019, Figure 5b), for the tic-tac-toe dataset

2.5 Formal Explainability

Formal explanation[8] approaches have been studied in a growing body of research in recent years[9]. Concretely, this paper uses the definition of *abductive explanation* (Ignatiev et al., 2019a) (AXp), which corresponds to a PI-explanation (Shih et al., 2018) in the case of boolean classifiers. AXp's represent prime implicants of the discrete-valued classifier function (which computes the predicted class)[10]. Throughout this paper we will opt to use the acronym AXp to refer to abductive explanations.

8. There is an extensive body of work on non-formal XAI approaches to XAI (Adadi and Berrada, 2018; Montavon et al., 2018; Samek et al., 2019; Guidotti et al., 2019; Samek et al., 2021; Tjoa and Guan, 2021; Holzinger et al., 2022, 2020; Ras et al., 2022).
9. A sample of references on formal explainability includes (Shih et al., 2018; Ignatiev et al., 2019a; Shih et al., 2019; Ignatiev et al., 2019b; Narodytska et al., 2019; Wolf et al., 2019; Audemard et al., 2020; Darwiche, 2020; Darwiche and Hirth, 2020; Shi et al., 2020; Rago et al., 2020; Boumazouza et al., 2020; Ignatiev et al., 2020b; Marques-Silva et al., 2020; Izza et al., 2020; Marques-Silva et al., 2021; Izza and Marques-Silva, 2021; Malfa et al., 2021; Huang et al., 2021b; Audemard et al., 2021; Ignatiev and Marques-Silva, 2021; Asher et al., 2021; Cooper and Marques-Silva, 2021; Boumazouza et al., 2021; Huang et al., 2021a; Rago et al., 2021; Liu and Lorini, 2021; Wäldchen et al., 2021; Darwiche and Marquis, 2021; Blanc et al., 2021; Arenas et al., 2021; Huang et al., 2022; Ignatiev et al., 2022; Marques-Silva and Ignatiev, 2022; Gorji and Rubin, 2022).
10. There exist also standard references with detailed overviews of the uses of prime implicants in the context of boolean functions (Hachtel and Somenzi, 2006; Crama and Hammer, 2011). Generalizations of prime implicants beyond boolean domains have been considered before (Marquis, 1991). Prime implicants have also been referred to as minimum satisfying assignments in first-order logic (FOL) (Dillig et al., 2012), and have been studied in modal and description logics (Bienvenu, 2009).



Classes	Coded Representation
{ripe, unripe}	$\{\mathbf{Y}, \mathbf{N}\}$

(a) Decision tree

(b) Mapping of classes

Feature	ID	Var.	Domain	Coded Domain
Texture	1	x_1	{clear, clightly blurry, blurry}	$\{1, 2, 3\}$
Root	2	x_2	{curly, slightly curly, curly}	$\{1, 2, 3\}$
Surface	3	x_3	{hard, soft}	$\{1, 2\}$
Color	4	x_4	{green, dark, light}	$\{1, 2, 3\}$
Sound	5	x_5	{crisp, muffled, dull}	$\{1, 2, 3\}$
Umbilicus	6	x_6	{flat, slightly hollow, hollow}	$\{1, 2, 3\}$

(c) Mapping of features

Figure 3. Decision tree adapted from (Zhou, 2021, Ch. 04, Fig. 4.3)

Let us consider a given classifier, computing a classification function κ on feature space $\mathbb{F}$, a point $\mathbf{v} \in \mathbb{F}$, with prediction $c = \kappa(\mathbf{v})$, and let $\mathcal{X}$ denote a subset of the set of features $\mathcal{F}$, $\mathcal{X} \subseteq \mathcal{F}$. $\mathcal{X}$ is a weak AXp for the instance $(\mathbf{v}, c)$ if,

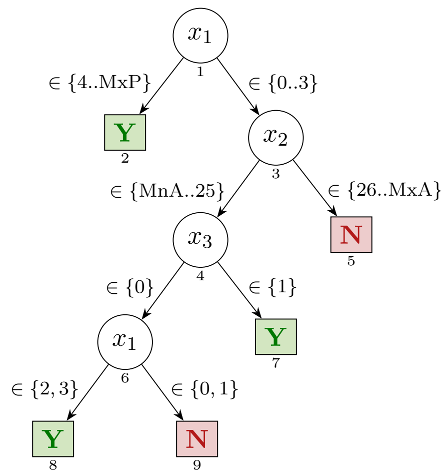
$$\mathsf{WeakAXp}(\mathcal{X}) \quad := \quad \forall (\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i)\right] \rightarrow (\kappa(\mathbf{x}) = c) \tag{3}$$

(We could highlight that $\mathsf{WeakAXp}$ is parameterized on κ, $\mathbf{v}$ and c, but opt not to clutter the notation, and so these dependencies will be left implicit.) Thus, given an instance $(\mathbf{v}, c)$, a (weak) AXp is a set of features which, if fixed to the values dictated by $\mathbf{v}$, then the prediction is guaranteed to be c, independently of the values assigned to the other features. $\mathcal{X}$ is an AXp if, besides being a weak AXp, it is also subset-minimal, i.e.

$$\mathsf{AXp}(\mathcal{X}) \quad := \quad \mathsf{WeakAXp}(\mathcal{X}) \wedge \forall (\mathcal{X}' \subsetneq \mathcal{X}). \neg \mathsf{WeakAXp}(\mathcal{X}') \tag{4}$$

An AXp can be viewed as a possible answer to a "**Why?**" question, i.e. why is the classifier's prediction c?

It should be plain in this work, but also in earlier work, that the representation of AXp's using subsets of features aims at simplicity. The sufficient condition for the prediction is evidently the conjunction of literals associated with the features contained in the AXp.



(a) Decision tree

Classes	Coded Representation
{Yes, No}	$\{\mathbf{Y}, \mathbf{N}\}$

(b) Mapping of classes

Feature	ID	Var.	Domain	Coded Domain
Priors	1	x_1	$\{0, \dots, \text{MxP}\}$	$\{0, \dots, \text{MxP}\}$
Age	2	x_2	$\{\text{MnA}, \dots, \text{MxA}\}$	$\{\text{MnA}, \dots, \text{MxA}\}$
Juvenile crimes	3	x_3	$\{0, 1\}$	$\{0, 1\}$

(c) Mapping of features

Figure 4. Decision tree adapted from (Rudin et al., 2021, Figure 2). According to the dataset, MnA = 18, MxA = 96 and MxP = 38, but the numbers are be left symbolic.

Similarly to the case of AXp's, one can define (weak) contrastive explanations (CXp's) (Miller, 2019; Ignatiev et al., 2020b). $\mathcal{Y} \subseteq \mathcal{F}$ is a weak CXp for the instance $(\mathbf{v}, c)$ if,

$$\mathsf{WeakCXp}(\mathcal{Y}) \quad := \quad \exists(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \notin \mathcal{Y}} (x_i = v_i)\right] \wedge (\kappa(\mathbf{x}) \neq c) \tag{5}$$

(As before, for simplicity we keep the parameterization of $\mathsf{WeakCXp}$ on κ, $\mathbf{v}$ and c implicit.) Thus, given an instance $(\mathbf{v}, c)$, a (weak) CXp is a set of features which, if allowed to take any value from their domain, then there is an assignment to the features that changes the prediction to a class other than c, this while the features not in the explanation are kept to their values (*ceteris paribus*).

Furthermore, a set $\mathcal{Y} \subseteq \mathcal{F}$ is a CXp if, besides being a weak CXp, it is also subset-minimal, i.e.

$$\mathsf{CXp}(\mathcal{Y}) \quad := \quad \mathsf{WeakCXp}(\mathcal{Y}) \wedge \forall(\mathcal{Y}' \subsetneq \mathcal{Y}). \neg \mathsf{WeakCXp}(\mathcal{Y}') \tag{6}$$

A CXp can be viewed as a possible answer to a "**Why Not?**" question, i.e. why isn't the classifier's prediction a class other than c? A different perspective for a contrastive explanation is as the answer to a *How?* question, i.e. how to change the features so as to change the prediction. In recent literature this alternative view has been investigated under the name *actionable recourse* (Ustun et al., 2019; Venkatasubramanian and Alfano, 2020; Karimi et al., 2021, 2020). It should be underlined that whereas AXp's correspond to prime implicants of the boolean function $(\kappa(\mathbf{x}) = c)$ that are consistent with some point $\mathbf{v} \in \mathbb{F}$, CXp are *not* prime implicates of function $(\kappa(\mathbf{x}) = c)$. Nevertheless, the concept of *counterexample* studied in formal explainability (Ignatiev et al., 2019a) corresponds to prime implicates of the function $(\kappa(\mathbf{x}) = c)$ (which are not restricted to be consistent with some specific point $\mathbf{v} \in \mathbb{F}$).

One important observation is that, independently of what κ represents, the WeakAXp and WeakCXp predicates (respectively defined using (3) and (5)) are *monotone*[11]. This means that the tests for minimality (i.e., respectively (4) and (6)) can be simplified to:

$$\mathsf{AXp}(\mathcal{X}) \quad := \quad \mathsf{WeakAXp}(\mathcal{X}) \wedge \forall(t \in \mathcal{X}).\neg\,\mathsf{WeakAXp}(\mathcal{X} \setminus \{t\}) \tag{7}$$

and,

$$\mathsf{CXp}(\mathcal{Y}) \quad := \quad \mathsf{WeakCXp}(\mathcal{Y}) \wedge \forall(t \in \mathcal{Y}).\neg\,\mathsf{WeakCXp}(\mathcal{Y} \setminus \{t\}) \tag{8}$$

Observe that, instead of considering all possible subsets of $\mathcal{X}$ (resp. $\mathcal{Y}$), it suffices to consider the subsets obtained by removing a single element from $\mathcal{X}$ (resp. $\mathcal{Y}$). This observation is at the core of the algorithms proposed in recent years for computing AXp's and CXp's of a growing range of families of classifiers (Ignatiev et al., 2019a,b; Narodytska et al., 2019; Marques-Silva et al., 2020; Izza et al., 2020; Marques-Silva et al., 2021; Izza and Marques-Silva, 2021; Malfa et al., 2021; Huang et al., 2021b; Ignatiev and Marques-Silva, 2021; Huang et al., 2021a).

Example 5. For the DT in Figure 1, consider the instance $((1, 1, 1), 1)$ (i.e. if Length is Short, and Thread is Follow-Up, and Author is Known, then predict Reads). The paths in $\mathcal{P}$ are: $\mathcal{P} = \{P_1, P_2\}$, with $P_1 = \langle 1, 3, 4\rangle$ and $P_2 = \langle 1, 3, 5, 7\rangle$. The paths in $\mathcal{Q}$ are: $\mathcal{Q} = \{Q_1, Q_2\}$, with $Q_1 = \langle 1, 2\rangle$ and $Q_2 = \langle 1, 3, 5, 6\rangle$. Path P_2 is consistent with the instance; all other paths are inconsistent with the instance. The features associated with P_2 are $\Phi(P_2) = \{1, 2, 3\}$, and the path literals associated with path P_2 are $\Lambda(P_2) = \{(x_1 \in \{1\}), (x_2 \in \{1\}), (x_3 \in \{1\})\}$. Nevertheless, from Figure 1, it is clear that $\mathcal{X} = \{1, 3\}$ is a weak AXp. Indeed, if feature 2 (feature variable x_2) is allowed to take any value in its domain, then the prediction remains unchanged. Hence, it is the case that, with $\mathbf{x} = (x_1, x_2, x_3)$, $\forall(\mathbf{x} \in \{0, 1\}^3).[(x_1) \wedge (x_3)] \rightarrow \kappa(\mathbf{x})$. Furthermore, $\mathcal{X}$ is minimal, since dropping either 1 or 3 from $\mathcal{X}$ will cause the weak AXp condition to fail.

CXp's can be computed in a similar way. One can also observe that if either x_1 or x_3 are allowed to take any value from their domains, then there is an assignment that causes the prediction to change. Thus, $\mathcal{Y}_1 = \{1\}$ or $\mathcal{Y}_2 = \{3\}$ are CXp's of the given instance. ◁

Given the definitions of AXp and CXp, and building on Reiter's seminal work (Reiter, 1987), recent work (Ignatiev et al., 2020b) proved the following duality between minimal hitting sets[12]:

11. Clearly, from the definition of WeakAXp (resp. WeakCXp), if WeakAXp($\mathcal{Z}$) (resp. WeakCXp($\mathcal{Z}$)) holds, then WeakAXp($\mathcal{Z}'$) (resp. WeakCXp($\mathcal{Z}'$)) also holds for any superset $\mathcal{Z}'$ of $\mathcal{Z}$. If WeakAXp($\mathcal{Z}$) (resp. WeakCXp($\mathcal{Z}$)) does not hold, then WeakAXp($\mathcal{Z}'$) (resp. WeakCXp($\mathcal{Z}'$)) also does not hold for any superset $\mathcal{Z}'$ of $\mathcal{Z}$.
12. Recall that a set $\mathcal{H}$ is a *hitting set* of a set of sets $\mathcal{S} = \{S_1, \ldots, S_k\}$ if $\mathcal{H} \cap S_i \neq \emptyset$ for $i = 1, \ldots, k$. $\mathcal{H}$ is a minimal hitting set of $\mathcal{S}$, if $\mathcal{H}$ is a hitting set of $\mathcal{S}$, and there is no proper subset of $\mathcal{H}$ that is also a hitting set of $\mathcal{S}$.

Proposition 1 (Minimal hitting-set duality between AXp's and CXp's)**.** AXp's are minimal hitting sets (MHSes) of CXp's and vice-versa.

We refer to Proposition 1 as MHS duality between AXp's and CXp's. The previous result has been used in more recent papers for enabling the enumeration of explanations (Marques-Silva et al., 2021; Ignatiev and Marques-Silva, 2021; Huang et al., 2021b). Furthermore, a consequence of Proposition 1 is the following result:

Lemma 1. Given a classifier function $\kappa : \mathbb{F} \to \mathcal{K}$, defined on a set of features $\mathcal{F}$, a feature $i \in \mathcal{F}$ is included in some AXp iff i is included in some CXp.

Another minimal hitting-set duality result, different from Proposition 1, was investigated in earlier work (Ignatiev et al., 2019b), and relates *global* AXp's (i.e. not restricted to be consistent with a specific point $\mathbf{v} \in \mathbb{F}$) and counterexamples (see Page 275).

Given the above, the universe of *explanation problems* is defined by $\mathbb{E}_I = \{\mathcal{E} \,|\, \mathcal{E} = (\mathcal{M}, (\mathbf{v}, c)), \mathcal{M} \in \mathbb{M}, \mathbf{v} \in \mathbb{F}, c \in \mathcal{K}, c = \kappa(\mathbf{v})\}$. As a result, a tuple $(\mathcal{M}, (\mathbf{v}, c))$ will allow us to unambiguously represent the classification problem $\mathcal{M}$ for which we will be computing AXp's and CXp's given the instance $(\mathbf{v}, c)$.

2.6 Summary of Notation

The notation used throughout the paper is summarized in Table 1 (see Page 277). (We should note that some of the notation introduced in this paper has also been used in a number of recent works [13].)

3. Duality of Explanations & Path-Based Explanations

This section builds on recent work on duality of explanations (Ignatiev et al., 2020b) (see Section 2.5), and makes the following contributions:

1. Explanations are generalized to explanation functions and conditions are outlined for minimal hitting-set (MHS) duality of explanations to hold in this more general setting.
2. Explanations are shown to respect a nesting property, with MHS duality holding for nested explanations.

Furthermore, the section highlights how the results above can be used for relating the computation of explanations of a DT with specific tree paths instead of being instance-specific.

3.1 Generalized Explanations & Duality

Explanation functions. Besides prime implicants of discrete-valued functions, we can envision a generalized explanation function $\xi : \mathbb{F} \to \{0, 1\}$, and redefine both weak AXp's and weak CXp's, assuming such a generalized explanation function[14]. However, we impose that ξ be parameterized on a selected subset $\mathcal{Z}$ of the features, and also on other parameters

13. See for example (Ignatiev et al., 2019a; Narodytska et al., 2019; Ignatiev et al., 2019b; Ignatiev, 2020; Marques-Silva et al., 2020; Ignatiev et al., 2020a; Izza et al., 2020; Marques-Silva et al., 2021; Izza and Marques-Silva, 2021; Ignatiev and Marques-Silva, 2021; Cooper and Marques-Silva, 2021; Huang et al., 2021b; Marques-Silva and Ignatiev, 2022; Huang et al., 2022; Ignatiev et al., 2022).
14. Explanation functions have been studied in earlier work on formal explainability (Wolf et al., 2019).

Symbol	Definition	Meaning
$\mathcal{F}$	$\{1,\ldots,m\}$	Set of features
$\mathcal{D}_i$	–	Domain of feature i
$\mathbb{D}$	$\mathbb{D}=(\mathcal{D}_1,\ldots,\mathcal{D}_m)$	Range of domains, $\mathcal{D}_i=\mathbb{D}(i)$
$\mathbb{U}$	$\mathbb{U}=\cup_{i\in\mathcal{F}}\mathcal{D}_i$	Union of domains
$\mathbb{F}$	$\mathcal{D}_1\times\mathcal{D}_2\times\ldots\times\mathcal{D}_m$	Feature space
x_i	$x_i\in\mathcal{D}_i$	Variable associated with feature i
$\mathcal{L}$	$\mathcal{L}=(x_i\in S_l)$	Literal, with $S_l\subsetneq\mathcal{D}_i$
$\mathbb{L}$	$\mathbb{L}=\{x_i\in S_l\}$	Sets of literals, $i\in\mathcal{F}\wedge S_l\subsetneq\mathcal{D}_i$
$\mathcal{K}$	$\{c_1,\ldots,c_K\}$	Set of classes
κ	$\kappa:\mathbb{F}\to\mathcal{K}$	Classification function
$\mathcal{I}$	$\mathcal{I}=(\mathbf{v},c)$	Instance, with $\mathbf{v}\in\mathbb{F},c\in\mathcal{K}$
$\mathbb{M}$	$\mathbb{M}=\{(\mathcal{F},\mathbb{D},\mathbb{F},\mathcal{K},\kappa)\}$	Universe of classification problems
$\mathbb{E}_I$	$\mathbb{E}_I=\{(\mathcal{M},(\mathbf{v},c))\}$	Explanation problems, $\mathcal{M}\in\mathbb{M},\mathbf{v}\in\mathbb{F},c\in\mathcal{K}$
ξ	$\xi:\mathbb{F}\to\{0,1\}$	Explanation function, $\xi(\mathbf{x};\mathcal{Z},\ldots)$, $\mathcal{Z}\subseteq\mathcal{F}$
$\mathbb{E}_S$	$\mathbb{E}_S=\{(\mathcal{M},(\xi,\mathcal{Z},c))\}$	XP problems, $\mathcal{M}\in\mathbb{M},\mathcal{Z}\subseteq\mathbb{F},c\in\mathcal{K}$, ξ: XP function
$\mathbb{E}_P$	$\mathbb{E}_P=\{(\mathcal{M},R_k)\}$	Path-related XP problems, $\mathcal{M}\in\mathbb{M},R_k\in\mathcal{R}$
$\mathcal{T}$	$\mathcal{T}=(V,E)$	Decision tree, with nodes V and edges E
V	$N\cup T$	Set of nodes in DT $\mathcal{T}$
T	–	Terminal nodes
ς	$\varsigma:T\to\mathcal{K}$	Class associated with each terminal node
N	–	Non-terminal nodes
ϕ	$\phi:N\to\mathcal{F}$	Feature associated with each non-terminal node
σ	$\sigma:N\to 2^V$	Child nodes of non-terminal node
ε	$\varepsilon:E\to\mathbb{L}$	Lit. $x_i\in S_l$ associated with edge (r,s), $i=\phi(r)$
$\mathcal{R}$	–	Paths in DT $\mathcal{T}$
R_k	$R_k=\langle r_1,\ldots,r_l\rangle$	Path in DT $\mathcal{T}$, with tree nodes $r_1,\ldots,r_l$
seq	–	Sequence of tree nodes in $R_k\in\mathcal{R}$
τ	$\tau:\mathcal{R}\to T$	Terminal node associated with path $R_k\in\mathcal{R}$
Φ	$\Phi:\mathcal{R}\to 2^{\mathcal{F}}$	Features associated with path R_k in $\mathcal{R}$
Λ	$\Lambda:\mathcal{R}\to 2^{\mathbb{L}}$	Literals associated with path R_k in $\mathcal{R}$
ρ	$\rho:\mathcal{F}\times\mathcal{R}\to 2^{\mathbb{U}}$	Values of feature i consistent with $R_k\in\mathcal{R}$
χ_I	$\chi_I:\mathbb{F}\times\mathcal{R}\to 2^{\mathcal{F}}$	Features that are inconsistent between instance and path
χ_P	$\chi_P:\mathcal{R}\times\mathcal{R}\to 2^{\mathcal{F}}$	Features that are inconsistent between two paths
$\mathcal{H}$	–	hard constraints/clauses
$\mathcal{B}$	–	soft constraints/clauses

Table 1: Summary of the notation used throughout the paper

which we may leave undefined, or instead opt to include. This parameterization will be represented by: $\xi(\mathbf{x};\mathcal{Z},\ldots)$. For example, if ξ represents a prime implicant that is sufficient for the prediction, the parameterization (as discussed in Section 2.5) is the restriction of the conjunction of literals to those features in $\mathcal{Z}$, where the literals are of the form $x_i = v_i$ (i.e. the parameterization on $\mathcal{Z}$ serves to select the coordinate values of $\mathbf{v}$ associated with the features in $\mathcal{Z}$). However, it is possible to consider explanation functions that involve other types of literals. Concretely, we will allow explanation functions to involve literals of the form $(x_i \in S_l)$.

Earlier work on formal explainability has most often considered as the underlying explanation function the prime implicants of discrete-valued functions, defined on arbitrary feature spaces. Hence, given an instance $(\mathbf{v}, c)$, a possible definition of explanation function is:

$$\xi(\mathbf{x}; \mathcal{Z}, \mathbf{v}) = \bigwedge_{i \in \mathcal{Z}} (x_i = v_i) \tag{9}$$

A clear limitation of using such prime implicants as the explanation function is that we are equating each feature with a *single* value from its domain. For categorical features this is not a major issue, but for ordinal features it can be too restrictive.

In the case of DT paths, a viable explanation function is:

$$\xi(\mathbf{x}; \mathcal{Z}, R_k, \Lambda(R_k)) = \bigwedge_{i \in \mathcal{Z}, (x_i \in S_l) \in \Lambda(R_k)} (x_i \in S_l) \tag{10}$$

(For simplicity, the parameterization on R_k could be ignored, since R_k is in fact a constant when computing explanations that relate with itself.)

Example 6. For the running example in Figure 4, consider the instance $((2, 25, 0), \mathbf{Y})$, consistent with path $P_1 = \langle 1, 3, 4, 6, 8 \rangle$. It is possible to conclude that a weak AXp is $\{1, 2\}$. Observe that there are three features with literals in the path, i.e. $\{1, 2, 3\} = \mathcal{F}$, and that changing the value of feature 3 does not change the prediction; hence a weak AXp is $\{1, 2\}$. Using the first explanation function above (see (9)), one could claim that $(x_1 = 2) \wedge (x_2 = 25)$ suffices for the prediction. However, using the second explanation function above (see (10)), one would be able to claim instead that $(x_1 \in \{2, 3\}) \wedge (x_2 \in \{\text{MnA}..25\})$ suffices for the prediction. Clearly, the second explanation function is markedly more informative regarding which values suffice for the prediction. (Another extension that this paper does not investigate, is that $(x_1 \in \{2..\text{MxP}\}) \wedge (x_2 \in \{\text{MnA}..25\})$ would also suffice for the prediction; this is the subject of future work.)
The two explanation functions above exhibit important properties, including duality relationships; this will be discussed later in this section. Nevertheless, other explanation functions could be envisioned. ◁

Generalizing AXp's and CXp's. Explanation functions serve to generalize weak AXp's and CXp's, as follows:

Definition 1 (WeakAXp and WeakCXp)**.** Given a classification problem $\mathcal{M} = (\mathcal{F}, \mathbb{D}, \mathbb{F}, \mathcal{K}, \kappa)$, an explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$, and an explanation function ξ, $\mathcal{Z} \subseteq \mathcal{F}$ is a weak abductive explanation if,

$$\mathsf{WeakAXp}(\mathcal{Z}) \quad := \quad \forall (\mathbf{x} \in \mathbb{F}). \xi(\mathbf{x}; \mathcal{Z}, \ldots) \rightarrow (\kappa(\mathbf{x}) = c) \tag{11}$$

$\mathcal{Z}$ is a weak contrastive explanation if,

$$\mathsf{WeakCXp}(\mathcal{Z}) \quad := \quad \exists (\mathbf{x} \in \mathbb{F}). \xi(\mathbf{x}; \mathcal{F} \setminus \mathcal{Z}, \ldots) \wedge (\kappa(\mathbf{x}) \neq c) \tag{12}$$

For simplicity, the parameterization of $\mathsf{WeakAXp}$ and $\mathsf{WeakCXp}$, on ξ, $\mathcal{M}$ and $\mathcal{E}$, $\mathbf{v}$, etc. is left implicit; this will be clear from the context.

A consequence of the definition of $\mathsf{WeakAXp}$ and $\mathsf{WeakCXp}$ is that we have the following immediate result:

Proposition 2. For any $\mathcal{Z} \subseteq \mathcal{F}$, it is the case that,

$$\mathsf{WeakAXp}(\mathcal{Z}) \leftrightarrow \neg\,\mathsf{WeakCXp}(\mathcal{F} \setminus \mathcal{Z})$$

Proof. $\mathsf{WeakAXp}(\mathcal{Z})$ states that,

$$\forall(\mathbf{x} \in \mathbb{F}).\xi(\mathbf{x}; \mathcal{Z}, \ldots) \rightarrow (\kappa(\mathbf{x}) = c)$$

whereas, $\mathsf{WeakCXp}(\mathcal{F} \setminus \mathcal{Z})$ states that,

$$\exists(\mathbf{x} \in \mathbb{F}).\xi(\mathbf{x}; \mathcal{Z}, \ldots) \land (\kappa(\mathbf{x}) \neq c)$$

which is the logical negation of $\mathsf{WeakAXp}(\mathcal{Z})$. Thus, if $\mathsf{WeakAXp}(\mathcal{Z})$ is true, then it must be the case that $\mathsf{WeakCXp}(\mathcal{F} \setminus \mathcal{Z})$ is false, and vice-versa. □

We will also need to consider sets of explanations and subset-minimal explanations. Hence, the following definitions are used:

Definition 2 ($\mathbb{S}_{\text{waxp}}$, $\mathbb{S}_{\text{wcxp}}$, $\mathbb{A}$, $\mathbb{C}$)**.** Given $\mathcal{M}$ and $\mathcal{E}$, the following sets of sets are defined:

$$\begin{aligned} \mathbb{S}_{\text{waxp}} &= \{\mathcal{Z} \in \mathcal{F} \,|\, \mathsf{WeakAXp}(\mathcal{Z})\} \\ \mathbb{S}_{\text{wcxp}} &= \{\mathcal{Z} \in \mathcal{F} \,|\, \mathsf{WeakCXp}(\mathcal{Z})\} \end{aligned} \tag{13}$$

The set $\mathbb{A}$ of the subset-minimal sets of $\mathbb{S}_{\text{waxp}}$ represents the AXp's, i.e.

$$\mathbb{A} = \{\mathcal{Z} \in \mathbb{S}_{\text{waxp}} \,|\, \forall(\mathcal{Z}' \subsetneq \mathcal{Z}).\neg\,\mathsf{WeakAXp}(\mathcal{Z}')\} \tag{14}$$

The set $\mathbb{C}$ of the subset-minimal sets of $\mathbb{S}_{\text{wcxp}}$ represents the CXp's, i.e.

$$\mathbb{C} = \{\mathcal{Z} \in \mathbb{S}_{\text{wcxp}} \,|\, \forall(\mathcal{Z}' \subsetneq \mathcal{Z}).\neg\,\mathsf{WeakCXp}(\mathcal{Z}')\} \tag{15}$$

Furthermore, we are especially interested in explanation functions that guarantee the monotonicity of $\mathsf{WeakAXp}$ and $\mathsf{WeakCXp}$. (As noted in Section 2.5, the monotonicity of these predicates enables devising more efficient algorithms for computing AXp's and CXp's.) Taking into consideration that, from (11) and (12), $\mathsf{WeakAXp}$ and $\mathsf{WeakCXp}$ (and so also AXp and CXp) are defined in terms of ξ, then we have the following definition:

Definition 3. An explanation function ξ is *monotone-inducing* if, given ξ:

1. $\mathsf{WeakAXp}(\emptyset) = 0$ and $\mathsf{WeakCXp}(\emptyset) = 0$;
2. $\mathsf{WeakAXp}(\mathcal{F}) = 1$ and $\mathsf{WeakCXp}(\mathcal{F}) = 1$;
3. Moreover, it holds that, for $\mathcal{A}_0 \subseteq \mathcal{F}$,

$$\begin{aligned} &\mathsf{WeakAXp}(\mathcal{A}_0) \rightarrow \forall(\mathcal{A}_1 \supseteq \mathcal{A}_0).\,\mathsf{WeakAXp}(\mathcal{A}_1) \\ &\mathsf{WeakCXp}(\mathcal{A}_0) \rightarrow \forall(\mathcal{A}_1 \supseteq \mathcal{A}_0).\,\mathsf{WeakCXp}(\mathcal{A}_1) \end{aligned}$$

(i.e. if $\mathcal{A}_0$ is a weak AXp (resp. weak CXp) then any of its supersets (resp. subsets) is also a weak AXp (resp. weak CXp).)

Example 7. The two explanation functions described in Example 6 are monotone-inducing. The fact that the explanation function associated with path literals is monotone-inducing will be pivotal for computing path explanations. ◁

Given the above, we can now state the main result of this section.

Proposition 3. Given $\mathcal{M}$ and $\mathcal{E}$, ξ is a monotone-inducing explanation function iff each element of $\mathbb{A}$ is an MHS of the elements of $\mathbb{C}$, and vice-versa. (This is to say that the AXp's of $\mathcal{E}$ are MHSes of the CXp's of $\mathcal{E}$ and vice-versa.)

Proof. The proof is split into cases:

i) If ξ is a monotone-inducing explanation function, then AXp's are MHSes of CXp's and vice-versa.
Let $\mathcal{A} \in \mathbb{A}$ be an AXp. Thus, $\mathcal{A}$ is a subset-minimal set such that (11) holds. We claim that $\mathcal{A}$ must hit every CXp $\mathcal{C}$ of $\mathbb{C}$. For the sake of contradiction, let us assume that this was not the case. Then, there would exist some $\mathcal{C} \in \mathbb{C}$, not hit by $\mathcal{A}$. As a result, $\mathcal{F} \setminus \mathcal{A}$ would necessarily contain $\mathcal{C}$. Since $\mathcal{C}$ is a CXp, then (12) would be satisfied. But this is impossible due to Proposition 2; a contradiction.
What remains to show is that the hitting set $\mathcal{A}$ is subset-minimal. Suppose it was not minimal. Then, we could create a minimal hitting set $\mathcal{A}' \subsetneq \mathcal{A}$, since $\mathcal{A}'$ would hit all the CXp's in $\mathbb{C}$, then (12) could be falsified by $\mathcal{F} \setminus \mathcal{A}'$. However, by Proposition 2, then $\mathcal{A}'$ would satisfy (11), and so $\mathcal{A}$ would not be minimal; a contradiction.
A similar argument can be used to prove that each $\mathcal{C} \in \mathbb{C}$ must hit every $\mathcal{A} \in \mathbb{A}$.

ii) If AXp's are MHSes of CXp's and vice-versa, then ξ is a monotone-inducing explanation function.
This follows from the definition of monotone-inducing explanation function. □

The result above can be related not only with recent results on the duality of explanations (Ignatiev et al., 2019b, 2020b), but also with other well-known results on duality in different areas (Reiter, 1987; Birnbaum and Lozinskii, 2003; Slaney, 2014). Finally, $\mathbb{E}_S$ will be used to denote the set of explanation problems given a classification problem $\mathcal{M}$, a subset $\mathcal{Z}$ of the features, and an explanation function ξ, parameterized on $\mathcal{Z}$ and other parameters: $\mathbb{E}_S = \{\mathcal{E} \,|\, \mathcal{E} = (\mathcal{M}, (\xi, \mathcal{Z}, c)), \mathcal{M} \in \mathbb{M}, \mathcal{Z} \subseteq \mathcal{F}, \xi \text{ is an explanation function}\}$.

3.2 Restricted Duality

This section investigates a restricted form of duality that results from AXp's exhibiting what can be viewed as a property of *nesting*. We consider an explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$ and a monotone-inducing explanation function ξ. Moreover, we let $\mathcal{Z} \subseteq \mathcal{F}$, with $\mathcal{Z} = \{i_1, i_2, \ldots, i_M\}$, represent a weak AXp, i.e.

$$\forall(\mathbf{x} \in \mathbb{F}). (\xi(\mathbf{x}; \mathcal{Z}, \ldots)) \rightarrow (\kappa(\mathbf{x}) = c) \tag{16}$$

Furthermore, let us define $\mathbb{F}_{\mathcal{Z}} = \mathcal{D}_{i_1} \times \mathcal{D}_{i_2} \times \cdots \times \mathcal{D}_{i_M}$, $\mathbb{D}_{\mathcal{Z}} = (\mathcal{D}_{i_1}, \mathcal{D}_{i_2}, \ldots, \mathcal{D}_{i_M})$, and let $\iota : Z = \{1, \ldots, M\} \to \mathcal{Z} = \{i_1, \ldots, i_M\}$ be a bijective function that maps coordinates 1 to M into the actual features' indices in $\mathcal{Z}$, i.e. $\iota(r) = i_r$, $r = 1, \ldots, M$ [15]. In addition,

15. With a slight abuse of notation, we will use $\iota^{-1}(\mathcal{Z})$ to denote the set $Z = \{1, \ldots, M\}$. We will also use $\iota(X) = \mathcal{Z} \subseteq \mathcal{Z}$ and $\iota^{-1}(\mathcal{X}) = X \subseteq Z$ to represent the mappings of sets of features.

we introduce the predicate $\text{prj}_{\mathcal{Z}}$, such that $\text{prj}_{\mathcal{Z}}(\mathbf{x}, \mathbf{y})$ holds when $\mathbf{y}$ is the projection of $\mathbf{x}$ on the coordinates specified by $\mathcal{Z}$, i.e. $y_j = x_{\iota(j)}$ for all $j \in \mathcal{Z}$. (Observe that $\text{prj}_{\mathcal{Z}}$ is effectively parameterized on ι, but this is left implicit.) In the concrete case of $\mathbf{v}$, we define $\mathbf{u} \in \mathbb{F}_{\mathcal{Z}}$, such that $\text{prj}_{\mathcal{Z}}(\mathbf{v}, \mathbf{u})$ is true. Moreover, define a binary classifier $\kappa_{\mathcal{Z}} : \mathbb{F}_{\mathcal{Z}} \to \{0, 1\}$, as follows:

$$\kappa_{\mathcal{Z}}(\mathbf{y}) = \begin{cases} 1, & \text{if } \forall(\mathbf{x} \in \mathbb{F}).\,[\text{prj}_{\mathcal{Z}}(\mathbf{x}, \mathbf{y}) \land \xi(\mathbf{x}; \mathcal{Z}, \ldots)] \rightarrow (\kappa(\mathbf{x}) = c) \\ 0, & \text{if } \exists(\mathbf{x} \in \mathbb{F}).\,[\text{prj}_{\mathcal{Z}}(\mathbf{x}, \mathbf{y}) \land \xi(\mathbf{x}; \mathcal{Z}, \ldots)] \land (\kappa(\mathbf{x}) \neq c) \end{cases} \tag{17}$$

Observe that, by definition of $\kappa_{\mathcal{Z}}$, one can conclude that $\kappa_{\mathcal{Z}}$ is independent of the features in $\mathcal{F} \setminus \mathcal{Z}$. Also note that $\kappa_{\mathcal{Z}}(\mathbf{y}) = 1$ only if $\kappa(\mathbf{x}) = c$ for all points $\mathbf{x} \in \mathbb{F}$ which project into $\mathbf{y} \in \mathbb{F}_{\mathcal{Z}}$.

Given the definition of $\kappa_{\mathcal{Z}}$, we can now define both a *restricted* classification problem $\mathcal{M}_{\mathcal{Z}} = (Z, \mathbb{D}_{\mathcal{Z}}, \mathbb{F}_{\mathcal{Z}}, \{0, 1\}, \kappa_{\mathcal{Z}})$, and associated explanation problem $\mathcal{E}_{\mathcal{Z}} = (\mathcal{M}_{\mathcal{Z}}, (\mathbf{u}, 1))$. Clearly, for the explanation problem $\mathcal{E}_{\mathcal{Z}}$, it must be the case that AXp's are the MHSes of the CXp's and vice-versa (Ignatiev et al., 2020b). Furthermore, it is plain that the AXp's and CXp's of $\mathcal{E}_{\mathcal{Z}}$ are subsets of Z.

Example 8. Consider the DT from Figure 2, and path $P_4 = \langle 1, 2, 5, 9 \rangle$, with $\varsigma(\tau(9)) = \mathbf{1}$. Let $\mathbf{v} = (0, 1, 0, 1)$, consistent with P_4. It is simple to conclude that $\mathcal{Z} = \{1, 2, 4\}$ is a weak AXp of $(\mathbf{v}, c)$. Moreover, we let $\iota(1) = 1, \iota(2) = 2, \iota(3) = 4$, with $Z = \{1, 2, 3\}$. Given the above, we can define $\kappa_{\mathcal{Z}}$.

y_1	y_2	y_3	$\kappa_{\mathcal{Z}}$
0	0	0,1	**0**
0	1	0	**0**
0	1	1	**1**
1	0,1	0,1	**1**

(Observe that the use of ',' in the rows serves solely to collapse multiple rows into one.) We can now compute the AXp's/CXp's for the explanation problem $(\mathcal{M}_{\mathcal{Z}}, (\mathbf{u}, \mathbf{1}))$, with $\mathbf{u} = (0, 1, 1)$, since $\text{prj}_{\mathcal{Z}}((0, 1, 0, 1), (0, 1, 1))$ holds.
Given the explanation problem $\mathcal{E}_{\mathcal{Z}}$, and from the definition of $\kappa_{\mathcal{Z}}$ in the table above, an AXp is $\{2, 3\}$. Clearly, the CXp's will be $\{2\}$ and $\{3\}$. We can now map the AXp's and CXp's of $\mathcal{E}_{\mathcal{Z}}$ to the features of $\mathcal{F}$. For the AXp, we get a set of features $\{2, 4\}$, which we will later argue that it is also an AXp of $\mathcal{E}$. For the CXp's, we get $\{2\}$ and $\{4\}$, which we will shortly argue that are subsets of CXp's of $\mathcal{E}$. Further, we will later argue that these sets of features relate with abductive and contrastive explanations associated with path P_4. ◁

Furthermore, given the definitions above, the following additional results also hold. Given a set $\mathcal{Z}$, and the resulting restricted binary classifier $\kappa_{\mathcal{Z}}$, there is a one to one mapping of AXp's between those of $\mathcal{E}_{\mathcal{Z}}$ and those of $\mathcal{E}$; however, each CXp of $\mathcal{E}_{\mathcal{Z}}$ is a subset of some CXp of $\mathcal{E}$.

Proposition 4. $X \subseteq Z = \iota^{-1}(\mathcal{Z})$ is an AXp of $\mathcal{E}_{\mathcal{Z}}$ iff $\mathcal{X} = \iota(X) \subseteq \mathcal{Z}$ is an AXp of $\mathcal{E}$.

Proof. Let $\mathbf{y}$ be a point in $\mathbb{F}_{\mathcal{Z}}$ consistent with the features in X, and so exhibiting prediction 1. Then, by definition of $\kappa_{\mathcal{Z}}$, it is the case that the prediction of κ for any $\mathbf{x}$, such that

$\text{prj}(\mathbf{x}, \mathbf{y})$ holds, must be c.
Similarly, let $\mathbf{x}$ be a point in $\mathbb{F}$ consistent with the features in $\mathcal{X}$, and so exhibiting prediction c. Then, by definition of $\kappa_{\mathcal{Z}}$, it is the case that the prediction of $\kappa_{\mathcal{Z}}$ for $\mathbf{y}$, such that $\text{prj}(\mathbf{x}, \mathbf{y})$ holds, must be 1.
Since by hypothesis, X is subset-minimal, then $\mathcal{X} = \iota(X)$ is subset-minimal. □

Proposition 5. Each CXp $Y \subseteq Z = \iota^{-1}(\mathcal{Z})$ of $\mathcal{E}_{\mathcal{Z}}$ is such that $\mathcal{Y} = \iota(Y) \subseteq \mathcal{Z}$ is a subset of some CXp of $\mathcal{E}$.

Proof. By definition, a CXp $Y \subseteq Z$ of $\mathcal{E}_{\mathcal{Z}}$ is a subset-minimal set of features in Z which, if allowed to take any value from their domains, suffice to change the prediction. However, for $\mathcal{E}$ and given $\mathbf{v}$, the features in $\mathcal{F} \setminus \mathcal{Z}$ take specific fixed values, dictated by $\mathbf{v}$. Hence, some of these features may be required to change their values for the prediction of κ to change from c to some of the class in $\mathcal{K} \setminus \{c\}$. This follows from the definition of $\kappa_{\mathcal{Z}}$ in (17). As a result, it may be necessary to add to $\mathcal{Y} = \iota(Y)$ additional features from $\mathcal{F} \setminus \mathcal{Z}$ so that the prediction changes. A minimal such set is a CXp of $\mathcal{E}$ and it represents a superset of $\mathcal{Y}$. Furthermore, no feature in Y (and so in the resulting $\mathcal{Y}$) is redundant, since Y is by definition a minimal set, even if the features not in $\mathcal{Z}$ are allowed to change their value. □

Furthermore, one additional result that is a consequence of the previous results is that the relationships between AXp's and CXp's can be stated in terms of AXp's and CXp's that are restricted to some *seed* set.

Definition 4 (Set-restricted AXp's/CXp's)**.** Let $\mathcal{E}$ be an explanation problem and let $\mathcal{Z} \in \mathcal{F}$ be a weak AXp of $\mathcal{E}$. The $\mathcal{Z}$-set-restricted AXp's are the AXp's of $\mathcal{E}_{\mathcal{Z}}$ mapped by ι to the indices of features in $\mathcal{F}$, and it is represented by $\mathbb{A}_{\mathcal{Z}}$. The $\mathcal{Z}$-set-restricted CXp's are the CXp's of $\mathcal{E}_{\mathcal{Z}}$ mapped by ι to the indices of features in $\mathcal{F}$, and it is represented by $\mathbb{C}_{\mathcal{Z}}$.

Given the definition of set-restricted AXp's/CXp's, we have the following result:

Proposition 6. Let $\mathcal{E}$ be an explanation problem, and let $\mathcal{Z}$ be a weak AXp for $\mathcal{E}$. Then, $\mathbb{A}_{\mathcal{Z}} \subseteq \mathbb{A}$, i.e. each $\mathcal{Z}$-set-restricted AXp is also an AXp. Furthermore, for each $\mathcal{W} \in \mathbb{C}_{\mathcal{Z}}$, there exists $\mathcal{Y} \in \mathbb{C}$ such that $\mathcal{W} \subseteq \mathcal{Y}$.

Proof. This result is a consequence of Propositions 4 and 5. □

Furthermore, due to MHS duality between AXp's and CXp's, we can compute all the AXp's of $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$ that are contained in $\mathcal{Z}$, by hitting set dualization using the CXp's in $\mathbb{C}_{\mathcal{Z}}$.

Proposition 7. Each element of $\mathbb{A}_{\mathcal{Z}}$ is a MHS of the elements in $\mathbb{C}_{\mathcal{Z}}$ and vice-versa.

Proof. This result follows from Propositions 1 and 4 to 6. □

Building on earlier results on duality of explanations (Ignatiev et al., 2019b, 2020b), Propositions 4 to 7 uncover yet another dimension of the duality of explanations. This new dimension reveals nesting properties of AXp's and CXp's.

Corollary 1. Let $\mathcal{W} \subseteq \mathcal{Z} \subseteq \mathcal{F}$. Then,

1. The $\mathcal{W}$-set-restricted AXp's are a subset of the $\mathcal{Z}$-set-restricted AXp's.
2. Each $\mathcal{W}$-set-restricted CXp is a subset of some $\mathcal{Z}$-set-restricted CXp.

3. The $\mathcal{W}$(or $\mathcal{Z}$)-set-restricted AXp's can be obtained from the $\mathcal{W}$(or $\mathcal{Z}$)-set-restricted CXp's by hitting set dualization, and vice-versa.

Example 9. Consider the running example from Figure 2, and the instance $((0, 1, 1, 1, 1), \mathbf{1})$ consistent with path $\langle 1, 2, 5\rangle$, and defining an explanation problem $\mathcal{E}$. Consider the set of features $\mathcal{W} = \{2, 4\}$. Clearly, $(x_2 = 1) \wedge (x_4 = 1)$ suffices for the prediction. We can also conclude that $\{2, 4\}$ is an AXp. Moreover, $\{2\}$ and $\{4\}$ are $\mathcal{W}$-set-restricted CXp's, and MHS duality is observed. Now consider the set of features $\mathcal{Z} = \{2, 3, 4, 5\}$. Clearly, $(x_2 = 1) \wedge (x_3 = 1) \wedge (x_4 = 1) \wedge (x_5 = 1)$ suffices for the prediction. In this case, careful analysis reveals that $\{2, 4\}$ and $\{3, 5\}$ are $\mathcal{Z}$-set-restricted AXp's of $\mathcal{E}$ (and so also plain AXp's of $\mathcal{E}$). As a result,$\{2, 3\}$, $\{2, 5\}$, $\{3, 4\}$, $\{4, 5\}$ are $\mathcal{Z}$-set-restricted CXp's, and again MHS duality is observed. As can be observed, for the subset $\mathcal{W}$ of $\mathcal{Z}$, the AXp's are a subset of the AXp's of $\mathcal{Z}$, and each CXp restricted to $\mathcal{W}$ is a subset of the CXp's restricted to $\mathcal{Z}$.
Another observation related with this example, is that although both $\{2, 4\}$ and $\{3, 5\}$ are AXp's of the original explanation problem, only the first one is clearly related with path P_4 of the DT. ◁

3.3 Path Explanations

Paths in DTs can contain literals for a subset of the features, and can be consistent with many (possibly uncountable) points in feature space. The goal of this section is to investigate *path explanations*; these represent sets of features such that (3) holds true for *any* instance consistent with some given path. We will consider both abductive and contrastive path explanations, but we will also investigate how enumeration of path explanations can be instrumented.

We will now show how the results in Sections 3.1 and 3.2 can be used to formalize path explanations and subsequently the concept of explanation redundancy in DT paths. First, Section 3.1 showed how to reason in terms of literals associated with paths and not literals associated with points in feature space. Second, Section 3.2 showed how to analyze duality of explanations in the case when sets of features (concretely those not tested in a given path) are excluded from explanations.

Consider a path $R_k \in \mathcal{R}$ in a DT $\mathcal{T}$. We define the following (path-based) explanation function, for $\mathcal{Z} \subseteq \Phi(\mathrm{R_k})$:

$$\xi(\mathbf{x}; \mathcal{Z}, \ldots) = \left[\bigwedge_{\substack{j\in\mathcal{Z}\\(x_j\in S_l)\in\Lambda(\mathrm{R_k})}} (x_j \in S_l)\right] \tag{18}$$

(Observe that this explanation function was first discussed in Example 6.) As a result, given the proposed explanation function ξ, and as outlined in Section 3.1 we can define both weak AXp's and CXp's.

Example 10. For the DT in Figure 2, we consider path $P_4 = \langle 1, 2, 5, 9\rangle$, and so with $c = \mathbf{1}$. In this case, we have that $\Phi(P_4) = \{1, 2, 4\}$. For $\mathcal{Z} = \Phi(P_4)$ we get,

$$\xi(\mathbf{x}; \mathcal{Z}, \ldots) = [(x_1 \in \{0\}) \wedge (x_2 \in \{1\}) \wedge (x_4 \in \{1\})]$$ ◁

In addition, path explanations are defined using the explanation function proposed in (18).

Definition 5 (Path Explanations)**.** A (weak) path AXp (resp. CXp) is a (weak) AXp (resp. CXp) given the explanation function (18).

A path AXp will be denoted an *abductive path explanation* (APXp); a path CXp will be denoted a *contrastive path explanation* (CPXp). An explanation problem associated with a path in a DT is represented by the tuple $(\mathcal{M}, R_k)$. Moreover, to distinguish the two kinds of explanations, those introduced in Section 2.5 will be referred to as *instance-based* explanations. Observe that the key difference between instance-based and path-based explanations are the literals used in the definition of explanation. For instance-based explanations, the literals are obtained from the point in feature space, whereas for path-based explanations, the literals are obtained from the conditions on features specified along the given path.

One alternative to path explanations would be to consider instance-based AXp's and CXp's, as introduced in Section 2.5, by considering some point in feature space consistent with the given path. However, such explanations offer information that might be too specific.

Example. Consider a classification problem $\mathcal{M}$ with $\mathcal{F} = \{1, 2, 3\}$, $\mathbb{D} = (\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3)$, with $\mathcal{D}_1 = \mathbb{R}$ and $\mathcal{D}_2 = \mathcal{D}_3 = \mathbb{N}_0$. Let the classifier be represented by a DT, with path $P_1 = \langle 1, 2, 3, 4 \rangle$ with $\Lambda(P_1) = \{(x_1 \in [\mathrm{V}_{\min}, 10]), (x_2 \in \{0, 1, 2, 3, 4\}), (x_3 \in \{0, 1\})$, and with $\phi(1) = 3$, $\phi(2) = 2$, $\phi(3) = 1$, and with $\varsigma(\tau(P_2)) = 1$. Given the instance $(\mathbf{v}, c) = ((0, 0), 1)$, let the AXp be $\{1, 2\}$. The information that the conjunction $(x_1 = 0) \wedge (x_2 = 0)$ represents a sufficient condition for the prediction to be 1, is clearly less instructive than the information that $(x_1 \in [\mathrm{V}_{\min}, 10]) \wedge (x_2 \in \{0, 1, 2, 3, 4\})$ also represents a sufficient condition for the prediction to be 1.

Moreover, from Proposition 7 one can readily conclude that path AXp's and CXp's exhibit MHS duality.

Proposition 8. For a DT $\mathcal{T}$ with set of paths $\mathcal{R}$, and a path $R_k \in \mathcal{R}$, the APXp's of R_k are the MHSes of the CPXp's of R_k and vice-versa.

Proof. This result instantiates, in the case of paths in DTs, the result of Proposition 7 for restricted duality. □

Given the generalized definition of weak AXp in Definition 1, it is plain that, for $\mathsf{WeakAXp}$ defined using ξ, $\mathsf{WeakAXp}(\mathcal{Z}) = 1$ and $\mathsf{WeakCXp}(\mathcal{Z}) = 1$ for $\Phi(P_k) \subseteq \mathcal{Z} \subseteq \mathcal{F}$, and so $\mathsf{WeakAXp}(\mathcal{F})$ and $\mathsf{WeakCXp}(\mathcal{F})$ are true. (Observe that it is assumed that the classifier is non-constant.) It is also clear that $\mathsf{WeakAXp}(\emptyset) = 0$ and $\mathsf{WeakCXp}(\emptyset) = 0$. Finally, one can also conclude that if $\mathsf{WeakAXp}(\mathcal{A}_0) = 1$, then $\mathsf{WeakAXp}(\mathcal{A}_1) = 1$ for $\mathcal{A}_1 \supseteq \mathcal{A}_0$. The same observation holds for weak CXp's. As a result, by Definition 3 we can conclude that ξ is monotone-inducing. Thus, by Proposition 3, there is duality between AXp's and CXp's given the explanation function ξ.

Despite ξ representing an explanation function, we must also understand how the explanations obtained with ξ relate with the explanations for the decision tree $\mathcal{T}$.

As shown next, we can relate path explanations and path explanation duality with restricted duality.

Proposition 9. Let $\mathcal{M}$ be the classification problem associated with DT $\mathcal{T}$, let $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$ denote an explanation problem given some instance $(\mathbf{v}, c)$ consistent with $P_k \in \mathcal{R}$, and let ξ be the explanation function associated with P_k, i.e. the conjunction of the literals in $\Lambda(P_k)$. Then,
1. Each APXp of P_k is an AXp for $\mathcal{E}$ that is contained in $\Phi(P_k)$;
2. Each CPXp of P_k is a subset of some CXp for $\mathcal{E}$ that is contained in $\Phi(P_k)$.

Proof. This result follows from the results in Sections 3.1 and 3.2 and the results earlier in this section. □

Example 11. We revisit Example 8. Let $\mathcal{Z} = \Phi(P_4) = \{1, 2, 4\}$: $\Lambda(P_4) = \{(x_1 \in \{0\}), (x_2 \in \{1\}), (x_4 \in \{1\})\}$ Thus, the explanation function can be defined as follows,

$$\xi(\mathbf{x}; \mathcal{Z}, \ldots) = \bigwedge_{j \in \mathcal{Z}, (x_j \in S_l) \in \Lambda(P_4)} (x_j \in S_l)$$

Given the definition of path explanations (and so of (generalized) AXp's and CXp's), we can conclude that $\mathcal{X} = \{2, 4\}$ is a path AXp for P_4. Moreover, $\mathcal{Y}_1 = \{2\}$ and $\mathcal{Y}_2 = \{4\}$ are path CXp's for P_4. It can be observed that $\mathcal{X}$ is an AXp for *any* instance $(\mathbf{v}, \mathbf{1})$, with $\mathbf{v}$ consistent with P_4. However, both $\mathcal{Y}_1$ and $\mathcal{Y}_2$ are subsets of CXp's of possible instances $(\mathbf{v}, \mathbf{1})$, consistent with P_4. For example, one can identify a CXp $\{2, 3\}$ and also a CXp $\{4, 5\}$. ◁

The fact that path explanations can be related with AXp's restricted to a specific set of features also signifies that not all instance-based explanations represent path explanations. This observation can be related with the distinction between path-restricted and path-unrestricted explanations first studied in (Izza et al., 2020).

Example 12. For the running example shown in Figure 2, we analyze the abductive explanations of path $P_4 = \langle 1, 2, 5, 9 \rangle$. Suppose we are given the instance is $(\mathbf{v}, c) = ((0, 1, 1, 1, 1), 1)$. An AXp is $\{3, 5\}$. However, this explanation offers little insight to why the prediction is **1** for the instances that are consistent with P_4. Using the nomenclature of earlier work (Izza et al., 2020), whereas $\{3, 5\}$ is a path-unrestricted explanation, $\{2, 4\}$ is a path-restricted explanation. In this paper, we consider only path explanations, and so $\{2, 4\}$ is the only path AXp we are interested in computing. ◁

Table 2 summarizes the kinds of explanations considered in this paper. APXp's and CPXp's are introduced in this paper and, in contrast with the other kinds of explanations, these are defined in terms of literals obtained from a specific DT path. Clearly, due to being instance-independent, path explanations offer a simpler solution to represent explanations of decision trees that only depend on the structure of the tree. Furthermore, a few additional results are consequences of the results presented in this section. For example, despite being based on a different semantics, there is a one-to-one mapping between the APXp's of R_k and the path-restricted AXp's of any instance consistent with R_k. The sole difference between path-restricted AXp's and APXp's is that the literals associated with APXp's are taken from the associated path, whereas the literals associated with path-restricted AXp's are obtained from a concrete instance (consistent with the path). Finally, $\mathbb{E}_P$ denotes the set of explanation problems given a classification problem $\mathcal{M}$, and a path $R_k \in \mathcal{R}$ in a decision

Explanation	**Definition**	**Literals used in ξ**	**Features containing XP**
AXp, path-unrestricted	(3)(7)	Instance-based	$\mathcal{F}$
CXp, path-unrestricted	(5)(8)	Instance-based	$\mathcal{F}$
AXp, path-restricted	(3)(7)	Instance-based	$\Phi(R_k)$
CXp, path-restricted	(5)(8)	Instance-based	$\Phi(R_k)$
APXp	Definition 5	Path-based	$\Phi(R_k)$
CPXp	Definition 5	Path-based	$\Phi(R_k)$

Table 2: Types of explanations considered in the paper, both for some path $R_k \in \mathcal{R}$ and for any instance $(\mathbf{v}, c)$ consistent with R_k

tree $\mathcal{T}$: $\mathbb{E}_P = \{\mathcal{E} \,|\, \mathcal{E} = (\mathcal{M}, R_k), \mathcal{M} \in \mathbb{M}, R_k \in \mathcal{R}\}$. In the rest of the paper, $\mathcal{M}$ is assumed to be such that the classification function κ is monotone-inducing,

4. Path Explanation Redundancy in Decision Trees

Given the definition of path explanations in Section 3.3, we can formalize the concept of path explanation redundancy.

Definition 6 (Explanation Redundant Path/Feature (XRP/XRF))**.** Given a DT $\mathcal{T}$, with set of paths $\mathcal{R}$, and a path $R_k \in \mathcal{R}$, R_k is an *explanation-redundant path* (or XRP) if $\Phi(R_k)$ does not represent a path AXp. Given a path AXp $\mathcal{X}$ for R_k, any feature $i \in \Phi(R_k)$ that is not included in $\mathcal{X}$ is a *explanation-redundant feature* (or XRF).

Feature redundancy is relative to a given APXp. Different APXp's can yield different redundant features. Clearly, one can consider the enumeration of APXp's to identify the set of features that is never-redundant, by enumerating all APXp's for a given path, and discarding any of the features deemed redundant for all of the APXp's.

4.1 Explanation Redundancy in Running Examples

The following examples illustrate path explanations and explanation redundancy.

Example 13. With respect to Example 2, with the DT shown in Figure 2, let the target path be $P_1 = \langle 1, 2, 4, 7, 10, 15 \rangle$. (In this case there is only one point in feature space consistent with P_1, i.e. $(0, 0, 1, 0, 1)$.) We claim that $\mathcal{X} = \{3, 5\}$ is a weak APXp, and so that P_1 is explanation-redundant. To prove the claim, we consider all the possible assignments to the other features:

Feature	Assignments							
x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_4	0	1	0	1	0	1	0	1
$\kappa(x_1, x_2, 1, x_4, 1)$	1	1	1	1	1	1	1	1

As can be concluded, as long as $x_3 = 1$ and $x_5 = 1$, then the prediction remains unchanged, since $\kappa(x_1, x_2, 1, x_4, 1)$ only takes value 1, for any assignment to x_1, x_2, x_4. In this case, we can observe that a path-based explanation of size 5 can be reduced to a (weak) abductive path explanation of size 2. Hence, there are (at least) 3 redundant features (namely features 1, 2 and 4) out of a total of 5 features included in path P_1. The redundant features represent **60%** of the original path length. As noted earlier, this DT was generated by the GOSDT/OSDT ((generalized scalable) optimal sparse decision trees) tools (Hu et al., 2019; Lin et al., 2020; Rudin et al., 2021), that specifically target interpretability. $\triangleleft$

Example 14. With respect to Example 3, with the DT shown in Figure 3, let the target path be $P_3 = \langle 1, 2, 6, 11, 12 \rangle$. (In this case there is only one point in feature space consistent with P_3: $(1, 2, 1, 2)$.) It is easy to conclude that $\mathcal{X} = \{1, 2, 3\}$ is a weak APXp, and so that P_3 is explanation-redundant. Indeed, if x_4 is allowed to take any value, then one can observe that the prediction remains unchanged. $\triangleleft$

Example 15. With respect to Example 4, with the DT shown in Figure 4, let the target path be $P_2 = \langle 1, 3, 4, 6, 8 \rangle$. (An example of a point in feature space consistent with P_2 is $(2, 20, 0)$.) It is easy to conclude that neither x_1 nor x_2 are allowed to take any value, whereas x_3 can be unrestricted. Hence, $\mathcal{X} = \{1, 2\}$ is a weak APXp. Since neither x_1 nor x_2 can be dropped, then $\{1, 2\}$ is an APXp. The literals associated with the APXp are $\{(x_1 \in \{2\}), (x_2 \in \{\text{MnA..25}\})\}$. $\triangleleft$

The examples above reveal that DTs taken from recent textbooks and papers often exhibit path explanation redundancy. Moreover, the examples above also show that DTs taken from papers that specifically address the learning of optimal sparse DTs (which aim at interpretability) can exhibit path explanation redundancy. In fact, some examples confirm that there can exist paths in optimal sparse decision trees for which there are more redundant features than non-redundant features. Section 4.2 offers a high-level perspective of the experimental results, which reveal that path explanation redundancy in DTs is indeed ubiquitous. Afterwards, Section 4.3 proves that there are functions for which path explanation redundancy is unavoidable, even in provably size-minimal DTs. These results and observations offer conclusive evidence regarding the significance of filtering path explanation redundancy from DT explanations, and further underline the critical importance of efficient algorithms for computing explanations in DTs.

4.2 Path Explanation Redundancy in Practice

This section summarizes some key takeaways that can be drawn from the experimental results (see Section 6), and which offer ample practical justification for computing AXp's of DTs (and so finding and filtering path explanation redundancy).

Path explanation redundancy in published examples. Table 11 (see Page 307) summarizes results on path explanation redundancy for DTs included in representative bibliography on DTs, namely textbooks and surveys[16]. The key observation is that path

16. A non-exhaustive list of references includes (Moret, 1982; Breiman et al., 1984; Quinlan, 1993; Breslow and Aha, 1997; Džeroski and Lavrač, 2001; Rokach and Maimon, 2008; Bessiere et al., 2009; Russell and Norvig, 2010; Berthold et al., 2010; Flach, 2012; Zhou, 2012; Kotsiantis, 2013; Alpaydin, 2014;

explanation redundancy is ubiquitous in most DTs that have been used as examples in textbooks and surveys over the years, going back to the inception of tree learning algorithms.

Path explanation redundancy in learned DTs. Table 6 and Table 7 (see Page 302 and Page 303) summarize the results obtained with two different, publicly available tree learning tools, namely Interpretable AI (IAI) (Bertsimas and Dunn, 2017; IAI, 2020) and ITI (Utgoff et al., 1997), on a large number of publicly available datasets. IAI is a recent tool that specifically targets the learning of *interpretable DTs*. As can be concluded from the results, for most datasets, the DTs learned by both algorithms exhibit a significant percentage of explanation redundant paths. Moreover, for paths that exhibit explanation redundancy, the number of redundant literals can also be significant.

Large-scale path explanation redundancy. Table 10 (see Page 306) shows results for DTs learned on more complex datasets (which are also publicly available). For these examples, the number of explanation redundant features can far exceed the number of explanation relevant features. Concretely for some examples, the number of explanation-redundant features is more than 7 times larger than the number of features used in an AXp.

Path explanation redundancy in optimal (sparse) DTs. Table 12 (see Page 307) shows results for DTs learned with recently proposed algorithms that specifically target the learning of optimal (and so indirectly *interpretable*) DTs, concretely (Hu et al., 2019; Lin et al., 2020; Rudin et al., 2021) and also (Verwer and Zhang, 2019). As can be observed, the *optimal sparse* DTs shown in earlier work exhibit a very significant number of redundant paths (between 55% and 75%). For explanation-redundant paths, the percentage of explanation-redundant features can reach 60% (as illustrated with Example 13 for the DT shown in Figure 2).

4.3 Path Explanation Redundancy in Theory

This section proves two results. First, we prove that there exist functions for which paths in smallest-size DTs will exhibit a number of explanation-redundant literals that grow linearly with the number of features. Second, we prove that, for a DT to be irredundant, then it must represent a generalized decision function (Huang et al., 2021a).

Optimal decision trees that exhibit redundancy. To simplify the statement of the main result, the following definitions and assumptions are used. A dataset is consistent if for any point $\mathbf{x}$ in feature space, contains an instance $(\mathbf{x}, c)$ for at most one class $c \in \mathcal{K}$. A classifier is exact if it correctly classifies any instance in training data, and that training data is consistent. (A classifier is perfect if it is exact and is of smallest size (Ignatiev et al., 2020a, 2021).) Furthermore, we assume that a DT learner will not branch on variables that take constant value on all the instances in training data that are consistent with the already chosen literals.

Shalev-Shwartz and Ben-David, 2014; Kelleher et al., 2020; Alpaydin, 2016; Valdes et al., 2016; Poole and Mackworth, 2017; Witten et al., 2017; Bramer, 2020; Zhou, 2021).

Proposition 10. Consider the boolean function,

$$f(x_1, x_2, \ldots, x_{m-1}, x_m) = \bigvee_{i=1}^{m} x_i$$

Then, given any DT learning algorithm that learns an exact DT (one that correctly classifies any point in feature space), the learned DT contains a path with m literals, for which there exists an AXp containing one single feature.

Before proving the claim of Proposition 10, it should be observed that a more general result could be stated, where the literals $(z_i = v_i)$ for a non-boolean feature i with domain $\mathcal{D}_i$ would replace the boolean literal x_i. However, the basic result remains unchanged, as it reveals in theory the need for explaining decision trees.

Proof. The AXp's for function f are easy to identify. For prediction 1, function f has m AXp's, namely $\mathcal{E}_{1,i} = \{i\}$ with $i = 1, \ldots, m$. For prediction 0, function f has one AXp, namely $\mathcal{E}_{0,1} = \{1, 2, \ldots, m\}$. Any other weak AXp will not be subset-minimal.

Next, we show that, no matter how the DT is constructed, there will always be at least one path that grows with m, and for which the size of the AXp is 1. Since the exercise is purely conceptual, we can assume that the dataset has size 2^m, representing the truth table of function f. We construct a DT as follows. At each step, we let some adversary pick any variable, among the variables that have not yet been picked, and then show that only one option exists to continue the construction of the DT. Let the first variable be x_{i_1}, with $1 \le i_1 \le m$. For $x_{i_1} = 1$, the prediction is 1, and so the DT must have a terminal node labeled 1. For $x_{i_1} = 0$, the resulting function f_{i_1} mimics f, but without variable x_{i_1}. Hence, we let again some adversary pick any variable among those not yet chosen. (Clearly, there is no reason to pick a variable already picked, since the function f_{i_1} does not depend on x_{i_1}.) Let the new chosen variable variable be x_{i_2}. The analysis for x_{i_2} is exactly the same as for x_{i_1}, and for $x_{i_2} = 0$, we get a new function f_{i_2}. After analyzing all features, the resulting DT has m paths with prediction 1 and 1 path with prediction 0. Thus, $\mathcal{P} = \{P_1, \ldots, P_m\}$ represents the paths with prediction 1, and $\mathcal{Q} = \{Q_1\}$ represents the path with prediction 0. Moreover, P_m has length m, with literals $\langle x_{i_1} = 0, x_{i_2} = 0, \ldots, x_{i_{n-1}} = 0, x_{i_m} = 1\rangle$. (The resulting DT and path P_m are shown in Figure 5.) For the instance $\{x_{i_m} = 1\} \cup \{x_{i_j} = 0, 1 \le j \le m-1\}$, path P_m is consistent with the instance and it has m literals. However, the AXp is $\{i_m\}$, denoting that $x_{i_m} = 1$ suffices for the prediction. The analysis and conclusion is independent of the order of features chosen. □

Although the proof analyzed AXp's, for the proposed function and resulting DT, the APXp's would be the same.

Corollary 2. There are DT classifiers, defined on m features, for which an instance has an AXp of size 1, and the consistent path has length m, and so it can be made larger by a factor of m than the size of an AXp.

Decision trees without path explanation redundancy. In this section we argue that for a DT not to exhibit redundancy then it must correspond to an irreducible generalized decision function (GDF) (Huang et al., 2021a). A GDF represents a multi-class classifier,

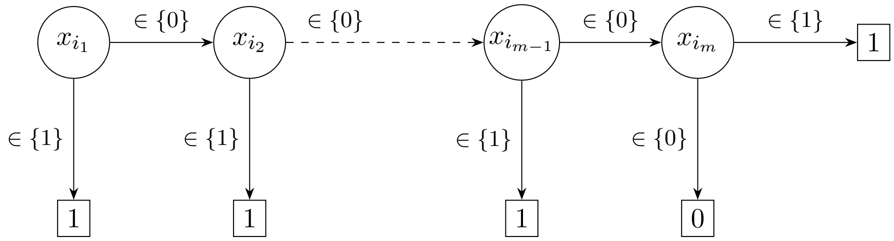


Figure 5. DT construction for proof of Proposition 10

with $\mathcal{K} = \{c_1, \ldots, c_K\}$, where each class $c_j \in \mathcal{K}$ is classified by a boolean function κ_j, such that set of boolean functions κ_i respects the following statement:

$$\forall(\mathbf{x} \in \mathbb{F}). \sum_{j=1,\ldots,K} \kappa_j(\mathbf{x}) = 1 \tag{19}$$

A GDF is represented by $\mathcal{G} = \{\kappa_1, \ldots, \kappa_K\}$. A DNF GDF is a set of boolean classifier functions κ_j, where each κ_j is represented by a disjunctive normal form (DNF) formula. A minimal DNF GDF is a set of boolean classifier functions where each κ_j is represented by an irredundant DNF formula φ_j, i.e. no term in the DNF φ_j is redundant, and no literal in any term of the DNF φ_j is redundant.

Lemma 2. A minimal DNF GDF $\mathcal{G}$ corresponds to a function representation where each term of each DNF for some κ_j is a prime implicant of κ_j.

Proof. Suppose a term t_r of the DNF representation of κ_j that is not a prime implicant of κ_j. Then, t_r can be simplified to t'_r, such that $t'_r \equiv t_r$. But then the DNF representation of $\mathcal{G}$ would not be minimal; a contradiction. □

Lemma 3. A DT does not exhibit path explanation redundancy iff the conjunction of the literals in each path to prediction $c \in \mathcal{K}$ represents a prime implicant for the boolean function $\kappa(\mathbf{x}) = c$.

Proof. If the conjunction of the literals in each path is a prime implicant for the boolean function $\kappa(\mathbf{x}) = c$, then no path in the DT exhibits path explanation redundancy; otherwise some path would not represent a prime implicant, as assumed by hypothesis.
If the DT exhibits no path path explanation redundancy, then we can represent the function $\kappa(\mathbf{x}) = c$ by a disjunction of the conjunctions of the literals in the paths predicting c. Each disjunct must be irreducible; otherwise we would be able to also reduce the explanation for some path. □

Proposition 11. A DT $\mathcal{T}$ does not exhibit path explanation redundancy iff there exists a minimal DNF GDF g that is equivalent to $\mathcal{T}$.

Proof. This result follows from Lemma 2 and Lemma 3. □

It should be underscored that minimal DNF GDFs represent a fairly restricted class of decision sets (DS) (Lakkaraju et al., 2016), namely minimal DSs exhibiting no overlap (Ignatiev et al., 2018b). The complexity of computing a minimal DS without overlap is not known, but it is conjectured to be hard for Σ_2^{P} (Ignatiev et al., 2018b). Furthermore, it is well-known that decision trees represent a far less expressive language than DSs (Rivest, 1987). Thus, most functions represented by DSs cannot be represented by DTs that correspond to minimal DNF GDFs. As a result, Proposition 11 offers further evidence that one should expect decision trees to be extremely unlikely to exhibit no path explanation redundancy in practice.

5. Computing Path Explanations in Decision Trees

Although the finding of formal explanations is computationally hard for a number of ML models (Ignatiev et al., 2019a; Barceló et al., 2020; Izza and Marques-Silva, 2021; Ignatiev and Marques-Silva, 2021; Audemard et al., 2021), it has been shown that for DTs, one AXp can be computed in polynomial time (Izza et al., 2020; Huang et al., 2021b)[17]. Moreover, and in the case of CXp's, recent work has shown that the total number of CXp's is polynomial, and that their enumeration runs in polynomial time (Huang et al., 2021b). This section refines these earlier results in several ways, proposing simpler and more efficient algorithms. More importantly, the section specifically considers algorithms for path explanations, as opposed to instance-based explanations. Nevertheless, the changes for computing (path restricted/unrestricted) AXp's/CXp's are straightforward.

We start by offering a simple approach supporting the rationale for polynomial-time explainability of DTs. Afterwards, we propose a simplified variant of an existing algorithm (Izza et al., 2020), and then detail a propositional logic Horn encoding for the problem of computing one AXp/APXp. The proposed encoding allows us to exploit existing algorithms for reasoning about propositional Horn formulas.

5.1 Abductive Path Explanations by Explicit Path Analysis

Since our goal is to compute a path explanation, we consider a concrete path P_k, a partition $(\mathcal{P}, \mathcal{Q})$ of the set of paths $\mathcal{R}$ in $\mathcal{T}$, with $P_k \in \mathcal{P}$ and with prediction $c = \varsigma(\tau(P_k))$ being the same for all paths in $\mathcal{P}$, and with the paths in $\mathcal{Q}$ yielding a prediction other than c. Let $F_k = \Phi(P_k)$, i.e. the set of features i associated with the edges of P_k. (For computing a path-unrestricted AXp, we would set $F_k = \mathcal{F}$.) Recall from Section 2.4 (and Table 1) that χ_I and χ_P represent, respectively, the set of features that are inconsistent between either a point or a path and some other path. For computing AXp's (i.e. given an instance) we will be interested in $\chi_I(\mathbf{v}, Q_l)$ for each $Q_l \in \mathcal{Q}$. For computing APXp's (i.e. given a path) we will be interested in $\chi_P(P_k, Q_l)$ for each $Q_l \in \mathcal{Q}$. Since the analysis is similar, we will focus on APXp's.

For the prediction to be guaranteed not to change, due to Q_l, at least one feature in $\chi_P(P_k, Q_l)$ must not be allowed to change value. Thus, one APXp is a (subset-)minimal hitting set of the sets $\chi_P(P_k, Q_l)$ ranging over the paths Q_l in $\mathcal{Q}$. Furthermore, it is well-

17. Furthermore, recent work has shown that computing a smallest size AXp is NP-hard (Barceló et al., 2020).

known that one subset-minimal hitting set can be computed in polynomial time (Eiter and Gottlob, 1995). For example, we can construct a set $\mathcal{X}$ containing the features in $\cup_{Q_l \in \mathcal{Q}} \chi_P(P_k, Q_l)$, and then iteratively remove one feature from $\mathcal{X}$ while the resulting set $\mathcal{X}$ is still a hitting set of all the $\chi_P(P_k, Q_l)$. (For AXp's, we would use a similar argument, but considering instead the sets $\chi_I(\mathbf{v}, Q_l)$.)

Example 16. Consider again the DT shown in Figure 1. For $P_2 = \langle 1, 3, 5, 7 \rangle$, we have that $\chi_P(P_2, Q_1) = \{1\}$ and $\chi_P(P_2, Q_2) = \{3\}$. Thus, the only minimal hitting set is $\{1, 3\}$, and so this represents the only APXp for P_2. Similarly, we could consider the instance $(\mathbf{v}, c) = ((1, 1, 1), 1)$, with $\chi_I((1, 1, 1), Q_1) = \{1\}$ and $\chi_I((1, 1, 1), Q_1) = \{3\}$, and so also obtain an AXp $\{1, 3\}$. Clearly, since P_2 is consistent with $\mathbf{v} = (1, 1, 1, 1)$, all the APXp's of P_2 should be AXp's of $\mathbf{v}$. ◁

Example 17. Consider again the DT shown in Figure 2. For path $P_1 = \langle 1, 2, 4, 7, 10, 15 \rangle$, we have that $\chi_P(P_1, Q_1) = \{3\}$, $\chi_P(P_1, Q_2) = \{5\}$, and $\chi_P(P_1, Q_3) = \{2, 5\}$. Clearly, the only minimal hitting set is $\{3, 5\}$ and so this represents the only APXp for P_1. Similarly, we could consider the instance $(\mathbf{v}, c) = ((0, 0, 1, 0, 1), 1)$ and so we would also obtain the AXp $\{3, 5\}$. ◁

Example 18. With respect to the DT shown in Figure 3, and for path $P_3 = \langle 1, 2, 6, 11, 12 \rangle$, we have that $\chi_P(P_3, Q_1) = \{3\}$, $\chi_P(P_3, Q_2) = \{2\}$, $\chi_P(P_3, Q_3) = \{1, 3\}$, and $\chi_P(P_3, Q_4) = \{1\}$. Clearly, the only minimal hitting set is $\{1, 2, 3\}$ and so this represents the only APXp for path P_3. ◁

The previous examples of explanations can also be viewed as *path-restricted* AXp's. The following example reveals the differences to path-unrestricted AXp's (Izza et al., 2020).

Example 19. Let us consider the example of Figure 1, and path $Q_1 = \langle 1, 2 \rangle$. In this case, we want to keep the paths P_1 and P_2 inconsistent. Hence, $\chi_P(Q_1, P_1) = \{1\}$ and $\chi_P(Q_1, P_3) = \{1\}$, and so the only APXp is $\{1\}$. Let us now consider the $((0, 1, 0), 0)$, which is consistent with path Q_1. In this case we get $\chi_I((0, 1, 0), P_1) = \{1, 2\}$ and $\chi_I((0, 1, 0), P_2) = \{1, 3\}$, and so the AXp's for the instance-based explanation problem are $\{1\}$ and $\{2, 3\}$. Observe that, given the instance, one can understand the AXp $\{2, 3\}$. However, in terms of explaining the sufficient conditions for the prediction to remain the same, given the values specified by the path, then it is clear that $\{1\}$ represents the only explanation of interest. ◁

An apparent drawback of computing explanations with the algorithm outlined in this section is that all DT paths must be explicitly listed, and these require worst-case quadratic space given the number of nodes in the DT. The next sections investigate alternative approaches, which perform better in practice.

5.2 Abductive Path Explanations by Tree Traversal

One approach to avoid the issue with explicit path representation is to iteratively traverse the DT as features are removed from the AXp, and checking whether the paths to predictions other than c remain inconsistent. This approach was first described in (Izza et al., 2020). Here, we describe a simpler variant.

Algorithm 1 summarizes the main steps of the proposed approach for computing an APXp for a concrete path P_k. (For computing a path-restricted AXp given an instance, we would just identify and use the same algorithm.) As shown, for APXp's (and also for path-

Algorithm 1: Computing one path explanation (or path-restricted AXp)

Function FindAPXp($\mathcal{T}, \mathcal{F}, P_k$)
  $\mathcal{U} \leftarrow \mathcal{F} \setminus \Phi(P_k)$; // Features $\notin$ path also $\notin$ APXp
  foreach $i \in \Phi(P_k)$ **do**
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{i\}$; // Tentatively drop i from APXp
    if ExistsConsistentQPath(P_k,$\mathcal{U}$, root($\mathcal{T}$)) **then**
      $\mathcal{U} \leftarrow \mathcal{U} \setminus \{i\}$; // Feature i must be included in APXp
  return $\mathcal{F} \setminus \mathcal{U}$; // Return APXp

Algorithm 2: Checking consistent path to prediction in $\mathcal{K} \setminus \{c\}$

Function ExistsConsistentQPath(P_k,$\mathcal{U}$, r)
  if $r \in T$ **then** // Decide return value if terminal
    if $\varsigma(r) \neq \varsigma(\tau(P_k))$ **then**
      return true; // Found consistent path to $d \neq c$
    else
      return false; // Not a consistent path to $d \neq c$
  $i \leftarrow \phi(r)$; // Pick feature associated with node r
  foreach $s \in \sigma(r)$ **do**
    // Recursively traverse child nodes, as long as
    // edge values exhibit consistent values
    if $(i \in \mathcal{U}) \vee (\rho(i, P_k) \cap \varepsilon((r,s)) \neq \emptyset)$ **then**
      if ExistsConsistentQPath(P_k,$\mathcal{U}$, s) **then**
        return true; // Found consistent path to $d \neq c$
  return false; // Unable to find consistent path to $d \neq c$

restricted AXp's given some instance), the features that are not tested in P_k are declared universal and added to a working set $\mathcal{U}$. (For computing a path-unrestricted AXp, the set $\mathcal{U}$ would be initialized to $\emptyset$.) The remaining features are analyzed one at a time. Each feature i is tentatively declared universal and Algorithm 1 then invokes a path traversal procedure (see Algorithm 2) for deciding whether there can exist a consistent path to a prediction other than c. If such a path exists, then the feature is added back to the set of features that must not be declared universal.

As can be observed, Algorithm 1 iteratively removes features from the set of features associated with P_k. For each feature i, Algorithm 1 then checks whether there exists some path in $\mathcal{Q}$ that can be made consistent. If such path exists, then i must be kept in the set of features sufficient for the prediction. Clearly, the tree traversal algorithm essentially tests whether the remaining set of features is still a hitting set of the paths in $\mathcal{Q}$, and so shares similarities with the algorithm described in Section 5.1, without exhibiting the drawback of explicitly enumerating all the paths in the DT.

The operation of both Algorithms 1 and 2 is summarized using the following example.

Example 20. We analyze the DT shown in Figure 4. Our goal is to find an APXp for path $P_3 = \langle 1, 3, 5 \rangle$. Let us assume that Algorithm 1 adds feature 1 to set $\mathcal{U}$, i.e. feature 1 is removed from the APXp being constructed. It is clear that, when the tree traversal is at node 1 (i.e. the root), it will take the left branch, and reach a terminal node with a prediction other that **N**; hence feature 1 must be removed from $\mathcal{U}$ and added to the APXp being constructed. $\triangleleft$

The running time of Algorithms 1 and 2 is clearly polynomial on the size of the DT. Given a path $R_k \in \mathcal{R}$, the algorithm analyzes the decision tree for each feature. Hence the running time is in $\mathcal{O}(|\mathcal{T}| \times |\mathcal{F}|)$. Moreover, Algorithm 1 can be run over all paths $\mathcal{R}$ in the DT $\mathcal{T}$. In this case, the running time is thus in $\mathcal{O}(|\mathcal{T}| \times |\mathcal{F}| \times |\mathcal{R}|)$. As the experimental results demonstrate, the running time of the algorithm is negligible (when compared with the time to learn the DT) almost without exception.

5.3 Abductive Path Explanations by Propositional Horn Encoding

One additional solution for computing an AXp is to formulate the problem as finding a minimal correction subset (MCS) of a propositional Horn formula, and then exploiting existing efficient algorithms (Arif et al., 2015; Marques-Silva et al., 2016). Besides enabling efficient implementations, the Horn encoding allows for integrating constraints that restrict the feature space by disallowing points in feature space that violate those constraints (Gorji and Rubin, 2021). As long as the added constraints are also Horn, and this is the case with propositional rules, then the complexity of reasoning is unaffected.

The general approach is to formulate a Horn optimization problem composed of a set of hard clauses $\mathcal{H}$ (which must be satisfied) and a set of soft clauses $\mathcal{B}$ (which ideally one would like to satisfy). Moreover, we seek an assignment to the variables that finds a subset-maximal set of clauses from $\mathcal{B}$ that are satisfied while satisfying the hard clauses. This problem can be solved in polynomial time in the case of Horn formulas (Arif et al., 2015; Marques-Silva et al., 2016), based on the fact that Horn formulas can be decided in linear time (Minoux, 1988). (Observe that finding a cardinality maximal solution, i.e. solving the MaxSAT problem for Horn formulas, is NP-hard (Jaumard and Simeone, 1987) and the respective decision problem is NP-complete. Similar results have been obtained for computing a smallest AXp (Barceló et al., 2020).)

It is straightforward to devise a naive Horn encoding that mimics the explicit path representation outlined above in Section 5.1. The dropping of each feature from the set of features in a APXp is represented by a boolean variable p_i. Ideally one would prefer to pick all features, and so the soft clauses are: $\mathcal{B} = \{(p_i) \,|\, i \in \Phi(P_k)\}$. Moreover, for each set $\chi_P(P_k, Q_l)$, with $Q_l \in \mathcal{Q}$, representing the features that are pairwise inconsistent between Q_l and P_k, one creates a Horn clause $(\vee_{i \in \chi_P(P_k,Q_l)} \neg p_i)$. Clearly, such an encoding does not offer any clear advantage with respect to the minimal hitting set algorithm, besides exploiting efficient Horn reasoners, since both approaches are based on explicit enumeration of all tree paths. A different approach, which avoids the worst-case quadratic representation on the size of the DT, is to devise a Horn encoding that bypasses the step of enumerating the paths in the DT. The main goal of this section is to propose such an encoding.

Let us consider a path $P_k \in \mathcal{P}$, with prediction $c \in \mathcal{K}$. Moreover, let $\mathcal{Q}$ denote the paths yielding a prediction other than c. Since the prediction is c, then any path in $\mathcal{Q}$ has some feature for which the allowed values are inconsistent with $\mathbf{v}$. We say that the paths in $\mathcal{Q}$ are *blocked*. (To be clear, a path is blocked as long as some of its literals are inconsistent.)

For each feature i associated with some node of path P_k, introduce a variable u_i. u_i denotes whether feature i is deemed *universal*, i.e. feature i is not included in the APXp that we will be computing. (Our goal is to find a subset maximal set of features that can be deemed universal, such that all the paths resulting in a prediction other than c remain blocked. Alternatively, we seek to find a subset-minimal set of features to declare non-universal or fixed, such that paths with a prediction other than c remain blocked.) Furthermore, for each DT node r, introduce variable b_r, denoting that all sub-paths from node r to any terminal node labeled $d \in \mathcal{K} \setminus \{c\}$ must be blocked, i.e. some literal in the sub-path must remain inconsistent. (Our goal is to guarantee that all paths to terminal nodes labeled $d \in \mathcal{K} \setminus \{c\}$ remain blocked even when some variables are allowed to become universal.)

We proceed to describe the proposed Horn encoding. Here, we opt to describe first the Horn encoding for computing a path-unrestricted AXp. Afterwards, we describe the Horn encoding for computing a path-restricted AXp (or an APXp).

First, for a path-unrestricted AXp, the soft clauses $\mathcal{B}$ are given by, $\{(u_i) \,|\, i \in \mathcal{F}\}$. In contrast, for APXp's and for path-restricted AXp's, the soft clauses $\mathcal{B}$ are given by, $\{(u_i) \,|\, i \in \Phi(P_k)\}$. In both cases, the goal is that one would ideally want to declare universal as many features as possible (among those that one can pick), thus minimizing the size of the explanation. (As noted above, we will settle for finding subset-maximal solutions.) We describe next the hard constraints $\mathcal{H}$ for representing consistent assignments to the u_i variables. For *path-unrestricted* AXp's (Izza et al., 2020), the hard constraints are created as follows:

H1. For the root node r, add the constraint $\top \rightarrow b_r$.
(The root node must be blocked.)

H2. For each terminal node r with prediction c, add the constraint $\top \rightarrow b_r$.
(Each terminal node with prediction c is also blocked. Also, observe that this condition is on the node, not on the path.)

H3. For each terminal node r with prediction $d \in \mathcal{K} \setminus \{c\}$, add the constraint $b_r \rightarrow \bot$.
(Terminal nodes predicting $d \neq c$ cannot be blocked. Also, and as above, observe that this condition is on the node, not on the path.)

H4. For a node r associated with feature i, and connected to the child node s, such that the edge value(s) is(are) *consistent* with the value of feature i in $\mathbf{v}$, add the constraint $b_r \rightarrow b_s$.
(If all sub-paths from node r must be blocked, then all sub-paths from node s must all be blocked, independently of the value taken by feature i.)

H5. For a node r associated with feature i, and connected to the child node s, such that the edge value(s) is(are) *inconsistent* with the value of feature i in $\mathbf{v}$, add the constraint $b_r \wedge u_i \rightarrow b_s$.
(In this case, the blocking condition along an edge inconsistent with the value of feature i in $\mathbf{v}$ is only relevant if the feature is deemed universal.)

Example 21. For the running example of Figure 2, let $(\mathbf{v}, c) = ((0,0,1,0,1),1)$. As dictated by the proposed Horn encoding, two sets of variables are introduced. The first set represents the variables denoting whether a feature is universal, corresponding to 5 variables: $\{u_1, u_2, u_3, u_4, u_5\}$. The second set represents the variables denoting whether a node is

Hard constraint type	Horn clauses
H1	$\{(b_1)\}$
H2	$\{(b_3),(b_9),(b_{11}),(b_{13}),(b_{15})\}$
H3	$\{(\neg b_6),(\neg b_{12}),(\neg b_{14})\}$
H4	$\{\,(b_1 \to b_2),(b_2 \to b_4),(b_4 \to b_7),(b_5 \to b_8),$ $(b_7 \to b_{10}),(b_8 \to b_{13}),(b_{10} \to b_{15})\,\}$
H5	$\{\,(b_1 \wedge u_1 \to b_3),(b_2 \wedge u_2 \to b_5),(b_4 \wedge u_3 \to b_6),$ $(b_5 \wedge u_4 \to b_9),(b_7 \wedge u_4 \to b_{11}),(b_8 \wedge u_5 \to b_{12}),$ $(b_{10} \wedge u_5 \to b_{14})\,\}$
Soft constraints, $\mathcal{B}$	$\{(u_1),(u_2),(u_3),(u_4),(u_5)\}$

Table 3: Horn clauses for the DT of Figure 2 for computing one AXp with $(\mathbf{v},c) = ((0,0,1,0,1),1)$

blocked, corresponding to 15 variables: $\{b_1,b_2,b_3,b_4,b_5,b_6,b_7,b_8,b_9,b_{10},b_{11},b_{12},b_{13},b_{14},b_{15}\}$. The resulting propositional Horn encoding contains hard ($\mathcal{H}$) and soft ($\mathcal{B}$) constraints, and it is organized as shown in Table 3.

It is easy to see that, if $u_1 = u_2 = u_3 = u_4 = u_5 = 1$, then $\mathcal{H}$ is falsified. Concretely, $(b_1) \wedge (b_1 \to b_2) \wedge (b_2 \to b_4) \wedge (u_3) \wedge (b_4 \wedge u_3 \to b_6) \wedge (\neg b_6) \nvDash \bot$. The goal is then to find a maximal subset $\mathcal{S}$ of $\mathcal{B}$ such that $\mathcal{S} \cup \mathcal{H}$ is consistent. (Alternatively, the algorithm finds a minimal set $\mathcal{C} \subseteq \mathcal{B}$, such that $\mathcal{B} \setminus \mathcal{C} \cup \mathcal{H}$ is consistent.) For this concrete example, one such minimal set is obtained by picking $u_1 = u_2 = u_4 = 1$ and $u_3 = u_5 = 0$, and by setting $b_1 = b_2 = b_3 = b_4 = b_5 = b_7 = b_8 = b_9 = b_{10} = b_{11} = b_{13} = b_{15} = 1$ and $b_6 = b_{12} = b_{14} = 0$. Hence, all clauses are satisfied, and so $\{3,5\}$ is a weak AXp. An MCS extractor (Marques-Silva et al., 2013a; Mencía et al., 2015, 2016) would confirm that $\{3,5\}$ is subset-minimal, and so it is an AXp. $\triangleleft$

Similarly, we can consider *path-restricted* AXp's (Izza et al., 2020) (or APXp's). As noted earlier, in this case, the soft clauses $\mathcal{B}$ are given by $\{(u_i)\,|\,i \in \Phi(P_k)\}$. The previous encoding can be modified to reflect the computation of a path-restricted AXp (and also an APXp), where a point $\mathbf{v} \in \mathbb{F}$ is no longer assumed. The changes to the previous encoding are as follows:

H′4. For a node r associated with feature i, and connected to the child node s, such that the edge value(s) is(are) *consistent* with the value of feature i tested in path P_k, or if feature i is not included in $\Phi(P_k)$, then add the constraint $b_r \to b_s$.
(If all sub-paths from node r must be blocked, then all sub-paths from node s must all be blocked, independently of the value taken by feature i.)

H′5. For a node r associated with feature i, and connected to the child node s, such that the edge value(s) is(are) *inconsistent* with the consistent values of feature i in path P_k, then add the constraint $b_r \wedge u_i \to b_s$.
(In this case, the blocking condition along an edge inconsistent with the consistent values of feature i along P_k is only relevant if the feature is deemed universal.)

Hard constraint type	Horn clauses
H1	$\{(b_1)\}$
H2	$\{(b_3), (b_9), (b_{11}), (b_{13}), (b_{15})\}$
H3	$\{(\neg b_6), (\neg b_{12}), (\neg b_{14})\}$
H′4	$\{\, (b_1 \rightarrow b_2), (b_2 \rightarrow b_4), (b_4 \rightarrow b_7), (b_5 \rightarrow b_8), (b_7 \rightarrow b_{10}), (b_8 \rightarrow b_{13}), (b_{10} \rightarrow b_{15}) \,\}$
H′5	$\{\, (b_1 \wedge u_1 \rightarrow b_3), (b_2 \wedge u_2 \rightarrow b_5), (b_4 \wedge u_3 \rightarrow b_6), (b_5 \wedge u_4 \rightarrow b_9), (b_7 \wedge u_4 \rightarrow b_{11}), (b_8 \wedge u_5 \rightarrow b_{12}), (b_{10} \wedge u_5 \rightarrow b_{14}) \,\}$
H′6	$\emptyset$ – all features in path
Soft constraints, $\mathcal{B}$	$\{(u_1), (u_2), (u_3), (u_4), (u_5)\}$

Table 4: Horn clauses for the DT of Figure 2 for computing one APXp with $P_1 = \langle 1, 2, 4, 7, 10, 15 \rangle$

H′6. For each feature i not included in $\Phi(P_k)$, add the unit clause (u_i). (Features not tested along P_k must not be included in the explanation.)

Concretely, the features not in the path must *not* be included in a path-restricted AXp or in an APXp.

Example 22. For the running example of Figure 2, and again with $(\mathbf{v}, c) = ((0, 0, 1, 0, 1), 1)$, the path consistent with $\mathbf{v}$ is $P_1 = \langle 1, 2, 4, 7, 10, 15 \rangle$. We use the same sets of variables as in Example 21. The resulting propositional Horn encoding contains hard ($\mathcal{H}$) and soft ($\mathcal{B}$) constraints, and consists of the following constraints shown in Table 4. i.e. the difference are the clauses forcing some features not to be included in explanations. ◁

Example 23. We use again the running example of Figure 2, but now we consider the path $P_4 = \langle 1, 2, 5, 9 \rangle$, e.g. by picking for example the instance $(\mathbf{v}, c) = ((0, 1, 1, 1, 0), 1)$. As before, we use the same sets of variables as in Example 21. The resulting propositional Horn encoding contains hard ($\mathcal{H}$) and soft ($\mathcal{B}$) constraints, and consists of the following constraints shown in Table 5. i.e. the difference are the clauses forcing some features not to be included in explanations.
As can be observed, any solution will set $u_3 = u_5 = 1$. It must also be the case that $u_2 = 0$ and $u_4 = 0$. However, we can safely set $u_1 = 1$. Hence the APXp is $\{2, 4\}$. ◁

Finally, and as hinted above, we observe that the same formulation can be used for computing a smallest AXp, by finding a cardinality-minimal instead of a subset-minimal set of true variables u_i. It is well-known that both problems, i.e. computing a smallest explanation and solving Horn MaxSAT, are hard for NP (Jaumard and Simeone, 1987; Barceló et al., 2020). Thus, we have the following result.

Proposition 12. Each maximum cost solution of the Horn formulation yields a cardinality-minimal AXp.

Hard constraint type	Horn clauses
H1	$\{(b_1)\}$
H2	$\{(b_3),(b_9),(b_{11}),(b_{13}),(b_{15})\}$
H3	$\{(\neg b_6),(\neg b_{12}),(\neg b_{14})\}$
H′4	$\{\,(b_1\rightarrow b_2),(b_2\rightarrow b_5),(b_4\rightarrow b_6),(b_4\rightarrow b_7),$ $(b_5\rightarrow b_9),(b_7\rightarrow b_{11}),(b_8\rightarrow b_{12}),(b_8\rightarrow b_{13}),$ $(b_{10}\rightarrow b_{14}),(b_{10}\rightarrow b_{15})\,\}$
H′5	$\{\,(b_1\wedge u_1\rightarrow b_3),(b_2\wedge u_2\rightarrow b_4),$ $(b_5\wedge u_4\rightarrow b_8),(b_7\wedge u_4\rightarrow b_{10})\,\}$
H′6	$\{(u_3),(u_5)\}$
Soft constraints, $\mathcal{B}$	$\{(u_1),(u_2),(u_3),(u_4),(u_5)\}$

Table 5: Horn clauses for the DT of Figure 2 for computing one APXp with $P_4=\langle 1,2,5,9\rangle$

Observe that, by enumerating Horn MaxSAT solutions, we are able to enumerate smallest AXp's.

5.4 Contrastive Path Explanations

Given a path $P_k\in\mathcal{R}$ in a DT, with prediction $c\in\mathcal{K}$, one can consider *any* instance consistent with P_k, and compute a CXp using the polynomial-time algorithm recently proposed in (Huang et al., 2021b). Since CXp's in DTs are constructed by path analysis, being limited to at most one per path with a different prediction, this immediately implies that their number is limited to the number of paths. Furthermore, a CXp associated with some path Q_l is declared redundant if some other path Q_s reveals a CXp with a stricter subset of the features provided by Q_l. Thus, we can conclude that the features associated with each CXp of an instance $(\mathbf{v},c)$ consistent with P_k must correspond to the CXp associated with some path Q_s.

Nevertheless, instance-based CXp's can contain features that are not even tested in path P_k. Given a path P_k with prediction c, a path Q_l to a prediction other than c may test a feature not tested in P_k. Hence, a CXp could report features not tested in P_k. Furthermore, using hitting set dualization for enumerating abductive explanations will require adapting existing algorithms to filter out features not tested in path P_k. This section details a more direct solution, one that takes into account both the generalized (literals obtained from those used in tree) and the restricted (literals obtained from those used in path P_k) aspects of path explanations in DTs. The computed explanations will be path contrastive explanations (CPXp's), and so subsets of actual CXp's for a concrete instance.

The proposed algorithm is based on earlier work (Huang et al., 2021b), with a few minor modifications:

1. Analyze each path Q_l in $\mathcal{Q}$ with prediction in $\mathcal{K}\setminus\{c\}$.

2. Traverse the path Q_l, ignore features that are not tested along P_k, and record in $\mathcal{C}$ the features with literals inconsistent with those in P_k, i.e. for a given feature i, $\rho(i, Q_l) \cap \rho(i, P_k) = \emptyset$.
3. Aggregate the computed sets of features $\mathcal{C}$, and keep the ones that are subset-minimal.

The previous algorithm runs in worst-case time $\mathcal{O}(m \times |\mathcal{Q}|)$. Furthermore, given recent results on the number of CXp's in DTs (Huang et al., 2021b), the number of CPXp's is bounded by $|\mathcal{Q}|$.

Example 24. With respect to the running example shown in Figure 2, and path $P_4 = \langle 1, 2, 5, 9 \rangle$, the algorithm would execute as follows:

Q_1: $\mathcal{C}_1 = \{2\}$.
Q_2: $\mathcal{C}_2 = \{2, 4\}$; drop $\mathcal{C}_2$.
Q_4: $\mathcal{C}_3 = \{4\}$.

Hence, the reported CPXp's would be: $\{\{2\}, \{4\}\}$. ◁

The fact that all CXp's can be enumerated in polynomial-time, offers an alternative to compute a smallest AXp that differs from the one proposed in Section 5.3. Indeed, a minimum-size (or minimum-cost) hitting set of the CXp's represents a smallest AXp.

Proposition 13. A minimum-cost hitting set of the CXp's is a smallest AXp, and vice-versa.

Thus, smallest AXp's can also be enumerated by enumerating minimum-cost hitting sets. (Also, there is a dual result regarding Proposition 13, its practical uses are unclear, since the number of AXp's may be exponentially large.)

5.5 Enumeration of Path Explanations

The enumeration of multiple (or all) abductive or contrastive explanations can help human decision makers to develop a better understanding for the reasons of some prediction, but also to gain a better perception of the underlying classifier. Recent work (Shih et al., 2018) compiles a decision function into a Sentential Decision Diagram (SDD), from which the enumeration of AXp's can be instrumented. Moreover, from a compiled representation of the AXp's, each AXp can be reported in polynomial time. The downside is that these representations are worst-case exponential in the size of the original ML model. Furthermore, it is unclear how compilation could be applied to the case of DTs. Another line of work for computing AXp's is based on iterative entailment checks using an NP-oracle (Ignatiev et al., 2019a), with enumeration studied in more recent work (Ignatiev et al., 2020b; Ignatiev and Marques-Silva, 2021). For classifiers for which AXp's and CXp's can be computed in polynomial time, a number of alternative algorithms have also been studied in recent work (Marques-Silva et al., 2020, 2021; Huang et al., 2021b), which guarantee that a single NP (in fact SAT) oracle call is required for each computed AXp or CXp. This section develops a solution for the enumeration of APXp's which builds on existing approaches for the enumeration of minimal hitting sets (MHSes). Despite a number of differences, the approach can be related with recent work (Huang et al., 2021b). A key insight is that exactly one call to a SAT oracle is required for each computed AXp, even if the computed AXp's are subset-minimal. This can in general be formalized as follows.

Proposition 14. If the computation of one AXp and one CXp runs in polynomial time, then there is an algorithm for the simultaneous enumeration of AXp's and CXp's that requires one SAT oracle call per computed AXp or CXp.

Proof. Consider the propositional encodings proposed in Section 5.3. We build $\mathcal{H}$ iteratively as follows. For each picked set of features representing an AXp, add a negative clause to $\mathcal{H}$, preventing the same AXp from being re-computed. For each picked set of features representing a CXp, add a positive clause to $\mathcal{H}$, requiring some of the non-picked features to be picked the next time. At each iteration, run a SAT oracle on $\mathcal{H}$. If the picked set of features is a weak AXp, then extract an AXp, and use it to add another clause to $\mathcal{H}$. If the picked set of features is a weak CXp, then extract a CXp, and use it to add another clause to $\mathcal{H}$. The algorithm iterates while there are additional AXp's or CXp's to enumerate. □

Moreover, given that the number of CPXp's is linear on the size of the decision tree (see Section 5.4, and given that an APXp must be a minimal hitting set of all the CPXp's (see Proposition 8), then we can construct the hypergraph of all CPXp's, which we can implement in polynomial time, and then exploit an existing hypergraph transversal (or hitting set dualization) approach (Bailey et al., 2003; Kavvadias and Stavropoulos, 2005; Khachiyan et al., 2006; Liffiton and Sakallah, 2008). Although some of these algorithms resort to NP oracles at each enumeration step (Liffiton and Sakallah, 2008) with promising experimental results, in theory each incremental step can be implemented in quasi-polynomial time (Fredman and Khachiyan, 1996).

The examples in Section 5.4 illustrate the use of hitting set dualization for computing APXp's from the complete set of CPXp's.

A SAT encoding. We consider the case of enumeration of APXp's from CPXp's; the case concerning the enumeration of (path (un)restricted) AXp's from CXp's would be similar. We associate a boolean variable p_i with each feature $i \in \mathcal{F}$, denoting (if equal to 1) whether the feature is picked to be included in some APXp. The CNF formula $\mathcal{H}$ is created as follows:

C1. For each CPXp $\mathcal{Y} = \{j_1, \ldots, j_r\}$, add a (positive) clause $(p_{j_1} \vee \ldots \vee p_{j_r})$ for $\mathcal{H}$, i.e. each APXp must hit all the CPXp's.

Furthermore, each time an APXp $\mathcal{X} = \{i_1, \ldots, i_s\}$ is computed, a new (negative) clause is added $(\neg p_{i_1} \vee \neg p_{i_2} \vee \ldots \vee \neg p_{i_s})$ to $\mathcal{H}$. While the formula $\mathcal{H}$ is satisfied, the computed model represents a superset of some APXp, that is not yet computed. As a result, we can then use a polynomial time algorithm for computing such an APXp, blocking it by adding a new (negative) clause to $\mathcal{H}$, and starting the process again. As can be concluded, the computation of each APXp requires one SAT oracle call, on a formula $\mathcal{H}$ whose size grows with the number of already computed CPXp's and the number of previously computed APXp's. Finally, we observe that, even though calling a SAT solver is computationally harder (in the worst-case) than a quasi-polynomial enumeration algorithm, e.g. the two algorithms proposed by M. Fredman and L. Khachiyan (Fredman and Khachiyan, 1996), existing practical evidence suggests otherwise (Liffiton and Sakallah, 2008).

Example 25. Consider the DT from Figure 2, and path $Q_2 = \langle 1, 2, 4, 7, 10, 14 \rangle$. By analyzing the paths with a different prediction we can identify the following weak CPXp's, from which CPXp's are then selected as follows:

Path	P_1	P_2	P_3	P_4	P_5	CXp's
Weak CPXp's	$\{5\}$	$\{4\}$	$\{2,5\}$	$\{2,4\}$	$\{1\}$	$\{\{1\},\{4\},\{5\}\}$

It is clear that the only APXp is $\{1,4,5\}$. The initial CNF formula $\mathcal{H}$ is: $\{(p_1),(p_4),(p_5)\}$. A SAT solver would compute an assignment that satisfies $\mathcal{H}$, e.g. $\{(p_1=1),(p_2=0),(p_3=1),(p_4=1),(p_5=1)\}$. From this satisfying assignment, we identify the Weak APXp: $\{1,3,4,5\}$, from which the APXp $\{1,4,5\}$ would then be extracted. As a result, $\mathcal{H}$ is extended with the clause $(\neg p_1 \vee \neg p_2 \vee \neg p_3)$. Clearly, with the new clause, the formula $\mathcal{H}$ becomes inconsistent, confirming that $\{1,4,5\}$ is the only APXp. $\triangleleft$

Finally, we observe that the proposed SAT encoding can be used for enumerating smallest AXp's, as a direct consequence of Proposition 13. For computing a smallest AXp,the hard clauses are the ones proposed above (see **C1** on Page 300), whereas the soft clauses are defined as follows:

C2. For each feature i, add a soft unit clause (p_i).

Thus, instead of just enumerating AXp's using a SAT formulation, the proposed MaxSAT formulation can be used for enumerating smallest AXp's, but also for enumerating AXp's by increasing size. Hence, we have the following result.

Proposition 15. The minimum-cost models of the propositional logic encoding summarized in **C1** and **C2** represent smallest AXp's. Given the MaxSAT encoding proposed above, each of its optimum solutions represents one smallest AXp. The enumeration of MaxSAT solutions by decreasing size will produce AXp's by increasing size.

6. Experimental Results

This section presents a summary of experimental evaluation of the explanation redundancy of two state-of-the-art heuristic DT classifiers and runtime assessment of the proposed algorithms to extract (path-restricted) AXp's from DTs, and also explanation redundancy in a range of DTs reported in the literature.

Experimental setup. We use the well-known DT learning tools *ITI* (*Incremental Tree Induction*) (Utgoff et al., 1997; ITI, 2020) and *IAI* (*Interpretable AI*) (Bertsimas and Dunn, 2017; IAI, 2020). ITI is run with the pruning option enabled, which helps avoiding overfitting and aims at constructing shallow DTs. To enforce IAI to produce shallow DTs and achieve high accuracy, it is set to use the optimal tree classifier method with the maximal depth of 6. This choice is motivated by our results, which confirm that larger maximal depths would in most cases increase the percentage of explanation redundant paths; on the other hand, a smaller maximal depth would not improve accuracy. The experiments consider datasets with categorical (non-binarized) data, which both ITI and IAI can handle. (Note that other known DT learning tools, including scikit-learn (Pedregosa et al., 2011) and DL8.5 (Aglin et al., 2020a; Verhaeghe et al., 2020b) can only handle numerical and binary features, respectively, and so could not be included in the experiments.) Furthermore, the experiments are performed on a MacBook Pro with a Dual-Core Intel Core i5 2.3GHz CPU with 8GByte RAM running macOS Catalina.

Dataset	IAI								
	D	**#N**	**%A**	**#P**	**%R**	**%C**	**%m**	**%M**	**%avg**
adult	6	83	78	42	33	25	20	40	25
ann-thyroid	6	61	97	31	25	30	20	50	36
anneal	6	29	99	15	26	16	16	33	21
backache	4	17	72	9	33	39	25	33	30
bank	6	113	88	57	5	12	16	20	18
biodegradation	5	19	65	10	30	1	25	50	33
cancer	6	37	87	19	36	9	20	25	21
car	6	43	96	22	86	89	20	80	45
colic	6	55	81	28	46	6	16	33	20
compas	6	77	34	39	17	8	16	20	17
contraceptive	6	99	49	50	8	2	20	60	37
dermatology	6	33	90	17	23	3	16	33	21
divorce	5	15	90	8	50	19	20	33	24
german	6	25	61	13	38	10	20	40	29
heart-c	6	43	65	22	36	18	20	33	22
heart-h	6	37	59	19	31	4	20	40	24
kr-vs-kp	6	49	96	25	80	75	16	60	33
lending	6	45	73	23	73	80	16	50	25
letter	6	127	58	64	1	0	20	20	20
lymphography	6	61	76	31	35	25	16	33	21
mushroom	6	39	100	20	80	44	16	33	24
pendigits	6	121	88	61	0	0	—	—	—
promoters	1	3	90	2	0	0	—	—	—
recidivism	6	105	61	53	28	22	16	33	18
seismic_bumps	6	37	89	19	42	19	20	33	24
shuttle	6	63	99	32	28	7	20	33	23
soybean	6	63	88	32	9	5	25	25	25
spambase	6	63	75	32	37	12	16	33	19
spect	6	45	82	23	60	51	20	50	35
splice	3	7	50	4	0	0	—	—	—

Table 6: Path explanation redundancy in decision trees obtained with IAI. The table shows tree statistics for IAI, namely, tree depth **D**, number of nodes **#N**, test accuracy **%A** and number of paths **#P**. The percentage of explanation-redundant paths (XRP's) is given as **%R** while the percentage of data instances (measured for the *entire* feature space) covered by XRP's is **%C**. Focusing solely on the XRP's, the average (min. or max., resp.) percentage of explanation-redundant features per path is denoted by **%avg** (**%m** and **%M**, resp.).

Benchmarks. The assessment is performed on a selection of 67 publicly available datasets, which originate from *UCI Machine Learning Repository* (UCI, 2020), *Penn Machine Learning Benchmarks* (PennML, 2020), and *OpenML repository* (OpenML, 2020). (We opt to report the results only for a subset of datasets. However, the results shown mimic the results for the complete benchmark set; these are provided as supplementary materials [18].) The number of features (data instances, resp.) in the benchmark suite vary from 2 to 58 (87 to 58000, resp.) with the average being 31.2 (6045.3, resp.).

18. https://github.com/yizza91/jair22sub

Dataset	ITI								
	D	**#N**	**%A**	**#P**	**%R**	**%C**	**%m**	**%M**	**%avg**
adult	6	83	78	42	33	25	20	40	25
ann-thyroid	6	61	97	31	25	30	20	50	36
anneal	6	29	99	15	26	16	16	33	21
backache	4	17	72	9	33	39	25	33	30
bank	6	113	88	57	5	12	16	20	18
biodegradation	5	19	65	10	30	1	25	50	33
cancer	6	37	87	19	36	9	20	25	21
car	6	43	96	22	86	89	20	80	45
colic	6	55	81	28	46	6	16	33	20
compas	6	77	34	39	17	8	16	20	17
contraceptive	6	99	49	50	8	2	20	60	37
dermatology	6	33	90	17	23	3	16	33	21
divorce	5	15	90	8	50	19	20	33	24
german	6	25	61	13	38	10	20	40	29
heart-c	6	43	65	22	36	18	20	33	22
heart-h	6	37	59	19	31	4	20	40	24
kr-vs-kp	6	49	96	25	80	75	16	60	33
lending	6	45	73	23	73	80	16	50	25
letter	6	127	58	64	1	0	20	20	20
lymphography	6	61	76	31	35	25	16	33	21
mushroom	6	39	100	20	80	44	16	33	24
pendigits	6	121	88	61	0	0	—	—	—
promoters	1	3	90	2	0	0	—	—	—
recidivism	6	105	61	53	28	22	16	33	18
seismic_bumps	6	37	89	19	42	19	20	33	24
shuttle	6	63	99	32	28	7	20	33	23
soybean	6	63	88	32	9	5	25	25	25
spambase	6	63	75	32	37	12	16	33	19
spect	6	45	82	23	60	51	20	50	35
splice	3	7	50	4	0	0	—	—	—

Table 7: Path explanation redundancy in decision trees obtained with ITI. Columns **D**, **#N**, **#P**, **%R**, **%C**, **%m**, **%M** and **%avg** have the same meaning as in Table 6.

Prototype implementation. The poly-time explanation-redundancy check algorithm presented in (Izza et al., 2020) and AXp extraction by Tree Traversal outlined in Section 5.2 are implemented in Perl. (An implementation using PySAT (Ignatiev et al., 2018a) toolkit and the solver Glucose, was instrumented in validating the results, but for the DTs considered, it was in general slower by at least one order of magnitude.) Additionally, the Propositional Horn Encoding approach outlined in Section 5.3 as well as the enumeration of AXp's/CXp's described in Section 5.5, are implemented in Python[19].

Results. Training DTs with IAI takes from 4s to 2310s with the average run time per dataset being 70s. In contrast, the time spent on eliminating explanation redundancy is *negligible*, taking from 0.026s to 0.4s per tree, with an average time of 0.06s. ITI runs

19. Sources are provided as a Python package and available in `https://github.com/yizza91/xpg`

Dataset	IAI							
	Traversal				Horn			
	m	**M**	**avg**	**Tot**	**m**	**M**	**avg**	**Tot**
adult	0.001	0.059	0.002	3.52	0.001	0.005	0.002	2.93
ann-thyroid	0.001	0.005	0.002	3.67	0.001	0.005	0.001	2.85
anneal	0.001	0.005	0.001	1.22	0.001	0.003	0.001	0.75
backache	0.001	0.001	0.001	0.13	0.000	0.001	0.001	0.09
bank	0.002	0.062	0.003	34.45	0.002	0.008	0.002	25.53
biodegradation	0.000	0.003	0.001	0.21	0.000	0.002	0.001	0.18
cancer	0.001	0.003	0.001	0.53	0.001	0.003	0.001	0.39
car	0.001	0.004	0.001	0.52	0.001	0.002	0.001	0.48
colic	0.001	0.005	0.002	0.71	0.001	0.002	0.001	0.43
compas	0.001	0.004	0.002	0.66	0.001	0.004	0.002	0.52
contraceptive	0.001	0.005	0.002	0.70	0.002	0.004	0.002	0.78
dermatology	0.001	0.005	0.001	0.50	0.001	0.002	0.001	0.31
divorce	0.000	0.002	0.001	0.10	0.000	0.001	0.001	0.08
german	0.001	0.004	0.001	0.76	0.001	0.002	0.001	0.65
heart-c	0.001	0.003	0.001	0.42	0.001	0.003	0.001	0.30
heart-h	0.001	0.004	0.001	0.39	0.001	0.005	0.001	0.27
kr-vs-kp	0.001	0.008	0.002	2.17	0.001	0.004	0.001	1.22
lending	0.001	0.003	0.001	1.99	0.001	0.003	0.001	1.51
letter	0.002	0.062	0.002	13.36	0.002	0.007	0.003	14.30
lymphography	0.001	0.007	0.002	0.32	0.001	0.003	0.001	0.20
mushroom	0.001	0.004	0.001	3.11	0.001	0.002	0.001	2.20
pendigits	0.002	0.063	0.003	10.03	0.002	0.007	0.003	8.45
promoters	0.000	0.000	0.000	0.02	0.000	0.000	0.000	0.02
recidivism	0.002	0.061	0.003	3.65	0.002	0.006	0.002	2.59
seismic_bumps	0.001	0.003	0.001	1.08	0.001	0.002	0.001	0.68
shuttle	0.001	0.006	0.001	22.65	0.001	0.005	0.001	22.35
soybean	0.001	0.058	0.002	1.17	0.001	0.005	0.001	0.82
spambase	0.001	0.008	0.003	3.43	0.001	0.003	0.001	1.81
spect	0.001	0.006	0.002	0.52	0.001	0.004	0.001	0.24
splice	0.000	0.001	0.000	0.22	0.000	0.002	0.000	0.30

Table 8: Assessing runtimes of the tree traversal algorithm and the propositional horn encoding approach for extracting one AXp. The table reports the results for DTs trained with IAI learning tool. Columns **m**, **M** and **avg** report, resp. , the minimal, maximal and average runtime (in second) to compute an AXp, while column **Tot** reports the total runtime (in second) of all tested instances in a dataset.

much faster than IAI and takes from 0.1s to 2s with 0.1s on average; the elimination of explanation redundancy is slightly more time consuming than for IAI, taking from 0.025s to 5.4s with 0.29s on average. This slowdown results from DTs learned with ITI being deeper on average, and features being tested multiple times along a same path.

Table 6 and Table 7 summarize, resp., the results of the explanation redundancy evaluation of IAI and ITI trees. Observe that despite the shallowness of the trees produced by IAI and ITI, for the majority of datasets and with a few exceptions, the paths in trees trained by both tools exhibit significant explanation redundancy. In particular, on average, 32.1% (46.9%, resp.) of paths are explanation redundant for the trees obtained by IAI (ITI, resp.). For some DTs, obtained with either IAI and ITI, more than 85% of tree paths are

Dataset	ITI							
	Traversal				Horn			
	m	**M**	**avg**	**Tot**	**m**	**M**	**avg**	**Tot**
adult	0.004	0.038	0.007	12.89	0.008	0.036	0.010	19.00
ann-thyroid	0.002	0.041	0.006	12.14	0.004	0.029	0.005	9.64
anneal	0.001	0.006	0.001	0.96	0.001	0.004	0.001	0.70
backache	0.000	0.001	0.000	0.08	0.000	0.001	0.000	0.07
bank	0.013	0.090	0.027	19.64	0.025	0.092	0.033	24.21
biodegradation	0.002	0.007	0.003	0.98	0.001	0.003	0.002	0.48
cancer	0.001	0.004	0.001	0.32	0.001	0.002	0.001	0.26
car	0.001	0.002	0.001	0.47	0.001	0.002	0.001	0.59
colic	0.000	0.001	0.001	0.24	0.000	0.001	0.000	0.18
compas	0.002	0.065	0.004	1.37	0.003	0.005	0.004	1.28
contraceptive	0.003	0.064	0.006	2.64	0.006	0.012	0.007	2.87
dermatology	0.000	0.001	0.001	0.27	0.000	0.001	0.001	0.19
divorce	0.000	0.001	0.000	0.03	0.000	0.002	0.000	0.04
german	0.002	0.007	0.003	3.19	0.002	0.003	0.002	2.03
heart-c	0.000	0.001	0.001	0.17	0.000	0.001	0.000	0.14
heart-h	0.001	0.001	0.001	0.21	0.001	0.001	0.001	0.18
kr-vs-kp	0.001	0.009	0.004	3.41	0.001	0.003	0.002	1.49
lending	0.004	0.030	0.006	9.13	0.008	0.039	0.010	15.99
letter	0.034	0.110	0.052	19.50	0.078	0.16	0.110	40.77
lymphography	0.000	0.002	0.001	0.13	0.001	0.001	0.001	0.09
mushroom	0.001	0.003	0.001	2.46	0.001	0.002	0.001	1.54
pendigits	0.008	0.047	0.011	37.59	0.015	0.056	0.019	61.59
promoters	0.000	0.001	0.000	0.04	0.000	0.001	0.000	0.04
recidivism	0.005	0.087	0.010	11.69	0.010	0.084	0.015	18.26
seismic_bumps	0.001	0.004	0.002	1.83	0.001	0.002	0.001	0.69
shuttle	0.002	0.061	0.002	2.85	0.003	0.060	0.003	3.56
soybean	0.001	0.005	0.003	1.65	0.001	0.003	0.002	0.95
spambase	0.002	0.069	0.009	11.20	0.003	0.062	0.003	4.18
spect	0.000	0.001	0.001	0.17	0.000	0.001	0.000	0.11
splice	0.001	0.064	0.002	1.48	0.003	0.069	0.004	3.43

Table 9: Assessing runtimes of the tree traversal algorithm and the propositional horn encoding approach for extracting one AXp. The table reports the results for DTs trained with ITI learning tool. Columns **m**, **M** , **avg** and **Tot** have the same meaning as in Table 8.

explanation redundant (XRP). Also, explanation redundant paths (XRP's) of the trees of IAI (ITI, resp.) cover on average 20.1% (37.7%, resp.) of feature space[20]. Moreover, in some cases, up to 89% and 98% of the entire feature space is covered by the XRP's for IAI and ITI, respectively. This means that DTs produced by IAI and ITI are unable to provide a user with a succinct explanation for the *vast majority* of data instances. In addition, the average number of explanation redundant features (XRF's) in XRP's for both IAI and ITI varies from 16% to 65%, but for some DTs it exceeds 80%.

To summarize, the numbers shown for the selected datasets and for the two state-of-the-art DT training tools (IAI and ITI) contrast with the common belief in the *inherent*

20. The coverage of a path is the feature space size of uninvolved/untested features in this path.

Dataset	DT			Path	AXp			
	D	**#N**	**%A**	**L**	**m**	**M**	**avg**	**n**
adult	17	509	73	16	8	8	8	2
				14	5	6	5.5	2
				16	5	5	5	1
allhyper	14	49	96	14	4	5	4.6	6
				9	4	5	4.5	8
				14	4	5	4.6	6
ann-thyroid	48	222	93	40	5	6	5.6	3
				36	5	6	5.6	3
				20	5	6	5.5	2
coil2000	12	177	91	10	2	4	3.8	39
				10	2	4	3.8	39
				10	2	4	3.8	30
fars	60	9969	76	29	11	11	11	2
				42	10	14	12.3	9
				48	9	9	9	1
kddcup	29	269	99	23	11	12	11.5	16
				23	11	12	11.5	16
				27	12	13	12.5	8

Table 10: Examples of 6 real-world datasets highlighting computed path AXp's (APXp's) in DTs learned with ITI, that require deep trees. For each dataset, the table displays 3 tested paths. Columns **D**, **#N** and **%A** denote, resp. depth, number of nodes and accuracy of the DT. Next, column **L** reports the path's length. Then, the average (min. or max., resp.) length of the computed APXp's, is denoted by **avg** (**m** and **M**, resp.). Finally, the total number of APXp's is shown in column **n**.

interpretability of decision tree classifiers. Perhaps as importantly, the performance figures confirm that the elimination of explanation redundancy in the DTs produced with available tools has *negligible* computational cost.

To demonstrate the effectiveness of the proposed algorithms, concretely tree traversal and propositional Horn encoding, we assess their running times to compute path-restricted AXp's from DTs obtained with ITI and IAI. The results are summarized in Table 8 and Table 9. As is quite evident from these results, the proposed solutions are effective in practice and the average running times are almost similar for all datasets and both DT learning tools. As final remark, we notice that in terms of comparison between the two algorithms, the Horn encoding approach is faster in 44/62 explained DTs trained with IAI and 41/62 DTs trained with ITI. Therefore, one can use a portfolio of the two approaches, terminating when one finishes.

Focusing merely on complex datasets that require deep trees, Table 10 shows results on computed path AXp's for a set of DTs generated by ITI. The results show that for these examples, paths can be much longer than path AXp's, namely, the number of explanation redundant features is bigger than the number of features involved in the explanation.

DT Ref	D	#N	#P	%R	%C	%m	%M	%avg
(Alpaydin, 2014, Ch. 09, Fig. 9.1)	2	5	3	33	25	50	50	50
(Alpaydin, 2016, Ch. 03, Fig. 3.2)	2	5	3	33	25	50	50	50
(Bramer, 2020, Ch. 01, Fig. 1.3)	4	9	5	60	25	25	50	36
(Breslow and Aha, 1997, Figure 1)	3	12	7	14	8	33	33	33
(Berthold et al., 2010, Ch. 08, Fig. 8.2)	3	7	4	25	12	50	50	50
(Breiman et al., 1984, Ch. 01, Fig. 1.1)	3	7	4	50	25	33	33	33
(Džeroski and Lavrač, 2001, Ch. 01, Fig. 1.2a)	2	5	3	33	25	33	33	33
(Džeroski and Lavrač, 2001, Ch. 01, Fig. 1.2b)	2	5	3	33	25	33	33	33
(Kelleher et al., 2020, Ch. 04, Fig. 4.14)	3	7	4	25	12	50	50	50
(Kelleher et al., 2020, Sec. 4.7, Ex. 4)	2	5	3	33	25	50	50	50
(Quinlan, 1993, Ch. 01, Fig. 1.3)	3	12	7	28	17	33	50	41
(Rokach and Maimon, 2008, Ch. 01, Fig. 1.5)	3	9	5	20	12	33	33	33
(Rokach and Maimon, 2008, Ch. 01, Fig. 1.4)	3	7	4	50	25	33	33	33
(Witten et al., 2017, Ch. 01, Fig. 1.2)	3	7	4	25	12	50	50	50
(Valdes et al., 2016, Figure 4)	6	39	20	65	63	20	40	33
(Flach, 2012, Ch. 02, Fig. 2.1(right))	2	5	3	33	25	50	50	50
(Kotsiantis, 2013, Figure 1)	3	10	6	33	11	33	33	33
(Moret, 1982, Figure 1)	3	9	5	80	75	33	50	41
(Poole and Mackworth, 2017, Ch. 07, Fig. 7.4)	3	7	4	50	25	33	33	33
(Russell and Norvig, 2010, Ch. 18, Fig. 18.6)	4	12	8	25	6	25	33	29
(Shalev-Shwartz and Ben-David, 2014, Ch. 18, Page 212)	2	5	3	33	25	50	50	50
(Zhou, 2012, Ch. 01, Fig. 1.3)	2	5	3	33	25	33	33	33
(Bessiere et al., 2009, Figure 1b)	4	13	7	71	50	33	50	36
(Zhou, 2021, Ch. 04, Fig. 4.3)	4	14	9	11	2	25	25	25

Table 11: Results on path explanation redundancy for example DTs found in the literature. Columns **D**, **#N**, **#P**, **%R**, **%C**, **%m**, **%M** and **%avg** have the same meaning as in Table 6.

Dataset	Tool	D	#N	%A	#P	%R	%C	%m	%M	%avg
monk1	BinOCT	3	13	91	7	28	11	66	66	66
	OSDT	5	13	100	7	57	41	33	33	33
tic-tac-toe	BinOCT	4	15	77	8	75	75	33	33	33
	OSDT	5	15	83	8	75	37	25	60	43
compas	OSDT	4	9	67	5	60	37	33	33	33
monk2	CART	6	31	69	16	62	22	20	66	33
	GOSDT	6	17	73	9	55	48	16	40	31

Table 12: Additional results on path explanation redundancy in (optimal) DTs, trained with different training tools: BinOCT (Verwer and Zhang, 2019), CART (Breiman et al., 1984), OSDT (Hu et al., 2019) and GOSDT (Lin et al., 2020), that have been presented in (Hu et al., 2019; Rudin et al., 2021). The results for CART are solely included for completeness. Columns **D**, **#N**, **%A**, **#P**, **%R**, **%C**, **%m**, **%M** and **%avg** hold the same meaning in Table 6.

Notably for some examples, the number of explanation redundant features is more than 7 times larger than the number of features belonging to the abductive explanation.

Finally, additional results on explanation redundancy of DTs reported in the literature are shown in Table 11 and Table 12. As can be seen, the same observations made for DTs of IAI and ITI hold for DTs obtained with different training tools existing in the literature.

More notably, these results demonstrate that also optimal (sparse) DTs, deemed succinctly explainable due to their shallowness, exhibit explanation-redundant paths/features.

7. Related Work

As indicated in Section 1, there exists a growing body of work on (optimally) learning DTs aiming for interpretability[21]. There is also general consensus on the interpretability of DTs (Breiman, 2001; Rudin, 2019; Molnar, 2020). The results in this paper prove that efforts for learning optimal DTs are necessarily incomplete, since the trees generated by such tools can (and inevitably will) exhibit path explanation redundancy. Furthermore, if interpretability is to be related with explanation succinctness, then our results prove (in theory and in practice) that learned optimal DTs should not in general be deemed interpretable, because more succinct (and in some cases far more succinct) explanations can be obtained with the algorithms proposed in this paper.

To our best knowledge, the assessment of path explanation redundancy in DTs when compared to AXp's has not been investigated in depth, besides our own work (Izza et al., 2020; Huang et al., 2021b) and results on the complexity of explaining DTs (Barceló et al., 2020) or the intelligibility of DTs (Audemard et al., 2021). However, some of the earlier results focus on boolean DT classifiers (Barceló et al., 2020; Audemard et al., 2021), and so the generalization to non-boolean DT classifiers is unclear. Moreover, recent work (Choi et al., 2020) outlines logical encodings of decision trees, but that is orthogonal to the work reported in this paper. It should be underscored that, in contrast with our own earlier work (Izza et al., 2020; Huang et al., 2021b), this paper highlights path explanations, both abductive and contrastive. In addition, there has been work on applying explainable AI (XAI) to decision trees (Lundberg et al., 2020), but with the focus of improving the quality of local (heuristic) explanations, where the goal is to relate a local approximate model against a reference model; hence there is no immediate relationship with the formal explanations studied in this paper. Similarly, one could consider exploiting non-formal model-agnostic explainers. There is a large body of work on non-formal model-agnostic XAI approaches (Adadi and Berrada, 2018; Montavon et al., 2018; Samek et al., 2019; Guidotti et al., 2019; Samek et al., 2021; Tjoa and Guan, 2021; Holzinger et al., 2022, 2020; Ras et al., 2022). Well-known examples include LIME (Ribeiro et al., 2016), SHAP (Lundberg and Lee, 2017) and Anchor (Ribeiro et al., 2018), for model-agnostic explanations, and sensitivity analysis (Simonyan et al., 2014) and LRP (Bach et al., 2015) in the case of saliency maps for neural networks. However, such model-agnostic explainers offer no guarantees of rigor. More importantly, the explanations computed by (non-formal) model-agnostic explainers can be unsound (Ignatiev et al., 2019c; Camburu et al., 2019; Ignatiev, 2020; Dimanov et al., 2020). In addition, the running times of these non-formal tools are not on par with the algorithms proposed in this paper, being in general orders of magnitude slower. There is recent work

21. Example references include (Nijssen and Fromont, 2007; Bessiere et al., 2009; Nijssen and Fromont, 2010; Bertsimas and Dunn, 2017; Verwer and Zhang, 2017; Narodytska et al., 2018; Verwer and Zhang, 2019; Hu et al., 2019; Avellaneda, 2019, 2020; Verhaeghe et al., 2020a; Aglin et al., 2020a; Lin et al., 2020; Janota and Morgado, 2020; Hu et al., 2020; Verhaeghe et al., 2020b; Aglin et al., 2020b; Demirovic and Stuckey, 2021; Schidler and Szeider, 2021; Ordyniak and Szeider, 2021; Shati et al., 2021; Alos et al., 2021; Demirovic et al., 2022; McTavish et al., 2022).

on approximate explanations with probabilistic guarantees (Wäldchen et al., 2021; Wang et al., 2021; Blanc et al., 2021), with initial results for DTs reported in (Izza et al., 2021).

8. Conclusions

This paper investigates path explanation redundancy in decision trees, i.e. the existence of features that are irrelevant for the prediction associated with a given path. In addition, the paper also shows that the computation of irredundant path explanations in DTs is tightly related with recent work on computing abductive explanations (Ignatiev et al., 2019a). Furthermore, the paper proposes several algorithms for computing path explanations, all of which run in worst-case polynomial time.

The experimental results offer conclusive evidence supporting the following claim: *DTs consistently exhibit path explanation redundancy, which is often significant, not only in the number of paths exhibiting explanation redundancy, but also in the number of features that can be deemed explanation-redundant for the path.* This claim is supported by the analysis of DTs used in a comprehensive range of examples taken from textbooks and survey papers, some of which dating back to the inception of well-known tree-learning algorithms (Breiman et al., 1984; Quinlan, 1993). This claim is also supported by the analysis of the DTs learned with well-known tree-learning algorithms, one of which explicitly targets interpretability (Bertsimas and Dunn, 2017; IAI, 2020). Finally, the claim is supported by the analysis of publicly available DTs generated with so-called optimal (sparse) decision tree learners (Verwer and Zhang, 2019; Hu et al., 2019; Lin et al., 2020; Rudin et al., 2021), which also explicitly target interpretability.

More importantly, the experimental results presented in this paper do *not* endorse the case made in recent research that DTs are intrinsically interpretable, concretely when interpretability correlates with succinctness of explanations. However, these same experimental results support making the alternative case: *that DTs require being explained in practice, that explaining DTs is computationally efficient in theory and in practice, and that explaining DTs must be a stepping stone for deploying ML in high-risk and safety-critical applications.* Moreover, we conjecture that the same case can be made for other classifiers that can be related with DTs in terms of the efficiency of computing explanations. Furthermore, we observe that the informal concept of interpretability in the case of DTs is justified not by the intrinsic property of explanations of DTs being succinct and irreducible, but by the fact that rigorous explanations can be efficiently computed, both in the case of DTs and possibly in the case of other related classifiers.

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, and by the H2020-ICT38 project COALA “Cognitive Assisted agile manufacturing for a Labor force supported by trustworthy Artificial intelligence”. This work received comments from several colleagues, including N. Asher, M. Cooper, E. Hebrard, X. Huang, C. Mencía, N. Narodytska, R. Passos and J. Planes.

References

Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160.

Aglin, G., Nijssen, S., and Schaus, P. (2020a). Learning optimal decision trees using caching branch-and-bound search. In *AAAI*, pages 3146–3153.

Aglin, G., Nijssen, S., and Schaus, P. (2020b). PyDL8.5: a library for learning optimal decision trees. pages 5222–5224.

Alos, J., Ansotegui, C., and Torres, E. (2021). Learning optimal decision trees using MaxSAT. *CoRR*, abs/2110.13854.

Alpaydin, E. (2014). *Introduction to machine learning*. MIT press.

Alpaydin, E. (2016). *Machine Learning: The New AI*. MIT Press.

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. (2016). Machine bias. propublica.org, https://bit.ly/3d8UZkJ.

Appuswamy, R., Franceschetti, M., Karamchandani, N., and Zeger, K. (2011). Network coding for computing: Cut-set bounds. *IEEE Trans. Inf. Theory*, 57(2):1015–1030.

Arenas, M., Baez, D., Barceló, P., Pérez, J., and Subercaseaux, B. (2021). Foundations of symbolic languages for model interpretability. In *NeurIPS*.

Arif, M. F., Mencía, C., and Marques-Silva, J. (2015). Efficient MUS enumeration of horn formulae with applications to axiom pinpointing. In *SAT*, pages 324–342.

Arrieta, A. B., Rodríguez, N. D., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 58:82–115.

Asher, N., Paul, S., and Russell, C. (2021). Fair and adequate explanations. In *CD-MAKE*, pages 79–97.

Audemard, G., Bellart, S., Bounia, L., Koriche, F., Lagniez, J., and Marquis, P. (2021). On the computational intelligibility of boolean classifiers. In *KR*, pages 74–86.

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

Avellaneda, F. (2019). Learning optimal decision trees from large datasets. *CoRR*, abs/1904.06314.

Avellaneda, F. (2020). Efficient inference of optimal decision trees. In *AAAI*, pages 3195–3202.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Bailey, J., Manoukian, T., and Ramamohanarao, K. (2003). A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns. In *ICDM*, pages 485–488.

Bandi, H. and Bertsimas, D. (2021). The price of diversity. *CoRR*, abs/2107.03900.

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

Belov, A., Lynce, I., and Marques-Silva, J. (2012). Towards efficient MUS extraction. *AI Commun.*, 25(2):97–116.

Berthold, M. R., Borgelt, C., Höppner, F., and Klawonn, F. (2010). *Guide to Intelligent Data Analysis - How to Intelligently Make Sense of Real Data*, volume 42 of *Texts in Computer Science*. Springer.

Bertsimas, D. and Dunn, J. (2017). Optimal classification trees. *Mach. Learn.*, 106(7):1039–1082.

Bertsimas, D., Dunn, J., Gibson, E., and Orfanoudaki, A. (2020a). Optimal survival trees. *CoRR*, abs/2012.04284.

Bertsimas, D., Dunn, J., and Mundru, N. (2019a). Optimal prescriptive trees. *INFORMS Journal on Optimization*, 1(2):164–183.

Bertsimas, D., Dunn, J., Pawlowski, C., Silberholz, J., Weinstein, A., Zhuo, Y. D., Chen, E., and Elfiky, A. A. (2018a). Applied informatics decision support tool for mortality predictions in patients with cancer. *JCO clinical cancer informatics*, 2:1–11.

Bertsimas, D., Dunn, J., Pawlowski, C., and Zhuo, Y. D. (2019b). Robust classification. *INFORMS Journal on Optimization*, 1(1):2–34.

Bertsimas, D., Dunn, J., Steele, D. W., Trikalinos, T. A., and Wang, Y. (2019c). Comparison of machine learning optimal classification trees with the pediatric emergency care applied research network head trauma decision rules. *JAMA pediatrics*, 173(7):648–656.

Bertsimas, D., Dunn, J., Velmahos, G. C., and Kaafarani, H. M. (2018b). Surgical risk is not linear: derivation and validation of a novel, user-friendly, and machine-learning-based predictive optimal trees in emergency surgery risk (potter) calculator. *Annals of surgery*, 268(4):574–583.

Bertsimas, D., Kung, J., Trichakis, N., Wang, Y., Hirose, R., and Vagefi, P. A. (2019d). Development and validation of an optimized prediction of mortality for candidates awaiting liver transplantation. *American Journal of Transplantation*, 19(4):1109–1118.

Bertsimas, D., Li, M. L., Paschalidis, I. C., and Wang, T. (2020b). Prescriptive analytics for reducing 30-day hospital readmissions after general surgery. *PloS one*, 15(9):e0238118.

Bertsimas, D., Masiakos, P. T., Mylonas, K. S., and Wiberg, H. (2019e). Prediction of cervical spine injury in young pediatric patients: an optimal trees artificial intelligence approach. *Journal of pediatric surgery*, 54(11):2353–2357.

Bertsimas, D., Orfanoudaki, A., and Weiner, R. B. (2019f). Personalized treatment for coronary artery disease patients: A machine learning approach. *CoRR*, abs/1910.08483.

Bertsimas, D., Orfanoudaki, A., and Weiner, R. B. (2020c). Personalized treatment for coronary artery disease patients: a machine learning approach. *Health Care Management Science*, 23(4):482–506.

Bertsimas, D., Pauphilet, J., Stevens, J., and Tandon, M. (2021). Predicting inpatient flow at a major hospital using interpretable analytics. *Manufacturing & Service Operations Management*.

Bertsimas, D. and Stellato, B. (2021). The voice of optimization. *Mach. Learn.*, 110(2):249–277.

Bertsimas, D. and Wiberg, H. (2020). Machine learning in oncology: Methods, applications, and challenges. *JCO Clinical Cancer Informatics*, 4:885–894.

Bessiere, C., Hebrard, E., and O'Sullivan, B. (2009). Minimising decision tree size as combinatorial optimisation. In *CP*, pages 173–187.

Bienvenu, M. (2009). Prime implicates and prime implicants: From propositional to modal logic. *J. Artif. Intell. Res.*, 36:71–128.

Biere, A., Heule, M., van Maaren, H., and Walsh, T., editors (2021). *Handbook of Satisfiability*. IOS Press.

Birnbaum, E. and Lozinskii, E. L. (2003). Consistent subsets of inconsistent systems: structure and behaviour. *J. Exp. Theor. Artif. Intell.*, 15(1):25–46.

Blanc, G., Lange, J., and Tan, L. (2021). Provably efficient, succinct, and precise explanations. In *NeurIPS*.

Blockeel, H. and Raedt, L. D. (1998). Top-down induction of first-order logical decision trees. *Artif. Intell.*, 101(1-2):285–297.

Boumazouza, R., Alili, F. C., Mazure, B., and Tabia, K. (2020). A symbolic approach for counterfactual explanations. In *SUM*, pages 270–277.

Boumazouza, R., Alili, F. C., Mazure, B., and Tabia, K. (2021). ASTERYX: A model-agnostic sat-based approach for symbolic and score-based explanations. In *CIKM*, pages 120–129.

Bramer, M. (2020). *Principles of Data Mining, 4th Edition*. Undergraduate Topics in Computer Science. Springer.

Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical science*, 16(3):199–231.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.

Breslow, L. A. and Aha, D. W. (1997). Simplifying decision trees: A survey. *Knowledge Eng. Review*, 12(1):1–40.

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

Camburu, O., Giunchiglia, E., Foerster, J., Lukasiewicz, T., and Blunsom, P. (2019). Can I trust the explainer? verifying post-hoc explanatory methods. *CoRR*, abs/1910.02065.

Cho, B.-J., Kim, K. M., Bilegsaikhan, S.-E., and Suh, Y. J. (2020). Machine learning improves the prediction of febrile neutropenia in korean inpatients undergoing chemotherapy for breast cancer. *Scientific reports*, 10(1):1–8.

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

Cook, S. A. (1971). The complexity of theorem-proving procedures. In Harrison, M. A., Banerji, R. B., and Ullman, J. D., editors, *STOC*, pages 151–158.

Cooper, M. C. and Marques-Silva, J. (2021). On the tractability of explaining decisions of classifiers. In Michel, L. D., editor, *CP*, pages 21:1–21:18.

Crama, Y. and Hammer, P. L. (2011). *Boolean Functions - Theory, Algorithms, and Applications*. Cambridge University Press.

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

Darwiche, A. and Hirth, A. (2020). On the reasons behind decisions. In *ECAI*, pages 712–720.

Darwiche, A. and Marquis, P. (2021). On quantifying literals in boolean logic and its applications to explainable AI. *J. Artif. Intell. Res.*

Demirovic, E., Lukina, A., Hebrard, E., Chan, J., Bailey, J., Leckie, C., Ramamohanarao, K., and Stuckey, P. J. (2022). Murtree: Optimal decision trees via dynamic programming and search. *J. Mach. Learn. Res.*, 23:26:1–26:47.

Demirovic, E. and Stuckey, P. J. (2021). Optimal decision trees for nonlinear metrics. In *AAAI*, pages 3733–3741.

Dillig, I., Dillig, T., McMillan, K. L., and Aiken, A. (2012). Minimum satisfying assignments for SMT. In *CAV*, pages 394–409.

Dimanov, B., Bhatt, U., Jamnik, M., and Weller, A. (2020). You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In *ECAI*, pages 2473–2480.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*.

Džeroski, S. and Lavrač, N., editors (2001). *Relational data mining*. Springer.

Eiter, T. and Gottlob, G. (1995). Identifying the minimal transversals of a hypergraph and related problems. *SIAM J. Comput.*, 24(6):1278–1304.

El Hechi, M. W., Maurer, L. R., Levine, J., Zhuo, D., El Moheb, M., Velmahos, G. C., Dunn, J., Bertsimas, D., and Kaafarani, H. M. (2021). Validation of the artificial intelligence-based predictive optimal trees in emergency surgery risk (potter) calculator in emergency general surgery and emergency laparotomy patients. *Journal of the American College of Surgeons*, 232(6):912–919.

Flach, P. A. (2012). *Machine Learning - The Art and Science of Algorithms that Make Sense of Data*. CUP.

Fletcher, S. and Islam, M. Z. (2019). Decision tree classification with differential privacy: A survey. *ACM Comput. Surv.*, 52(4):83:1–83:33.

Fredman, M. L. and Khachiyan, L. (1996). On the complexity of dualization of monotone disjunctive normal forms. *J. Algorithms*, 21(3):618–628.

Freitas, A. A. (2013). Comprehensible classification models: a position paper. *SIGKDD Explorations*, 15(1):1–10.

Gennatas, E. D., Friedman, J. H., Ungar, L. H., Pirracchio, R., Eaton, E., Reichmann, L. G., Interian, Y., Luna, J. M., Simone, C. B., Auerbach, A., et al. (2020). Expert-augmented machine learning. *Proceedings of the National Academy of Sciences*, 117(9):4571–4577.

Gorji, N. and Rubin, S. (2021). Sufficient reasons for classifier decisions in the presence of constraints. *CoRR*, abs/2105.06001.

Gorji, N. and Rubin, S. (2022). Sufficient reasons for classifier decisions in the presence of domain constraints. In *AAAI*.

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.

Hachtel, G. D. and Somenzi, F. (2006). *Logic synthesis and verification algorithms*. Springer.

Holzinger, A., Goebel, R., Fong, R., Moon, T., Müller, K., and Samek, W., editors (2022). *xxAI - Beyond Explainable AI - International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*, volume 13200 of *Lecture Notes in Computer Science*. Springer.

Holzinger, A., Saranti, A., Molnar, C., Biecek, P., and Samek, W. (2020). Explainable AI methods - A brief overview. In *xxAI*, pages 13–38.

Hu, H., Siala, M., Hebrard, E., and Huguet, M. (2020). Learning optimal decision trees with MaxSAT and its integration in AdaBoost. In *IJCAI*, pages 1170–1176.

Hu, X., Rudin, C., and Seltzer, M. I. (2019). Optimal sparse decision trees. In *NeurIPS*, pages 7265–7273.

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., Cooper, M. C., Asher, N., and Marques-Silva, J. (2022). Tractable explanations for d-DNNF classifiers. In *AAAI*.

Huang, X., Izza, Y., Ignatiev, A., and Marques-Silva, J. (2021b). On efficiently explaining graph-based classifiers. In *KR*, pages 356–367.

Hyafil, L. and Rivest, R. L. (1976). Constructing optimal binary decision trees is NP-complete. *Inf. Process. Lett.*, 5(1):15–17.

IAI (2020). Interpretable AI. https://www.interpretable.ai/.

Ignatiev, A. (2020). Towards trustable explainable AI. In *IJCAI*, pages 5154–5158.

Ignatiev, A., Cooper, M. C., Siala, M., Hebrard, E., and Marques-Silva, J. (2020a). Towards formal fairness in machine learning. In *CP*, pages 846–867.

Ignatiev, A., Izza, Y., Stuckey, P., and Marques-Silva, J. (2022). Using MaxSAT for efficient explanations of tree ensembles. In *AAAI*.

Ignatiev, A. and Marques-Silva, J. (2021). SAT-based rigorous explanations for decision lists. In *SAT*, pages 251–269.

Ignatiev, A., Marques-Silva, J., Narodytska, N., and Stuckey, P. J. (2021). Reasoning-based learning of interpretable ML models. In *IJCAI*, pages 4458–4465.

Ignatiev, A., Morgado, A., and Marques-Silva, J. (2018a). PySAT: A python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437.

Ignatiev, A., Narodytska, N., Asher, N., and Marques-Silva, J. (2020b). From contrastive to abductive explanations and back again. In *AIxIA*, pages 335–355.

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

Ignatiev, A., Narodytska, N., and Marques-Silva, J. (2019b). On relating explanations and adversarial examples. In *NeurIPS*, pages 15857–15867.

Ignatiev, A., Narodytska, N., and Marques-Silva, J. (2019c). On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509.

Ignatiev, A., Pereira, F., Narodytska, N., and Marques-Silva, J. (2018b). A SAT-based approach to learn explainable decision sets. In *IJCAR*, pages 627–645.

ITI (2020). Incremental Decision Tree Induction. https://www-lrn.cs.umass.edu/iti/.

Izza, Y., Ignatiev, A., and Marques-Silva, J. (2020). On explaining decision trees. *CoRR*, abs/2010.11034.

Izza, Y., Ignatiev, A., Narodytska, N., Cooper, M. C., and Marques-Silva, J. (2021). Efficient explanations with relevant sets. *CoRR*, abs/2106.00546.

Izza, Y. and Marques-Silva, J. (2021). On explaining random forests with SAT. In *IJCAI*, pages 2584–2591.

Janota, M. and Morgado, A. (2020). SAT-based encodings for optimal decision trees with explicit paths. In *SAT*, pages 501–518.

Jaumard, B. and Simeone, B. (1987). On the complexity of the maximum satisfiability problem for horn formulas. *Inf. Process. Lett.*, 26(1):1–4.

Karimi, A., Barthe, G., Schölkopf, B., and Valera, I. (2020). A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *CoRR*, abs/2010.04050. Accepted for publications at ACM Computing Surveys.

Karimi, A., Schölkopf, B., and Valera, I. (2021). Algorithmic recourse: from counterfactual explanations to interventions. In *FAccT*, pages 353–362.

Kavvadias, D. J. and Stavropoulos, E. C. (2005). An efficient algorithm for the transversal hypergraph generation. *J. Graph Algorithms Appl.*, 9(2):239–264.

Kelleher, J. D., Mac Namee, B., and D'arcy, A. (2020). *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies.* MIT Press.

Khachiyan, L., Boros, E., Elbassioni, K. M., and Gurvich, V. (2006). An efficient implementation of a quasi-polynomial algorithm for generating hypergraph transversals and its application in joint generation. *Discret. Appl. Math.*, 154(16):2350–2372.

Kotsiantis, S. B. (2013). Decision trees: a recent overview. *Artif. Intell. Rev.*, 39(4):261–283.

Lakkaraju, H., Bach, S. H., and Leskovec, J. (2016). Interpretable decision sets: A joint framework for description and prediction. In *KDD*, pages 1675–1684.

Liffiton, M. H., Previti, A., Malik, A., and Marques-Silva, J. (2016). Fast, flexible MUS enumeration. *Constraints An Int. J.*, 21(2):223–250.

Liffiton, M. H. and Sakallah, K. A. (2008). Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reason.*, 40(1):1–33.

Lin, J., Zhong, C., Hu, D., Rudin, C., and Seltzer, M. I. (2020). Generalized and scalable optimal sparse decision trees. In *ICML*, pages 6150–6160.

Lipton, Z. C. (2018). The mythos of model interpretability. *Commun. ACM*, 61(10):36–43.

Liu, X. and Lorini, E. (2021). A logic for binary classifiers and their explanation. In *CLAR*.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence*, 2(1):56–67.

Lundberg, S. M. and Lee, S. (2017). A unified approach to interpreting model predictions. In *NeurIPS*, pages 4765–4774.

Malfa, E. L., Michelmore, R., Zbrzezny, A. M., Paoletti, N., and Kwiatkowska, M. (2021). On guaranteed optimal robust explanations for NLP models. In *IJCAI*, pages 2658–2665.

Marques-Silva, J., Gerspacher, T., Cooper, M. C., Ignatiev, A., and Narodytska, N. (2020). Explaining naive bayes and other linear classifiers with polynomial time and delay. In *NeurIPS*.

Marques-Silva, J., Gerspacher, T., Cooper, M. C., Ignatiev, A., and Narodytska, N. (2021). Explanations for monotonic classifiers. In *ICML*, pages 7469–7479.

Marques-Silva, J., Heras, F., Janota, M., Previti, A., and Belov, A. (2013a). On computing minimal correction subsets. In *IJCAI*, pages 615–622.

Marques-Silva, J. and Ignatiev, A. (2022). Delivering trustworthy AI through formal XAI. In *AAAI*.

Marques-Silva, J., Ignatiev, A., Mencía, C., and Peñaloza, R. (2016). Efficient reasoning for inconsistent Horn formulae. In *JELIA*, pages 336–352.

Marques-Silva, J., Janota, M., and Belov, A. (2013b). Minimal sets over monotone predicates in boolean formulae. In *CAV*, pages 592–607.

Marques-Silva, J., Janota, M., and Mencía, C. (2017). Minimal sets on propositional formulae. problems and reductions. *Artif. Intell.*, 252:22–50.

Marques-Silva, J. and Mencía, C. (2020). Reasoning about inconsistent formulas. In *IJCAI*, pages 4899–4906.

Marquis, P. (1991). Extending abduction from propositional to first-order logic. In *FAIR*, pages 141–155.

Maurer, L. R., Bertsimas, D., Bouardi, H. T., El Hechi, M., El Moheb, M., Giannoutsou, K., Zhuo, D., Dunn, J., Velmahos, G. C., and Kaafarani, H. M. (2021). Trauma outcome predictor: An artificial intelligence interactive smartphone tool to predict outcomes in trauma patients. *Journal of Trauma and Acute Care Surgery*, 91(1):93–99.

McTavish, H., Zhong, C., Achermann, R., Karimalis, I., Chen, J., Rudin, C., and Seltzer, M. (2022). How smart guessing strategies can yield massive scalability improvements for sparse decision tree optimization. In *AAAI*.

Mencía, C., Ignatiev, A., Previti, A., and Marques-Silva, J. (2016). MCS extraction with sublinear oracle queries. In *SAT*, pages 342–360.

Mencía, C., Previti, A., and Marques-Silva, J. (2015). Literal-based MCS extraction. In *IJCAI*, pages 1973–1979.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81–97.

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

Minoux, M. (1988). LTUR: A simplified linear-time unit resolution algorithm for horn formulae and computer implementation. *Inf. Process. Lett.*, 29(1):1–12.

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

Molnar, C. (2020). *Interpretable Machine Learning*. Leanpub. http://tiny.cc/6c76tz.

Montavon, G., Samek, W., and Müller, K. (2018). Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.*, 73:1–15.

Moret, B. M. E. (1982). Decision trees and diagrams. *ACM Comput. Surv.*, 14(4):593–623.

Narodytska, N., Ignatiev, A., Pereira, F., and Marques-Silva, J. (2018). Learning optimal decision trees with SAT. In *IJCAI*, pages 1362–1368.

Narodytska, N., Shrotri, A. A., Meel, K. S., Ignatiev, A., and Marques-Silva, J. (2019). Assessing heuristic machine learning explanations with model counting. In *SAT*, pages 267–278.

Nijssen, S. and Fromont, É. (2007). Mining optimal decision trees from itemset lattices. In *KDD*, pages 530–539.

Nijssen, S. and Fromont, É. (2010). Optimal constraint-based decision tree induction from itemset lattices. *Data Min. Knowl. Discov.*, 21(1):9–51.

Ong, C. J., Orfanoudaki, A., Zhang, R., Caprasse, F. P. M., Hutch, M., Ma, L., Fard, D., Balogun, O., Miller, M. I., Minnig, M., Saglam, H., Prescott, B., Greer, D. M., Smirnakis, S., and Bertsimas, D. (2020). Machine learning and natural language processing methods to identify ischemic stroke, acuity and location from radiology reports. *PLoS One*, 15(6):e0234908.

OpenML (2020). OpenML: Machine learning, better, together. https://www.openml.org/.

Ordyniak, S. and Szeider, S. (2021). Parameterized complexity of small decision tree learning. In *AAAI*, pages 6454–6462.

Orfanoudaki, A., Chesley, E., Cadisch, C., Stein, B., Nouh, A., Alberts, M. J., and Bertsimas, D. (2020). Machine learning provides evidence that stroke risk is not linear: The non-linear framingham stroke risk score. *PloS one*, 15(5):e0232414.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830.

PennML (2020). Penn Machine Learning Benchmarks. https://github.com/EpistasisLab/pmlb.

Poole, D. and Mackworth, A. K. (2017). *Artificial Intelligence - Foundations of Computational Agents*. CUP.

Quinlan, J. R. (1986). Induction of decision trees. *Mach. Learn.*, 1(1):81–106.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan-Kaufmann.

Rago, A., Cocarascu, O., Bechlivanidis, C., Lagnado, D. A., and Toni, F. (2021). Argumentative explanations for interactive recommendations. *Artif. Intell.*, 296:103506.

Rago, A., Cocarascu, O., Bechlivanidis, C., and Toni, F. (2020). Argumentation as a framework for interactive explanations for recommendations. In *KR*, pages 805–815.

Ras, G., Xie, N., van Gerven, M., and Doran, D. (2022). Explainable deep learning: A field guide for the uninitiated. *J. Artif. Intell. Res.*, 73:329–396.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In *KDD*, pages 1135–1144.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *AAAI*, pages 1527–1535.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Rivest, R. L. (1987). Learning decision lists. *Mach. Learn.*, 2(3):229–246.

Rokach, L. and Maimon, O. Z. (2008). *Data mining with decision trees: theory and applications*. World scientific.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., and Zhong, C. (2021). Interpretable machine learning: Fundamental principles and 10 grand challenges. *CoRR*, abs/2103.11251. Accepted for publication in Statistics Surveys.

Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence - A Modern Approach*. Pearson Education.

Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., and Müller, K. (2021). Explaining deep neural networks and beyond: A review of methods and applications. *Proc. IEEE*, 109(3):247–278.

Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K., and Müller, K., editors (2019). *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer.

Schidler, A. and Szeider, S. (2021). SAT-based decision tree learning for large data sets. In *AAAI*, pages 3904–3912.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press.

Shati, P., Cohen, E., and McIlraith, S. A. (2021). SAT-based approach for learning optimal decision trees with non-binary features. In *CP*, pages 50:1–50:16.

Shi, W., Shih, A., Darwiche, A., and Choi, A. (2020). On tractable representations of binary neural networks. In *KR*, pages 882–892.

Shih, A., Choi, A., and Darwiche, A. (2018). A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, pages 5103–5111.

Shih, A., Choi, A., and Darwiche, A. (2019). Compiling bayesian network classifiers into decision graphs. In *AAAI*, pages 7966–7974.

Siers, M. J. and Islam, M. Z. (2021). Class imbalance and cost-sensitive decision trees: A unified survey based on a core similarity. *ACM Trans. Knowl. Discov. Data*, 15(1):4:1–4:31.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*.

Slaney, J. (2014). Set-theoretic duality: A fundamental feature of combinatorial optimisation. In *ECAI*, pages 843–848.

Sosa-Hernández, V. A., Monroy, R., Medina-Pérez, M. A., Loyola-González, O., and Herrera, F. (2021). A practical tutorial for decision tree induction: Evaluation measures for candidate splits and opportunities. *ACM Comput. Surv.*, 54(1):18:1–18:38.

Tjoa, E. and Guan, C. (2021). A survey on explainable artificial intelligence (XAI): toward medical XAI. *IEEE Trans. Neural Networks Learn. Syst.*, 32(11):4793–4813.

UCI (2020). UCI Machine Learning Repository. https://archive.ics.uci.edu/ml.

Ustun, B., Spangher, A., and Liu, Y. (2019). Actionable recourse in linear classification. In *FAT*, pages 10–19.

Utgoff, P. E., Berkman, N. C., and Clouse, J. A. (1997). Decision tree induction based on efficient tree restructuring. *Mach. Learn.*, 29(1):5–44.

Valdes, G., Luna, J. M., Eaton, E., Simone, C. B., Ungar, L. H., and Solberg, T. D. (2016). MediBoost: a patient stratification tool for interpretable decision making in the era of precision medicine. *Scientific reports*, 6(1):1–8.

Venkatasubramanian, S. and Alfano, M. (2020). The philosophical basis of algorithmic recourse. In *FAT*, pages 284–293.

Verhaeghe, H., Nijssen, S., Pesant, G., Quimper, C., and Schaus, P. (2020a). Learning optimal decision trees using constraint programming. *Constraints An Int. J.*, 25(3-4):226–250.

Verhaeghe, H., Nijssen, S., Pesant, G., Quimper, C., and Schaus, P. (2020b). Learning optimal decision trees using constraint programming (extended abstract). In *IJCAI*, pages 4765–4769.

Verwer, S. and Zhang, Y. (2017). Learning decision trees with flexible constraints and objectives using integer optimization. In *CPAIOR*, pages 94–103.

Verwer, S. and Zhang, Y. (2019). Learning optimal classification trees using a binary linear program formulation. In *AAAI*, pages 1625–1632.

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.

Wang, E., Khosravi, P., and den Broeck, G. V. (2021). Probabilistic Sufficient Explanations. In *IJCAI*, pages 3082–3088.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2017). *Data Mining*. Morgan Kaufmann.

Wolf, L., Galanti, T., and Hazan, T. (2019). A formal approach to explainability. In *AIES*, pages 255–261.

Wu, X. and Kumar, V., editors (2009). *The top ten algorithms in data mining*. CRC press.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A. F. M., Liu, B., Yu, P. S., Zhou, Z., Steinbach, M. S., Hand, D. J., and Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37.

Zhou, Z. (2021). *Machine Learning*. Springer.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press.