ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230384

From Decision Trees to Explained Decision Sets

Xuanxiang Huang[a;*] **and Joao Marques-Silva**[b]

[a]University of Toulouse, France
[b]IRIT, CNRS, Toulouse, France
ORCiD ID: Xuanxiang Huang https://orcid.org/0000-0002-3722-7191,
Joao Marques-Silva https://orcid.org/0000-0002-6632-3086

Abstract. Recent work demonstrated that path explanation redundancy is ubiquitous in decision trees, i.e. most often paths in decision trees include literals that are redundant for explaining a prediction. The implication of this result is that decision trees must be explained. Nevertheless, there are applications of DTs where running an explanation algorithm is impractical. For example, in settings that are time or power constrained, running software algorithms for explaining predictions would be undesirable. Although the explanations for paths in DTs do not generally represent themselves a decision tree, this paper shows that one can construct a decision set from some of the decision tree explanations, such that the decision set is not only explained, but it also exhibits a number of properties that are critical for replacing the original decision tree.

1 Introduction

Recent years witness groundbreaking advances in machine learning (ML) [4]. However, these advances raise concerns about whether the operation of complex ML models can be understood and trusted by human decision makers. Such concerns are at the core of ongoing efforts on understanding the operation of ML models, e.g. stability of predictions and rationale for predictions. Moreover, the ongoing efforts towards understanding the rationale of predictions broadly represent the burgeoning field of eXplainable Artificial Intelligence (XAI). XAI is characterized by a number of different approaches for tackling the problem of explaining ML models [18]. One important approach is referred to as *intrinsic interpretability* [41], where so-called interpretable models are used, and where the model is itself the explanation.

Decision trees (DTs) have long been deemed interpretable [5], and are at the core of proposals for the use of interpretable models, especially in high-risk applications of ML [44, 45]. Explanations in DTs are apparently very easy to derive, in that the literals in the path consistent with the input represent the explanation. Unfortunately, recent work demonstrated that paths in DTs can be arbitrarily redundant when compared with logically sufficient (abductive) explanations for a prediction [29]. The main consequence of these recent results is that, similarly to other ML models, decision trees must be explained. (It should be noted that this consequence hinges on the assumption that explanation succinctness matters. However, succinctness *must* always matter when it comes to explainability, since otherwise one could argue that the input to the ML model would suffice as an explanation.) Redundancy has also been observed in other so-called interpretable models, including decision lists [38].

There exist very efficient polynomial-time algorithms for computing abductive explanations in DTs [29]. However, one immediate question is whether one can remove explanation redundancy from paths, so that decision makers have immediate access to the actual explanations. (Also, in some settings, the iterated computation of explanations might be unrealistic, due to constraints on available resources.) Unfortunately, the removal of redundancy breaks the structure of DTs. In addition, it is known that the family of DTs that do not exhibit explanation redundancy is very restricted [29].

Since mapping a DT to an explanation-irredundant DT is unachievable, this paper proposes a different solution. Concretely, the paper proposes an algorithm for mapping a DT into a decision set (DS), but such that the resulting DS exhibits a number of critical properties, which ensures it operates as a DT. Since DSs are unordered, they can display a number of fundamental issues. Firstly, DSs can exhibit overlap, and thus may not even compute a classification function. Secondly, for DSs that exhibit no overlap, the classification function may not be total, i.e. for some inputs there is no prediction. In this case, the use of a default rule requires special handing, so that the default rule is only used when no other rule applies. Lastly, DSs require being explained, and explanations for DSs are harder to compute than for DTs [22, 2]. Furthermore, the paper indirectly proposes a practical solution to the abstract goal of *intrinsic interpretability* [44, 41, 46], where the classifier is itself the explanation. Indeed, the algorithm proposed in this paper offers a solution to deliver a classifier where the explanation can be extracted by manual inspection from the classifier. The experimental results validate the scalability of the proposed algorithm, and offer comprenhensive evidence to the quality of the obtained DSs, with a key metric being the total number of literals used for explaining the DT paths.

A generalization of (exact) abductive explanations are probabilistic (abductive) explanations [48, 3, 26, 27], which aim at providing decision makers with shorter explanations (which are easier to grasp) and which, albeit not as rigorous, still offer strong probabilistic guarantees of rigor. As an additional contribution, and in the case of probabilistic explanations, the paper shows that the properties of the DSs obtained from DTs no longer hold. As a result, the paper outlines a simple, albeit less compact, solution that can be employed in the case of probabilistic explanations.

The paper is organized as follows. The definitions and notations used throughout the paper are introduced in Section 2, which also briefly overviews related work. Section 3 develops the algorithm for

*

mapping a DT into an (explained) DS, proves the key properties of the resulting DS, and investigates the limitations of the algorithm in the case of probabilistic explanations. Section 4 presents experimental results that confirm the properties of DSs obtained from explaining DTs. The results confirm the explained DSs offer significantly more compact (and subset-minimal) explanations than the explanations obtained from the paths in the original DTs. Section 5 concludes the paper.

2 Preliminaries

2.1 Basic Concepts

Classification problems. In this paper, a classification problem is defined on a set of features $\mathcal{F} = \{1, \ldots, m\}$ and a set of classes $\mathcal{K} = \{c_1, \ldots, c_K\}$. Each feature $i \in \mathcal{F}$ takes values from a domain $\mathbb{D}_i$, where domains can be categorical or ordinal. Feature space is the cartesian product of the domains of the features, $\mathbb{F} = \mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_m$. A classifier $\mathcal{M}$ realizes a non-constant classification function $\kappa : \mathbb{F} \to \mathcal{K}$. An instance is a pair $(\mathbf{v}, c)$, such that $\mathbf{v} \in \mathbb{F}$, $c \in \mathcal{K}$, and $c = \kappa(\mathbf{v})$. Finally, we associate a tuple $(\mathcal{F}, \mathbb{F}, \mathcal{K}, \kappa)$ with each classifier.

Throughout this paper, we assume that a classifier is total, i.e. κ is a total function. (Recall that a function is total if it is defined on all points of its domain. Furthermore, and because κ is a function, there is a single value associated with $\kappa(\mathbf{v})$ for each $\mathbf{v} \in \mathbb{F}$; otherwise κ would have to be defined as a relation, and not as a function.)

Decision trees (DTs). A DT $\mathcal{T}$ is a directed acyclic graph $G = (V, E)$. V is partitioned into a set N of non-terminal nodes, and a set T of terminal nodes. With the exception of the root of $\mathcal{T}$, all nodes have one incoming edge. The terminal nodes have no outgoing edges, and each is associated with a class from $\mathcal{K}$. The non-terminal nodes are associated with a single feature $i \in \mathcal{F}$ (i.e. univariate DTs), and the outgoing edges are associated with sets that partition the domain of i. In this paper, the domain of each feature is partitioned by using literals of the form $x_i = d_i$, with $d_i \in \mathbb{D}_i$, or $x_i \in \mathbb{E}_i$, with $\mathbb{E}_i \subseteq \mathbb{D}_i$. (The set notation represents the more general setting proposed in [29]. Also, literals of the form $x_i = d_i$ can easily be transformed into $x_i \in \{d_i\}$.) Moreover, each node of V is assigned a number (usually 1 is assigned to the root). Paths in the DT are represented as a sequence of numbers, e.g. $\mathcal{P} = \langle r_1, r_2, \ldots, r_n \rangle$, such that each pair (r_j, r_{j+1}) denotes an edge of $\mathcal{T}$. The set of paths of $\mathcal{T}$ is denoted by $\mathbb{P}$ (where the dependency on $\mathcal{T}$ is omitted for simplicity). Given a path $\mathcal{P}_k$, the features tested in the non-terminal nodes of $\mathcal{P}_k$ are represented by $\Phi(\mathcal{P}_k)$. Similarly, the features *not* tested along $\mathcal{P}_k$ are denoted by $\Psi(\mathcal{P}_k)$.

Also, for a path $\mathcal{P}_k$ of $\mathcal{T}$, and a set of features $\mathcal{X} \subseteq \Phi(\mathcal{P}_k) \subseteq \mathcal{F}$, $\Lambda(\mathcal{P}_k, \mathcal{X})$ denotes the set of literals associated with the features in $\mathcal{X}$ along path $\mathcal{P}_k$. (The definition of Λ accounts for situations where a feature is tested more than once, but we will not delve into that in this paper. Concretely, for each feature $i \in \Phi(\mathcal{P}_k)$, we have literals $x_i \in \mathbb{E}_i$, where $\mathbb{E}_i \subseteq \mathbb{D}_i$ is the intersection of the sets in each of the literals of $\mathcal{P}_k$ on feature i. Furthermore, since each node partitions the domain of the feature, then no two paths can be consistent for the same point in feature space. In addition, for any path, it is assumed that there exists at least one point $\mathbf{v}$ in feature space for which the path's literals are consistent with $\mathbf{v}$. Finally, it is assumed that DTs are organized such that the computed classification function is total. (Evidently, DTs can be envisioned for which κ is not total [29, page 270], or for which κ is not a function, but it is instead a relation [29], e.g. when node domain splits do not form a partition.)

Decision sets (DSs). A decision set is a set of *unordered* rules [39, 40, 33] of the form:

$$\text{IF} \quad \tau \quad \text{THEN} \quad \kappa(\cdot) = c \tag{1}$$

where the rule's condition (τ) is generally assumed to be a conjunction of literals, defined on the features of the classification problem. The semantics is that when τ is true, then the rule is said to *fire*, and the prediction is c. It should be noted that, since the rules are unordered, any rule for which the condition is true will fire. For some of the examples, we will use a more compact notation for rules, of the form:

$$\tau \rightarrowtail c \tag{2}$$

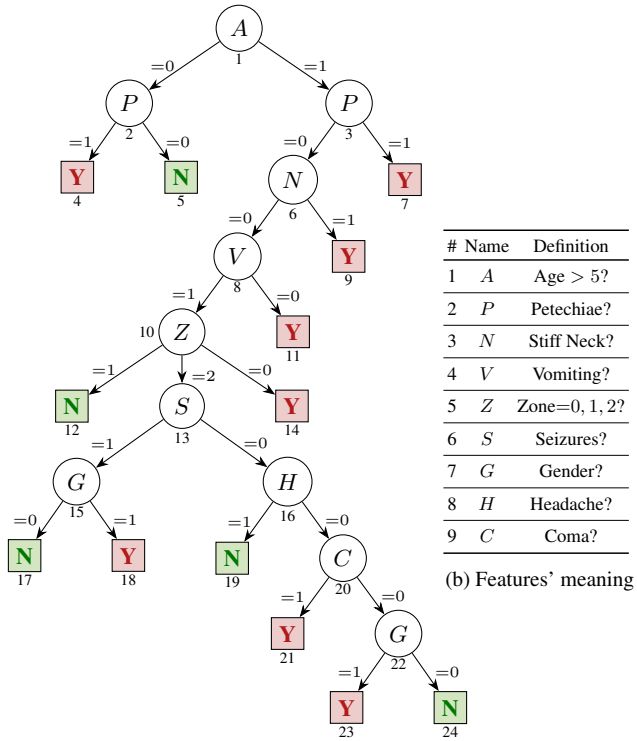
This more compact representation has the same interpretation as before, i.e. if the condition τ is true, then the prediction is c. Boolean literals in the conditions of rules will be represented by variables, e.g. x, or their negations, e.g. $\neg x$ or $\overline{x}$. (For the more complex examples, we will opt for the more compact notation, i.e. $\overline{x}$.)

DSs are distinguished from decision lists (DLs) [43] in that DLs are ordered, i.e. impose an order among the rules. Although DSs and DLs are generally considered to be interpretable [33, 25, 44, 41, 45, 46], there is recent evidence to the contrary [22, 38]. Besides the need for being explained, DSs exhibit a number of additional issues. First, there may exist overlap between rules predicting different classes, i.e. two or more rules that fire and predict different classes. In such a situation, the classifier does not compute a classification function. Removing overlap in general is believed to be computationally hard [25]. One proposed solution [33] considers the fairly restricted case of eliminating overlap *only* on training data, while giving no guarantees of non-overlap beyond training data. One alternative is to allow for the classification function not to be total, or to introduce a default rule [25], i.e. a catch-all rule with no condition, with the semantics that it fires only when the other rules do not fire.

Running examples. Throughout the paper, we will use the following two running examples.

Example 1. The first running example is the DT of Figure 1, which is adapted from [34]. The DT serves to diagnose the most severe case of meningitis, Meningococcal Disease (MD), without invasive tests. Clearly, $\mathcal{F} = \{1, \ldots, 9\}$, $\mathcal{K} = \{\mathbf{Y}, \mathbf{N}\}$, $\mathbb{D}_i = \{0, 1\}$ for $i = \{1, 2, 3, 4, 6, 7, 8, 9\}$, and $\mathbb{D}_5 = \{0, 1, 2\}$. (Observe that `Age` is ordinal (integer or real), but we only test whether the value is greater than 5.) Moreover, we will consider the instance $(\mathbf{v}, c) = ((A = 1, P = 0, N = 0, V = 1, Z = 0, S = 0, H = 0, C = 0, G = 1), \mathbf{Y})$. The paths predicting $\mathbf{Y}$ are numbered $\mathcal{P}_1, \ldots, \mathcal{P}_8$. The paths predicting $\mathbf{N}$ are numbered $\mathcal{Q}_1, \ldots, \mathcal{Q}_5$. The paths and their numberings are obtained from an in-order traversal of the tree. For example $\mathcal{P}_1 = \langle 1, 2, 4 \rangle$, $\mathcal{Q}_1 = \langle 1, 2, 5 \rangle$, $\mathcal{Q}_3 = \langle 1, 3, 6, 8, 10, 13, 15, 17 \rangle$, $\mathcal{Q}_5 = \langle 1, 3, 6, 8, 10, 13, 16, 20, 22, 24 \rangle$, $\mathcal{P}_5 = \langle 1, 3, 6, 8, 10, 14 \rangle$, and $\mathcal{P}_7 = \langle 1, 3, 6, 9 \rangle$. For $\mathcal{P}_5$, $\Phi(\mathcal{P}_5) = \{1, 2, 3, 4, 5\}$, where the mapping of features is as shown in Figure 1, i.e. feature 1 is A, feature 2 is P, and so on. In addition, and also for $\mathcal{P}_5$, $\Lambda(\mathcal{P}_5, \{1, 4, 5\}) = \{(A), (\overline{V}), (Z = 0)\}$.

Example 2. The second running example is shown in Figure 2. This second running example will be used to illustrate the computation of probabilistic abductive explanations. The figure also shows the truth table for the DT, and for each row in the table, the number of points in feature space consistent with that row.

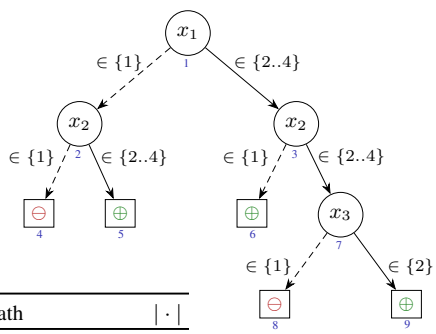


(a) Decision tree

#	Name	Definition
1	A	Age > 5?
2	P	Petechiae?
3	N	Stiff Neck?
4	V	Vomiting?
5	Z	Zone=0, 1, 2?
6	S	Seizures?
7	G	Gender?
8	H	Headache?
9	C	Coma?

(b) Features' meaning

Figure 1. First example DT, adapted from [34]



x_1	x_2	x_3	$\kappa(\mathbf{x})$	Path	$\lvert\cdot\rvert$
1	1	1	$\ominus$	$\mathcal{Q}_1 = \langle 1, 2, 4\rangle$	1
1	1	2	$\ominus$	$\mathcal{Q}_1$	1
1	2..4	1	$\oplus$	$\mathcal{P}_1 = \langle 1, 2, 5\rangle$	3
1	2..4	2	$\oplus$	$\mathcal{P}_1$	3
2..4	1	1	$\oplus$	$\mathcal{P}_2 = \langle 1, 3, 6\rangle$	3
2..4	1	2	$\oplus$	$\mathcal{P}_2 = \langle 1, 3, 6\rangle$	3
2..4	2..4	1	$\ominus$	$\mathcal{Q}_2 = \langle 1, 3, 7, 8\rangle$	9
2..4	2..4	2	$\oplus$	$\mathcal{P}_3 = \langle 1, 3, 7, 9\rangle$	9

(a) Decision tree

(b) Truth table

Figure 2. Second example DT, adapted from [27, 26]

Logic-based explainability. We adopt a formal definition of explanation, as studied in recent works [47, 24, 37, 36].

Given an instance $(\mathbf{v}, c)$, an explanation problem $\mathcal{E}$ is a tuple $(\mathcal{M}, (\mathbf{v}, c))$. Moreover, a weak abductive explanation (WAXp) is a set of features $\mathcal{X} \subseteq \mathcal{F}$ which, if assigned the values dictated by $\mathbf{v}$, then the prediction is c. Formally,

$$\mathsf{WeakAXp}(\mathcal{X}; \mathcal{M}, (\mathbf{v}, c)) := \forall(\mathbf{x} \in \mathbb{F}).\ (\wedge_{i\in\mathcal{X}} x_i = v_i) \rightarrow (\kappa(\mathbf{x}) = c) \quad (3)$$

(where the parametrization on $\mathcal{M}$ and $(\mathbf{v}, c)$ is shown.) Moreover, an abductive explanation (AXp) is a subset-minimal weak AXp:

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

(where the parametrization on $\mathcal{M}$ and $(\mathbf{v}, c)$ is left implicit; we will do this henceforth.)

Because the definition of WAXp is monotonic [24, 37, 36], then AXp's can be computed more efficiently:

$$\mathsf{AXp}(\mathcal{X}) := \mathsf{WeakAXp}(\mathcal{X}) \wedge \forall(i \in \mathcal{X}).\neg\,\mathsf{WeakAXp}(\mathcal{X} \setminus \{i\}) \quad (5)$$

These latter definitions are at the core of algorithms for computing AXp's. Besides abductive explanations, contrastive explanations can be formally defined [23]. These will not be used in this paper.

In this paper we use the restriction of AXp's to the case when features must be taken from the path. Such AXp's are referred to as *path AXp's* [29]. Throughout the paper, path AXp's are AXp's, but where the features that can be included in the explanation must exist in the path.

Example 3. For the running example (see Figure 1), and the instance $((A = 1, P = 0, N = 0, V = 1, Z = 0, S = 0, H = 0, C = 0, G = 1), \mathbf{Y})$, this point corresponds to the path $\langle 1, 3, 6, 8, 10, 14\rangle$. We can show that one AXp is $\{A, Z\}$ (technically, we should write $\{1, 5\}$).

To offer a more detailed insight into the process of computing this path AXp, one possible computation is summarized next. While the we will argue that the order of features $\{S, H, C, G\}$ does not matter, the remaining features are analyzed in order $\langle A, P, N, V, Z\rangle$. (Depending on the explanation problem, the order features may or may not matter.)

1. As the features in $\{S, H, C, G\}$ do not appear in the path, we can assign any value to these features. As a result, during the computation of this path's AXp's, these features are not taken into consideration.
2. Let feature A take any possible value from its domain. In this case, we can find a point $(A = 0, P = 0, N = 0, V = 1, Z = 0, S = 0, H = 0, C = 0, G = 1)$ that makes the path $\langle 1, 2, 5\rangle$ consistent, which predicts a different class $\mathbf{N}$. Thus, this violates the definition of path AXp. Hence, feature A must be *fixed* to the value 1.
3. Let feature P take any possible value from its domain. In this case, we cannot find a point in the feature space that makes consistent some path predicting the different class $\mathbf{N}$. As a result, feature P can be declared *free*, allowing it to take any value from its domain.
4. For the same reason, features in $\{N, V\}$ can also be freed, i.e. no path predicting the different class $\mathbf{N}$ can be made consistent when these features are allowed to take any values from their domains.
5. Finally, let feature Z take any possible value from its domain. In this case, we can find a point $(A = 1, P = 0, N = 0, V = 1, Z = 1, S = 0, H = 0, C = 0, G = 1)$ that makes the path $\langle 1, 3, 6, 8, 10, 12\rangle$ consistent, which predicts the different class $\mathbf{N}$. As before, this violates the definition of path AXp. Hence, feature Z must be fixed to the value 0.
6. In summary, we obtain the path AXp $\{A, Z\}$.

Thus, we can confidently state the following rule, representing a suf-

ficient condition for predicting $\mathbf{Y}$,

$$\text{IF} \quad (\text{Age} > 5) \land (\text{Zone} = 0) \quad \text{THEN} \quad \kappa(\cdot) = \mathbf{Y}$$

Using the more compact notation proposed earlier, we could also write, $A \land Z = 0 \;\rightarrowtail\; \mathbf{Y}$. The use of explanations allows identifying possible model learning issues with the example DT; this is further discussed elsewhere [36].

There has been rapid progress in logic-based explainability in recent years, which is overview for example in [37, 36] and the references therein.

Probabilistic abductive explanations (PAXp's). Building on earlier work [48, 1, 26, 27], we define *weak probabilistic* AXp (or weak PAXp) $\mathcal{X} \subseteq \mathcal{F}$ as a set of fixed features for which the probability of predicting the correct class c, for points consistent with the values of $\mathcal{X}$ in $\mathbf{v}$, exceeds $\delta > 0$, with $c = \kappa(\mathbf{v})$. Thus, $\mathcal{X} \subseteq \mathcal{F}$ is a weak PAXp if the following predicate holds true,

$$\begin{aligned}&\mathsf{WeakPAXp}(\mathcal{X}; \mathbb{F}, \kappa, \mathbf{v}, c, \delta)\\ &\quad := \Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = c \,|\, \mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}}) \geq \delta \qquad (6)\\ &\quad := \frac{|\{\mathbf{x} \in \mathbb{F} : \kappa(\mathbf{x}) = c \land (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}|}{|\{\mathbf{x} \in \mathbb{F} : (\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}})\}|} \geq \delta\end{aligned}$$

(Where the restriction of $\mathbf{x}$ to the variables with indices in $\mathcal{X}$ is represented by $\mathbf{x}_{\mathcal{X}} = (x_i)_{i \in \mathcal{X}}$. Concretely, the notation $\mathbf{x}_{\mathcal{X}} = \mathbf{v}_{\mathcal{X}}$ represents the constraint $\land_{i \in \mathcal{X}} x_i = v_i$.) The condition above means that the fraction of the number of points predicting the target class and consistent with the fixed features (represented by $\mathcal{X}$), given the total number of points in feature space consistent with the fixed features, must exceed δ. We can adapt (4) to define a PAXp given the definition of WPAXp. Since the definition of weak PAXp (see Eq. (6)) is non-monotonic, then the computation of PAXp's cannot be simplified [26, 27] using (5) [26, 27].

For DTs with categorical features, and for each pick of fixed features, one can compute the conditional probability in polynomial-time [26, 27]. For the purposes of this paper, we will use the truth table of Figure 2.

Example 4. For the DT of Figure 2, let us consider the instance $(\mathbf{v}, c) = ((1, 1, 1), \ominus)$ and $\mathcal{X} = \{1, 2\}$. Then, the number of points where $x_1 = 1 \land x_2 = 1$ is 2. Moreover, for all those points, $\kappa(\cdot) = \ominus$. Thus, $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \ominus \,|\, \mathbf{x}_{\{1,2\}} = \mathbf{v}_{\{1,2\}}) = 1$. Moreover, when $(\mathbf{v}, c) = ((2, 2, 2), \oplus)$ and $\mathcal{X} = \{3\}$, the number of points where $x_3 = 2$ is 16 and, among these, the number of points for which $\kappa(\cdot) = \oplus$ is 15. Thus, $\Pr_{\mathbf{x}}(\kappa(\mathbf{x}) = \oplus \,|\, \mathbf{x}_{\{3\}} = \mathbf{v}_{\{3\}}) = {}^{15}/_{16}$. Finally, with $\delta = 0.9$, it is the case that $\{1, 2\}$ is a WPAXp for (any instance of) $\mathcal{Q}_1$, and $\{3\}$ is a WPAXp for (any instance of) $\mathcal{P}_3$. It is simple to show that both WPAXp's are PAXp's [26, 27].

2.2 Related Work

Decision trees. As indicated earlier, it is generally assumed that DTs compute total functions, but this may not always be the case [29, page 270]. Without exception, tree induction algorithms guarantee that the resulting DT computes a classification function. However, it is possible to force a greedy tree induction algorithm to generate a DT that does not compute a total function. Nevertheless, deciding whether a DT computes a total function can be formulated as a decision problem, and answered with an automated reasoner. In the rest of the paper, we assume that such checking has been performed, and so DTs are assumed to compute a total function.

Despite the recent interest in computing explanations for DTs [28, 3, 19, 2, 29, 1], there seems to be no simple way to remove the redundancy from DT paths. Observe that the set of explanations associated with paths in a DT most often does not represent a DT, and attempts at constructing a DT from such explanations would necessarily re-introduce redundancy.

Decision sets & lists. Decision sets and lists have been studing since the 1960s [39], with extensive work in the 80s and 90s [40, 6, 7, 8, 9, 11]. The learning DLs and DSs is still an ongoing theme of research [12, 13, 33, 20, 25]. As noted above, DSs exhibit a number of limitations, the most important of which being overlap between rules predicting different classes. There exist solutions which guarantee that overlap is non-existing [25], but the computed classification function is either not total, or require the use of a default rule with a dedicated semantics.

The use of a default rule with a dedicated semantics complicates interpretability or approaches for computing explanations. The alternative solution, i.e. allowing for the classification function not to be total, is also not desirable. To the best of our knowledge, there is no solution for learning a decision set that produces DSs that exhibit no overlap, require no default rule, and which offer explanations by inspection (i.e. guarantee that there is no need for computing explanations).

Despite being considered interpretable models, DTs, DLs and DSs have been shown to require the computation of explanations [22, 29, 38], most often because of explanation redundancy.

3 From Decision Trees to (Explained) Decision Sets

To the best of our knowledge, there is no simple way to remove redundancy from a DT such that some DT can be reconstructed. As a result, one solution is to consider removing redundancy from a DT such that a different ML model is obtained. However, one key requirement for such ML model is that it must allow for explanations to be easily extracted, i.e. no algorithm is to be executed. The next section shows one basic approach to obtain such an ML model. Afterwards, we discuss extensions to the basic approach, their limitations, and alternative solutions.

3.1 Mapping a DT into a DS

This section develops an algorithm which, given a DT computing a total classification function, creates a DS with the following key properties:

1. The DS does not include a default rule;
2. The DS does not exhibit overlap (i.e. it computes a classification function);
3. The DS computes a total classification function; and
4. Each rule is a path AXp of the original DT.

Given the above properties, the DSs obtained with the algorithm described below will be referred to as *explained decision sets*.

The above properties are critically important, since the created DS does not exhibit *any* of the issues that are problematic for existing implementations of DSs. Furthermore, for any point in feature space, if some rule $\mathcal{R}_j$ fires, then the condition of the rule represents a path AXp of the original DT, i.e. there is *no* need for computing AXp's.

Algorithm 1 represents the proposed solution for constructing a DS starting from a DT. As can be observed, for each path in the DT, the algorithm computes one AXp. This AXp is then used for constructing a decision rule, using the literals obtained from the literals

Algorithm 1 Converting DT to DS

Input: Decision Tree $\mathcal{T}$ with classification function κ

function DT2DS($\mathcal{T}$)
  $\mathcal{S} \leftarrow \emptyset$ ▷ $\mathcal{S}$: DS to be constructed
  $\mathbb{P} \leftarrow$ AllPaths($\mathcal{T}$) ▷ $\mathbb{P}$: set of all paths in $\mathcal{T}$
  while $\mathbb{P} \neq \emptyset$ **do**
    $\mathcal{P}_k \leftarrow$ PickPath($\mathbb{P}$) ▷ $\mathcal{P}_k$: some path not yet explained
    $\mathbb{P} \leftarrow \mathbb{P} \setminus \{\mathcal{P}_k\}$
    $\mathcal{X} \leftarrow$ FindPathAXp($\mathcal{P}_k$) ▷ E.g. algorithms from [29]
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\text{IF } \wedge_{l \in \mathcal{A}(\mathcal{P}_k, \mathcal{X})} l \text{ THEN } \kappa(\cdot) = c\}$
  $\mathcal{S} \leftarrow$ RemoveDuplicateRules($\mathcal{S}$)
  return $\mathcal{S}$

$$
\begin{aligned}
&\mathrm{R}_{01}\colon \text{IF } [P] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathrm{R}_{02}\colon \text{IF } [\overline{A} \land \overline{P}] \text{THEN } \kappa(\cdot) = \mathbf{N}\\
&\mathrm{R}_{03}\colon \text{IF } [\overline{P} \land \overline{N} \land V \land Z = 1] \text{ THEN } \kappa(\cdot) = \mathbf{N}\\
&\mathrm{R}_{04}\colon \text{IF } [\overline{P} \land \overline{N} \land V \land Z = 2 \land S \land \overline{G}] \text{ THEN } \kappa(\cdot) = \mathbf{N}\\
&\mathrm{R}_{05}\colon \text{IF } [A \land Z = 2 \land S \land G] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathrm{R}_{06}\colon \text{IF } [\overline{P} \land \overline{N} \land V \land Z = 2 \land \overline{S} \land H] \text{ THEN } \kappa(\cdot) = \mathbf{N}\\
&\mathrm{R}_{07}\colon \text{IF } [A \land Z = 2 \land \overline{S} \land \overline{H} \land C] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathrm{R}_{08}\colon \text{IF } [A \land Z = 2 \land \overline{H} \land G] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathrm{R}_{09}\colon \text{IF } [\overline{P} \land \overline{N} \land V \land Z = 2 \land \overline{C} \land \overline{G}] \text{ THEN } \kappa(\cdot) = \mathbf{N}\\
&\mathrm{R}_{10}\colon \text{IF } [A \land Z = 0] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathrm{R}_{11}\colon \text{IF } [A \land \overline{V}] \text{ THEN } \kappa(\cdot) = \mathbf{Y}\\
&\mathrm{R}_{12}\colon \text{IF } [A \land N] \text{ THEN } \kappa(\cdot) = \mathbf{Y}
\end{aligned}
\tag{7}
$$

included in the AXp. In the end, duplicate rules are removed. The algorithm used for computing one path AXp, i.e. FindPathAXp, can be any of the algorithms proposed in earlier work [28, 29].

Example 5. Table 1 summarizes the execution of Algorithm 1 on the example DT of Figure 1. Each row lists: (a) the list of path nodes; (b) the features in the explanation when the path represents the explanation; (c) the condition of the rule that would be obtained in such a situation; (d) the features in some path AXp; and (e) the condition of the rule obtained from the features in the AXp. In total, the DT has 13 paths, and so Table 1 summarizes the algorithm's execution for each of the 13 paths. As can be observed, the last row creates a rule which is a duplicate, and so it will not be added to the DS. As a result, the DS consists of 12 rules (i.e. obtained from the first 12 executions of the algorithm's main loop). These are shown in (7). The order of the rules can be any, since the result is a decision set. The order shown in (7) is taken from the order in which paths are analyzed in Table 1, with duplicate rules removed.

As will be demonstrated in the next section, each rule is itself an abductive explanation (of the original DT), the DS computes a function (i.e. it exhibits *no* overlap), and the computed function is total. More importantly, as an be observed in Table 1, while the path literals for the DT total 75 literals, the explanations obtained from the DS total 44 literals, representing a reduction of more than 40% on the total number of literals used in explanations.

Extensions. The proposed algorithm leaves some flexibility on how to compute each AXp. One solution is to compute one subset-minimal AXp, since there are polynomial-time algorithms in the case of DTs [28, 19, 29]. Alternatively, one can consider computing cardinality-minimal AXp's, thus obtaining shortest explanations for each path. It is well-known that computing one cardinality-minimal AXp is NP-hard [3], but with a decision problem in NP. Thus, given proposed Horn encodings [29], a cardinality-minimal AXp can be computed by solving Horn Maximum Satisfiability (MaxSAT).

There has been recent work on inferring and exploiting constraints on the inputs when computing abductive explanations [17, 51]. It is also immediate to account for constraints on the inputs when computing explanations. Thus, Algorithm 1 can be used to produce an explained DS that takes input constraints into account.

Finally, we should observe that for each path in the DT there can be more than one path AXp. Indeed, in the worst-case, the number of path AXp's can be exponential on the number of features. As a result, the proposed algorithm can be adapted to allow for the (restricted) navigation of the space of AXp's for each path, thus enabling a human decision maker to select which AXp to associate with each path. It should be noted that in the case of the DT from Figure 1, and given the AXp's computed in Example 5, none of the paths exhibits more than one AXp.

Algorithm's complexity. The complexity of Algorithm 1 is linear on the complexity of computing abductive explanations. For plain subset-minimal AXp's, Algorithm 1 runs in polynomial-time, since there exist polynomial-time algorithms for computing one AXp [28, 29].

When the DS is to be constructed from cardinality-minimal AXp's, Algorithm 1 computes one such explanation a number of times that is linear with the nodes in the DT. However, computing one smallest AXp in the case of DTs is NP-hard [3]. Moreover, the algorithms for computing cardinality-minimal AXp's will require at most a logarithmic number of calls to an NP oracle in the worst-case. In the case of cardinality-minimal AXp's for DTs, a DS is obtained solving Horn MaxSAT a number of times that grows with the number of (terminal) nodes in the DT. Finally, navigation of the space of AXp's will also impact complexity, depending on how many AXp's are to be enumerated.

3.2 Properties of Explained Decision Sets

As suggested in earlier sections, we now prove the key properties of the DS created with Algorithm 1. First, we prove that each rule condition in the DS maps to a path AXp of the original DT. Then, we prove that Algorithm 1 creates a DS that computes a (classification) function, i.e. there is no overlap between rules with different predictions. Finally, we prove that the DS computes a total function, i.e. for any point in feature space, at least one rule fires.

Proposition 1. The conditions of each rule in the DS represent a path AXp of the original DT.

Proof. This result is an immediate consequence of Algorithm 1, since each rule in the DS is obtained from a path AXp of the original DT. □

As a consequence of Proposition 1, each rule corresponds to some path(s) of the original DT. Furthermore, and under the hypothesis that the original DT computes a total function, we can prove the following results. The first result ensures that the DS computes a function (i.e. there is no overlap). The second results ensures that the computed function is total (i.e. there is a prediction for every point in feature space).

Proposition 2. The constructed DS is non-overlapping.

Proof. Suppose the constructed DS is overlapping, which means

Table 1: Summary of the execution of Algorithm 1 for the DT in Figure 1. The rule extracted from each AXp is added to the DS, with duplicates removed.

Path	Path Xp	Rule condition from path	Path AXp	Rule condition from path AXp
$\langle 1,2,4\rangle$	$\{1,2\}$	$\overline{A}\land P$	$\{2\}$	P
$\langle 1,2,5\rangle$	$\{1,2\}$	$\overline{A}\land\overline{P}$	$\{1,2\}$	$\overline{A}\land\overline{P}$
$\langle 1,3,6,8,10,12\rangle$	$\{1,2,3,4,5\}$	$A\land\overline{P}\land\overline{N}\land V\land Z=1$	$\{2,3,4,5\}$	$\overline{P}\land\overline{N}\land V\land Z=1$
$\langle 1,3,6,8,10,13,15,17\rangle$	$\{1,2,3,4,5,6,7\}$	$A\land\overline{P}\land\overline{N}\land V\land Z=2\land S\land\overline{G}$	$\{2,3,4,5,6,7\}$	$\overline{P}\land\overline{N}\land V\land Z=2\land S\land\overline{G}$
$\langle 1,3,6,8,10,13,15,18\rangle$	$\{1,2,3,4,5,6,7\}$	$A\land\overline{P}\land\overline{N}\land V\land Z=2\land S\land G$	$\{1,5,6,7\}$	$A\land Z=2\land S\land G$
$\langle 1,3,6,8,10,13,16,19\rangle$	$\{1,2,3,4,5,6,8\}$	$A\land\overline{P}\land\overline{N}\land V\land Z=2\land\overline{S}\land H$	$\{2,3,4,5,6,8\}$	$\overline{P}\land\overline{N}\land V\land Z=2\land\overline{S}\land H$
$\langle 1,3,6,8,10,13,16,20,21\rangle$	$\{1,2,3,4,5,6,8,9\}$	$A\land\overline{P}\land\overline{N}\land V\land Z=2\land\overline{S}\land\overline{H}\land C$	$\{1,5,6,8,9\}$	$A\land Z=2\land\overline{S}\land\overline{H}\land C$
$\langle 1,3,6,8,10,13,16,20,22,23\rangle$	$\{1,2,3,4,5,6,7,8,9\}$	$A\land\overline{P}\land\overline{N}\land V\land Z=2\land\overline{S}\land\overline{H}\land\overline{C}\land G$	$\{1,5,7,8\}$	$A\land Z=2\land\overline{H}\land G$
$\langle 1,3,6,8,10,13,16,20,22,24\rangle$	$\{1,2,3,4,5,6,7,8,9\}$	$A\land\overline{P}\land\overline{N}\land V\land Z=2\land\overline{S}\land\overline{H}\land\overline{C}\land\overline{G}$	$\{2,3,4,5,7,9\}$	$\overline{P}\land\overline{N}\land V\land Z=2\land\overline{C}\land\overline{G}$
$\langle 1,3,6,8,10,14\rangle$	$\{1,2,3,4,5\}$	$A\land\overline{P}\land\overline{N}\land V\land Z=0$	$\{1,5\}$	$A\land Z=0$
$\langle 1,3,6,8,11\rangle$	$\{1,2,3,4\}$	$A\land\overline{P}\land\overline{N}\land\overline{V}$	$\{1,4\}$	$A\land\overline{V}$
$\langle 1,3,6,9\rangle$	$\{1,2,3\}$	$A\land\overline{P}\land N$	$\{1,3\}$	$A\land N$
$\langle 1,3,7\rangle$	$\{1,2\}$	$A\land P$	$\{2\}$	P

there exist at least two rules with non-contradicting *conditions* that predict two *different classes*. In such a case, there would exist a point in the feature space that is consistent with two paths of the original DT leading to two different classes, which contradicts with the hypothesis that the DT computes a total function. Hence, DS is non-overlapping. □

Proposition 3. The constructed DS is total.

Proof. If the constructed DS is not total, it implies the existence of a point in the feature space that is not consistent with any rule in the DS. As a result, this point is also not consistent with any path of the original DT, which contradicts the hypothesis that the original DT is total. Thus, the constructed DS is total. □

3.3 Limitations & Solutions

The basic algorithm proposed in the previous section (see Algorithm 1) allows mapping DTs to (explained) DSs when an path AXp is associated with each path. This section investigates limitations of the proposed algorithm and outlines possible solutions.

Probabilistic explanations. There has been recent work on computing rigorous probabilistic explanations [48, 30, 31, 32, 1, 26]. Similarly to computing AXp's, Algorithm 1 could be instrumented to compute probabilistic AXp's (PAXp's) or locally minimal PAXp's (LmPAXp's) [26]. Thus, a DS would be constructed using different notions of probabilistic explanations instead of plain abductive explanations. Unfortunately, in this case the resulting DS would not respect the properties established in Section 3.2, in that overlap is no longer guaranteed not to exist. The following example illustrates the issue of overlap that PAXp's can induce.

Example 6. We use the example DT in Figure 2, and the WPAXp's studied in Example 4 to convey the issues raised by probabilistic explanations.

Let $\delta = 0.9$. For the path $\langle 1,2,4\rangle$, the (only) PAXp is $\{1,2\}$, which would yield the rule $x_1\in\{1\}\land x_2\in\{1\}\ \rightarrowtail\ \ominus$. Moreover, for the path $\langle 1,3,7,9\rangle$ a PAXp is $\{3\}$, which would yield the rule $x_3\in\{2\}\ \rightarrowtail\ \oplus$. It is plain to conclude that there exists overlap between the two rules.

DT annotation. In the case of probabilistic explanations, there is a simple, but less compact, approach to pre-compute the explanations of each path. The solution is to annotate the terminal nodes of DTs with the computed explanations. In the case of probabilistic explanations, it suffices to compute a (Lm)PAXp for each path, and then annotate the terminal node of the path with that explanation. It is plain that the DT guarantees the non-existence of overlap. Moreover, annotating the terminal nodes will have no effect on whether the DT computes a total function.

There are downsides to this solution, which Algorithm 1 addresses. First, DT annotation yields a less compact representation of both the ML model and a possible universe of explanations. Second, a human decision maker will be expected to be able to relate computed explanations with the paths the explanations are associated with.

4 Experiments

This section presents experimental results that evaluate the practical efficiency of the proposed approach for mapping a Decision Tree into a Decision Set. It is important to emphasize that the experiment did not consider computing cardinality-minimal AXp's for extracting rules and constructing DS.

Experimental setup. The evaluation comprises 44 datasets that are originate from Penn ML Benchmarks [42]. The datasets used in the paper consist of features with either categorical or ordinal domains (i.e. integer or real-valued). The number of features ranges from 6 to 240, while the number of classes varies from 2 to 26, with an average of 47 features and 5 classes. Each dataset is divided into training and testing sets, with 80% of the data used for training and 20% used for testing. DTs are learned using Orange3 [10], the maximum depth set to 9 while the minimal test accuracy set to 70%. It is worth noting that

Table 2: The table shows statistics for datasets, decision trees and the resulting decision sets. The table includes the number of features (m) and the number of classes (K) for each dataset. For each DT, the table reports the tree depth (D), the number of nodes (#N), test accuracy (A%), the number of paths ($|\mathbb{P}|$), and $|\Lambda(\mathbb{P})|$, which is the total number of literals in all tree paths. For the resulting DS, the table reports the number of rules ($|\mathcal{S}|$) and $|\Lambda(\mathcal{S})|$, which is the total number of literals in all rules. Moreover, the table reports the maximum (max R%) and average (avg. R%) path redundancy ratio, which refers to the proportion of literals that can be removed from a decision path to convert it into a decision rule over the total number of literals included in that path. The last column reports the runtime (in seconds) for converting DT into DS.

Dataset	m	K	DT					DS				Runtime (s)
			D	#N	A%	$\|\mathbb{P}\|$	$\|\Lambda(\mathbb{P})\|$	$\|\mathcal{S}\|$	$\|\Lambda(\mathcal{S})\|$	max R%	avg. R%	
adult	14	2	9	151	86.0	76	639	43	212	62.5	37.8	0.21
analcatdata_authorship	70	4	5	25	95.3	13	51	13	49	20.0	3.1	0.02
ann_thyroid	21	3	8	25	99.7	13	66	11	33	60.0	33.2	0.02
breast_cancer_wisconsin	30	2	6	13	92.1	7	27	7	18	50.0	26.0	0.01
car_evaluation	21	4	9	117	96.2	59	478	46	299	50.0	17.8	0.13
chess	36	2	9	43	99.4	22	155	21	94	66.7	32.5	0.05
churn	20	2	9	93	92.8	47	303	43	178	60.0	34.8	0.09
coil2000	85	2	9	117	93.9	59	467	44	254	66.7	23.9	0.18
connect_4	42	3	9	689	72.4	345	3021	259	1915	44.4	15.0	1.55
corral	6	2	5	27	100.0	14	56	6	12	60.0	46.1	0.01
dermatology	34	6	7	17	95.9	9	42	9	40	14.3	3.2	0.02
dna	180	3	9	149	92.5	75	563	73	468	50.0	14.9	0.22
ionosphere	34	2	7	29	93.0	15	73	13	38	71.4	36.3	0.03
kr_vs_kp	36	2	9	39	98.9	20	135	20	89	66.7	31.0	0.05
letter	16	26	9	499	73.9	250	2122	229	1514	50.0	21.6	0.87
mfeat_factors	216	10	9	155	84.8	78	579	78	544	25.0	5.5	0.21
mfeat_fourier	76	10	9	217	75.0	109	826	107	692	44.4	13.5	0.30
mfeat_karhunen	64	10	9	253	77.2	127	980	125	816	42.9	14.8	0.34
mfeat_pixel	240	10	9	185	87.0	93	658	93	637	22.2	2.9	0.26
mfeat_zernike	47	10	9	293	74.2	147	1187	145	1076	33.3	7.7	0.46
mofn_3_7_10	10	2	7	93	97.4	47	294	46	174	57.1	38.9	0.17
molecular_biology_promoters	57	2	4	15	72.7	8	26	8	21	50.0	16.7	0.01
movement_libras	90	15	9	127	73.6	64	414	64	401	28.6	2.9	0.16
mux6	6	2	6	55	100.0	28	141	15	46	50.0	34.7	0.04
optdigits	64	10	9	353	89.3	177	1433	177	1367	25.0	4.3	0.61
pendigits	16	10	9	235	94.0	118	903	115	776	44.4	10.6	0.33
ring	20	2	9	107	83.6	54	394	48	219	77.8	37.1	0.32
satimage	36	6	9	333	86.2	167	1351	154	980	66.7	20.9	0.47
sonar	60	2	7	27	78.6	14	68	14	52	42.9	19.4	0.02
soybean	35	18	9	79	84.4	40	260	38	212	33.3	10.6	0.07
spambase	57	2	9	141	92.0	71	509	68	379	62.5	20.1	0.19
spect	22	2	9	77	79.6	39	268	34	170	66.7	23.2	0.09
spectf	44	2	9	29	85.7	15	91	15	53	66.7	41.1	0.04
splice	60	3	9	91	89.2	46	319	46	257	50.0	20.1	0.12
texture	40	11	9	167	90.4	84	660	81	552	33.3	12.3	0.24
threeOf9	9	2	8	63	100.0	32	185	23	75	57.1	34.6	0.06
tic_tac_toe	9	2	9	113	91.7	57	389	48	240	55.6	23.8	0.14
tokyo1	44	2	9	27	94.3	14	91	12	46	44.4	30.9	0.03
twonorm	20	2	9	351	83.8	176	1409	171	1232	44.4	9.3	0.69
vote	16	2	8	19	93.1	10	50	10	35	40.0	25.0	0.02
waveform_21	21	3	9	343	75.6	172	1375	154	1026	44.4	15.5	0.56
waveform_40	40	3	9	391	75.0	196	1596	180	1244	44.4	14.3	0.70
wdbc	30	2	6	17	89.5	9	40	9	28	50.0	25.2	0.01
xd6	9	2	9	79	100.0	40	243	30	90	66.7	44.2	0.10

Orange3 is capable of handling features with categorical or ordinal domains. Furthermore, a prototype of the proposed algorithm was implemented in Python, and is publicly available in the repository [1]. Finally, the experiments were performed on a MacBook Pro with a 6-Core Intel Core i7 2.6 GHz processor with 16 GByte RAM, running macOS Monterey.

Results. Table 2 summarizes the results of mapping DTs to DS. For the learned DTs, 34 out of the 44 DTs have depth more than 7, 33 out of the 44 DTs achieve a test accuracy of over 80%. Moreover, the number of nodes in the learned DTs ranges from 13 to 689, with 22 out of 44 models having more than 100 nodes. Besides, the number of tree paths for DTs varies from 7 to 345, with an average of 74 paths. The total number of literals in tree paths varies from 26 to 3021, with an average of 566 literals.

Through the transformation of DTs into DSes, we have successfully reduced the number of rules and literals required for decision-making. The number of rules for the DSes ranged from 6 to 259, with an average of 66 rules. Besides, the total number of literals in the rules varied from 12 to 1915, with an average of 423 literals. On average, roughly 10% tree paths are redundant, and roughly 25% literals are redundant.

More specifically, for the *adult*, *car_evaluation*, *coil2000*, *connect_4*, and *corral* datasets, we observe that 43.4%, 22%, 25.4%, 24.9%, and 57.1%, respectively, of the tree paths are redundant. Although the tree paths for the *breast_cancer_wisconsin*, *molecular_biology_promoters*, *sonar*, *spectf*, and *wdbc* datasets are not redundant, a non-negligible ratio of redundant literals exists within the tree paths. Specifically, for these datasets, the maximal ratio of redundant literals in the tree paths is 50%, 50%, 42.9%, 66.7%, and 50%, respectively, while the average ratios of redundant literals in the tree paths are 26%, 16.7%, 19.4%, 41.1%, and 25.2%, respectively. An additional observation is that in 34 out of the 44 DTs, the maximal ratio of redundant literals in the tree paths is at least 40%, and in some cases, can exceed 70% (e.g., datasets *ionosphere* and *ring*). Additionally, in 24 out of 44 DTs, the average ratio of redundant literals in the tree paths is at least 20%. However, there are indeed some DTs where the average ratio of redundant literals in the tree paths is small, such as *analcatdata_authorship* and *dermatology*.

Finally, the table shows that the runtime for mapping DTs into DSes is negligible, as indicated in the last column. This can be attributed to the algorithm used, which leverages a polynomial-time method for extracting one path AXp from each tree path.

5 Conclusions

This paper demonstrates that (non-explained) decision trees can be mapped onto (explained) decision sets, such that the obtained decision sets exhibit all the key properties of decision trees. The paper proposes an algorithm that constructs an explained decision set starting from a decision tree. In contrast with other algorithms for constructing decision sets proposed in the recent past [33, 35, 25, 16, 14, 49, 15, 50, 21], the algorithm proposed in this paper ensures that the resulting decision sets compute a total function, such that the condition of each rule is the explanation for the prediction when the rule fires. Given the existing proposals for intrinsic interpretability [44, 41, 46], the algorithm proposed in this paper offers a solution to deliver a classifier where the explanation is extracted, by inspection, from the classifier. The experiments demonstrate not only the scalability of the proposed algorithm, but also the significantly tighter representations that explainable decision sets achieve.

Future work will investigate the impact on the size of the explained DT of computing smallest explanations for each path in the DT. Similarly, additional heuristics for selecting the explanations to consider for each path in the DT will be investigated. For example, one could give preference to picking explanations that match already picked explanations, so as to minimize the total number of rules in the DS.

[1] https://github.com/XuanxiangHuang/dtree2dset

Acknowledgements

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”. JMS also acknowledges the incentive provided by the ERC who, by not funding this research nor a handful of other grant applications between 2012 and 2022, has had a lasting impact in framing the research presented in this paper.

References

[1] Marcelo Arenas, Pablo Barceló, Miguel Romero, and Bernardo Subercaseaux, ‘On computing probabilistic explanations for decision trees’, in *NeurIPS*, (2022).
[2] Gilles Audemard, Steve Bellart, Louenas Bounia, Frédéric Koriche, Jean-Marie Lagniez, and Pierre Marquis, ‘On the computational intelligibility of boolean classifiers’, in *KR*, pp. 74–86, (2021).
[3] Pablo Barceló, Mikaël Monet, Jorge Pérez, and Bernardo Subercaseaux, ‘Model interpretability through the lens of computational complexity’, in *NeurIPS*, (2020).
[4] Yoshua Bengio, Yann LeCun, and Geoffrey E. Hinton, ‘Deep learning for AI’, *Commun. ACM*, **64**(7), 58–65, (2021).
[5] Leo Breiman, ‘Statistical modeling: The two cultures’, *Statistical science*, **16**(3), 199–231, (2001).
[6] Peter Clark and Tim Niblett, ‘The CN2 induction algorithm’, *Machine Learning*, **3**, 261–283, (1989).
[7] William W. Cohen, ‘Efficient pruning methods for separate-and-conquer rule learning systems’, in *IJCAI*, pp. 988–994, (1993).
[8] William W. Cohen, ‘Fast effective rule induction’, in *ICML*, pp. 115–123, (1995).
[9] William W. Cohen and Yoram Singer, ‘A simple, fast, and effictive rule learner’, in *AAAI*, pp. 335–342, (1999).
[10] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan, ‘Orange: Data mining toolbox in python’, *Journal of Machine Learning Research*, **14**, 2349–2353, (2013).
[11] Johannes Fürnkranz, ‘Separate-and-conquer rule learning’, *Artif. Intell. Rev.*, **13**(1), 3–54, (1999).
[12] Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrac, *Foundations of Rule Learning*, Cognitive Technologies, Springer, 2012.
[13] Johannes Fürnkranz and Tomás Kliegr, ‘A brief overview of rule learning’, in *RuleML*, pp. 54–69, (2015).
[14] Bishwamittra Ghosh, Dmitry Malioutov, and Kuldeep S. Meel, ‘Classification rules in relaxed logical form’, in *ECAI*, pp. 2489–2496, (2020).
[15] Bishwamittra Ghosh, Dmitry Malioutov, and Kuldeep S. Meel, ‘Efficient learning of interpretable classification rules’, *J. Artif. Intell. Res.*, **74**, 1823–1863, (2022).
[16] Bishwamittra Ghosh and Kuldeep S. Meel, ‘IMLI: an incremental framework for MaxSAT-based learning of interpretable classification rules’, in *AIES*, pp. 203–210, (2019).
[17] Niku Gorji and Sasha Rubin, ‘Sufficient reasons for classifier decisions in the presence of domain constraints’, in *AAAI*, (February 2022).
[18] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi, ‘A survey of methods for explaining black box models’, *ACM Comput. Surv.*, **51**(5), 93:1–93:42, (2019).

[19] Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, and Joao Marques-Silva, 'On efficiently explaining graph-based classifiers', in *KR*, pp. 356–367, (2021).

[20] Eyke Hüllermeier, Johannes Fürnkranz, Eneldo Loza Mencía, Vu-Linh Nguyen, and Michael Rapp, 'Rule-based multi-label classification: Challenges and opportunities', in *RuleML+RR*, pp. 3–19, (2020).

[21] Alexey Ignatiev, Edward Lam, Peter J. Stuckey, and João Marques-Silva, 'A scalable two stage approach to computing optimal decision sets', in *AAAI*, pp. 3806–3814, (2021).

[22] Alexey Ignatiev and Joao Marques-Silva, 'SAT-based rigorous explanations for decision lists', in *SAT*, pp. 251–269, (2021).

[23] Alexey Ignatiev, Nina Narodytska, Nicholas Asher, and Joao Marques-Silva, 'From contrastive to abductive explanations and back again', in *AIxIA*, pp. 335–355, (2020).

[24] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva, 'Abduction-based explanations for machine learning models', in *AAAI*, pp. 1511–1519, (2019).

[25] Alexey Ignatiev, Filipe Pereira, Nina Narodytska, and João Marques-Silva, 'A SAT-based approach to learn explainable decision sets', in *IJCAR*, pp. 627–645, (2018).

[26] Yacine Izza, Xuanxiang Huang, Alexey Ignatiev, Nina Narodytska, Martin C. Cooper, and João Marques-Silva, 'On computing probabilistic abductive explanations', *CoRR*, **abs/2212.05990**, (2022).

[27] Yacine Izza, Xuanxiang Huang, Alexey Ignatiev, Nina Narodytska, Martin C. Cooper, and João Marques-Silva, 'On computing probabilistic abductive explanations', *Int. J. Approx. Reason.*, **159**, 108939, (2023).

[28] Yacine Izza, Alexey Ignatiev, and Joao Marques-Silva, 'On explaining decision trees', *CoRR*, **abs/2010.11034**, (2020).

[29] Yacine Izza, Alexey Ignatiev, and João Marques-Silva, 'On tackling explanation redundancy in decision trees', *J. Artif. Intell. Res.*, **75**, 261–321, (2022).

[30] Yacine Izza, Alexey Ignatiev, Nina Narodytska, Martin C. Cooper, and João Marques-Silva, 'Efficient explanations with relevant sets', *CoRR*, **abs/2106.00546**, (2021).

[31] Yacine Izza, Alexey Ignatiev, Nina Narodytska, Martin C. Cooper, and João Marques-Silva, 'Provably precise, succinct and efficient explanations for decision trees', *CoRR*, **abs/2205.09569**, (2022).

[32] Yacine Izza and João Marques-Silva, 'On computing relevant features for explaining nbcs', *CoRR*, **abs/2207.04748**, (2022).

[33] Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec, 'Interpretable decision sets: A joint framework for description and prediction', in *KDD*, pp. 1675–1684, (2016).

[34] Viviane Maria Lelis, Eduardo Guzmán, and María-Victoria Belmonte, 'Non-invasive meningitis diagnosis using decision trees', *IEEE Access*, **8**, 18394–18407, (2020).

[35] Dmitry Malioutov and Kuldeep S. Meel, 'MLIC: A maxsat-based framework for learning interpretable classification rules', in *CP*, pp. 312–327, (2018).

[36] Joao Marques-Silva, 'Logic-based explainability in machine learning', in *Reasoning Web*, pp. 24–104, (2022).

[37] Joao Marques-Silva and Alexey Ignatiev, 'Delivering trustworthy AI through formal XAI', in *AAAI*, (2022).

[38] Joao Marques-Silva and Alexey Ignatiev, 'No silver bullet: interpretable ml models must be explained', *Frontiers in AI*, (2023). In press.

[39] Ryszard S. Michalski, 'On the quasi-minimal solution of the general covering problem', in *ISIP*, pp. 125–128, (1969).

[40] Ryszard S. Michalski, Igor Mozetic, Jiarong Hong, and Nada Lavrac, 'The multi-purpose incremental learning system AQ15 and its testing application to three medical domains', in *AAAI*, pp. 1041–1047, (1986).

[41] Christoph Molnar, *Interpretable Machine Learning*, Leanpub, 2020. http://tiny.cc/6c76tz.

[42] Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore, 'PMLB: a large benchmark suite for machine learning evaluation and comparison', *BioData Mining*, **10**(1), 36, (2017).

[43] Ronald L. Rivest, 'Learning decision lists', *Mach. Learn.*, **2**(3), 229–246, (1987).

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

[45] Cynthia Rudin, 'Why black box machine learning should be avoided for high-stakes decisions, in brief', *Nature Reviews Methods Primers*, **2**(1), 1–2, (2022).

[46] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong, 'Interpretable machine learning: Fundamental principles and 10 grand challenges', *Statistics Surveys*, **16**, 1–85, (2022).

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

[48] Stephan Wäldchen, Jan MacDonald, Sascha Hauch, and Gitta Kutyniok, 'The computational complexity of understanding binary classifier decisions', *J. Artif. Intell. Res.*, **70**, 351–387, (2021).

[49] Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, and Pierre Le Bodic, 'Computing optimal decision sets with SAT', in *CP*, ed., Helmut Simonis, pp. 952–970, (2020).

[50] Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, and Pierre Le Bodic, 'Learning optimal decision sets and lists with SAT', *J. Artif. Intell. Res.*, **72**, 1251–1279, (2021).

[51] Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, Nina Narodytska, and Joao Marques-Silva, 'Eliminating the impossible, whatever remains must be true: On extracting and applying background knowledge in the context of formal explanations', in *AAAI*, (2023).