Logic-Based Explainability in Machine Learning – A Glimpse of the Field

Antonio Morgado[a], Jordi Planes[b], Ramon Béjar[b], Joao Marques-Silva[c]

[a] *INESC-ID / IST Universidade Lisboa, R. Alves Redol 9 Lisboa, 1000-029, Portugal*
[b] *Univ. Lleida, C/ Jaume II, 69, Lleida, 25001, Spain*
[c] *ICREA, Univ. Lleida, C/ Jaume II, 69, Lleida, 25001, Spain*

Abstract

The ongoing remarkable progress in Machine Learning (ML) is not without drawbacks. It is well-known that the most promising ML models are inscrutable in their operation. The need to debug ML-based systems, but also the need to trust those systems, makes it crucial that human decision makers can validate, or even understand, the operation of such systems. In ML-based systems that impact humans, validation of the predictions made is absolutely paramount. The purpose of eXplainable Artificial Intelligence (XAI) is to explain the operation of complex ML models. The importance of XAI is underscored by its rapid growth, being generally considered a stepping stone for trustworthy AI. Unfortunately, the vast majority of work on XAI is based on informal approaches, that offer no guarantees of rigor. In contrast, recent years witnessed the emergence of rigorous, logic-based, explainability. Logic-based explainability offers absolute guarantees of rigor, and has served to demonstrate several misconceptions common in informal XAI. This paper offers an account of the emerging field of logic-based explainability.

Keywords:
Explainability, Machine Learning, Automated Reasoning, Logic-Based Explainability

---

Contents

1. Introduction

The ongoing progress in Artificial Intelligence (AI), especially in Machine Learning (ML), is widely regarded as monumental. Not only AI/ML have had recent profound impact, but are generally expected to have far-reaching societal impact in the near future. The immense success of AI/ML is not without downsides, and one of the most visible is arguably the lack of trust triggered by the opaque operation of complex models [23, 24].

This perceived lack of trust has motivated a number of efforts in recent years, aiming at delivering trustworthy AI. Examples of such efforts include the assessment of robustness and fairness, but also the emergence of eXplainable AI (XAI), which broadly represents solutions for aiding human decision-makers understanding the operation of complex black-box AI/ML models. The importance of XAI is underscored not only by existing and forthcoming regulation [17, 18], but also by an ever-increasing range of publications [22].

Methods of XAI can be broadly categorized as those targeting feature attribution and those targeting feature selection [59]. The goal of feature attribution is to assign a score to each feature, measuring the feature's contribution to a given prediction. Examples of feature attribution methods are LIME [63] and SHAP [49]. Both methods attempt to learn scores for the features, but while LIME is based on iterative sampling, SHAP is based on approximate computation of Shapley values [72]. Other well-known examples of XAI methods include variants of saliency maps [68].

In contrast, the purpose of feature selection is to identify a set of features that is sufficient for the prediction, i.e. if the set of features is fixed to their given values, then the prediction is known, independently of the values of the other features. An example of a feature selection method is ANCHOR [64], which similar to other methods is based on sampling.

Besides feature attribution and selection, another approach to explainability is intrinsic interpretability [66, 59, 67], where the ML model, due to being interpretable, serves as the explanation. For example, this is the case of decision trees [12], decision sets [45], or other graph-based ML models [27], which are considered to be naturally interpretable. In such cases, some sort of manual inspection of the model itself produces an explanation for the prediction. Despite the apparent simplicity of the models, recent works [43, 56] have demonstrated that even intrinsic interpretable models should be explained (at least in terms of obtaining shorter and succinct explanations).

Unfortunately, most of the previous work on XAI offers no guarantees of rigor. We refer to these as non-formal approaches to explainability. Non-formal approaches to explainability are beyond the scope of this paper.[1] In spite of the many proposed non-formal approaches to explainability, recent works have demonstrated several shortcomings of non-formal explanation approaches, which makes them inadequate in high-risk settings [38, 32, 61, 31, 52].

The alternative to non-formal XAI is the more recently proposed formal logic-based XAI (FXAI) [55, 50]. Formal XAI is based on rigorously defined (and so formal) explanations, ensuring a level of rigor that directly correlates with the logic languages used for representing ML models. Being logic-based by nature, FXAI relies in most cases on efficient automated reasoning tools for the computation of the explanations. Example of such tools include Boolean Satisfiability (SAT), and Satisfiability Modulo Theories (SMT) reasoners, both of which represent highly-successful instantiations of symbolic computation. This paper surveys FXAI, covering its core definitions, and outlining widely used algorithms. In contrast to earlier work, this paper opts to introduce a more general explainability framework, which illustrates that past work can be applied in a wider range of use cases.

The paper is organized as follows. Section 2 introduces the notations and definitions used throughout the paper. Section 3 defines formal explainability, and briefly surveys the progress observed in recent years. Section 4 details a more general explainability framework, where inputs of ML models may be unspecified, and classification functions are replaced by classification relations, thereby allowing more than one prediction to be considered when some inputs are unspecified. Section 5 outlines well-known FXAI algorithms but taking into account the generalized FXAI framework. Section 6 details a few case studies. Afterwards, Section 7 details well-known explainability queries, but also takes into consideration the generalized framework proposed in the paper. The paper concludes in Section 8.

2. Preliminaries

The paper assumes basic knowledge of computational complexity, namely the classes of decision problems P and NP [4], and their function variants. The paper also assumes basic knowledge of propositional logic, including the

[1]Detailed surveys on the topic can be found elsewhere [22, 59, 60, 68, 69, 70].

Boolean satisfiability (SAT) decision problem for propositional logic formulas in conjunctive normal form (CNF), and the use of SAT solvers as oracles. The interested reader is referred to textbooks on these topics [4, 9].

2.1. Classification Problems in ML

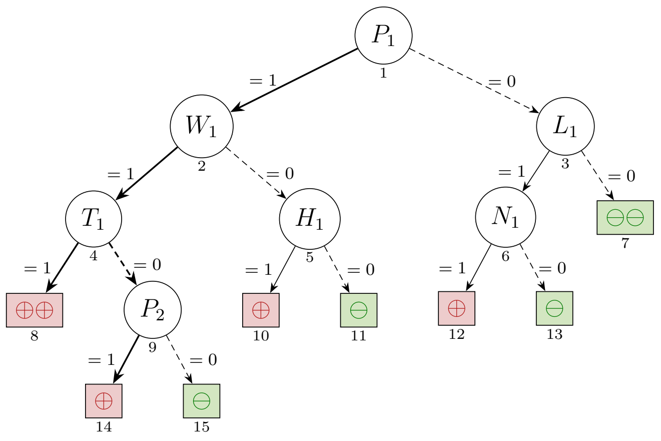
Classification problems in ML are defined on a set of m features (or attributes) $\mathcal{F} = \{1, \ldots, m\}$ and a set of K classes $\mathcal{K} = \{c_1, \ldots, c_K\}$. Each feature $i \in \mathcal{F}$ takes values from its associated domain D_i. Domains can be categorical or ordinal, with values that can be Boolean, integer or real. The feature space $\mathbb{F}$ is defined by the Cartesian product of the domains as $\mathbb{F} = D_1 \times \ldots \times D_m$. The notation $\mathbf{x} = (x_1, \ldots, x_m)$ represents an arbitrary point in the feature space, where each x_i is a variable taking values from D_i. Moreover, $\mathbf{v} = (v_1, \ldots, v_m)$ represents a specific point in the feature space, where each v_i is such that $v_i \in D_i$ (a concrete value). An ML classifier $\mathcal{M}$ is characterized by a (non-constant) classification function $\kappa : \mathbb{F} \to \mathcal{K}$ mapping feature space points into classes. A classifier $\mathcal{M}$ is represented by the tuple $\mathcal{M} = (\mathcal{F}, \mathbb{F}, \mathcal{K}, \kappa)$. Observe that, the classification function represents the operation (or behavior) of the ML classifier, i.e. the machine learning model. Different ML models can be considered, e.g. neural networks, tree ensembles, decision lists, etc.

Given a classifier $\mathcal{M}$, a point $\mathbf{v} \in \mathbb{F}$, and a class $c \in \mathcal{K}$ such that $c = \kappa(v)$, an instance (or sample) is defined as $(\mathbf{v}, c)$. In this paper, we consider explanation problems that given a classifier $\mathcal{M}$ and an instance $(\mathbf{v}, c)$, provide an explanation for the instance. Thus, we denote an Explanation Problem $\mathcal{E}$ by the tuple $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$.

A related problem is the Classifier Decision Problem (CDP), in which given a target class $c \in \mathcal{K}$ decide whether there exists some point in the feature space $\mathbf{x} \in \mathbb{F}$ for which the prediction is c, that is, the formula $\exists(\mathbf{x} \in \mathbb{F}).(\kappa(x) = c)$ is true.

2.2. Examples of Classifiers

Formal explainability has been studied in the context of different types of classifiers. For example for Decision Trees [42, 28], Decision Sets and Lists [34], Boosted Trees and Tree Ensembles [37, 32, 33], Random Forests [44], Decision Diagrams and Graphs and Explanation Graphs [28], Propositional Languages [26], Naive Bayes classifiers [53], Monotonic classifiers [54], Neural Networks [30], Bayesian Network classifiers [73]. Additional information on classifiers studied in formal explainability can be found in [51, 55], and



Features & Predicates		
$\mathcal{F}$	Pred.	Def.
1	H_1	HCT ≤ 41.2
2	L_1	Lymp. ≤ 0.58
3	N_1	Neutr. ≤ 4.9
4	P_1	PLT ≤ 193
4	P_2	PLT ≤ 143
5	T_1	Temp. > 37.4
6	W_1	WBC ≤ 6.0

Figure 1: Example DT from the domain of medical diagnosis. Each internal node is associated with a predicate defined on one of the features.

corresponding references. In order to demonstrate the results in the paper we consider two running examples for classifiers that are based on Decision Lists and Neural Networks.

2.2.1. Decision Trees

Decision Trees (DT) are one of the most widely used classifiers in ML. A decision tree is a directed acyclic graph, with one root node. Each non-terminal node is associated with a single feature[2], and terminal nodes are associated with classes from $\mathcal{K}$ (the predicted class by the node). Each edge is associated with a literal defined on the values from the domain of the feature (of the origin node of the edge).

We consider as a running example a decision tree for a medical diagnosis example adapted from [75, Fig. 1], which targets the prediction of dengue fever.

Example 1. *The example decision tree is shown in Figure 1. The set of features is* $\mathcal{F} = \{1, 2, 3, 4, 5, 6\}$*, each of which with a real-valued domain. The set of classes is* $\mathcal{K} = \{\ominus\ominus, \ominus, \oplus, \oplus\oplus\}$*. Each class has the following interpretation: i)* $\oplus\oplus$ *indicates that dengue fever is probable (or very likely);*

[2]In this work we are considering univariate DTs.

ii) $\oplus$ *that dengue fever is likely; iii)* $\ominus$ *that non-dengue fever is likely; and iv)* $\ominus\ominus$ *that non-dengue fever is probable.*[3]

2.2.2. Neural Networks

To illustrate our algorithms, we consider a simple architecture for a Neural Network (NN), concretely feed-forward neural network. An NN is composed of a number of layers of neurons. The output values of the neurons in a given layer are computed given the output values of the neurons in the previous layer, up to a number L of layers, and such that the inputs represent layer 0. Each neuron computes an intermediate value given the output values of the neurons in the previous layer, and the weights of the connections between neurons. For classification problems, there are different mechanisms to predict the actual class associated with the computed output values. One option is to have one output of the final layer for each different class. Another option is to pick the class depending on the value taken by a single output variable. The running example presented in this section follows the first approach.

Example 2. *In this example we consider a simple neural network to predict whether a paper is accepted (A), rejected (R), or considered to be a borderline (B) paper at a conference. The network has the following four input features:*

1. *whether the paper has novelty or not (1 or 0)*
2. *does the paper present a good experimental evaluation (1 or 0)*
3. *does the paper properly cite all the relevant works in the area (1 or 0)*
4. *is the conference a non-top conference (1 or 0)*

The classes of the network are $\mathcal{K} = \{A, B, R\}$*, and the features are Boolean features corresponding to the previous items respectively, that is,* $\mathcal{F} = \{1, 2, 3, 4\}$ *with domains* $D_i = \{0, 1\}$ *for* $i \in \{1, 2, 3, 4\}$*.*

After training, the neural network predicts the paper to get accepted if the majority of the four input values is 1, rejected if the majority is 0, and borderline if there is a tie. As examples $\kappa(0, 0, 0, 0) = R$*,* $\kappa(0, 1, 0, 1) = B$*,*

[3]Although orthogonal to the example, the original names of the features are as follows (from [75]): HCT: hematocrit, Lymp.: total number of lymphocytes, Neutr.: total number of neutrophils, PLT: platelet count, Temp.: body temperature, WBC=white blood cell count.

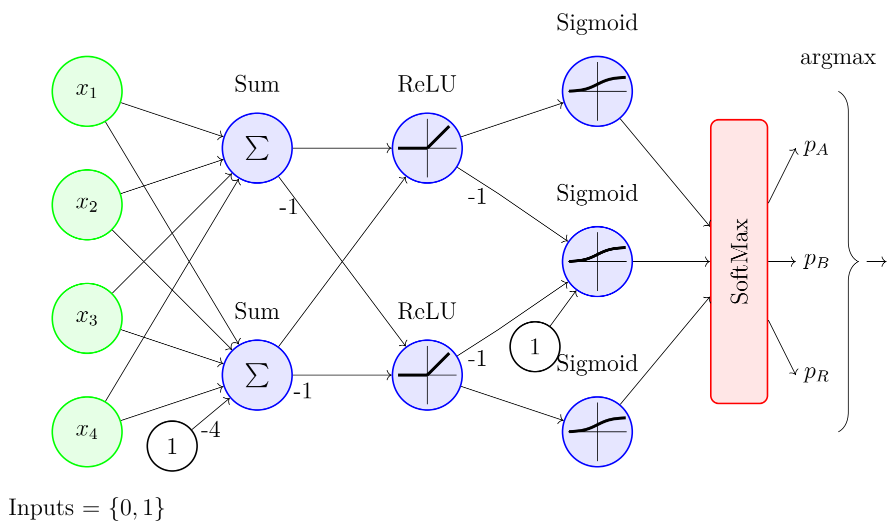


Figure 2: Simple Neural Network for predicting acceptance of a paper.

and $\kappa(1,1,1,0) = A$. *The network is represented in Figure 2. There are five layers in the network. Any connection between two units of the network is associated with a weight. If the weight is not shown explicitly in the figure, the weight is assumed to be +1. The first one is the input layer, with four binary (0/1) inputs that represent the four input features of our problem. In the second layer, one unit sums the values of all the inputs (upper unit in the second layer), and the other unit performs the same sum but it also adds a bias term equal to -4. This way, the value of the first unit will be equal to the number of true inputs (1) and the value of the second unit will be equal to the number of false inputs (0) multiplied by -1. Then, in the next layer we use two units with the ReLU activation function* ($ReLU(x) = max(0, x)$) *to decide whether there are more true inputs than false inputs* ($ReLU(\sum x_i + \sum x_i - 4)$) *or more false inputs than positive inputs* ($ReLU(-\sum x_i + -(\sum x_i - 4))$). *In the next layer the outputs of these ReLU units are combined to transform these decision values to three explicit values in the range* $[0, 1]$, *using the sigmoid activation function* ($\sigma(x) = \frac{1}{1+e^{-x}}$).[4] *The upper sigmoid*

[4]In standard neural network architectures, layers with activation units are usually used after layers that perform linear transformations (like it happens in the second and third layers of our network). However, in this example network a linear layer between the ReLU layer (third layer) and the sigmoid layer (fourth layer) is not really necessary, so for the

unit encodes whether the true inputs are majority, the middle sigmoid unit encodes whether there are ties between true and false inputs, and the lower sigmoid unit encodes whether the false inputs are majority. Finally, these three values (call them $\sigma_1, \sigma_2, \sigma_3$) from the sigmoid units are passed to a softmax layer, that transforms these values to a normalized version :

$$softmax(i) = \frac{e^{\sigma_i}}{\sum_{j=1}^{3} e^{\sigma_j}}$$

The previous neural network example uses linear transformation layers and activation layers that are commonly used when learning neural networks. But once we have learned such a neural network, we can modify the functions used in some layers to have a network that computes the same function, but in such a way that the network can be transformed to an equivalent boolean/integer model that can be used in our logic-based explainability framework.

In the previous example, this can be achieved by two simple modifications:

1. Change the $sigmoid(x)$ function by the function:

$$step(x) = \begin{cases} 0 & if \ x \leq 0 \\ 1 & otherwise \end{cases}$$

2. Change the $softmax([x_0, x_1, \ldots, x_{k-1}])$ function by the function:

$$pickmax([x_0, x_1, \ldots, x_{k-1}]) = \begin{cases} 0 & if \ \sum_j b_{0,j} \geq k \\ 1 & if \ \sum_j b_{1,j} \geq k \\ \ldots & \\ k-1 & if \ \sum_j b_{k-1,j} \geq k \end{cases}$$

   where $b_{i,j} = 1 \leftrightarrow x_i \geq x_j$. This function represents the final decision function for a set of k target classes (three in our previous example). For problems where there can be ties between the inputs with maximum value, ties can be broken in favor of the first x_i that satisfies its inequality.

sake of simplification we have omitted such intermediate layers.

3. Logic-Based Explainability

Logic-based (or formal) explanation approaches are characterized by providing guarantees of rigor in the explanations produced, given the underlying ML model. This is in direct contrast with the non-formal approaches. This section reviews logic-based explanations, and provides an overview of the state of the art of the field.

3.1. Abductive Explanations

One of the questions posed to a ML model upon a prediction to a given instance, is *"Why"* did the model report such a prediction. As a way to answer this question, [73] and [36] have proposed Prime Implicant (PI) explanations and Abductive explanations (AXp's).[5] These correspond to a minimal set of literals, setting features to values from the instance, that force the output of the ML model to correspond to the predicted output, thus explaining the prediction. Formally, we start by presenting a weak version of AXp's. Given an instance $(\mathbf{v}, c)$, where $\mathbf{v} \in \mathcal{F}$, $c \in \mathcal{K}$ such that $c = \kappa(\mathbf{v})$, a WeakAXp is a subset of features $\mathcal{X} \subseteq \mathcal{F}$ for which the following predicate holds:

$$\mathsf{WeakAXp}(\mathcal{X}) := \forall(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i)\right] \rightarrow (\kappa(\mathbf{x}) = c) \tag{1}$$

Observe that only features are part of the WeakAXp, not values, since the values are determined by the instance. An AXp is then a subset-minimal WeakAXp, that is, $\mathcal{X} \subseteq \mathcal{F}$ is an AXp iff the following predicate holds:

$$\mathsf{AXp}(\mathcal{X}) := \mathsf{WeakAXp}(\mathcal{X}) \wedge \forall(\mathcal{X}' \subsetneq \mathcal{X}).\neg\mathsf{WeakAXp}(\mathcal{X}') \tag{2}$$

One important aspect of the WeakAXp predicate is that it is monotone.[6] The monotonicity of WeakAXp allows for a simplified definition of the AXp predicate as in (3).

$$\mathsf{AXp}(\mathcal{X}) := \mathsf{WeakAXp}(\mathcal{X}) \wedge \forall(i \in \mathcal{X}).\neg\mathsf{WeakAXp}(\mathcal{X} \setminus \{i\}) \tag{3}$$

[5] PI explanations were proposed in [73] in the context of Boolean classifiers. Independently [36] studied PI explanations in the case of for more general classification functions, i.e., not necessarily Boolean, and related instead explanations with abduction and called explanations as Abductive explanations (AXp's). We follow the formalization used in recent works [50, 55].

[6] WeakAXp is monotone because if $\mathsf{WeakAXp}(\mathcal{X})$ holds for $\mathcal{X} \subseteq \mathcal{F}$, then for any $\mathcal{X}' \subseteq \mathcal{F}$ such that $\mathcal{X} \subseteq \mathcal{X}'$, it holds that $\mathsf{WeakAXp}(\mathcal{X}')$.

Observe that (3) requires a linear number of checks to the WeakAXp predicate, while (2) checks an exponential number of subsets. The algorithms to compute AXp's are then based on (3).

Example 3 (Abductive Explanations - Decision Trees)**.** *Consider the DT in Example 1 for predicting dengue fever. Let a patient be represented by the point in the feature space* $\mathbf{v} = (35.0, 0.25, 5.1, 100, 37.1, 5.2)$.[7] *The predicted instance is* $(\mathbf{v}, \oplus)$.

Two examples of WeakAXp's are the sets $\mathcal{X}_a = \{1, 2, 3, 4, 5, 6\}$ *and* $\mathcal{X}_b = \{4, 5, 6\}$. X_a *corresponds to all the features, which means if we set all features to the values of the instance, the output will be for sure the same as the output of the instance. For* $\mathcal{X}_b$*, note that any point respecting PLT set to 100, Temp. set to 37.1 and WBC set to 5.2, will be classified by the DT with the leaf node 14, with class* $\oplus$.

Observe that, $\mathcal{X}_a$ *is not subset-minimal, since* $\mathcal{X}_b \subset \mathcal{X}_a$ *and both are WeakAXp's. Thus,* $\mathcal{X}_a$ *is not an AXp. On the other hand,* $\mathcal{X}_b$ *is an AXp because it is subset-minimal. If we try to remove any feature from* X_b *we can obtain points in the feature space that are classified with other classes not necessarily* $\oplus$*. Another example of an AXp is the set* $\mathcal{X}_c = \{1, 4, 5\}$*, because any point respecting the values of the instance for the features in* $\mathcal{X}_c$*, will either be classified by the DT with the leaf nodes 10 or 14 (with class* $\oplus$*). On the other hand, a similar analysis allows us to check that* $\mathcal{X}_c$ *is indeed subset-minimal.*

Example 4 (Abductive Explanations - Neural Networks)**.** *Consider the NN in Example 2 for predicting the acceptance of a paper in a conference. Let the paper be represented by the point in the feature space* $\mathbf{v} = (1, 1, 0, 1)$*, and the instance* $(\mathbf{v}, A)$*. There is only one AXp which is* $\mathcal{X} = \{1, 2, 4\}$*. The reason being that it is enough to consider three features with value 1, in order for the paper to be accepted. Since in* $\mathbf{v}$*, the only three features containing value 1 are 1, 2, 4, then they are all part of the AXp. On the other hand, if any of the features is disregarded, then the paper could be considered a borderline paper or even rejected.*

3.2. Contrastive Explanations

Related to the "Why" question, a complementary question that has been posed to ML models, is the "Why not". The idea is that given the instance,

[7]Values of the points in the feature space are represented in the order of the features.

show which features are necessary to change its value in order to be able to change the output of the prediction. To answer the *"Why not"* question, [35] proposed the use of contrastive explanations (CXp's). These represent a minimal set of features whose change in value, allows a change in the prediction. Formally, and similar to the AXp case, we start by first showing a weak version of a CXp. Given an instance $(\mathbf{v}, c)$, such that $c = \kappa(\mathbf{v})$, a WeakCXp is a subset of features $\mathcal{X} \subseteq \mathcal{F}$, such that the following predicate holds:

$$\mathsf{WeakCXp}(\mathcal{X}) := \exists(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{F} \setminus \mathcal{X}} (x_i = v_i)\right] \wedge (\kappa(\mathbf{x}) \neq c) \tag{4}$$

Observe that as in the case of WeakAXp's, a WeakCXp contains only features, no values are included, since the required values (for features not in the WeakCXp) are set by the instance. A CXp is then a minimal-subset WeakCXp, that is, $\mathcal{X} \subseteq F$ is a CXp iff the following predicate holds:

$$\mathsf{CXp}(\mathcal{X}) := \mathsf{WeakCXp}(\mathcal{X}) \wedge \forall(\mathcal{X}' \subsetneq \mathcal{X}).\neg\mathsf{WeakCXp}(\mathcal{X}') \tag{5}$$

The WeakCXp predicate is also a monotone predicate. Similar to AXp's, the monotonicity of the WeakCXp predicate allows for a simplified definition of the CXp predicate as in (6).As such, the algorithms to compute CXp's are based in (6).

$$\mathsf{CXp}(\mathcal{X}) := \mathsf{WeakCXp}(\mathcal{X}) \wedge \forall(i \in \mathcal{X}).\neg\mathsf{WeakCXp}(\mathcal{X} \setminus \{i\}) \tag{6}$$

Example 5 (Contrastive Explanations - Decision Trees)**.** *Consider the DT in Example 1 for predicting dengue fever. Let a patient be again represented by the point in the feature space* $\mathbf{v} = (35.0, 0.25, 5.1, 100, 37.1, 5.2)$ *as in Example 3. The instance is* $(\mathbf{v}, \oplus)$*.*

Two examples of WeakCXp's are $\mathcal{X}_a = \{1, 2, 3, 4, 5, 6\}$*, and* $\mathcal{X}_b = \{1, 6\}$*.* $\mathcal{X}_a$ *corresponds to the feature set, which means if we allow all features to take any value, and since there is more than one class in the DT, then there are points in the feature space for which the prediction changes. For the case of* $\mathcal{X}_b$*, if we fix the values of the features 2,3,4, and 5 to the values in* $\mathbf{v}$*, and change the values of features 1 and 6 to 50.0 and 7.0 (for example), then the point* $(50.0, 0.25, 5.1, 100, 37.1, 7.0)$ *is classified as* $\ominus$ *(changing the prediction).*

As can easily be seen, $\mathcal{X}_a$ *is not a CXp, because* $\mathcal{X}_b \subset \mathcal{X}_a$ *and both are WeakCXp's, meaning that* $\mathcal{X}_a$ *is not subset-minimal. Nevertheless,* $\mathcal{X}_b$ *is an*

CXp, because if we try to disregard from $\mathcal{X}_b$ any of the features (1 or 6), then the leaf nodes reached in the DT will either be node 10 or node 14 (depending on the feature that is disregarded), and both nodes classify as $\oplus$ meaning the prediction would not change. Thus, $\mathcal{X}_b$ is subset-minimal.

This example, contains two more CXp's, which are $\mathcal{X}_c = \{4\}$ and $\mathcal{X}_d = \{5\}$, since it is enough to consider new values for those features (independently) to change the prediction, and both are subset-minimal. No other CXp's exist for this example.

Example 6 (Contrastive Explanations - Neural Networks)**.** *Consider the NN in Example 2 for predicting the acceptance of a paper, and let the instance be $(\mathbf{v}, A)$, with $\mathbf{v} = (1, 1, 0, 1)$ (as in Example 4). This example contains three CXp with the features $\mathcal{X}_a = \{1\}$, $\mathcal{X}_b = \{2\}$, and $\mathcal{X}_c = \{4\}$. This is because in order to change the prediction, it is enough to change one of those feature values (on their own). Feature 3 is not a CXp since changing its value to 1 would still predict to accept the paper (not changing it). For feature 3 to be included in a CXp, then other features would be included as well, but then the CXp would not be subset-minimal, since those features belong to CXps of size one.*

3.3. Duality between Explanations

Given an explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$ (where $c = \kappa(\mathbf{v})$), we consider the set of all the AXp's $\mathbb{A}$ and the set of all CXp's $\mathbb{C}$ as follows:

$$\mathbb{A}(\mathcal{E}) = \{\mathcal{X} \subseteq \mathcal{F} \mid \mathsf{AXp}(\mathcal{X}) = \mathit{true}\} \tag{7}$$
$$\mathbb{C}(\mathcal{E}) = \{\mathcal{X} \subseteq \mathcal{F} \mid \mathsf{CXp}(\mathcal{X}) = \mathit{true}\} \tag{8}$$

One property that has been proven about the two sets is that they are dual to each other with regards to minimum hitting sets[8], which builds on Reiter's seminal work on model-based diagnosis [62].

Proposition 1 (Duality between AXp's and CXp's [35])**.** *Given an explanation problem $\mathcal{E}$, and the sets $\mathbb{A}(\mathcal{E})$ and $\mathbb{C}(\mathcal{E})$, then:*

- *each AXp in $\mathbb{A}(\mathcal{E})$ is a minimum hitting set of the sets in $\mathbb{C}(\mathcal{E})$*

[8] A minimum hitting set $\mathcal{X}$ of a set of sets $\mathbb{S} = \{\ldots, \mathcal{S}_i, \ldots\}$ is a subset-minimal set such that, for any $\mathcal{S}_i \in \mathbb{S}$, it holds that $\mathcal{X} \cap \mathcal{S}_i \neq \emptyset$.

	AXp's $\mathbb{A}$		CXp's $\mathbb{C}$
Example 3	$\{\{1,4,5\},\{4,5,6\}\}$	Example 5	$\{\{4\},\{5\},\{1,6\}\}$
Example 4	$\{\{1,2,4\}\}$	Example 6	$\{\{1\},\{2\},\{4\}\}$

Table 1: AXp's and CXp's from the running examples. Each AXp is a mhs of the CXp's and vice-versa.

- *each CXp in $\mathbb{C}(\mathcal{E})$ is a minimum hitting set of the sets in $\mathbb{A}(\mathcal{E})$*

The importance of this result is related to the ability to compute/enumerate explanations once the dual has been obtained.

Example 7 (Duality Property)**.** *Recall the Examples 3 and 5. We have shown that the set of all CXp's is the set $\mathbb{C} = \{\{4\},\{5\},\{1,6\}\}$. Observe that, any mhs of $\mathbb{C}$ must contain features 4 and 5 since they are part of unit/singleton sets in $\mathbb{C}$. Since a mhs must hit the set $\{1,6\}$, then the alternatives for an mhs candidate is to contain feature 1 or feature 6 or both, but the mhs must be subset-minimal, then the only mhs's of $\mathbb{C}$ are the sets $\{1,4,5\}$ and $\{4,5,6\}$. These correspond to the AXp's found in Example 3. A similar analysis could be obtained the other way around, starting from the set of all AXp's to obtain the set of all CXp's.*

We can also verify the duality property from the Examples 4 and 6. A summary of the AXp's and CXp's from the different running examples is shown on Table 1.

3.4. Timeline and Current Status of Logic-based Explainability

Figure 3 presents a timeline of research in the field of formal XAI. In contrast, Figure 4 overviews the progress observed in algorithms for computing one (abductive or contrastive) explanation.

Regarding Figure 3, the initial focus was on proposing definitions and establishing duality between abductive and contrastive explanations [73, 36, 37, 35]. A number of polynomial-time algorithms for computing one explanation were proved soon after, for several families of classifiers. As shown in Figure 4 (in green), these include NBCs [53], (univariate) decision trees [42, 27, 43], monotonic classifiers [54], graph-based classifiers [27], restricted propositional language classifiers [26], but also several other families of classifiers [15, 16, 13]. For some other families of classifiers, computational hardness results were established, but logic encodings were also devised [44, 34, 33], which

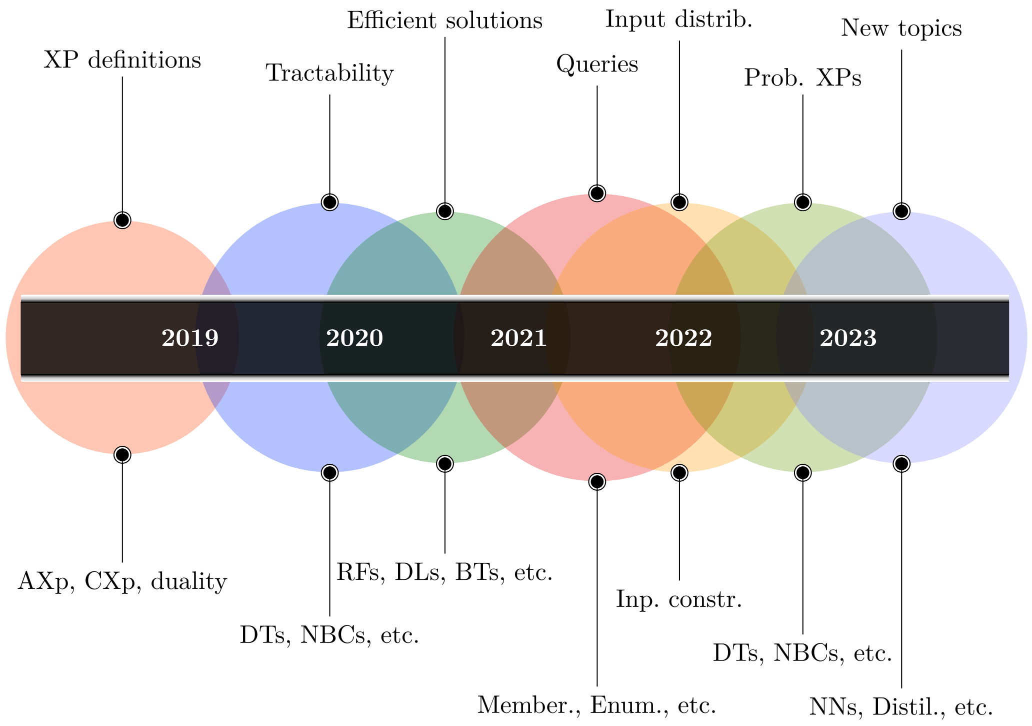


Figure 3: Timeline of research on formal XAI

in turn enable very efficient computation of explanations. As shown in Figure 4 (in blue), these include decision lists [34], random forests [44], boosted trees and tree ensembles [33]. For some families of classifiers, e.g. random forests and tree ensembles in general, it is the case that large complex models can now be explained. Additional work considered different explainability queries [7, 27, 5, 6, 29, 25]. A number of known shortcomings of formal XAI have been studied in recent years. The handling of constraints on the inputs, i.e. when not all points in feature space are allowed, was studied in a number of works [21, 81]. Probabilistic explanations were proposed to enable the computation of smaller explanations [76, 39]. Initial results on computing explanations using simpler (or distilled) surrogate models have also been obtained [11]. In more recent work, the relationship between explainability and adversarial examples, which was known since 2019 [37], has been exploited for the efficient computation of explanations in the case of NNs [30, 77, 41] was demonstrated. This line of work is expected to enable the efficient computation of abductive explanations using robustness tools, especially for large

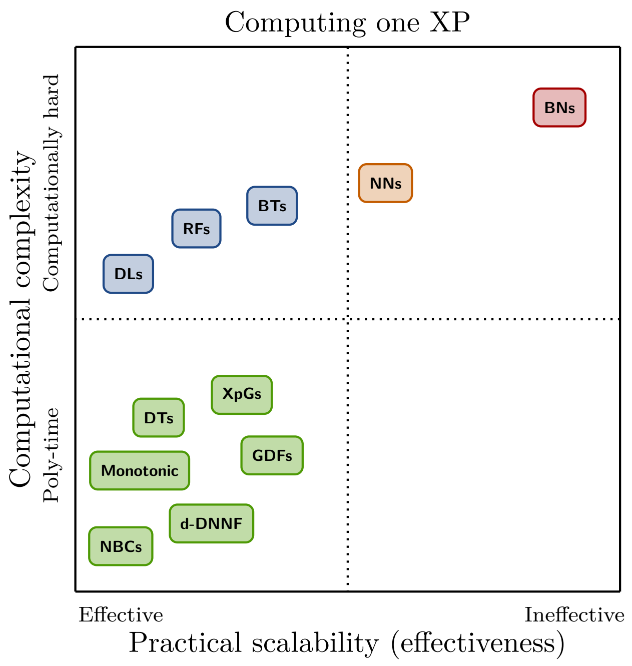


Figure 4: Progress in computing one AXp/CXp

deep neural networks.

Additional promising lines of research include: i) further expanding the scalability of explainers based on finding adversarial examples [41]; ii) relating explanations by feature selection (e.g. abductive and contrastive explanations) with explanations by featureattribution [10, 79]; iii) exploiting explanations of surrogate models as explanations for complex models [11];

4. A General Framework

The previous section presented the state of the art of logic-based explainability. Given an ML model, a (fully specified) input instance and the corresponding (single) output, a user seeks to obtain a formal explanation for the output with guarantees of rigor. This section considers a generalized framework of the explanations, where the user is allowed to bypass some of the inputs by leaving them unspecified, when obtaining the explanations. Additionally, the framework generalizes to the cases of multiple output classes. Unspecified inputs in the instance considering multiple output classes allow the user to disregard the effect of certain (unwanted) features and to focus on more expressive explanations, capable to cope with several outputs simultaneously.

4.1. Partially-Specified Inputs and Multiple Classes

We consider the situation of partially specified inputs to an ML model, i.e. not all features of the input instance $\mathbf{v}$ are assigned values, in which case we say that there exist *missing feature values* (or simply missing inputs). To account for these missing values, we extend the definition of a classifier to allow for *partially-specified inputs*, as follows. Each domain $\mathbb{D}_i$ is extended with a distinguished value $\mathfrak{u}$, not in any $\mathbb{D}_i$, which indicates that feature i is left unspecified, i.e. it could possibly take *any* value from its domain. Concretely, $\mathbb{D}'_i = \mathbb{D}_i \cup \{\mathfrak{u}\}$. Feature space (with unspecified inputs) becomes $\mathbb{F}' = \mathbb{D}'_1 \times \cdots \times \mathbb{D}'_m$. A completely specified point in feature space $\mathbf{v} \in \mathbb{F}$ is *covered* by $\mathbf{z} \in \mathbb{F}'$, written $\mathbf{v} \sqsubseteq \mathbf{z}$, if $\forall(i \in \mathcal{F}).\,[(v_i = z_i) \lor (z_i = \mathfrak{u})]$. Given $\mathbf{z} \in \mathbb{F}'$, $\mathcal{F}$ is partitioned into $\mathcal{U}$ and $\mathcal{S}$, containing respectively the unspecified ($\mathcal{U}$) and specified($\mathcal{S}$) features.

It will also be convenient to generalize the classification function to map points in $\mathbb{F}'$ into a set $\mathcal{T} \subseteq \mathcal{K}$[9] of (target) classes, i.e. $\kappa' : \mathbb{F}' \to 2^{\mathcal{K}}$. Thus, given a partially specified input $\mathbf{z} \in \mathbb{F}'$, the (generalized) prediction is some $\mathcal{T} \subsetneq \mathcal{K}$. Hence, κ' is defined as follows,

$$\kappa'(\mathbf{z}) = \{c \in \mathcal{K} \,|\, \exists(\mathbf{v} \in \mathbb{F}).\mathbf{v} \sqsubseteq \mathbf{z} \land c = \kappa(\mathbf{v})\} \tag{9}$$

In general, we denote $\kappa'(\mathbf{z})$ by $\mathcal{T} \subsetneq \mathcal{K}$, and refer to $\mathcal{T}$ as the (generalized) prediction given $\mathbf{z}$. Finally, a generalized (i.e. with partially specified inputs) explanation problem is defined as a tuple $\mathcal{E}' = (\mathcal{M}, (\mathbf{z}, \mathcal{T}))$.

Example 8 (Missing Inputs Instance - Decision Trees)**.** *Consider the DT in Example 1 for predicting dengue fever. Instead of a single patient (as in Examples 3 and 5), in this example we consider a set of patients. The considered set of patients are predicted to have some form of dengue fever; very likely ($\oplus\oplus$) or likely ($\oplus$). The user wishes to obtain an explanation for this set of patients to have any form of dengue.*

The example set of patients is represented by the following point in the feature space (with unspecified inputs) $\mathbf{z} = (35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.2)$*. Some of the features are unspecified (marked* $\mathfrak{u}$*), namely features 3 (for Neutr.)*

[9]The case where $\mathcal{T} = \mathcal{K}$ is uninteresting for our purposes and so it is not considered.

and 5 (for Temp.). The other features are assigned the values shown. $\mathcal{F}$ is partitioned into $\mathcal{U} = \{3, 5\}$, and $\mathcal{S} = \{1, 2, 4, 6\}$, and $\kappa'(\mathbf{z}) = \{\oplus, \oplus\oplus\}$. Moreover, we assume that the user considers the set of possible predictions $\mathcal{T} = \{\oplus, \oplus\oplus\}$ to be satisfactory, and seeks an explanation for $\mathcal{T}$. The predicted input instance to explain is then $(\mathbf{z}, \mathcal{T})$.

Example 9 (Missing Inputs Instance - Neural Networks)**.** *Consider the NN from Example 2 for predicting the acceptance of a paper in a conference. We want to obtain an explanation for a paper to have been predicted to have any chance (accept or borderline) while disregarding the ranking of the conference (the value of feature 4). As such, we consider the same feature point as in Examples 4 and 6, with $\mathfrak{u}$ as value of feature 4, that is, $\mathbf{z} = (1, 1, 0, \mathfrak{u})$. The instance to explain is then $(\mathbf{z}, \{A, B\})$.*

4.2. Selecting Multiple Classes

When trying to get an explanation, what should the value of $\mathcal{T}$ be? One possible solution is for $\mathcal{T}$ to be user-specified, e.g. it represents a set of classes (or a class) of interest to the user. Another solution is that, if $\mathbf{z} \in \mathbb{F}'$ is given, then use the definition above (see (9)) for computing $\kappa'(\mathbf{z})$. One obstacle to exploiting this solution in practice is that the number of (completely specified) points to analyze is worst-case unbounded on the size of $\mathcal{U}$.[10] In practice, a simpler solution can be applied to infer $\mathcal{T} = \kappa'(\mathbf{z})$ with the following steps:

1. Set $\mathcal{T} = \emptyset$;
2. For each $c \in \mathcal{K}$, decide whether the following statement is consistent:

$$\exists(\mathbf{x} \in \mathbb{F}).\,(\wedge_{i\in\mathcal{S}}(x_i = z_i)) \wedge (c = \kappa(\mathbf{x})) \tag{10}$$

3. If the statement above is consistent, then add c to $\mathcal{T}$.

For example, for families of classifiers for which computing one explanation is in P [73, 54, 43], then the proposed solution computes $\mathcal{T}$ in polynomial time, even if the number of points in feature space covered by $\mathbf{z}$ is unbounded/exponential in the worst case.

Observe that considering missing inputs with several classes in $\mathcal{T}$, allows for a more general (still rigorous) explanation, able to explain all the classes in $\mathcal{T}$ (simultaneously). Otherwise, for each of the classes in $\mathcal{T}$, the user would

[10]Observe that, for features with discrete finite domains, the worst-case would be exponential. However, we also account for real-valued features. In such a situation, the worst-case number of points in feature space is unbounded.

have to obtain completely specified instances (for example by imputation of the missing values), and only then compute explanations (obtaining one explanation for each of the specified instances, i.e, one explanation for each class in $\mathcal{T}$). The user would end up with a set of explanations, instead of only one.

The flexibility in the choice of $\mathcal{T}$, i.e. either inferred from $\mathbf{z}$ or specified by the user, is illustrated in the previous Examples 8 and 9. For the DT Example 8, and the NN Example 9, the selected instances to explain consider the respective $\mathcal{T}$'s to be inferred from the feature points in the instance. Concretely, the DT example has $\mathbf{z} = (35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.2)$, with $\mathcal{T} = \{\oplus, \oplus\oplus\}(= \kappa'(\mathbf{z}))$ to explain why is the prediction any form of dengue fever.

4.3. General Explanations

Depending on the selected set of specified features, such set may or may not suffice for a given prediction, i.e. there are not (or there are) assignments to the non-specified features that cause the prediction to change.

Definition 1. *Given $\mathbf{z} \in \mathbb{F}'$, and associated partition of $\mathcal{F}$, $\mathcal{U}$ and $\mathcal{S}$, we say that $\mathbf{z}$ is* sufficient for the prediction *$\mathcal{T}$ if,*

$$\forall(\mathbf{x} \in \mathbb{F}).\,(\wedge_{i \in \mathcal{S}}(x_i = z_i)) \rightarrow (\kappa(\mathbf{x}) \in \mathcal{T}) \tag{11}$$

If (11) does not hold, then $\mathbf{z}$ is not sufficient for the prediction.

In the remainder of the paper, we focus on the cases where $\mathbf{z}$ is sufficient for the prediction given $\mathcal{T}$. Otherwise, the model and $\mathbf{z}$ do not suffice to answer ”Why?” and ”Why Not?” questions regarding the prediction, i.e. we would have to impute values to the features in $\mathcal{U}$. An alternative would be to expand $\mathcal{T}$ so that $\mathbf{z}$ becomes sufficient for the prediction.

Given a point $\mathbf{z}$ sufficient for the prediction $\mathcal{T}$, a set $\mathcal{X}$ of (specified) features, $\mathcal{X} \subseteq \mathcal{S}$, is an abductive explanation (AXp) if $\mathcal{X}$ is sufficient for the prediction, and $\mathcal{X}$ is irreducible. Thus, an AXp is a minimal set $\mathcal{X} \subseteq \mathcal{S} \subseteq F$ such that,

$$\forall(\mathbf{x} \in \mathbb{F}).\,(\wedge_{i \in \mathcal{X}}(x_i = z_i)) \rightarrow (\kappa(\mathbf{x}) \in \mathcal{T}) \tag{12}$$

Any set $\mathcal{X}$ respecting the statement above is a *weak* AXp (or WAXp). Similarly, we can define a contrastive explanation given some (specified) features. A CXp is a minimal set $\mathcal{Y} \subseteq \mathcal{S} \subseteq F$ such that,

$$\exists(\mathbf{x} \in \mathbb{F}).\,\big(\wedge_{i \in \mathcal{S} \setminus \mathcal{Y}}(x_i = z_i)\big) \wedge (\kappa(\mathbf{x}) \notin \mathcal{T}) \tag{13}$$

i.e. the features in $\mathcal{Y}$ (and also in $\mathcal{U}$) are allowed to take any value in their domain, and this suffices for obtaining a prediction not in $\mathcal{T}$. Any set $\mathcal{Y}$ respecting the statement above is a *weak* CXp (or WCXp). We use predicates $\mathsf{WeakAXp}$ and $\mathsf{WeakCXp}$ to refer to sets that may or may not satisfy the definitions of weak AXp/CXp.

One key observation is that the definitions above continue to exhibit important properties of explanations that have been identified in earlier work.

Proposition 2 (Monotonicity)**.** *Given* $\mathbf{z}$ *(and so a partition of* $\mathcal{F}$*,* $\mathcal{S}$ *and* $\mathcal{U}$*) that is sufficient for the prediction* $\mathcal{T} \subsetneq \mathcal{K}$*, then* $\mathsf{WeakAXp}$ *and* $\mathsf{WeakCXp}$ *are monotone and up-closed.*

Proof. By definition of $\mathbf{z}$ being sufficient for the prediction, and because κ is surjective, then it is immediate that $\mathsf{WeakAXp}(\mathcal{S})$ and $\mathsf{WeakCXp}(\mathcal{S})$ hold. For both $\mathsf{WeakAXp}$ and $\mathsf{WeakCXp}$, it is also plain that if the predicates hold for some set $\mathcal{I} \subseteq \mathcal{S}$, then the predicates will hold for any $\mathcal{I} \subseteq \mathcal{I}' \subseteq \mathcal{S}$. In the case of weak AXp's, if $\mathcal{I}' \supseteq \mathcal{I}$, then any point $\mathbf{x} \in \mathbb{F}$ consistent with $\mathcal{I}'$ will also be consistent with $\mathcal{I}$. Since $\mathsf{WeakAXp}(\mathcal{I})$ holds, then $\mathsf{WeakAXp}(\mathcal{I}')$ also holds, i.e. the prediction does not change. In the case of weak CXp's, if $\mathcal{I}' \supseteq \mathcal{I}$, then any point $\mathbf{x} \in \mathbb{F}$ consistent with $\mathbb{F} \setminus \mathcal{I}$ will also be consistent with $\mathbb{F} \setminus \mathcal{I}'$. Since $\mathsf{WeakCXp}(\mathcal{I})$ holds, then $\mathsf{WeakCXp}(\mathcal{I}')$ also holds, i.e. the prediction can be changed. □

This property allows defining AXp's/CXp's using (3) and (6) as before but considering the generalizations proposed as follows:

$$\mathsf{AXp}(\mathcal{X}) := \mathsf{WeakAXp}(\mathcal{X}) \wedge_{t\in\mathcal{X}} \neg\mathsf{WeakAXp}(\mathcal{X} \setminus \{t\}) \tag{14}$$
$$\mathsf{CXp}(\mathcal{Y}) := \mathsf{WeakCXp}(\mathcal{Y}) \wedge_{t\in\mathcal{Y}} \neg\mathsf{WeakCXp}(\mathcal{Y} \setminus \{t\}) \tag{15}$$

Let $\mathbf{z} \in \mathbb{F}'$ (with associated partition of $\mathcal{F}$ into $\mathcal{U}$ and $\mathcal{S}$), and let $\mathcal{S}$ be sufficient for $\mathcal{T} \subseteq \mathcal{K}$. Then, for the generalized explanation problem $\mathcal{E}' = (\mathcal{M}, (\mathbf{z}, \mathcal{T}))$ we define,

$$\mathbb{A}_{\mathtt{u}}(\mathcal{E}') = \{\mathcal{X} \subseteq \mathcal{S} \,|\, \mathsf{AXp}(\mathcal{X})\} \tag{16}$$
$$\mathbb{C}_{\mathtt{u}}(\mathcal{E}') = \{\mathcal{Y} \subseteq \mathcal{S} \,|\, \mathsf{CXp}(\mathcal{Y})\} \tag{17}$$

Given the above, the important property of duality between explanations is generalized in the following, but before presenting the duality result (and its proof), we present two lemmas used in the proof of the duality result.

Lemma 1. *If $\mathcal{Y}$ is a hitting set of $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$, then $\mathcal{Y}$ is a weak CXp of $\mathcal{E}'$.*

Proof. Suppose that $\mathcal{Y}$ is a hitting set of $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$. By contradiction suppose that:

$$\forall(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{S}\setminus\mathcal{Y}} (x_i = z_i)\right] \rightarrow (\kappa(\mathbf{x}) \in \mathcal{T})$$

but, then this means that $\mathcal{S} \setminus \mathcal{Y}$ is a weak AXp of $\mathcal{E}'$, and there exists an AXp in $\mathcal{S} \setminus \mathcal{Y}$ that belongs to $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$. Since $\mathcal{Y}$ is a hitting set of $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$ then $\mathcal{Y} \cap (\mathcal{S} \setminus \mathcal{Y}) \neq \emptyset$, which is impossible. Thus:

$$\exists(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{S}\setminus\mathcal{Y}} (x_i = z_i)\right] \land (\kappa(\mathbf{x}) \notin \mathcal{T})$$

that is, $\mathcal{Y}$ is a weak CXp of $\mathcal{E}'$. □

Lemma 2. *If $\mathcal{X}$ is hitting set of $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$, then $\mathcal{X}$ is a weak AXp of $\mathcal{E}'$.*

Proof. Suppose that $\mathcal{X}$ is hitting set of $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$. By contradiction suppose that:

$$\exists(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{X}} (x_i = z_i)\right] \land (\kappa(\mathbf{x}) \notin \mathcal{T})$$

since $\mathcal{X} = \mathcal{S} \setminus (\mathcal{S} \setminus \mathcal{X})$, then it is the same as:

$$\exists(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{S}\setminus(\mathcal{S}\setminus\mathcal{X})} (x_i = z_i)\right] \land (\kappa(\mathbf{x}) \notin \mathcal{T})$$

this means that $\mathcal{S}\setminus\mathcal{X}$ is a weak CXp of $\mathcal{E}'$, and there exists a CXp inside $\mathcal{S}\setminus\mathcal{X}$ that belongs to $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$. Since $\mathcal{X}$ is a hitting set of $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$ then $\mathcal{X}\cap(\mathcal{S}\setminus\mathcal{X}) \neq \emptyset$, which is impossible. Thus:

$$\forall(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \in \mathcal{X}} (x_i = z_i)\right] \rightarrow (\kappa(\mathbf{x}) \in \mathcal{T})$$

that is, $\mathcal{X}$ is a weak AXp of $\mathcal{E}'$. □

Proposition 3 (Duality & missing data)**.** *Given an explanation problem $\mathcal{E}'$, and the sets $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$ and $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$, then:*

- *each AXp in $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$ is a minimum hitting set of the sets in $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$*
- *each CXp in $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$ is a minimum hitting set of the sets in $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$*

Proof. Consider $\mathcal{X} \in \mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$, and $\mathcal{Y} \in \mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$. Suppose by contradiction that $\mathcal{X} \cap \mathcal{Y} = \emptyset$. Because $\mathcal{X} \in \mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$, then it happens that:

$$\forall(\mathbf{x} \in \mathbb{F}).\left[\bigwedge\nolimits_{i\in\mathcal{X}}(x_i = z_i)\right]\rightarrow(\kappa(\mathbf{x}) \in \mathcal{T})$$

Since $\mathcal{X} \cap \mathcal{Y} = \emptyset$ (and both $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{S}$), then $\mathcal{X} \subseteq \mathcal{S} \setminus \mathcal{Y}$, and:

$$\forall(\mathbf{x} \in \mathbb{F}).\left[\bigwedge\nolimits_{i\in\mathcal{S}\setminus\mathcal{Y}}(x_i = z_i)\right]\rightarrow(\kappa(\mathbf{x}) \in \mathcal{T})$$

But this contradicts the fact that $\mathcal{Y} \in \mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$. Thus any $\mathcal{X} \in \mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$ is a hitting set of any $\mathcal{Y} \in \mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$, and vice-versa.

What is missing to prove is the minimality of the hitting sets. Suppose that $\mathcal{Y} \in \mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$ is a hitting set of $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$. Suppose by contradiction $\mathcal{Y}$ is not subset minimal hitting set, that is, there exists $\mathcal{Z} \subsetneq \mathcal{Y}$ such that $\mathcal{Z}$ is a hitting set of $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$. By Lemma 1, then $\mathcal{Z}$ is a weak CXp of $\mathcal{E}'$, and there exists an CXp inside $\mathcal{Z}$, but this contradicts the subset minimality of the CXp $\mathcal{Y}$ because $\mathcal{Z} \subsetneq \mathcal{Y}$, thus $\mathcal{Y} \in \mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$ is a minimal hitting set of $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$.

Suppose now that $\mathcal{X} \in \mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$ is a hitting set of $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$. Suppose by contradiction $\mathcal{X}$ is not subset minimal hitting set, that is, there exists $\mathcal{Z} \subsetneq \mathcal{X}$ such that $\mathcal{Z}$ is a hitting set of $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$. By Lemma 2, then $\mathcal{Z}$ is a weak AXp of $\mathcal{E}'$, and there exists an AXp inside $\mathcal{Z}$, but this contradicts the subset minimality of the AXp $\mathcal{X}$ because $\mathcal{Z} \subsetneq \mathcal{X}$, thus $\mathcal{X} \in \mathbb{A}_{\mathfrak{u}}(\mathcal{E}')$ is a minimal hitting set of $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}')$. □

Example 10 (General Framework Explanations - Decision Trees)**.** *With respect to the DT of Example 1, the point* $\mathbf{z} = (35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.0)$*, and the target classes* $\mathcal{T} = \{\oplus, \oplus\oplus\}$ *from Example 8, we can identify by inspection weak AXp's and also plain AXp's. A possible WAXp is of course* $\mathcal{S} = \{1, 2, 4, 6\}$*, i.e. the non-missing feature values. But this is not subset-minimal, thus not an AXp. The set of AXps include* $\mathcal{X}_a = \{4, 6\}$ *(representing the literals* PLT $= 100$ *and* WBC $= 5.0$*), and also* $\mathcal{X}_b = \{1, 4\}$ *(representing the literals* HCT $= 35.0$ *and* PLT $= 100$*). Given the set of AXp's* $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}') = \{\{4, 6\}, \{1, 4\}\}$ *then, by MHS duality, we obtain* $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}') = \{\{4\}, \{1, 6\}\}$*. As an example, if we allow* PLT *(feature 4) to take any value, then by picking* PLT > 193 *and by setting* Neutr. *(feature 3, which is unspecified) to a suitable value (e.g.* > 4.9*), we manage to change the prediction to value* $\ominus$ *not in* $\{\oplus, \oplus\oplus\}$*.*

Example 11 (General Framework Explanations - Neural Networks)**.** *Consider the nn from Example 2 for predicting the acceptance of a paper. Let the instance to explain be* $(\mathbf{z}, \mathcal{T})$ *from Example 9 with* $\mathbf{z} = (1, 1, 0, \mathfrak{u})$ *and* $\mathcal{T} = \{A, B\}$*.*

The predictions of the nn correspond to a majority voting on the values of the features. In order for a paper to be accepted or classified as borderline, then the sum of feature values needs to be at least of two. Since in the point $\mathbf{z}$ *the only specified features that have value 1 are features 1 and 2, then in order to guarantee a prediction in* $\mathcal{T} = \{A, B\}$*, we need to fix the values of those two features, that is,* $\mathcal{X} = \{1, 2\}$ *is a WAXp for* $(\mathbf{z}, \mathcal{T})$*. Moreover, since none of those features is enough to guarantee the classification required on its own, then* $\mathcal{X} = \{1, 2\}$ *is subset-minimal and is a AXp.*

By duality, both $\mathcal{Y}_a = \{1\}$ *and* $\mathcal{Y}_a = \{2\}$ *are the CXps of the example. Changing only the value of those features will allow to change the prediction to a class not in* $\mathcal{T}$*. For example* $\kappa((0, 1, 0, 0)) = R \notin \mathcal{T}$*.*

4.4. Additional Results

This section uncovers additional properties of explanations, namely when inputs are partially specified. These properties allow relating explanations with different subsets of unspecified features, and can be used for example for the on-demand enumeration of explanations.

Throughout this section we consider a generalized explanation problem $\mathcal{E}'_r = (\mathcal{M}, (\mathbf{z}, \mathcal{T}))$, with $\kappa'(\mathbf{z}) = \mathcal{T}$. The index r denotes an order relation on the number of unspecified features, i.e. for $\mathcal{E}'_{r_1}$ and $\mathcal{E}'_{r_2}$, with $r_1 < r_2$, then $\mathcal{E}'_{r_1}$ has no more unspecified features than $\mathcal{E}'_{r_2}$. We start by investigating properties of AXp's and CXp's for explanation problems that refine $\mathcal{E}'_r$, i.e. the set of specified features $\mathcal{S}_j$ is contained in $\mathcal{S}$. Afterwards, we investigate properties of AXp's and CXp's for explanation problems that relax $\mathcal{E}'$, i.e. the set of specified features $\mathcal{S}_j$ contains $\mathcal{S}_r$.

Let $(\mathbf{v}, c)$ be such that $\mathbf{v} \sqsubseteq \mathbf{z}$, $c \in \mathcal{T}$, and $\kappa(\mathbf{v}) = c$ ($\kappa'(\mathbf{v}) = \{c\} \subseteq \mathcal{T}$). Moreover, let $\mathcal{E}'_0 = (\mathcal{M}, (\mathbf{v}, \mathcal{T}))$ be an explanation problem given $\mathcal{E}'_r$ and concretized $\mathbf{v}$. We can thus establish the following:

Proposition 4. *Given the definitions of* $\mathcal{E}'_r$ *and* $\mathcal{E}'_0$ *above, then:*

1. *If* $\mathcal{X} \in \mathbb{A}_{\mathfrak{u}}(\mathcal{E}'_r)$*, then* $\mathcal{X} \in \mathbb{A}_{\mathfrak{u}}(\mathcal{E}'_0)$*, i.e.* $\mathcal{X}$ *is an AXp for* $\mathcal{E}'_0$ *and* $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}'_r) \subseteq \mathbb{A}_{\mathfrak{u}}(\mathcal{E}'_0)$*.*
2. *If* $\mathcal{Y} \in \mathbb{C}_{\mathfrak{u}}(\mathcal{E}'_r)$*, then there exists* $\mathcal{Z} \subseteq \mathcal{U}$ *such that* $\mathcal{W} = \mathcal{Y} \cup \mathcal{Z} \in \mathbb{C}_{\mathfrak{u}}(\mathcal{E}'_0)$*, i.e. each* $\mathcal{Y} \cup \mathcal{Z}$ *is a CXp for* $\mathcal{E}'_0$ *and* $\forall(\mathcal{Y} \in \mathbb{C}_{\mathfrak{u}}(\mathcal{E}'_r))\exists(\mathcal{W} \in \mathbb{C}_{\mathfrak{u}}(\mathcal{E}'_0)).[\mathcal{Y} \subseteq \mathcal{W}]$*.*

Proof. *(4.1)* Suppose $\mathcal{X} \in \mathbb{A}_{\mathtt{u}}(\mathcal{E}'_r)$, then $\mathcal{X}$ is a weakAXp of $\mathcal{E}'_0$ because $\mathbf{v} \sqsubseteq \mathbf{z}$. To prove the minimality we proceed by contradiction. Suppose by contradiction there exists an AXp $\mathcal{W}$ of $\mathcal{E}'_0$, such that, $\mathcal{W} \subsetneq \mathcal{X}$, then:

$$\forall(\mathbf{x} \in \mathbb{F}).\ \left[\bigwedge\nolimits_{i\in\mathcal{W}}(x_i = v_i)\right] \rightarrow (\kappa(\mathbf{x}) \in \mathcal{T})$$

Then, because $\mathcal{W} \subsetneq \mathcal{X} \subset \mathcal{S}$:

$$\forall(\mathbf{x} \in \mathbb{F}).\ \left[\bigwedge\nolimits_{i\in\mathcal{W}}(x_i = z_i)\right] \rightarrow (\kappa(\mathbf{x}) \in \mathcal{T})$$

that is, $\mathcal{W}$ is a weakAXp of $\mathcal{E}'_r$, which is a contradiction of minimality of the AXp $\mathcal{X}$ for $\mathcal{E}'_r$. Thus $\mathbb{A}_{\mathtt{u}}(\mathcal{E}'_r) \subseteq \mathbb{A}_{\mathtt{u}}(\mathcal{E}'_0)$.

(4.2) Consider $\mathcal{Y} \in \mathbb{C}_{\mathtt{u}}(\mathcal{E}'_r)$, then let $\mathbf{x} \in \mathbb{F}$ be such that:

$$\left(\wedge_{i\in\mathcal{S}\setminus\mathcal{Y}}(x_i = z_i)\right) \wedge (\kappa(\mathbf{x}) \notin \mathcal{T})$$

Because $\mathbf{v} \sqsubseteq \mathbf{z}$, then $\mathbf{x}$ also satisfies:

$$\left(\wedge_{i\in\mathcal{F}\setminus(\mathcal{Y}\cup\mathcal{U})}(x_i = v_i)\right) \wedge (\kappa(\mathbf{x}) \notin \mathcal{T})$$

that means, $\mathcal{Y} \cup \mathcal{U}$ is a weakCXp of $\mathcal{E}'_0$. From the weakCXp $\mathcal{Y} \cup \mathcal{U}$ we can obtain CXp's contained in $\mathcal{Y} \cup \mathcal{U}$.
What is left to prove is that $\mathcal{Y} \subseteq \mathcal{W}$ for any CXp $\mathcal{W} \subseteq \mathcal{Y} \cup \mathcal{U}$. Consider a CXp $\mathcal{W} \subseteq \mathcal{Y}\cup\mathcal{U}$. Let $I_{\mathcal{Y}} = \mathcal{Y}\setminus\mathcal{W}$, and $\mathcal{Z} = \mathcal{W}\setminus\mathcal{Y}$. Suppose by contradiction that $I_{\mathcal{Y}} \neq \emptyset$. Observe that $I_{\mathcal{Y}} \not\subseteq \mathcal{W}$, $\mathcal{Z} \subseteq \mathcal{U}$, and that $\mathcal{W} = (\mathcal{Y} \setminus I_{\mathcal{Y}}) \cup \mathcal{Z}$. Because $\mathcal{W}$ is a CXp of $\mathcal{E}'_0$, there exists $\mathbf{x} \in \mathbb{F}$ such that:

$$\left(\wedge_{i\in\mathcal{F}\setminus\mathcal{W}}(x_i = v_i)\right) \wedge (\kappa(\mathbf{x}) \notin \mathcal{T})$$

then it is also true that $\mathbf{x}$ satisfies:

$$\left(\wedge_{i\in\mathcal{F}\setminus[(\mathcal{Y}\setminus I_{\mathcal{Y}})\cup\mathcal{U}]}(x_i = z_i)\right) \wedge (\kappa(\mathbf{x}) \notin \mathcal{T})$$

that is:

$$\left(\wedge_{i\in\mathcal{S}\setminus(\mathcal{Y}\setminus I_{\mathcal{Y}})}(x_i = z_i)\right) \wedge (\kappa(\mathbf{x}) \notin \mathcal{T})$$

that means $\mathcal{Y}\setminus I_{\mathcal{Y}}$ is a weakCXp for $\mathcal{E}'_r$, which is a contradiction of minimality of CXp $\mathcal{Y}$ for $\mathcal{E}'_r$. Thus $\mathcal{Y} \subseteq \mathcal{W}$ for any CXp $\mathcal{W} \subseteq \mathcal{Y} \cup \mathcal{U}$ of $\mathcal{E}'_0$. □

Nested duality. Proposition 3 and Proposition 4 reveal an important property of duality among AXp's and CXp's that holds in general. The rest of this section assumes the following explanation scenario. We consider the computation of an AXp $\mathcal{X}$ of some explanation problem $\mathcal{E}'$, starting from $\mathbf{z}$, that partitions $\mathcal{F}$ into $\mathcal{S} = \mathcal{S}_r$ and $\mathcal{U} = \mathcal{U}_r$. Moreover, let $r \le j \le r+s$. In the process of computing the AXp $\mathcal{X}$, let the computed weak AXp's be given by the sequence $\langle \mathcal{X}_r = \mathcal{S}_r, \mathcal{X}_{r+1}, \ldots, \mathcal{X}_{r+s} = \mathcal{X} \rangle$. (At each step of executing the AXp extraction algorithm, the set of fixed features is $\mathcal{X}_j$.) Since for each $\mathcal{X}_j$, problem $\mathcal{E}'_j$, and CXp's, $\mathbb{A}_{\mathtt{u}}(\mathcal{E}'_j)$ and $\mathbb{C}_{\mathtt{u}}(\mathcal{E}'_j)$. For each $\mathcal{E}'_j$, the specified features are $\mathcal{S}_j = \mathcal{X}_j$ and the unspecified features are $\mathcal{U}_j = \mathcal{F} \setminus \mathcal{X}_j$. Clearly, for $j = r$, $\mathcal{S}_r = \mathcal{S}$ and $\mathcal{U}_r = \mathcal{U}$. Given the explanation problem $\mathcal{E}'$, and the sequence $\langle \mathcal{X}_r = \mathcal{S}, \mathcal{X}_{r+1}, \ldots, \mathcal{X}_{r+s} = \mathcal{X} \rangle$, then the following results hold.

Proposition 5 (Relating AXp's and CXp's)**.** *Given the assumptions above, it is the case that,*

1. *If $\mathcal{W} \in \mathbb{A}_{\mathtt{u}}(\mathcal{E}'_k)$, then $\mathcal{W} \in \mathbb{A}_{\mathtt{u}}(\mathcal{E}'_j)$, for any j, k with $r \le j \le k \le r+s$;*
2. *If $\mathcal{Y} \in \mathbb{C}_{\mathtt{u}}(\mathcal{E}'_k)$, then there exists $\mathcal{Z}_r \subseteq \mathcal{X}_r$ such that $\mathcal{Y} \cup \mathcal{Z}_r \in \mathbb{C}_{\mathtt{u}}(\mathcal{E}'_j)$, for any j, k with $r \le j \le k \le r+s$.*

Proof. (5.1) Consider the process of computing $\mathcal{X}$, and the computed weak AXps given by the sequence $\langle \mathcal{X}_r = \mathcal{S}, \mathcal{X}_{r+1}, \ldots, \mathcal{X}_{r+s} = \mathcal{X} \rangle$. Between each pair $(\mathcal{X}_j,\ \mathcal{X}_{j+1})$ $(r \le j < r+s)$, a feature $i_j (\in \mathcal{X}_j)$ is tested to be left unspecified, that is, check if the predicate $\mathbb{P}_{\text{axp}}(\mathcal{X}_j \setminus \{i_j\})$ is true or not. If the predicate is false, then $\mathcal{X}_{(j+1)} = \mathcal{X}_j$, and obviously, $\mathbb{A}_{\mathtt{u}}(\mathcal{E}'_{j+1}) = \mathbb{A}_{\mathtt{u}}(\mathcal{E}'_j)$.

If the predicate is true, then $\mathcal{X}_{j+1} = \mathcal{X}_j \setminus \{i_j\}$. Consider $\mathcal{W} \in \mathbb{A}_{\mathtt{u}}(\mathcal{E}'_{j+1})$, then $\mathcal{W} \subseteq \mathcal{X}_{j+1}$ and $\mathsf{AXp}(\mathcal{W}) = \top$. As such, $\mathcal{W} \subseteq \mathcal{X}_j$, and $\mathcal{W}$ is subset minimal verifying $\mathsf{WeakAXp}(\mathcal{W}) = \top$; that is $\mathcal{W} \in \mathbb{A}_{\mathtt{u}}(\mathcal{E}'_j)$.

We just proved that $\mathbb{A}_{\mathtt{u}}(\mathcal{E}'_{j+1}) \subseteq \mathbb{A}_{\mathtt{u}}(\mathcal{E}'_j)$ for $r \le j < r+s$. By transitivity, if $\mathcal{W} \in \mathbb{A}_{\mathtt{u}}(\mathcal{E}'_k)$, then $\mathcal{W} \in \mathbb{A}_{\mathtt{u}}(\mathcal{E}'_j)$, with $r \le j \le k \le r+s$.

(5.2) As before consider the process of computing $\mathcal{X}$, and the sequence $\langle \mathcal{X}_r = \mathcal{S}, \mathcal{X}_{r+1}, \ldots, \mathcal{X}_{r+s} = \mathcal{X} \rangle$. Between each pair $(\mathcal{X}_j,\ \mathcal{X}_{j+1})$ $(r \le j < r+s)$, a feature $i_j (\in \mathcal{X}_j)$ is tested to be left unspecified, that is, check if the predicate $\mathbb{P}_{\text{axp}}(\mathcal{X}_j \setminus \{i_j\})$ is true or not. If the predicate is false, then $\mathcal{X}_{j+1} = \mathcal{X}_j$, and obviously, $\mathbb{C}_{\mathtt{u}}(\mathcal{E}'_{j+1}) = \mathbb{C}_{\mathtt{u}}(\mathcal{E}'_j)$. Let $\mathcal{Z}_j = \emptyset (\subseteq \mathcal{X}_j)$, then if $\mathcal{Y} \in \mathbb{C}_{\mathtt{u}}(\mathcal{E}'_{j+1})$ then $\mathcal{Y} \cup \mathcal{Z}_j \in \mathbb{C}_{\mathtt{u}}(\mathcal{E}'_j)$.

If the predicate is true, then $\mathcal{X}_{j+1} = \mathcal{X}_j \setminus \{i_j\}$. Consider $\mathcal{Y} \in \mathbb{C}_{\mathtt{u}}(\mathcal{E}'_{j+1})$, then $\mathcal{Y} \subseteq \mathcal{X}_{j+1}$ is a subset minimal set satisfying $\mathsf{WeakCXp}(\mathcal{Y}) = \top$, that is:

$$\exists (\mathbf{x} \in \mathbb{F}). \left(\wedge_{i \in \mathcal{X}_{j+1} \setminus \mathcal{Y}} (x_i = z_i) \right) \wedge (\kappa(\mathbf{x}) \notin \mathcal{T})$$

There are two cases to consider, depending on the necessity of including feature i_j in the CXp for the set of fixed features $\mathcal{X}_j$. In the first case, feature i_j is required in the CXp. This can happen because i_j is not a part of $\mathcal{X}_{j+1}$, as such in this case, it happens that:

$$\forall(\mathbf{x} \in \mathbb{F}).\left(\wedge_{i\in\mathcal{X}_j\setminus\mathcal{Y}}(x_i = z_i)\right) \rightarrow (\kappa(\mathbf{x}) \in \mathcal{T})$$

In this case, let $\mathcal{Z}_j = \{i_j\}(\subseteq \mathcal{X}_j)$, then we can consider $\mathcal{Y} \cup \mathcal{Z}_j$, and because $\mathsf{WeakCXp}(\mathcal{Y}) = \top$, we know that:

$$\exists(\mathbf{x} \in \mathbb{F}).\left(\wedge_{i\in\mathcal{X}_j\setminus(\mathcal{Y}\cup\mathcal{Z}_j)}(x_i = z_i)\right) \wedge (\kappa(\mathbf{x}) \notin \mathcal{T})$$

The subset minimality of $\mathcal{Y} \cup \mathcal{Z}_j$ is due to the subset minimality of $\mathcal{Y}$ in $\mathcal{X}_{j+1}$.

In the second case, feature i_j is not required in the CXp, that is, it is enough to consider $\mathcal{Y}$ in the set $\mathcal{X}_j$ so that:

$$\exists(\mathbf{x} \in \mathbb{F}).\left(\wedge_{i\in\mathcal{X}_j\setminus\mathcal{Y}}(x_i = z_i)\right) \wedge (\kappa(\mathbf{x}) \notin \mathcal{T})$$

Let $\mathcal{Z}_j = \emptyset(\subseteq \mathcal{X}_j)$, then if $\mathcal{Y} \in \mathbb{C}_{\text{u}}(\mathcal{E}'_{j+1})$ then $\mathcal{Y} \cup \mathcal{Z}_j \in \mathbb{C}_{\text{u}}(\mathcal{E}'_j)$.

We have proven that for each pair $(\mathcal{X}_j,\ \mathcal{X}_{j+1})$ $(r \le j < r+s)$, if $\mathcal{Y} \in \mathbb{C}_{\text{u}}(\mathcal{E}'_{j+1})$, then there exists $\mathcal{Z}_j \subseteq \mathcal{X}_j$ such that $\mathcal{Y} \cup \mathcal{Z}_j \in \mathbb{C}_{\text{u}}(\mathcal{E}'_j)$.

Consider now j, k such that $r \le j \le k \le r+s$, and $\mathcal{Y} \in \mathbb{C}_{\text{u}}(\mathcal{E}'_k)$. Consider the sequence of pairs $\langle(\mathcal{X}_{k-1}, \mathcal{X}_k), \ldots, (\mathcal{X}_j, \mathcal{X}_{j+1})\rangle$. Starting from the first pair, apply the previous proven result and obtain that there exists $\mathcal{Z}_{k-1} \subseteq \mathcal{X}_{k-1}$ such that $\mathcal{Y} \cup \mathcal{Z}_{k-1} \in \mathbb{C}_{\text{u}}(\mathcal{E}'_{k-1})$. For the next pair use the new CXp $\mathcal{Y} \cup \mathcal{Z}_{k-1}$ and obtain that there exists $\mathcal{Z}_{k-2} \subseteq \mathcal{X}_{k-2}$ such that $\mathcal{Y} \cup \mathcal{Z}_{k-1} \cup \mathcal{Z}_{k-2} \in \mathbb{C}_{\text{u}}(\mathcal{E}'_{k-2})$. Proceed in a similar way until the last pair to obtain that there exists $\mathcal{Z}_j \subseteq \mathcal{X}_j$ such that $\mathcal{Y} \cup \mathcal{Z}_{k-1} \cup \ldots \mathcal{Z}_j \in \mathbb{C}_{\text{u}}(\mathcal{E}'_j)$. Since $\mathcal{X}_{k-1} \subseteq \ldots \subseteq \mathcal{X}_j$, and $\mathcal{Z}_{k-1} \cup \ldots \mathcal{Z}_j \subseteq \mathcal{X}_j$, then we can consider $\mathcal{Z} = \bigcup_{t=j}^{k-1} \mathcal{Z}_t$, $\mathcal{Z} \subseteq \mathcal{X}_j$, and $\mathcal{Y} \cup \mathcal{Z} \in \mathbb{C}_{\text{u}}(\mathcal{E}'_j)$ as required. □

Moreover, we also have the following duality result.

Proposition 6 (Nested Duality)**.** *Each AXp in $\mathbb{A}_{\text{u}}(\mathcal{E}'_j)$ is a minimal hitting set of the CXp's in $\mathbb{C}_{\text{u}}(\mathcal{E}'_j)$ and vice-versa, for $r \le j \le r+s$.*

Proof. Consider the process of computing $\mathcal{X}$, and the computed weak AXps given by the sequence $\langle\mathcal{X}_r = \mathcal{S}, \mathcal{X}_{r+1}, \ldots, \mathcal{X}_{r+s} = \mathcal{X}\rangle$. Between each pair $(\mathcal{X}_j,\ \mathcal{X}_{j+1})$ $(r \le j < r+s)$, a feature $i_j(\in \mathcal{X}_j)$ is tested to be left unspecified, that is, check if the predicate $\mathbb{P}_{\text{axp}}(\mathcal{X}_j \setminus \{i_j\})$ is true or not. If the predicate is true then $\mathcal{X}_{j+1} = \mathcal{X}_j \setminus \{i_j\}$, otherwise $\mathcal{X}_{j+1} = \mathcal{X}_j$.

Consider j such that $r \leq j \leq r+s$. The set $\mathcal{X}_j$ divide $\mathcal{F}$ in two, the set of fixed features $\mathcal{S} = \mathcal{X}_j$ and the set of unspecified features $\mathcal{U} = \mathcal{F} \setminus \mathcal{X}_j$. Let $\mathbf{z}'$ be such that $z'_i = z_i$ if $i \in \mathcal{X}_j$, undefined otherwise. Then, $\mathcal{E}'_j = (\mathcal{M}, (\mathbf{z}', \mathcal{T}))$ is a generalized explanation problem. Due to Proposition 3, then each AXp in $\mathbb{A}_{\mathfrak{u}}(\mathcal{E}'_j)$ is a minimal hitting set of the CXp's in $\mathbb{C}_{\mathfrak{u}}(\mathcal{E}'_j)$, and vice-versa. $\square$

5. General Algorithms

In the previous sections, we have introduced the concepts of abductive explanations (AXp's) and contrastive explanations (CXp's) as formal predicates that can be applied over different classes of machine learning models. In this section, we present two general approach algorithms. The first one is for computing one AXp or one CXp. The second one is for enumerating the whole set of AXps or CXps. The algorithms presented here are general in the sense that can be applied to different classes of ML models, assuming certain basic predicates can be implemented with these ML models.

5.1. Computing One Explanation

We start with a general approach for computing one explanation, based on exploiting automated reasoners, such as SAT, MILP, or SMT solvers as oracles. Consistency checking with a reasoner for a theory $\mathbb{T}$ on a $\mathbb{T}$-formula $\varphi_{\mathbb{T}}$ can be used to check whether there exists a model in φ given $\mathbb{T}$, and is represented as $\mathbf{CO}(\varphi_{\mathbb{T}})$. These theory reasoners operate on formulas of a logic language. Given some logic formula φ, $[\![\varphi]\!]_{\mathbb{T}}$ denotes the encoding of φ in the logic language of $\mathbb{T}$. For simplicity we use $[\![\varphi]\!]$. Specialized encodings for the different classifiers are handled in Section 6. In this sections an encoding is assumed for each of the classifiers.

Consider a feature point (with missing inputs) $\mathbf{z}$, the associated partition of the set of features in specified and unspecified features $(\mathcal{S},\mathcal{U})$ $(\mathcal{F} = \mathcal{S} \cup \mathcal{U})$, and a target predicted set of classes to explain $\mathcal{T} \subset \mathcal{K}$. Given a set $\mathcal{W} \subseteq \mathcal{S}$, we can ask a reasoner to check for the consistency of the following statement:

$$\mathbf{CO}\left(\left[\!\!\left[\bigwedge_{i \in \mathcal{W}} (x_i = z_i) \land \kappa(x) \notin \mathcal{T}\right]\!\!\right]\right) \tag{18}$$

where $\kappa(x)$ is the classifier. The function $\mathbf{CO}$ is a consistency oracle that checks whether the formula in theory $\mathbb{T}$ is satisfiable, i.e. it returns true ($\top$) or false ($\bot$).

The explanations in the previous section can be decided with calls to the consistency oracle. In the case of AXps, by double-negating the formula of a WeakAXp (12), one gets:

$$\neg\exists(x \in \mathbb{F}).\left[\!\!\left[\bigwedge_{i\in\mathcal{X}}(x_i = v_i) \land \kappa(x) \notin \mathcal{T}\right]\!\!\right] \tag{19}$$

By setting $\mathcal{W} = \mathcal{X}$, it is clear that (19) holds if (18) does not hold.

In the case of CXps, we directly encode the formula WeakCXp (13) as a consistency check by setting $\mathcal{W} = \mathcal{Y}$.

The computation of a single AXp or a single CXp can be achieved by adapting existing algorithms provided a few requirements are met. First, reasoning in theory $\mathbb{T}$ must be monotone, i.e. consistency is preserved if constraints are removed from the formula, and inconsistency is preserved if constraints are added to the formula. Second, for the computation of one AXp, the predicate to consider is:

$$\mathbb{P}_{axp}(\mathcal{X}) \triangleq \neg\mathbf{CO}\left(\left[\!\!\left[\bigwedge_{i\in\mathcal{X}}(x_i = v_i) \land \kappa(x) \notin \mathcal{T}\right]\!\!\right]\right) \tag{20}$$

For the computation of one CXp, the predicate to consider is:

$$\mathbb{P}_{cxp}(\mathcal{Y}) \triangleq \mathbf{CO}\left(\left[\!\!\left[\bigwedge_{i\in\mathcal{S}\setminus\mathcal{Y}}(x_i = v_i) \land \kappa(x) \notin \mathcal{T}\right]\!\!\right]\right) \tag{21}$$

Given that reasoning in theory $\mathbb{T}$ is monotone, the computation of (20) and (21) is also monotone with respect to the set of features $\mathcal{S}$. This allows extending the algorithms for computing one explanation.

Algorithm 1 computes a single AXp or CXp, depending on the predicate $\mathbb{P}$ given as input, either $\mathbb{P}_{axp}$ or $\mathbb{P}_{cxp}$ respectively. The initial seed set $\mathcal{R}$ contains the specified features to consider, that is $\mathcal{R} \subseteq \mathcal{S}$. As long as precondition $\mathbb{P}(\mathcal{R})$ holds (Line 2), any set $\mathcal{R}$ can be considered. At every iteration of the loop, the algorithm checks if the current set $\mathcal{W}$ whitout feature i is still an Xp (Line 4) according to predicate $\mathbb{P}$. If so, the feature is removed from the set $\mathcal{W}$, otherwise the feature is necessary and it is kept. When all features have been tested, the algorithm returns the resulting set $\mathcal{W}$ as the Xp.

Example 12 (Compute One AXp – Decision Trees)**.** *Taking Example 1 with the DT in Figure 1, and the instance in Example 8, an example execution of*

Algorithm 1 Finding one AXp/CXp

```
    Input:  Seed $\mathcal{R} \subseteq \mathcal{S}$, Predicate $\mathbb{P}$
    Output: One XP $\mathcal{W}$

1: procedure oneXP($\mathcal{R}$, $\mathbb{P}$)
2:     $\mathcal{W} \leftarrow \mathcal{R}$                              ▷ Initialization: $\mathbb{P}(\mathcal{W})$ holds
3:     for $i \in \mathcal{R}$ do                          ▷ Loop invariant: $\mathbb{P}(\mathcal{W})$ holds
4:         if $\mathbb{P}(\mathcal{W} \setminus \{i\})$ then
5:             $\mathcal{W} \leftarrow \mathcal{W} \setminus \{i\}$              ▷ If $\mathbb{P}(\mathcal{W} \setminus \{i\})$, update $\mathcal{W}$
6:     return $\mathcal{W}$                         ▷ Returned set $\mathcal{W}$: $\mathbb{P}(\mathcal{W})$ holds
```

Iter.	$i/Pred.$	$\mathbb{P}_{axp}$	False Nodes	$\mathcal{W}$
				$\{1,2,4,6\}$
1	$1/H_1$	$\top$		$\{2,4,6\}$
2	$2/L_1$	$\top$		$\{4,6\}$
3	$4/P_1, P_2$	$\bot$	7, 13, 15	$\{4,6\}$
4	$6/W_1$	$\bot$	11	$\{4,6\}$

Table 2: Execution of Algorithm 1 for computing an AXp for the decision tree in Figure 1 and instance to explain $((35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.2), \{\oplus, \oplus\oplus\})$.

Algorithm 1 for computing an AXp is as in Table 2. In Table 2, the False Nodes column show the number of the terminal nodes in the DT that do not verify the predicate.

The example instance is $\mathbf{z} = (35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.2)$*, which means the specified features are* $\mathcal{S} = \{1,2,4,6\}$*, and the unspecified features as* $\mathcal{U} = \{3,5\}$*. For the specified features we have the values* $HCT = 35.0$*,* $Lump. = 0.25$*,* $PLT = 100.0$*,* $WBC = 5.2$*. In terms of predicates this means* $H_1 = L_1 = P_1 = P_2 = W_1 = 1$*. The target classes for explaining the prediction is* $\mathcal{T} = \{\oplus, \oplus\oplus\}$*.*

In the example execution, the algorithm starts with a seed $\mathcal{R}$ *equal to set of all specified features* $\mathcal{S} = \{1,2,4,6\}$ *and builds an AXp in* $\mathcal{W}$*, initially* $\mathcal{W} = \mathcal{R} = \mathcal{S}$*. Iteratively, it tests removing a feature with the predicate* $\mathbb{P}_{axp}$*. It will remove the feature from* $\mathcal{W}$ *if the predicate is true.*

In our example, in both iterations 1 and 2, the predicate $\mathbb{P}_{axp}$ *returns* *true, so the tested features 1 and 2 are dropped from $\mathcal{W}$. For iteration 1, the predicate is called with features 2, 4, and 6 ($\mathbb{P}_{axp}(\{2,4,6\})$. By looking at the DT (Figure 1), we can see that starting from the root node and fixing the selected specified features (2, 4, and 6), the reachable nodes are only nodes 8 and 14, each classified as $\oplus\oplus$ and $\oplus$ respectively. Both these classes are in $\mathcal{T}$, meaning any point in the feature space respecting the values of the specified selected features (according to $\mathbf{z}$) is classified with a class in $\mathcal{T}$. Since $\mathbb{P}_{axp}(\mathcal{X})$ is true iff there is no point in the feature space that respects the values of features in $\mathcal{X}$ whose classification is outside $\mathcal{T}$, then $\mathbb{P}_{axp}(\{2,4,6\})$ is true. Similarly, for iteration 2, the predicate is called with $\mathbb{P}_{axp}(\{4,6\})$, and the terminal nodes reachable from the root are the same 8 and 14. By the same reasoning, $\mathbb{P}_{axp}(\{4,6\})$ is true.*

On the other hand, the predicate in iterations 3 and 4 returns false, so the tested features 4 and 6 are kept in $\mathcal{W}$. For iteration 3, the predicate is called with feature 6 only ($\mathbb{P}_{axp}(\{6\})$). By looking at the DT, we can see that fixing only the value of feature 6, then from the root, the reachable nodes are 7, 8, 12, 13, 14 and 15. But, nodes 7, 13 and 15 are classified with $\ominus\ominus$ or $\ominus$, meaning there exist at least one point in the feature space that respects the selected specified feature 6 and classifies outside the target class $\mathcal{T}$ (e.g. $v_1 = (41, 0.6, 4, 200, 37, 5.2)$, $\kappa(v_1) = \ominus\ominus$), thus $\mathbb{P}_{axp}(\{6\})$ is false. Similarly, for iteration 4, the predicate is called with feature 4 only ($\mathbb{P}_{axp}(\{6\})$). The terminal nodes reachable from the root are 8, 10, 11, 14. But node 14 is classified with $\ominus$, which means there exist at least one point in the feature space that respects the selected specified feature 4 and classifies outside the target class $\mathcal{T}$ (e.g. $v_2 = (42, 0.6, 4, 100, 37, 6.2)$, $\kappa(v_2) = \ominus$), thus $\mathbb{P}_{axp}(\{4\})$ is false.

The algorithm then terminates then with $\mathcal{W} = \{4,6\}$ as the reported AXp.

Example 13 (Compute One CXp – Decision Trees)**.** *Taking Example 1 with the DT in Figure 1, and the same instance in Example 8, an example execution of Algorithm 1 for computing a CXp is as in Table 3. In Table 3, the True Nodes column show the number of the terminal nodes in the DT that do verify the predicate.*

The example instance considered is the same as in the previous example $((35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.2), \{\oplus, \oplus\oplus\})$. In the example execution, the algorithm starts with a seed $\mathcal{R}$ equal to the set of all specified features $\mathcal{S} = \{1,2,4,6\} = \mathcal{R}$ and builds an CXp in $\mathcal{W}$, initially $\mathcal{W} = \mathcal{R}$. Iteratively, it tests removing a feature with the given predicate $\mathbb{P}_{cxp}$, removing from $\mathcal{W}$ if

Iter.	$i/Pred.$	$\mathbb{P}_{cxp}$	True Nodes	$\mathcal{W}$
				$\{1,2,4,6\}$
1	$1/H_1$	$\top$	7,13,15	$\{2,4,6\}$
2	$2/L_1$	$\top$	13,15	$\{4,6\}$
3	$4/P_1,P_2$	$\bot$		$\{4,6\}$
4	$6/W_1$	$\top$	13,15	$\{4\}$

Table 3: Execution of Algorithm 1 for computing an CXp for the DT in Figure 1 and instance to explain $((35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.2), \{\oplus, \oplus\oplus\})$.

Iter.	i	$\mathbb{P}_{axp}$	$\mathcal{W}$
			$\{1,2,3\}$
1	1	$\bot$	$\{1,2,3\}$
2	2	$\bot$	$\{1,2,3\}$
3	3	$\top$	$\{1,2\}$

(a)

Iter.	i	$\mathbb{P}_{cxp}$	$\mathcal{W}$
			$\{1,2,3\}$
1	1	$\top$	$\{2,3\}$
2	2	$\bot$	$\{2,3\}$
3	3	$\top$	$\{2\}$

(b)

Table 4: Example executions of Algorithm 1 for the neural network in Example 2 with instance $((1, 1, 0, \mathfrak{u}), \{A, B\})$. (a) Computing one AXp (b) Computing one CXp

the predicate is true. In our example, at iteration 3 the predicate is not satisfied, so features 4 is not removed from $\mathcal{W}$, unlike the other features in the remaining iterations. Thus, the algorithm terminates with $\mathcal{W} = \{4\}$ as the reported CXp.

Example 14 (Compute One Xp – Neural Networks)**.** *Consider the NN in Example 2 for which we wish to compute an explanation for the point $(1, 1, 0, \mathfrak{u})$ with prediction $\{A, B\}$ as in Example 9. Example executions of Algorithm 1 for computing XPs are shown in Table 4. In the two executions, initially the specified features are $\mathcal{S} = \{1, 2, 3\}$ and the unspecified feature is $\mathcal{U} = \{4\}$.*

In the example execution for computing an AXp (in Table 4(a)), the predicate $\mathbb{P}_{\text{axp}}$ checks if there exists a point in the feature space that classifies in $\mathcal{T} = \{A, B\}$. The algorithm starts with the seed $\mathcal{R} = \mathcal{S} = \{1, 2, 3\}$. In iteration 1, the predicate is called with features $\{2, 3\}$, and observing the neural network classifier we can see that it classifies in $\{A, B, R\}$. If feature 4 and

1 are 1, the classifier classifies as A, if feature 4 and 1 are 0, the classifier classifies as R, otherwise the classifier classifies as B. Since there are points classifying out of $\mathcal{T}$, the predicate is false and the feature is not removed from $\mathcal{W}$. In iteration 2, the predicate is called with features $\{1, 3\}$, and observing the neural network classifier we can see that it classifies in $\{A, B, R\}$. The reasoning is the same as in iteration 1. Since there are points classifying out of $\mathcal{T}$, the predicate is false and the feature is not removed from $\mathcal{W}$. In iteration 3, the predicate is called with feature $\{1, 2\}$, and observing the neural network classifier we can see that it classifies in $\{A, B\}$. Since there are no points classifying out of $\mathcal{T}$, the predicate is true and the feature is removed from $\mathcal{W}$. The algorithm terminates with $\mathcal{W} = \{1, 2\}$ as the reported AXp.

In the example execution for computing an CXp (in Table 4(b)), the predicate $\mathbb{P}_{\text{cxp}}$ checks if there exists a point in the feature space that classifies different than $\mathcal{T} = \{A, B\}$. The algorithm starts with the seed $\mathcal{R} = \mathcal{S} = \{1, 2, 3\}$. In iteration 1, the predicate is called with features $\{2, 3\}$, and observing the neural network classifier we can see that it classifies in $\{A, B, R\}$. Since there are points classifying out of $\mathcal{T}$, the predicate is true and the feature is removed from $\mathcal{W}$. In iteration 2, the predicate is called with feature $\{3\}$, and observing the neural network classifier we can see that it classifies in $\{A, B\}$. Since there are no points classifying out of $\mathcal{T}$, the predicate is false and the feature is not removed from $\mathcal{W}$. In iteration 3, the predicate is called with feature $\{2\}$, and observing the neural network classifier we can see that it classifies in $\{A, B, R\}$. Since there are points classifying out of $\mathcal{T}$, the predicate is true and the feature is removed from $\mathcal{W}$. The algorithm terminates with $\mathcal{W} = \{2\}$ as the reported CXp.

5.2. Enumerating Explanations

In this section, we present an algorithm for computing all the explanations. Given an explanation problem, and a set of computed explanations (AXp or CXp), the enumeration of explanations can be done by computing a new explanation among those not in the set of computed explanations. One solution for enumerating AXp is to compute all CXp, and then use hitting set dualization for computing the AXp. Unfortunately, this approximation could be costly because the number of CXp is often exponential. From a work on the enumeration of MUSes [47] (Algorithm MARCO), the solution is to iteratively compute AXp/CXp by exploiting hitting set duality, using a SAT solver for iteratively picking a set of features to be a seed for computing one AXp or one CXp. The enumeration of AXp/CXp builds on the general

Algorithm 2 Enumerating all AXp/CXp

```
1: procedure enumXP()
2:     $\mathcal{H} \leftarrow \emptyset$
3:     repeat
4:         $(\mathsf{outc}, \boldsymbol{s}) \leftarrow \mathsf{SAT}(\mathcal{H})$
5:         if $\mathsf{outc} = \top$ then
6:             $\mathcal{R} \leftarrow \{i \in \mathcal{S} \mid s_i = 0\}$        ▷ $\mathcal{R}$: seed, *fixed* features
7:             $\mathcal{Q} \leftarrow \{i \in \mathcal{S} \mid s_i = 1\}$        ▷ $\mathcal{Q}$: *universal* features
8:             if $\mathbb{P}_{\mathrm{cxp}}(\mathcal{Q}) = \top$ then        ▷ $\mathcal{Q} \supseteq$ CXp
9:                 $\mathcal{P} \leftarrow \mathsf{oneXP}(\mathcal{Q}, \mathbb{P}_{\mathrm{cxp}})$
10:                $\mathsf{reportCXp}(\mathcal{P})$
11:                $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\vee_{i\in\mathcal{P}} \neg s_i)\}$
12:            else        ▷ $\mathcal{R} \supseteq$ AXp
13:                $\mathcal{P} \leftarrow \mathsf{oneXP}(\mathcal{R}, \mathbb{P}_{\mathrm{axp}})$
14:                $\mathsf{reportAXp}(\mathcal{P})$
15:                $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\vee_{i\in\mathcal{P}} s_i)\}$
16:    until $\mathsf{outc} = \bot$
```

purpose algorithm for computing one AXp/CXp from a seed. The SAT oracle returns an output ($\top$ or $\bot$) and an assignment $\boldsymbol{\mu}$ in case the output is $\top$.

Consider a feature point (with missing inputs) $\mathbf{z}$, the associated partition of the set of features in specified and unspecified features $(\mathcal{S},\mathcal{U})$ $(\mathcal{F} = \mathcal{S} \cup \mathcal{U})$, and a target predicted set of classes to explain $\mathcal{T} \subset \mathcal{K}$. A general purpose approach for the enumeration of explanations is shown in Algorithm 2. Unspecified features are allowed to take any value from their domain, and so are treated as universal features. The algorithm iteratively calls a SAT solver to compute a candidate WeakCXp/WeakAXp (line 4). The candidate is checked for being a WeakCXp (line 8). If so, then the `oneXP` function is called with the candidate as seed and the $\mathbb{P}_{cxp}$ predicate as inputs, to compute a new CXp (line 9). The new CXp explanation is reported (line 10), and a new clause is added to the set $\mathcal{H}$, to block the CXp explanation (line 11).

Otherwise, the `oneXP` function is called with the candidate as seed and the $\mathbb{P}_{axp}$ predicate as inputs, to compute a new AXp (line 13). The new AXp explanation is reported (line 14), and a new clause is added to the set $\mathcal{H}$, to block the AXp explanation (line 15).

When the SAT solver returns $\bot$, the algorithm terminates with all expla-

Iter.	outc, $\boldsymbol{s}$	$\mathcal{R} \mid \mathcal{Q}$	$\mathbb{P}_{cxp}(\mathcal{Q})$	$\mathcal{P}_{(a/c)}$	Add to $\mathcal{H}$
					$\emptyset$
1	$\top, \{s_1 = s_2 = s_4 = s_6 = 0\}$	$\{1,2,4,6\} \mid \{\}$	$\bot$	$\{4,6\}_{(a)}$	$(s_4 \vee s_6)$
2	$\top, \{s_1 = s_2 = s_6 = 0, s_4 = 1\}$	$\{1,2,6\} \mid \{4\}$	$\top$	$\{4\}_{(c)}$	$(\neg s_4)$
3	$\top, \{s_1 = s_2 = s_4 = 0, s_6 = 1\}$	$\{1,2,4\} \mid \{6\}$	$\bot$	$\{1,4\}_{(a)}$	$(s_1 \vee s_4)$
4	$\top, \{s_2 = s_4 = 0, s_1 = s_6 = 1\}$	$\{2,4\} \mid \{1,6\}$	$\top$	$\{1,6\}_{(c)}$	$(\neg s_1 \vee \neg s_6)$
5	$\bot$				

Table 5: Execution of Algorithm 2 for enumerating explanations for the DT in Figure 1 and instance to explain $((35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.2), \{\oplus, \oplus\oplus\})$. For each iteration, the table shows a column with the following information: the iteration number; the values of variables `outc` and $\boldsymbol{s}$ (line 4); the sets $\mathcal{R}$ and $\mathcal{Q}$ that result from the current $\boldsymbol{s}$ (lines 6 and 7); the value of the predicate $\mathbb{P}_{cxp}$ applied to the current $\mathcal{Q}$ (line 8); the value of variable $\mathcal{P}$ indexed with $_a$ or $_c$ depending on the variable obtaining a value in line 9 or line 13, and corresponding to an AXp or CXp respectively; a clause added to $\mathcal{H}$ in the end of the iteration.

nations reported.

In the examples that follow, we assumed a SAT solver biased towards false variables.[11]

Example 15 (Enumerate Xps - Decision Trees). *Consider the Example 1 with the DT in Figure 1, and the instance in Example 8, an example execution of Algorithm 2 for enumerating explanations is as in Table 5.*

The algorithm starts by computing a satisfying assignment with all variables assigned false (considering a biased SAT solver). It computes a candidate WeakCXp in variable $\mathcal{Q}$, which is empty. Since, empty is not a viable WeakCXp, then the candidate WeakAXp $\mathcal{R} = \{1, 2, 4, 6\}$ is used for computing the AXp $\{4, 6\}$, and the clause $(s_4 \vee s_6)$ is added to the CNF formula $\mathcal{H}$ to block the AXp from being computed again. The next three iterations follow a similar procedure, reporting the CXps $\{4\}$ and $\{1, 4\}$, and the AXp $\{1, 4\}$. Additionally, the clauses $(\neg s_4)$, $(s_1 \vee s_4)$, and $(\neg s_1 \vee \neg s_6)$ are added to $\mathcal{H}$, as blocking clauses.

[11] If a formula is satisfiable independently of the value of a variable x, then the SAT solver returns an assignment with value false for x. This is done to simplify the demonstration, it is not a requirement.

Iter.	outc, s	$\mathcal{R}\|\mathcal{Q}$	$\mathbb{P}_{cxp}(\mathcal{Q})$	$\mathcal{P}_{(a/c)}$	Add to $\mathcal{H}$
					$\emptyset$
1	$\top, \{s_o = s_h = s_w = 0\}$	$\{o, h, w\}\|\{\}$	$\bot$	$\{o, h\}_{(a)}$	$(s_o \vee s_h)$
2	$\top, \{s_h = s_w = 0, s_o = 1\}$	$\{h, w\}\|\{o\}$	$\top$	$\{o\}_{(c)}$	$(\neg s_o)$
3	$\top, \{s_o = s_w = 0, s_h = 1\}$	$\{o, w\}\|\{h\}$	$\top$	$\{h\}_{(c)}$	$(\neg s_h)$
4	$\bot$				

Table 6: Execution of Algorithm 2 for enumerating explanations for the DL in Figure B.7 and instance to explain $((\mathit{sunny}, \mathfrak{u}, \mathit{high}, \text{false}), \{\mathit{classes} = -\})$. The table shows similar information as Table 5.

In iteration five the SAT solver reports formula $\mathcal{H}$ to be unsatisfiable (all AXps and CXps have been reported) and the algorithm terminates.

For the following example executions of Algorithm 2 with the remaining running examples, we will only present a summary of the full resulting iterations, since the execution procedures are similar to the previous example.

Example 16 (Enumerate Xps - Decision Lists)**.** *Consider the Example 24 with the DL in Figure B.7, and the instance in Example 27, an example execution of Algorithm 2 for enumerating explanations is as in Table 6.*

The algorithm starts by computing a satisfying assignment with all variables assigned false (considering a biased SAT solver). It computes a candidate WeakCXp in variable $\mathcal{Q}$, which is empty. Since, empty is not a viable WeakCXp, then the WeakAXp $\mathcal{R} = \{o, h, w\}$ is used for computing the AXp $\{o, h\}$, and the clause $(s_o \vee s_h)$ is added to SAT formula $\mathcal{H}$ to block the AXp from being recomputed. The next two iterations follow a similar procedure, but with the respective candidates $\mathcal{Q}$ being reported as WeakCXps, and used as seeds for computing the CXps $\{o\}$ and $\{h\}$. Additionally, the clauses $(\neg s_o)$, and $(\neg s_h)$ are added to $\mathcal{H}$, as blocking clauses.

In the final iteration, the SAT solver reports formula $\mathcal{H}$ to be unsatisfiable (all AXps and CXps have been reported) and the algorithm terminates.

Example 17 (Enumerate Xps - Monotonic Classifier)**.** *Consider the Example 32 with the monotonic classifier in Example 32, and the instance in Example 35, an example execution of Algorithm 2 for enumerating explanations is as in Table 7.*

Iter.	outc, s	$\mathcal{R}\vert\mathcal{Q}$	$\mathbb{P}_{cxp}(\mathcal{Q})$	$\mathcal{P}_{(a/c)}$	Add to $\mathcal{H}$
					$\emptyset$
1	$\top, \{s_Q = s_E = s_P = 0\}$	$\{Q, E, P\}\vert\{\}$	$\bot$	$\{E, P\}_{(a)}$	$(s_E \vee s_P)$
2	$\top, \{s_Q = s_P = 0, s_E = 1\}$	$\{Q, P\}\vert\{E\}$	$\top$	$\{E\}_{(c)}$	$(\neg s_E)$
3	$\top, \{s_Q = s_E = 0, s_P = 1\}$	$\{Q, E\}\vert\{P\}$	$\top$	$\{P\}_{(c)}$	$(\neg s_P)$
4	$\bot$				

Table 7: Execution of Algorithm 2 for enumerating explanations for the monotonic classifier in Example 32 and instance to explain $((10, 10, \mathfrak{u}, 10), \{A, B\})$. The table shows similar information as Table 5.

Iter.	outc, s	$\mathcal{R}\vert\mathcal{Q}$	$\mathbb{P}_{cxp}(\mathcal{Q})$	$\mathcal{P}_{(a/c)}$	Add to $\mathcal{H}$
					$\emptyset$
1	$\top, \{s_1 = s_2 = s_3 = 0\}$	$\{1, 2, 3\}\vert\{\}$	$\bot$	$\{1, 2\}_{(a)}$	$(s_1 \vee s_2)$
2	$\top, \{s_2 = s_3 = 0, s_1 = 1\}$	$\{2, 3\}\vert\{1\}$	$\top$	$\{1\}_{(c)}$	$(\neg s_1)$
3	$\top, \{s_1 = s_3 = 0, s_2 = 1\}$	$\{1, 3\}\vert\{2\}$	$\top$	$\{2\}_{(c)}$	$(\neg s_2)$
4	$\bot$				

Table 8: Execution of Algorithm 2 for enumerating explanations for the NN in Figure 2 and instance to explain $((1, 1, 0, \mathfrak{u}), \{A, B\})$. The table shows similar information as Table 5.

The algorithm starts by computing a satisfying assignment with all variables assigned false (considering a biased SAT solver). It computes a candidate WeakCXp in variable $\mathcal{Q}$, which is empty. Since, empty is not a viable WeakCXp, then the WeakAXp $\mathcal{R} = \{Q, E, P\}$ is used for computing the AXp $\{E, P\}$, and the clause $(s_E \vee s_P)$ is added to SAT formula $\mathcal{H}$ to block the AXp from being computed again. The next two iterations follow a similar procedure, but with the respective candidates $\mathcal{Q}$ being reported as WeakCXps, and used as seeds for computing the CXps $\{E\}$ and $\{P\}$. Additionally, the clauses $(\neg s_E)$, and $(\neg s_P)$ are added to $\mathcal{H}$, as blocking clauses.

In the final iteration, the SAT solver reports formula $\mathcal{H}$ to be unsatisfiable (all AXps and CXps have been reported) and the algorithm terminates.

Example 18 (Enumerate Xps - Neural Networks)**.** *Consider the Example 2* *with the NN in Figure 2, and the instance in Example 9, an example execution of Algorithm 2 for enumerating explanations is as in Table 8.*

The algorithm starts by computing a satisfying assignment with all variables assigned false. It computes a candidate WeakCXp in variable $\mathcal{Q}$, which is empty. Since, empty is not a viable WeakCXp, then the WeakAXp $\mathcal{R} = \{1, 2, 3\}$ is used for computing the AXp $\{1, 2\}$, and the clause $(s_1 \lor s_2)$ is added to SAT formula $\mathcal{H}$ to block the AXp from being recomputed. The next two iterations follow a similar procedure, but with the respective candidates $\mathcal{Q}$ being reported as WeakCXps, and used as seeds for computing the CXps $\{1\}$ and $\{2\}$. Additionally, the clauses $(\neg s_1)$, and $(\neg s_2)$ are added to $\mathcal{H}$, as blocking clauses.

In the final iteration, the SAT solver reports formula $\mathcal{H}$ to be unsatisfiable (all AXps and CXps have been reported) and the algorithm terminates.

6. Algorithms for Specific Classifiers

In this section, we present additional details for computing explanations for the two different classes of ML models we have considered in our running examples. For decision trees we outline a polynomial-time algorithm for computing one explanation. Finally, for neural networks, we briefly analyze different approaches for the computation of explanations and generalized versions of explanations.

6.1. Decision Trees

In the case of DTs, for deciding $\mathbb{P}_{\text{axp}}$, two main approaches have been proposed in the literature. One approach proposes a propositional Horn logic encoding [43]. An earlier approach proposes iterative traversals of the tree [42]. We briefly overview in this section how this can be instrumented. We consider both complete instances $(\mathbf{v}, c)$ and the partially-specified instances with multiple classes $(\mathbf{z}, \mathcal{T})$ of the general framework of Section 4.

From Algorithm 1, and given a set $\mathcal{W} \subseteq \mathcal{F}$ of features fixed to the values dictated by $\mathbf{v}$ in a complete instance $(\mathbf{v}, c)$, we traverse the paths in the DT, such that a path is declared inconsistent only if for some tree node of the path, the tested feature is fixed in $\mathcal{W}$ and the tree node is inconsistent with the value specified by $\mathbf{v}$. Furthermore, if there is a path to a prediction other than c that is not inconsistent (change of prediction), then $\mathbb{P}_{\text{axp}}$ fails for the current $\mathcal{W}$, indicating that for some assignment to the non-fixed features, a prediction other than c is obtained. For an instance $(\mathbf{z}, \mathcal{T})$ from the more general framework of Section 4, the same basic algorithm is used, but a change of prediction is decided when there exists a non-inconsistent path, with respect to the fixed features $\mathcal{W}$, to a prediction not in $\mathcal{T}$. Clearly, this algorithm will traverse the DT in the worst-case, and so it runs in polynomial-time on the size of the DT.

Example 19. *We consider the DT from Figure 1. Consider the instance* $((35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.0), \{\oplus, \oplus\oplus\})$*, and the set* $\mathcal{W} = \{1, 4\}$ *(with* $\text{HCT} = 35$ *and* $\text{PLT} = 100$*). Clearly, any of the paths with a prediction not in* $\mathcal{T} = \{\oplus, \oplus\oplus\}$ *remains inconsistent, and so* $\mathcal{W}$ *is a weak AXp. Furthermore, if we allow either feature 1 or 4 to take any value, then it is also clear that a path with a prediction not in* $\mathcal{T}$ *will be consistent for some points in feature space. Hence,* $\mathcal{W}$ *is an AXp.*

6.1.1. *Use Cases with Decision Tree Classifiers*

This subsection details two experimental use cases of the general framework using decision tree classifiers.

Setup. To illustrate use cases of the logic-based explainability general framework, we consider an explainer for decision trees that is based on navigating the DT [42]. As in earlier work, the running times for computing one AXp/CXp are always negligible, since the explainer runs in polynomial time on the tree size. Also, the explainer has been implemented in Python. The explainer allows a user to declare unspecified features as well as to list accepted *target* classes (all other classes are referred to as *unwanted*).

Target/Unwanted Classes. The goal of our first experiment is to compute smaller (and so easier to understand [57]) AXp's, such that a given set of target classes is guaranteed to occur. For example, in the context of classifying the risk associated to the insurance of an individual, we may want to explain why there is some degree of risk associated with the insurance, but we do not necessarily care about the actual degree of risk.

Furthermore, the experiments consider example decision trees, obtained with a commercial tool.[12] A maximum depth of 8 was chosen, so as to keep the trees sufficiently shallow. We consider a number of well-known publicly available datasets, namely `dermatology`, `student-por`, `auto`, `soybean`, and `zoo`, for each of which all features are categorical.[13]

For the purpose of the paper, we will not dwell on the meaning ascribed to each feature, i.e. features will be denoted by fi, with i ranging over the number of features. On the other hand, the output classes will be referred by numbers. Tables 9 to 11 show the mapping between classes names and numbers used in the paper. For the remaining datasets, i.e. `auto` and `student-por`, the classes are already numeric. In fact, the classes of the benchmark `student-por` represent a marking in the range $[0-15]$. The classes of the benchmark `auto` represent the assurance risk in the range $[(-3)-3]$.

[12]We opted to use the optimal decision trees induced with the tree learner from Interpretable AI [8], which is available (on request) from `https://www.interpretable.ai/`.

[13]The datasets were obtained from the UCI repository (`https://archive.ics.uci.edu/ml/`), in the case of `dermatology`, `student-por`, `soybean` and `zoo`, and from the PennML repository (`https://epistasislab.github.io/pmlb/`), in the case of `auto`. Motivated by the experiments in this section, we opted for multi-class datasets.

Name	Number
psoriasis	1
seboreic dermatitis	2
lichen planus	3
pityriasis rosea	4
cronic dermatitis	5
pityriasis rubra pilaris	6

Table 9: Class mapping for `dermatology`

Name	Number
mammal	1
reptile	2
bug	3
bird	4
invertebrate	5
amphibian	6
fish	7

Table 10: Class mapping for `zoo`

Name	Number
diaporthe-stem-canker	1
charcoal-rot	2
rhizoctonia-root-rot	3
phytophthora-rot	4
brown-stem-rot	5
powdery-mildew	6
downy-mildew	7
brown-spot	8
bacterial-blight	9
bacterial-pustule	10
purple-seed-stain	11
anthracnose	12
phyllosticta-leaf-spot	13
alternarialeaf-spot	14
frog-eye-leaf-spot	15
diaporthe-pod-and-stem-blight	16
cyst-nematode	17
2-4-d-injury	18
herbicide-injury	19

Table 11: Class mapping for `soybean`

Moreover, for each dataset, two sets of (resp. target/unwanted) classes were chosen. Concretely, for `dermatology` the unwanted class was 3 (i.e. *lichen planus*). For `student-por` the unwanted classes were all the grades between 0 and 13 (i.e. the target are good or above grades). For `auto` the unwanted classes were those of no risk (i.e. -3, -2 and -1, but it should be noted that the dataset does not have entries predicting -3 or -2). For `soybean`, the unwanted classes were rot diseases (i.e. 1, 2, 3 and 4). Finally, for `zoo`, the unwanted class was *mammal*, i.e. we are interested in understanding which features suffice for not predicting a *mammal*. In all cases, and given a target terminal node in the DT, the specified features are selected as those tested in the path from the root to the terminal node. The unspecified features are all remaining ones.

Table 12 summarizes the results of computing one AXp. As can be observed, starting from a non-negligible number of features (i.e. a weak AXp given the unspecified features), we are able to obtain fairly small AXp's, which explain what must not be changed so that the unwanted classes are

Dataset	|DT|	Avg Depth	Term Node	Target/Unwanted	Specified Features	AXp
`dermatology`	51	6.2	32	$\{1,2,4,5,6\}/\{3\}$	{f4, f7, f13, f19, f26, f27, f33}	{f7, f26}
`student-por`	287	7.5	48	{14..19}/{0..13}	{f2, f9, f12, f24, f29, f31}	{f31}
`auto`	33	4.4	26	{0..3}/{-3..-1}	{f5, f8, f12, f18}	{f8, f18}
`soybean`	109	6.8	106	$\{0,5..18\}/\{1,2,3,4\}$	{f1, f2, f14, f25, f26, f28, f29}	{f14, f28}
`zoo`	41	5.3	26	{2..7}/{1}	{f0, f4, f6, f8, f10, f12, f15}	{f0, f15}

Table 12: Summary of results

not predicted. Such small explanations are important, since it is generally accepted that smaller explanations are simpler for human decision makers to comprehend [57].

Assessing an ML classifier. As a second experiment, we detail a case study of a recently proposed decision tree for non-invasive diagnosis of Meningococcal Disease (MD) meningitis [46, Figure 9]. The DT is shown in Figure 5, and is to be used on suspected cases of meningitis, with the goal to decide whether the type is MD or not. Some features refer to traits of a patient (e.g. age, gender, and place where she lives), and some other refer to symptoms that the patient exhibits (e.g. being in a coma, headache, seizures, vomiting, stiff neck, and existence of petechiae). For this DT, we have used our explainer as well as selected a set of unspecified features to prove that patients exhibiting no symptoms can be diagnosed with MD meningitis. This example illustrates a novel use case of logic-based explainability in assessing how realistic, and possibly misleading, are the predictions of an ML classifier.

We wish to assess the quality of the classifier, and so we seek to understand whether a patient can be diagnosed with MD meningitis without exhibiting any of the symptoms. The proposed procedure operates as follows. For each leaf predicting the class of interest (i.e. a diagnosis of MD meningitis) we identify the features representing symptoms and, among these, we pick the ones that are active (i.e. the patient exhibits the symptom). We then declare the other features (non-active or non-symptom) as specified, and leave the active symptom features as unspecified. If the specified features are sufficient for the prediction, then we know that a patient can be diagnosed with MD meningitis despite not exhibiting any symptoms. Notwithstanding, we proceed to compute an AXp, since this offers an irreducible set of features sufficient for the unwanted prediction. (Although we consider a concrete use case, the procedure outlined above is completely general as an arbitrary number of identified features could be considered.)

For example, for a patient that conforms to the following path of the DT that predicts MD meningitis:

$$\begin{aligned}\langle\quad & (\mathsf{Age} > 5) = 1, \mathsf{Petechiae} = 0, \mathsf{Stiff\ Neck} = 0, \\ & \mathsf{Vomiting} = 1, \mathsf{Zone} = 2, \mathsf{Seizures} = 0, \\ & \mathsf{Headache} = 0, \mathsf{Coma} = 0, \mathsf{Gender} = 1 \quad\rangle\end{aligned}$$

We leave as unspecified features, only the active symptom (Vomiting) to look for an AXP. The remaining symptons and traits are considered specified (to their values). The AXP we obtain is:

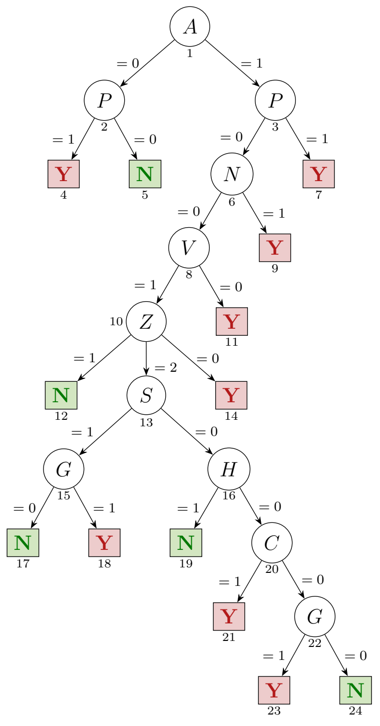
$$\{(\mathsf{Age} > 5), \mathsf{Gender}, \mathsf{Headache}, \mathsf{Zone}\}$$

That is, the unique active symptom in this path (Vomiting) is not really necessary to predict a patient as having MD meningitis, as it is only necessary for the patient to have age > 5, reside in zone 2, to exhibit no headache, and have gender 1 (male). This behavior is obtained also with other paths of the DT, and they are not easily spotted by a visual inspection of the DT.

Thus, the DT can serve to diagnose MD meningitis for patients without any active (i.e. true) symptom, among the symptoms considered. The active symptom feature in the path is only misleadingly being used for the prediction; as the computed AXp confirms, the active feature is redundant, and so irrelevant for the prediction. The ensuing conclusion is that the DT proposed in [46] exhibits unrealistic diagnoses. As demonstrated by the computed AXp, any human being suspect of having meningitis, older than 5 years of age, that is a male, with an urban residence (i.e. Zone=2), and exhibiting no headache will be diagnosed as having MD meningitis! By applying a similar analysis on path $\langle 1, 3, 6, 8, 11\rangle$, we also conclude that any patient older than 5 years of age that is not Vomiting will also be diagnosed with MD meningitis.

Finally, a stronger analysis can be made with respect to path $\langle 1, 3, 6, 8, 10, 14\rangle$. In this case, we can declare *any* of the symptoms unspecified, and still conclude that A and Z are sufficient for the prediction, that is, the resulting AXp is $\{A, Z\}$. Hence, the prediction of MD meningitis involves testing no symptoms whatsoever.

The identified problems can result from a very unbalanced dataset, an incorrect tree inducer, or some critical features that are not being accounted for. It seems unlikely that the issues with the DT might result from a flipped test, e.g. the test on V, since flipping the values of the test on V would yield the same conclusions. As can be observed, the issue with the DT of Figure 5 is not easily spotted by inspection, concretely for the longer path. Furthermore, such analysis by inspection would be far more challenging on larger DTs. More importantly, the proposed analysis is only made feasible by the generalized explainability solution proposed.



Feat.#	Name	Meaning	Definition	Domain	Trait/Symp.
1	A	Age	Age > 5?	$\{0, 1\}$	T
2	P	Petechiae	Petechiae?	$\{0, 1\}$	S
3	N	Stiff Neck	Stiff Neck?	$\{0, 1\}$	S
4	V	Vomiting	Vomiting?	$\{0, 1\}$	S
5	Z	Zone	Zone=?	$\{0, 1, 2\}$	T
6	S	Seizures	Seizures?	$\{0, 1\}$	S
7	G	Gender	Gender?	$\{0\,(\text{F}), 1\,(\text{M})\}$	T
8	H	Headache	Headache?	$\{0, 1\}$	S
9	C	Coma	Coma?	$\{0, 1\}$	S

Path & Prediction	$\langle 1, 3, 6, 8, 10, 13, 16, 20, 22, 23 \rangle$ & **Y**
Features in path	$\{A, P, N, V, Z, S, H, C, G\}$
Symptoms in path	$\{P, N, V, S, H, C\}$
Active symptoms	$\{V\}$
Computed AXp	$\{A, G, H, Z\}$
Interpretation	Age$\geq$5 $\land$ Male $\land$ no Headache $\land$ Zone=2
Conclusion	MD meningitis can be predicted without symptoms

Figure 5: DT for non-invasive diagnosis of MD meningitis.

6.2. Neural Networks

The use of mixed-integer linear programming (MILP) for deciding $\mathbb{P}_{\text{axp}}$ was proposed with the first inroads into formal XAI [36]. However, scalability was an important issue, and so AXps could be computed only for modest size NNs. Furthermore, the use of MILP imposed key constraints on the organization of NNs that could be analyzed, e.g. the use of ReLU as the activation units.

Recent work proposed a fundamentally different approach for computing explanations in the case of NNs. First, existing definitions of explanations subject to constraints [81] were adapted to enable the definition of distance-restricted AXps and CXps [30]. Second, duality between distance-restricted AXps and CXps was established, which then enabled defining the computation of AXps in terms of finding adversarial examples. As a result, recent results [30] report the computation of distance-restricted explanations for NNs having more than one order of magnitude more activation units than earlier results [36].

7. Generalizations: The Case of XAI Queries

This section surveys generalizations of Logic-based explainability considering the framework of Section 4. The surveyed generalizations are both in terms of the type of explanations considered, as well as considering the participation of features in explanations. In the literature, the problem of determining the participation of a feature in explanations is referred to as explainability queries. We then detail solutions for the presented explainability queries considering the generalized framework of Section 4.

Logic-based explanations often assume a uniform distribution over the inputs, i.e. all points in feature space are equally likely and possible. Recent work proposes to account for explicit constraints on the inputs [21, 80, 81, 15] by considering in the definition of a weak AXp an additional map $\varsigma : \mathbb{F} \to \{0, 1\}$ constraining the point (with 0 denoting a disallowed point, and 1 denoting an allowed point). With this modified definition of WeakAXp, it is immediate that, given a generalized explanation problem $\mathcal{E}' = (\mathcal{M}, (\mathbf{z}, \mathcal{T}))$, a WeakAXp with input constraints can be generalized as follows:

$$\mathsf{WeakAXp}(\mathcal{X}) := \forall \mathbf{x} \in \mathbb{F}. \left[\varsigma(\mathbf{x}) \wedge \bigwedge_{i \in \mathcal{X}} (x_i = z_i) \right] \to \kappa(\mathbf{x}) \in \mathcal{T} \tag{22}$$

In a similar vein, it is apparent (and so it is conjectured) that the computation of probabilistic explanations [76, 40, 39, 3] can be handled in a similar way. Recently, [40] proposed the computation of Probabilistic Abductive Explanations. The authors introduce the concept of weak Probabilistic Abductive Explanation $\mathcal{X}$ ($\mathsf{WPAXp}(\mathcal{X})$), which represents the probability of a variable x having the same classification as a given instance $\mathbf{v}$, conditioned to x agreeing with the values of $\mathbf{v}$ in the features of $\mathcal{X}$, being larger than a given δ. Given a generalized explanation problem $\mathcal{E}' = (\mathcal{M}, (\mathbf{z}, \mathcal{T}))$, the definition of $\mathsf{WPAXp}(\mathcal{X})$ can be generalized as follows:

$$\mathsf{WPAXp}(\mathcal{X}) := \ Pr_x(\kappa(x) \in \mathcal{T} \mid x_{\mathcal{X}} = z_{\mathcal{X}}) \geq \delta \tag{23}$$

The generalization of predicted class proposed in Section 4 requires that the classifier computes a total function, i.e. each point in feature space maps to a concrete class. However, the results from Section 4 can be generalized to a more general set up where κ can associate multiple classes for each point in feature space, but also when the class is left unspecified, i.e. κ is a map from $\mathbb{F}$ into $2^{\mathcal{K}'}$, with $\mathcal{K}' = \mathcal{K} \cup \mathfrak{u}$, and where $\mathfrak{u}$ denotes any class. For example, this generalization allows for assigning *don't care* classes to points in feature space which are known not to be feasible, offering an alternative to recent work [21].

XAI Queries. A generalization that can be made is with regards to querying the participation of features in explanations. Namely the queries of necessity, relevancy and irrelevancy [25], that is, whether a feature participates in all the explanations of the output class (the feature is necessary), or the feature belongs to at least one explanation of the output classification class (the feature is relevant), or the feature does not belong to any explanation of the output classification class (the feature is irrelevant).

Example 20 (Necessary, Relevant and Irrelevant Features - Decision Trees)**.** *From Example 3 we can see that, for a set of features* $\mathcal{F} = \{1, 2, 3, 4, 5, 6\}$*, the set of the abductive explanations is* $\mathbb{A} = \{\{1, 4, 5\}, \{4, 5, 6\}\}$ *(considering the instance to explain* $((35.0, 0.25, 5.1, 100, 37.1, 5.2), \oplus)$*). In this case, we can see that features* 2 *and* 3 *do not belong to any explanation, thus they are deemed to be irrelevant. On the other hand both features* 4 *and* 5 *belong to all the explanations, thus they are deemed to be necessary features. Finally, features* 1 *and* 6 *belong to (at least) one explanation each, thus they are deemed to be relevant features.*

Consider the instance $((35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.2), \{\oplus, \oplus\oplus\})$*, with missing inputs from Example 8. The set of features is the same set* $\mathcal{F}$ *as before, but in this case we must divide the features in specified and unspecified features (according to the instance point). As such, we only refer to necessary, relevant or irrelevant features for the specified features* $\mathcal{S} = \{1, 2, 4, 6\}$ *(*$\mathcal{U} = \{3, 5\}$ *since they are valued* $\mathfrak{u}$ *in the instance point). In this case, the set of abductive explanations is* $\mathbb{A}_{\mathfrak{u}} = \{\{1, 4\}, \{4, 6\}\}$*. Similarly, features* 2 *and* 5 *are irrelevant, feature* 4 *is a necessary feature, and features* 1 *and* 6 *are relevant features.*

Example 21 (Necessary, Relevant and Irrelevant Features - Neural Networks)**.** *Consider the neural network in Example 2, and the instance* $(\mathbf{z}, \mathcal{T})$ *with* $\mathbf{z} = (1, 1, 0, \mathfrak{u})$*,* $\mathcal{T} = \{A, B\}$ *with missing inputs from Example 9. The set of specified features is* $\mathcal{S} = \{1, 2, 3\}$*, and the set of explanations is* $\mathbb{A}_{\mathfrak{u}} = \{\{1, 2\}\}$ *(Example 17).*

Feature 3 *does not belong to the explanations, and so it is an irrelevant feature. On the other hand, both features* 1 *and* 2*, belong to the only explanation, and so are both necessary and relevant features.*

The general complexity of necessity and (ir)relevancy has been studied in the context of logic-based abduction [19, 20, 71]. [25] makes an in-depth study of the complexity of querying a classifier for necessity and (ir)relevancy of a feature. In this work we show how to query a classifier for the relevancy of a feature considering explanations with unspecified inputs. Consider a feature point (with missing inputs) $\mathbf{z}$, the associated partition of the set of features in specified and unspecified features $(\mathcal{S}, \mathcal{U})$ $(\mathcal{F} = \mathcal{S} \cup \mathcal{U})$, and a target predicted set of classes to explain $\mathcal{T} \subset \mathcal{K}$. Algorithm 3 proposes an adaptation of the algorithm in [25] taking into account the unspecified inputs. Similar to the algorithms in Section 5, we use the predicate $\mathbb{P}_{axp}(\mathcal{X})$ of (20), to check if $\mathcal{X}$ is a WeakAXp.

The algorithm receives as input the target feature $t \in \mathcal{S}$ we wish to query the model about its (ir)relevancy. The algorithm maintains a CNF formula $\mathcal{H}$ which is used by the SAT solver to determine WeakAXp candidates. Each variable of the formula corresponds to a selector variable for the specified features. Initially, $\mathcal{H}$ contains only one unit clause with the selector variable associated to the feature t. This ensures that any candidate will contain feature t.

At each iteration, the SAT solver is asked for a new candidate WeakAXp, by determining the satisfiability of the formula $\mathcal{H}$. If the formula is not

Algorithm 3 Deciding feature relevancy

```
    Input:  Target feature $t \in \mathcal{S}$
    Output: $\top$ if $t$ is a relevant; $\bot$ otherwise

 1: procedure isRelevant($t$)
 2:     $\mathcal{H} \leftarrow \{(s_t)\}$                                   ▷ $\mathsf{Vars}(\mathcal{H}) = \{s_i \mid i \in \mathcal{S}\}$
 3:     repeat
 4:         $(\mathsf{outc}, \mathsf{s}) \leftarrow \mathsf{SAT}(\mathcal{H})$
 5:         if $\mathsf{outc}$ = true then
 6:             $\mathcal{P} \leftarrow \{i \in \mathcal{S} \mid s_i = 1\}$
 7:             $\mathcal{D} \leftarrow \{i \in \mathcal{S} \mid s_i = 0\}$
 8:             if $\neg\mathbb{P}_{axp}(\mathcal{P})$ then
 9:                 $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\vee_{i\in\mathcal{D}} s_i)\}$          ▷ Block non-WAXp
10:             else if $\neg\mathbb{P}_{axp}(\mathcal{P} \setminus \{t\})$ then
11:                 return true                                          ▷ $t$ is relevant
12:             else                                       ▷ Block $t$-irrelevant WAXp
13:                 $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\vee_{i\in\mathcal{P}\setminus\{t\}} \neg s_i)\}$
14:     until $\mathsf{outc}$ = false
15:     return false                                                     ▷ $t$ is irrelevant
```

satisfiable, then the algorithm terminates with t as an irrelevant feature, and returns false.

If the formula is satisfiable then the SAT solver provides an assignment for the selector variables which corresponds to a new candidate WeakAXp. The candidate is obtained in variable $\mathcal{P}$ as the set of features whose selector variables are assigned true. The algorithm then checks if the new candidate is indeed a WeakAXp, by calling the predicate $\mathbb{P}_{axp}(\mathcal{P})$. If not, then a new clause is added to $\mathcal{H}$ to block the non-WeakAXp candidate to be recomputed. The blocking clause is build by requiring at least one of the selector variables of the features not in the candidate to be true.

If the candidate is a WeakAXp, then the algorithm checks if the candidate is a WeakAXp for which t is relevant, that is the algorithm calls the predicate $\mathbb{P}_{axp}(\mathcal{P} \setminus \{t\})$. If $\mathcal{P} \setminus \{t\}$ is not a WeakAXp, then t is required to be present for the candidate to be a WeakAXp (and an AXp must exist containing t). That is, t a relevant feature and the algorithm returns true. Otherwise, a new clause is add to $\mathcal{H}$ to block the irrelevant WeakAXp candidate to be recomputed. The blocking clause is build by requiring at least one of the

Iter.	outc, s	$\mathcal{P}$	$\neg\mathbb{P}_{axp}(\mathcal{P})$	$\neg\mathbb{P}_{axp}(\mathcal{P}\setminus\{t\})$	Add to $\mathcal{H}$
					(s_1)
1	$\top, \{s_2 = s_4 = s_6 = 0, s_1 = 1\}$	$\{1\}$	$\top$		$(s_2 \vee s_4 \vee s_6)$
2	$\top, \{s_4 = s_6 = 0, s_2 = s_1 = 1\}$	$\{1, 2\}$	$\top$		$(s_4 \vee s_6)$
3	$\top, \{s_2 = s_6 = 0, s_4 = s_1 = 1\}$	$\{1, 4\}$	$\bot$	$\top$	

Table 13: Execution of Algorithm 3 for deciding the (ir)relevancy of feature 1 of the DT in Figure 1, with instance to explain $((35.0, 0.25, \mathfrak{u}, 100.0, \mathfrak{u}, 5.2), \{\oplus, \oplus\oplus\})$.

selector variables of the features in the candidate, other than t, to be false.

In the following we present some example executions of checking the (ir)relevancy of features for the running examples. Similar to previous example executions, for demonstration purposes, we assume a SAT solver biased towards unsatisfying variables.

Example 22 ((Ir)Relevancy - Decision Trees)**.** *Consider the Example 1 with the DT in Figure 1, and the instance in Example 8, for which we wish to compute the (ir)relevancy of feature 1. An example execution of Algorithm 3 is as in Table 13.*

In the first iteration $\mathcal{H}$ *contains only* (s_1) *since 1 is the feature to check for relevancy. The formula is satisfiable and the SAT solver returns the assignment unsatisfying all selector variables except for* s_1*. The algorithm computes as candidate WeakAXp the set* $\mathcal{P}$ *containing only feature 1, and calls the predicate* $\mathbb{P}_{axp}(\{1\})$*. We can observe from the Figure 1, that fixing only feature 1 to the value of HCT=35.0, then starting from the root, the reachable nodes are the nodes 7, 8, 10, 12, 13, 14 and 15. But nodes 7, 13 and 15 are classified outside the target classes* $\mathcal{T} = \{\oplus\oplus, \oplus\}$*, meaning there is at least one point in the feature space that agrees with the instance point in feature 1 and is classified with a class not in* $\mathcal{T}$ *(e.g.* $v_1 = (35.0, 0.6, 4.9, 200, 37, 6)$ *and* $\kappa(v_1) = \ominus\ominus$*). Thus the predicate returns false and its negation is true. The algorithm adds to* $\mathcal{H}$ *the blocking clause* $(s_2 \vee s_4 \vee s_6)$*.*

In the second iteration, formula $\mathcal{H}$ *is satisfiable, and this time the SAT solver returns the assignment satisfying both* s_1 *and* s_2 *(for example). The new candidate* $\mathcal{P}$ *is then the set* $\{1, 2\}$*, and the algorithm calls the predicate* $\mathbb{P}_{axp}(\{1, 2\})$*. This time fixing both features 1 with the value HCT=35.0 and feature 2 with Lymp.=0.25, then starting from the root node, the reachable*

Iter.	outc, s	$\mathcal{P}$	$\neg\mathbb{P}_{axp}(\mathcal{P})$	$\neg\mathbb{P}_{axp}(\mathcal{P} \setminus \{t\})$	Add to $\mathcal{H}$
					(s_2)
1	$\top, \{s_1 = s_3 = 0, s_2 = 1\}$	$\{2\}$	$\top$		$(s_1 \vee s_3)$
2	$\top, \{s_3 = 0, s_1 = s_2 = 1\}$	$\{1, 2\}$	$\bot$	$\top$	

Table 14: Execution of Algorithm 3 for deciding the (ir)relevancy of feature 2 of the NN in Figure 2, with instance to explain $((1, 1, 0, \mathfrak{u}), \{A, B\})$.

nodes are 8, 10, 12, 13, 14 and 15. Since nodes 13 and 15 are classified outside the target classes $\mathcal{T}$, then there is at least one point in the feature space that agrees with the instance point in features 1 and 2, and it is classified with a class not in $\mathcal{T}$ (e.g. $v_2 = (35.0, 0.25, 5.1, 200, 37, 6)$ and $\kappa(v_2) = \ominus$). Thus the predicate returns false and its negation is true. The algorithm adds to $\mathcal{H}$ the new blocking clause $(s_4 \vee s_6)$.

In the third iteration, formula $\mathcal{H}$ is again satisfiable, and this time the SAT solver returns the assignment satisfying both s_1 and s_4 (for example). The new candidate $\mathcal{P}$ is then the set $\{1, 4\}$, and the algorithm calls the predicate $\mathbb{P}_{axp}(\{1, 4\})$. Fixing both features 1 with the value HCT=35.0 and feature 4 with PLT=25.0, then starting from the root node, the reachable nodes are 8, 10 and 14, each of which is classified with a class in $\mathcal{T}$. This means that there are no points in the feature space agreeing with the instance in features 1 and 4 whose classification is outside $\mathcal{T}$, that is, $\mathbb{P}_{axp}(\{1, 4\})$ is true and its negation is false. As such the candidate $\{1, 4\}$ is a Weak AXp. The algorithm proceeds to check whether $\{1, 4\}$ requires feature 1 in order to be a WeakAXp and contain an AXp, that is, it calls the predicate $\mathbb{P}_{axp}(\{4\})$. This time fixing only feature 4 with PLT=25.0, then starting from the root node, the reachable nodes are 8, 10, 11 and 14. Node 11 is classified outside the target classes $\mathcal{T}$, then there is at least one point in the feature space that agrees with the instance point in feature 4, and it is classified with a class not in $\mathcal{T}$ (e.g. $v_3 = (42.0, 0.25, 5.1, 100.0, 37, 6.1)$ and $\kappa(v_3) = \ominus$). Thus the predicate returns true and its negation is false, meaning $\{1, 4\}$ requires feature 1 to be a WeakAXP and contain an AXp, so 1 is a relevant feature and the algorithm terminates.

Example 23 ((Ir)Relevancy - Neural Networks)**.** *Consider the neural network in Example 2, and the instance in Example 9, for which we wish to* *compute the (ir)relevancy of feature* 2 *(how relevant is it for a paper to have a good experimental evaluation). An example execution of Algorithm 3 is as in Table 14.*

In the first iteration $\mathcal{H}$ *contains only* (s_2) *since* 2 *is the feature to check for relevancy. The formula is satisfiable and the SAT solver returns the assignment unsatisfying all selector variables except for* s_2. *The algorithm computes the candidate set* $\mathcal{P}$ *containing only feature* 2*, and calls the predicate* $\mathbb{P}_{axp}(\{2\})$ *to check whether the candidate is a WeakAXp. The predicate asks for the inexistance of a point in the feature space that sets the value of feature* 2 *to 1 and yet the paper is neither accepted nor considered a borderline paper, that is, the paper is rejected. The point* $v_1 = (0, 1, 0, 0)$ *does respect the value of feature* 2 *to 1. Since after training the NN predicts the paper to be rejected if the majority of the values of the features is 0, then we can see that* v_1 *is predicted (by the NN) to be rejected. Thus, the predicate is false and its negation is true. This means the candidate is not a WeakAXp, and the algorithm adds the clause* $(s_1 \vee s_3)$ *to* $\mathcal{H}$.

In the second iteration, formula $\mathcal{H}$ *is satisfiable and the SAT solver returns an assignment that satisfies both variables* s_1 *and* s_2 *(for example). The candidate corresponds to the set* $\{1, 2\}$*, and the algorithm asks if* $\{1, 2\}$ *is a WeakAXp, that is, it calls the predicate* $\mathbb{P}_{axp}(\{1, 2\})$. *Similar to before, the predicate asks for the inexistance of a point in the feature space that sets the value of both features* 1 *and* 2 *to 1 and yet the paper is rejected. Since after training the NN corresponds to a majority voting on the values of the inputs, then any point with both the first and the second values set to 1, will have either a tie (two 1's and two 0's) or a win (majority of 1's), meaning the paper is either predicted to be borderline or accepted, that is, classified in* $\mathcal{T}$. *So the predicate is true (it's negation is false), and the candidate is a WeakAXp. The algorithm proceeds by checking if removing feature* 2*, the candidate is still a WeakAXp, that is, it calls* $\mathbb{P}_{axp}(\{1\})$. *This time the predicate asks for the inexistance of a point in the feature space that sets the value of feature* 1 *to 1 and yet the paper is rejected. The point* $v_2 = (1, 0, 0, 0)$ *does respect the value of feature* 1 *to 1. The majority voting on the inputs (performed by the NN) predicts the paper to be rejected. Thus, the predicate is false, meaning* $\{1\}$ *is not a WeakAXp, and feature* 2 *is necessary for* $\{1, 2\}$ *to be a WeakAXp. Meaning there is an AXp in* $\{1, 2\}$ *that contains* 2*. The algorithm terminates with* 2 *as a relevant feature.*

8. Conclusions

Motivated by the fundamental importance of XAI for the widespread adoption of AI/ML, but also by the limitations of a number of informal non-rigorous approaches to XAI, this paper surveys the emerging field of formal XAI (FXAI), and documents the progress that has been observed in recent years. Furthermore, the paper shows that most of past work on formal XAI can be framed in a more general explainability framework, which enables a larger number of practical use cases. Finally, the paper also identifies areas of research of formal XAI where future advances are to be expected.

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", and funded by the Spanish Ministry of Science and Innovation (MICIN) under projects PID2019-111544GB-C22 and PID2022-139835NB-C22, and by a María Zambrano fellowship and a Requalification fellowship financed by Ministerio de Universidades of Spain and by European Union – NextGenerationEU.

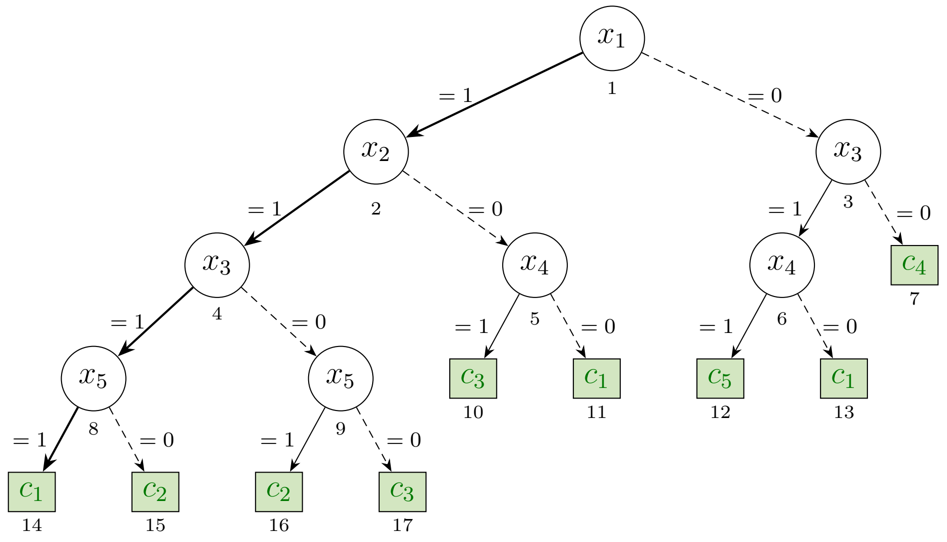
Appendix A. Illustrative Example

In this section we present an example to show the potential of the general framework. Namely, we show an example where without the general framework a much weaker explanation is obtained, that is only able to explain a subset of the query asked. On the other hand, with the general framework a single and smaller explanation is able fully explain the instance, with a much wider reach.

Consider a Decision Tree classifier model where the set of features is $\mathcal{F} = \{1, 2, 3, 4, 5\}$. Let all features be binary, that is, $D_i = \{0, 1\}$ ($i \in \{1, 2, 3, 4, 5\}$). The set of classes is $\mathcal{K} = \{c_1, c_2, c_3, c_4, c_5\}$. Figure A.6 presents the tree of the classifier, in particular.[14] From the figure, we can also observe that it is enough to set feature 1 to value 1 in order to obtain a classification of either classes c_1, c_2, or c_3.

Consider the point where the values of the features are all 1, $\mathbf{v} = (1, 1, 1, 1, 1)$. Then, $\kappa(\mathbf{v}) = c_1$. We will be interested in obtaining an explanation for the previous point $\mathbf{v}$.

[14]It is assumed that the feature of a node has value 1 in the left subtree (of the node), and value 0 in the right subtree.



$$\mathcal{K} = \{c_1, c_2, c_3, c_4, c_5\}$$

Figure A.6: Example of a DT Model

For the purpose of the example, classifying as c_1, c_2 or c_3 is equally valid, meaning we do not differentiate between each of those classes. In our general framework, the previous is obtained by considering the instance $(\mathbf{v}, \{a, b, c\})$. Note that in this case, the point $\mathbf{z} \in \mathbb{F}'$ considered is $\mathbf{z} = \mathbf{v}$ (containing no unspecified input value), and the selected target class $\mathcal{T} \subseteq \mathcal{K}$ was provided by the user as $\mathcal{T} = \{a, b, c\}$. As observed before, we can see that a computed explanation in the general framework would be simply $\{1\}$.

On the other hand, if we disregard the general framework, the instance considered can only be $(\mathbf{v}, c_1)$. By inspection of the tree, we can say that a computed explanation is $\{1, 2, 3, 5\}$, where disregarding any of the values of these features leads to a classification other than c_1.

As can be observed, the explanation $\{1, 2, 3, 5\}$ is much larger than $\{1\}$, and covers much less leaves of the tree. We could also consider to simulate the expressiveness of the explanations obtained with the general framework proposed, but without it. However, as the example shows, it is unclear how in general $\{1\}$ might be computed if for point $\mathbf{v}$ we accept any of the classes c_1, c_2, or c_3 .

Appendix B. Decision Lists

Decision lists have been deeply studied in the literature [65], and decision list classifiers have been used in a wide range of ML applications [2, 1, 14]. Decision lists (DL) represent an ordered set of rules. A DL can be represented in the following form:

$$\begin{array}{rl} R_0: & \texttt{If} \quad (cond_0) \quad \texttt{Then} \quad cls_0 \\ & \vdots \\ R_j: & \texttt{Else If} \quad (cond_j) \quad \texttt{Then} \quad cls_j \\ & \vdots \\ R_{def}: & \texttt{Else} \quad cls_{def} \end{array}$$

where for rule R_j, $cond_j$ represents a Boolean expression defined on the features and their domains, and $cls_j \in \mathcal{K}$. A rule R_j is said to be fired if its $cond_j$ is satisfied and none of the previous rules is fired. Rule R_{def} is the last rule and is called the default rule because it is always fired if none of the previous rules is fired.

Example 24. *As a running example we consider a DL obtained by using the CN2 Rule Induction algortihm [14] from Orange using the Python script from*

$$\begin{array}{rl}
R_0: & \texttt{If} \quad (o = overcast) \quad \texttt{Then} \quad classes = + \\
R_1: & \texttt{Else If} \quad (t = hot) \quad \texttt{Then} \quad classes = - \\
R_2: & \texttt{Else If} \quad (\neg(h = high)) \wedge (\neg(o = rain)) \quad \texttt{Then} \quad classes = + \\
R_3: & \texttt{Else If} \quad (\neg(o = rain)) \quad \texttt{Then} \quad classes = - \\
R_4: & \texttt{Else If} \quad (w = \text{false}) \quad \texttt{Then} \quad classes = + \\
R_5: & \texttt{Else If} \quad (o = rain) \quad \texttt{Then} \quad classes = - \\
R_{def}: & \texttt{Else} \quad classes = +
\end{array}$$

Figure B.7: Rules for the meteo DL.

the xdl-tool[15] *as interface. The data was originaly obtained in from [58]. We refer to the resulting DL as the meteo DL. Given some weather condition, the objective of the meteo DL is to determine whether student classes for playing tennis will be given ($classes = +$) or not ($classes = -$). The classifier considers four features: "outlook"; "temperature"; "humidity"; and "windy", which we refer by their first letter, that is $\mathcal{F} = \{o, t, h, w\}$. The domains of the features are as follows:*

$$\begin{array}{lcl}
D_o & = & \{overcast, rain, sunny\} \\
D_t & = & \{hot, mild, cold\} \\
D_h & = & \{high, normal\} \\
D_w & = & \{false, true\}
\end{array}$$

and the output of the DL is given by the target feature "classes" ($classes = +$, or $classes = -$). The rules of the meteo DL are shown in Figure B.7.

Example 25 (Abductive Explanations - Decision Lists)**.** *Consider the DL from Example 24 of predicting classes depending on weather conditions. Let the weather conditions of the day be represented by the point in the feature space $\mathbf{v} = (sunny, mild, high, false)$, and the instance $(\mathbf{v}, classes = -)$ (due to firing rule R_3).*

An example of an AXp is $\mathcal{X} = \{o, h\}$, because any point respecting $o = sunny$ and $h = high$ will either fire rule R_1 or rule R_3, classifying with $classes = -$. On the other hand $\mathcal{X}$ is subset-minimal, since if we disregard feature o, then rule R_0 could be fired with $classes = +$; if we disregard feature h, then rule R_2 could be fired with $classes = +$.

[15] Located at https://github.com/alexeyignatiev/xdl-tool/blob/main/cn2-tool/cn2.py

	AXp's $\mathbb{A}$		CXp's $\mathbb{C}$
Example 25	$\{\{o,h\}\}$	Example 26	$\{\{o\},\{h\}\}$

Table B.15: AXp's and CXp's from the DL running example. Each AXp is a mhs of the CXp's and vice-versa.

Example 26 (Contrastive Explanations - Decision Lists)**.** *Consider the DL from Example 24 of predicting classes depending on weather conditions, and let the instance be* $(\mathbf{v}, \mathit{classes} = -)$ *with* $\mathbf{v} = (\mathit{sunny}, \mathit{mild}, \mathit{high}, \mathit{false})$ *(as in Example 25).*

This example contains only two CXp's, with one feature each. These are the CXp's $\mathcal{X}_a = \{o\}$ *and* $\mathcal{X}_b = \{h\}$*. For example, the point* $(\mathit{overcast}, \mathit{mild}, \mathit{high}, \mathit{false})$ *only differs from* $\mathbf{v}$ *in the value of feature* o*, yet the classification is* $\mathit{classes} = +$*. Similarly, the point* $(\mathit{sunny}, \mathit{mild}, \mathit{normal}, \mathit{false})$ *only differs from* $\mathbf{v}$ *in the value of feature* h*, and due to rule* R_2 *the classification is* $\mathit{classes} = +$*. On the other hand, both* $\mathcal{X}_a$ *and* $\mathcal{X}_b$ *are subset-minimal, thus making them CXp's.*

Table B.15 shows both the AXp's (Example 25) and CXp's (Example 26) from Example 24. The table shows clearly that the duality property is satisfied.

Example 27 (Missing Inputs Instance - Decision Lists)**.** *Consider the DL from Example 24 of predicting students having classes depending on weather conditions. For this example we want to obtain an explanation for a group of days, for which classes were predicted not to be given* $(\mathit{classes} = -)$*. Additionally, we want to disregard from the model feature "temperature". The remaining weather condition feature's values are the same as in Examples 25 and 26. The weather conditions for the group of days is then represented by the point in the feature space* $\mathbf{z} = (\mathit{sunny}, \mathfrak{u}, \mathit{high}, \mathit{false})$*, and the instance is* $(\mathbf{z}, \{\mathit{classes} = -\})$*.*

The DL example has $\mathbf{z} = (\mathit{sunny}, \mathfrak{u}, \mathit{high}, \mathfrak{u})$, with $\mathcal{T} = \{\mathit{classes} = -\}(= \kappa'(\mathbf{z}))$ matching the inferred value from the classification of $\mathbf{z}$; and the NN example has $\mathbf{z} = (1, 1, 0, \mathfrak{u})$, with $\mathcal{T} = \{A, B\}$ corresponding to $\kappa'(\mathbf{z})$.

Example 28 (General Framework Explanations - Decision Lists)**.** *Consider the DL of Example 24, and the instance to explain* $(\mathbf{z}, \mathcal{T})$ *from Example 27 with* $\mathbf{z} = (\mathit{sunny}, \mathfrak{u}, \mathit{high}, \mathit{false})$ *and* $\mathcal{T} = \{\mathit{classes} = -\}$*. By inspection of the*

Iter.	i	$\mathbb{P}_{axp}$	$\mathcal{W}$
			$\{o, h, w\}$
1	o	$\perp$	$\{o, h, w\}$
2	h	$\perp$	$\{o, h, w\}$
3	w	$\top$	$\{o, h\}$

(a)

Iter.	i	$\mathbb{P}_{cxp}$	$\mathcal{W}$
			$\{o, h, w\}$
1	o	$\top$	$\{h, w\}$
2	h	$\perp$	$\{h, w\}$
3	w	$\top$	$\{h\}$

(b)

Table B.16: Example executions of Algorithm 1 for the DL in Example 24 with instance $((sunny, \mathfrak{u}, high, \text{false}), \{classes = -\})$. (a) Computing one AXp (b) Computing one CXp

DL we can see that there is only one AXp $\mathcal{X} = \{o, h\}$. This is because any point respecting $o = sunny$ and $h = high$ will either fire rule R_1 or rule R_3, with a classification of $classes = -$ $(\in \mathcal{T})$ (similarly to Example 25). On the other hand, setting either of the features alone ($o = sunny$ or $h = high$) is not enough to guarantee the classification to be $classes = -$, that is, $\mathcal{X}$ is subset-minimal. Additionally, setting $w = false$ alone will not force the classification to $classes = -$, and the feature can be disregarded from the AXp.

By duality, or by inspection of the DL, we can see that the set of CXps contains both $\mathcal{Y}_a = \{o\}$ and $\mathcal{Y}_b = \{h\}$. Forcing the other features to the values in the instance except for the features in the CXps will allow the classification to change to $classes = +$ $(\notin \mathcal{T})$.

For the following example executions of Algorithm 1 with the remaining running examples, we will only present a summary of the full resulting iterations, since the execution procedures are similar to the examples in Section 5.1.

Example 29 (Compute One Xp – Decision Lists)**.** *Consider the DL in Example 24 for which we wish to compute an explanation for the point $(sunny, \mathfrak{u}, high, false)$ with prediction $\{classes = -\}$ as in Example 27. Example executions of Algorithm 1 for computing Xps are shown in Table B.16. In the two executions, initially the specified features are $\mathcal{S} = \{o, h, w\}$ and the unspecified feature is $\mathcal{U} = \{t\}$.*

In the example execution for computing an AXp (in Table B.16(a)), the predicate $\mathbb{P}_{\text{axp}}$ checks if there exists a point in the feature space that classifies in $\{classes = -\}$. The algorithm starts with the seed $\mathcal{R} = \mathcal{S} = \{o, h, w\}$. In iteration 1, the predicate is called with features $\{h, w\}$, and observing the DL *we can see the rules that can be triggered are R_0, R_1, R_3 and R_4 that classify as $(classes = -)$ and $(classes = +)$. Since there are points classifying to class $(classes = +)$, then the predicate is false and the feature is not removed from $\mathcal{W}$. In iteration 2 we observe the predicate is also false, since the rules that can triggered are R_1, R_2 and R_3 that classify as $(classes = -)$ and $(classes = +)$, and the feature h is not removed. On the contrary, in iteration 3 the rules that could be triggered are R_1 and R_3 that classify as $(classes = -)$, and make the predicate true, which leads to the removal of the feature w from the set $\mathcal{W}$. The algorithm terminates with $\mathcal{W} = \{o, h\}$ as the reported AXp.*

In the example execution for computing a CXp (in Table B.16(b)), the predicate $\mathbb{P}_{\text{cxp}}$ checks if there exists a point in the feature space that classifies as $\{classes = +\}$. The algorithm starts with the seed $\mathcal{R} = \mathcal{S} = \{o, h, w\}$. (Observe that the set of fixed features for $\mathbb{P}_{\text{cxp}}$ is $\mathcal{S} \setminus \mathcal{R}$, which is empty at this point). In iteration 1, the predicate is called with features $\{h, w\}$, and observing the DL we can see the rules that can be triggered are R_1, R_2, R_3, R_4 and R_5 that classify as $(classes = -)$ and $(classes = +)$. Since there are points classifying to class $(classes = +)$, the predicate is true and the feature o is removed from $\mathcal{W}$. In iteration 2 we observe the predicate is called with feature $\{w\}$. In this case, the rules R_1 and R_3 could be triggered, that classify as $\{classes = -\}$. So the predicate is false, and the feature h is not removed. Finally, in iteration 3, the predicate is called with $\{h\}$, so that rules R_1, R_2 and R_3 could be triggered, and classify as $(classes = -)$ and $(classes = +)$. So the predicate is true, and the feature w is removed. The algorithm terminates with $\mathcal{W} = \{h\}$ as the reported CXp.

We present additional detail for computing explanations in general framework. We summarize the logic encoding of $\mathbb{P}_{\text{axp}}$, thus allowing the use of the algorithms proposed in Section 5.

For decision lists, we briefly summarize the logic encodings detailed in [34, 50]. We consider a generalized instance $(\mathbf{z}, \mathcal{T})$. The condition for a prediction to change is that one of the rules R_j that predicts one of the classes not in $\mathcal{T}$ must fire, and none of the rules that predicts a class in $\mathcal{T}$ and that precedes R_j must not fire. The previous condition can be encoded in a suitable logic formula, the expressivity of which will depend on the domains of the features.

A generalization that can be made is with regards to querying the participation of features in explanations. Namely the queries of necessity, relevancy and irrelevancy [25].

Iter.	outc, s	$\mathcal{P}$	$\neg\mathbb{P}_{axp}(\mathcal{P})$	$\neg\mathbb{P}_{axp}(\mathcal{P}\setminus\{t\})$	Add to $\mathcal{H}$
					(s_w)
1	$\top, \{s_o = s_h = 0, s_w = 1\}$	$\{w\}$	$\top$		$(s_o \vee s_h)$
2	$\top, \{s_h = 0, s_o = s_w = 1\}$	$\{o, w\}$	$\top$		(s_h)
3	$\top, \{s_o = 0, s_h = s_w = 1\}$	$\{h, w\}$	$\top$		(s_o)
4	$\top, \{s_o = s_h = s_w = 1\}$	$\{o, h, w\}$	$\bot$	$\bot$	$(\neg s_o \vee \neg s_h)$
5	$\bot$				

Table B.17: Execution of Algorithm 3 for deciding the (ir)relevancy of feature w of the DL in Figure B.7, with instance to explain $((\mathit{sunny}, \mathfrak{u}, \mathit{high}, \text{false}), \{\mathit{classes} = -\})$.

Example 30 (Necessary, Relevant and Irrelevant Features - Decision Lists)**.** *Consider Example 24 with the DL in Figure B.7, and the instance* $(\mathbf{z}, \mathcal{T})$ *with* $\mathbf{z} = (\mathit{sunny}, \mathfrak{u}, \mathit{high}, \mathit{false})$*,* $\mathcal{T} = \{\mathit{classes} = -\}$ *with missing inputs from Example 27. The set of specified features is* $\mathcal{S} = \{o, h, w\}$*, and the set of explanations is* $\mathbb{A}_{\mathfrak{u}} = \{\{o, h\}\}$ *(Example 16).*

Feature w does not belong to the explanations, and so it is an irrelevant feature. On the other hand, both features o and h, belong to the only explanation, and so are both necessary and relevant features.

In the following we present some example executions of checking the consistency of features for the running examples. Similar to previous example executions, for demonstration purposes, we assume a SAT solver biased towards unsatisfying variables.

Example 31 ((Ir)Relevancy - Decision Lists)**.** *Consider the Example 24 with the DL in Figure B.7, and the instance in Example 27, for which we wish to compute the (ir)relevancy of feature w. An example execution of Algorithm 3 is as in Table B.17.*

In the first iteration $\mathcal{H}$ contains only (s_w) since w is the feature to check for relevancy. The formula is satisfiable and the SAT solver returns the assignment unsatisfying all selector variables except for s_w. The algorithm computes as candidate WeakAXp the set $\mathcal{P}$ containing only feature w, and calls the predicate $\mathbb{P}_{axp}(\{w\})$. We can observe from the rules of the DL in Figure B.7, that by fixing the value of feature w to false, then rule R_0 can be fired with for example the point $v_1 = (\mathit{overcast}, \mathit{hot}, \mathit{high}, \mathit{false})$. v_1 is *classified outside the target set $\mathcal{T}$ ($\kappa(v_1) = (classes = +)$). Then there is at least one point in the feature space that agrees with the instance point in feature w, and it is classified with a class not in $\mathcal{T}$. Thus the predicate returns false and its negation is true. $\{w\}$ is not a WeakAXp, and the algorithm adds to $\mathcal{H}$ the new blocking clause $(s_o \vee s_h)$.*

Both iteration 2 and 3 are similar to iteration 1. In iteration 2, $\mathcal{H}$ is satisfiable and the SAT solver returns the assignment satisfying both variables s_o and s_w. The candidate $\mathcal{P}$ is then $\{o, w\}$, which calling the predicate $\mathbb{P}_{axp}(\{o, w\})$ is false because considering for example the point $v_2 = (sunny, mild, normal, false)$ respects the values of the features o and w in the instance, and yet the classification is $\kappa(v_2) = (classes = +)$ (rule R_3 is fired) which is not a part of $\mathcal{T}$. Since the predicate is false, then its negation is true. $\{o, w\}$ is not a WeakAXp, and the algorithm adds to $\mathcal{H}$ the new blocking clause (s_h).

In iteration 3, $\mathcal{H}$ is again satisfiable and the SAT solver returns the assignment satisfying both variables s_h and s_w. The candidate $\mathcal{P}$ is then $\{h, w\}$, which calling the predicate $\mathbb{P}_{axp}(\{h, w\})$ is false because for example the point $v_3 = (overcast, mild, high, false)$ respects the values of the features h and w in the instance, and yet the classification is $\kappa(v_3) = (classes = +)$ (rule R_0 is fired) which is not a part of $\mathcal{T}$. Since the predicate is false, then its negation is true. $\{h, w\}$ is not a WeakAXp, and the algorithm adds to $\mathcal{H}$ the new blocking clause (s_o).

In iteration 4, $\mathcal{H}$ is again satisfiable but this time the SAT solver returns the assignment satisfying all variables s_o, s_h and s_w. The candidate $\mathcal{P}$ is $\{o, h, w\}$. Calling the predicate $\mathbb{P}_{axp}(\{o, h, w\})$ returns true, because the only rules that can be fired (respecting the values of the instance for the features o, h and w) are o the rules R_1 and R_3, both of which classify the points in classes of $\mathcal{T}$. Since the predicate is true, then its negation is false, meaning the candidate is a WeakAXp. The algorithm proceeds by checking if the candidate WeakAXp is indeed a WeakAXp without feature w, that is, it calls the predicate $\mathbb{P}_{axp}(\{o, h\})$. Similarly, any point respecting the values of $o = sunny$ and $h = high$ are classified either by firing rules R_1 or R_3, which again classify the points within $\mathcal{T}$. Thus the candidate WeakAXp does not require feature w to be a WeakAXp, and the algorithm adds the clause $(\neg s_o \vee \neg s_h)$.

Finally, in iteration 5, formula $\mathcal{H}$ is unsatisfiable and the algorithm returns with w being an irrelevant feature.

Appendix C. Monotonic Classifiers

Monotonic classifiers have been extensively studied in recent years [19, 48, 74, 78]. Monotonic classifiers consider that classes in $\mathcal{K}$ are ordered according to an order $\preceq$. Without loss of generality we assume $c_1 \preceq \ldots \preceq c_k$. Additionally, for each $i \in \mathcal{F}$, the domain D_i is limited by a lower bound $\lambda(i)$, and an upper bound $\mu(i)$, that is, $v_i \in D_i$ is such that $\lambda(i) \leq v_i \leq \mu(i)$. Given $\mathbf{v}^a, \mathbf{v}^b \in \mathbb{F}$, we say that $\mathbf{v}^a \leq \mathbf{v}^b$ iff for any $i \in \mathcal{F}$, then $v_i^a \leq v_i^b$. A classifier is said to be monotonic if $\mathbf{v}^a \leq \mathbf{v}^b$ implies that $\kappa(\mathbf{v}^a) \preceq \kappa(\mathbf{v}^b)$.

Example 32. *The monotonic example classifier is adapted from the example in [54]. The objective is to predict the grade of a student in a course. The grades are the classes in $\mathcal{K} = \{A, B, C, D, E, F\}$ with the (usual) ordering $F \preceq E \preceq D \preceq C \preceq B \preceq A$. The features are the components of the assessment and correspond to: Quiz, Exam, Homework, Project. These will be referred by their first letter, that is , $\mathcal{F} = \{Q, E, H, P\}$. The domains of the features are all equal to $D_i = \{0, \ldots, 10\}$ (for $i \in \mathcal{F}$). The auxiliary function $Score(\mathbf{x})$ containing the final score is considered, where $Score(\mathbf{x}) = (0.1x_Q + 0.4x_E + 0.2x_H + 0.3x_P)$. The classification function is given by the following:*

$$\begin{aligned}\kappa(\mathbf{x}) \quad = \quad & ITE(Score(\mathbf{x}) \geq 9, A,\\ & \quad ITE(Score(\mathbf{x}) \geq 7, B,\\ & \qquad ITE(Score(\mathbf{x}) \geq 5, C,\\ & \qquad\quad ITE(Score(\mathbf{x}) \geq 4, D,\\ & \qquad\qquad ITE(Score(\mathbf{x}) \geq 2, E, F)))))\end{aligned}$$

Observe that in the example the lower and upper bounds of all features values are 0 and 10. Also observe that, the example classification function is indeed monotonic. As an example consider the points $\mathbf{v}^b = (5, 5, 5, 5)$ and $\mathbf{v}^a = (10, 10, 10, 10)$. $\mathbf{v}_i^b = 5 \leq 10 = \mathbf{v}_i^a$ (for any $i \in \mathcal{F}$) and $\kappa(\mathbf{v}^b) = C \preceq A = \kappa(\mathbf{v}^a)$.

Example 33 (Abductive Explanations - Monotonic Classifiers)**.** *Consider the monotonic classifier from Example 32 for predicting the grade of students with regards to the components of the assessment. Consider that the assessment components of a given (perfect) student is represented by the point in the feature space $\mathbf{v} = (10, 10, 10, 10)$, and the instance $(\mathbf{v}, A)$.*

An example of an AXp is $\{E, H, P\}$, because any student having all grades 10 in Exam, Homework, and Project will have a Score of at least 9, thus the

	AXp's $\mathbb{A}$		CXp's $\mathbb{C}$
Example 33	$\{\{E, H, P\}\}$	Example 34	$\{\{E\}, \{H\}, \{P\}\}$

Table C.18: AXp's and CXp's from the monotonic classifier running example. Each AXp is a mhs of the CXp's and vice-versa.

student will be graded with an A. On the other hand, if we disregard any of the grades of Exam, Homework, or Project, then the values of the other features are not enough to guarantee that the overall Score will be of at least 9.

Example 34 (Contrastive Explanations - Monotonic Classifiers)**.** *Consider the monotonic classifier from Example 32 for predicting the grade of students with regards to the components of the assessment, and let the instance be* $(\mathbf{v}, A)$ *with* $\mathbf{v} = (10, 10, 10, 10)$ *(as in Example 33).*

This example only contains three CXp's with one feature each. These are the CXp's $\mathcal{X}_a = \{E\}$*,* $\mathcal{X}_b = \{H\}$ *and* $\mathcal{X}_c = \{P\}$*. For example, the points* $(10, 0, 10, 10)$*,* $(10, 10, 0, 10)$*, and* $(10, 10, 10, 0)$ *differ from* $\mathbf{v}$ *only in the features* E*,* H *and* P *respectively, and the classification is different from* A *in all cases (*C*,* B*,* B *respectively). Also,* $\mathcal{X}_a$*,* $\mathcal{X}_b$ *and* $\mathcal{X}_c$ *are all subset-minimal, thus making them CXp's.*

Table C.18 shows both the AXp's (Example 33) and CXp's (Example 34) from Example 32. As with the example with decision lists, the table shows clearly that the duality property is satisfied also in this case.

Example 35 (Missing Inputs Instance - Monotonic Classifiers)**.** *Consider the monotonic classifier from Example 32 for predicting the grade of students with regards to the components of the assessment. Similar to Examples 33 and 34, we consider good students with very good scores, but in this example we want to disregard the effect of the work done at home, that is, the Homework component of the assessment. The target group of students are represented with the assessment components given by the point in the feature space* $\mathbf{z} = (10, 10, \mathfrak{u}, 10)$*, and the instance to explain is* $(\mathbf{z}, \{A, B\})$*.*

Example 36 (General Framework Explanations - Monotonic Classifiers)**.** *Consider the monotonic classifier from Example 32 for predicting the grade of students. Let the instance to explain be* $(\mathbf{z}, \mathcal{T})$ *from Example 35 with* $\mathbf{z} = (10, 10, \mathfrak{u}, 10)$ *and* $\mathcal{T} = \{A, B\}$*.*

Iter.	i	$\mathbb{P}_{axp}$	$\mathcal{W}$
			$\{Q, E, P\}$
1	Q	$\top$	$\{E, P\}$
2	E	$\bot$	$\{E, P\}$
3	P	$\bot$	$\{E, P\}$

(a)

Iter.	i	$\mathbb{P}_{cxp}$	$\mathcal{W}$
			$\{Q, E, P\}$
1	Q	$\top$	$\{E, P\}$
2	E	$\top$	$\{P\}$
3	P	$\bot$	$\{P\}$

(b)

Table C.19: Example executions of Algorithm 1 for the monotonic classifier in Example 32 with instance $((10, 10, \mathfrak{u}, 10), \{A, B\})$. (a) Computing one AXp (b) Computing one CXp

In order to get a classification of A or B the Score of the student has to be of at least 7. Given the definition of the Score function, in order to guarantee such score of 7, then we need to select both the features E and P ($0.1x_Q + 0.4 \times 10 + 0.2 \times x_H + 0.3 \times 10 \geq 7$, for any $x_Q, x_H \in \{1, \ldots, 10\}$). That means, $\mathcal{X} = \{E, P\}$ is a WAXp, and since none of the features are enough on its own to guarantee the Score of 7, then $\mathcal{X}$ is subset-minimal and is a AXp. Setting the value of feature Q does not force the prediction to be A or B. Thus $\mathcal{X} = \{E, P\}$ is the only AXp.

By inspection, or by duality, both $\mathcal{Y}_a = \{E\}$ and $\mathcal{Y}_a = \{P\}$ are the CXps of the example, that is, setting the values of all other specified features, other than the ones in $\mathcal{Y}_a$ or $\mathcal{Y}_b$ (on their own), allows a change in the prediction to be outside $\mathcal{T} = \{A, B\}$. For example, a student with grades $(10, 10, 0, 0)$, has the same grade values in the features Q and E, which are the specified features of the instance to explain other than P, and its classification is $\kappa((10, 10, 0, 0)) = C \notin \mathcal{T}$.

On the other hand, the Monotonic classifier Example 35 considers $\mathbf{z} = (10, 10, \mathfrak{u}, 10)$, and $\kappa'(\mathbf{z}) = \{A\}$, which is a subset of the target classes ($\mathcal{T} = \{A, B\}$) considered for explanation by user request.

For the following example executions of Algorithm 1 with the remaining running examples, we will only present a summary of the full resulting iterations, since the execution procedures are similar to the examples in Section 5.1.

Example 37 (Compute One Xp – Monotonic Classifier)**.** *Consider the monotonic classifier in Example 32 for which we wish to compute an explanation for the point $(10, 10, \mathfrak{u}, 10)$ with prediction $\{A, B\}$ as in Example 35. Example executions of Algorithm 1 for computing Xps are shown in Table C.19.*

In the two executions, initially the specified features are $\mathcal{S} = \{Q, E, P\}$ and the unspecified feature is $\mathcal{U} = \{H\}$.

In the example execution for computing an AXp (in Table C.19(a)), the predicate $\mathbb{P}_{\text{axp}}$ checks if there exists a point in the feature space that classifies different than $\mathcal{T} = \{A, B\}$. The algorithm starts with the seed $\mathcal{R} = \mathcal{S} = \{Q, E, P\}$. In iteration 1, the predicate is called with features $\{E, P\}$, and observing the monotonic classifier we can see the range of Score is $[7 - 10]$, that classify in $\{A, B\}$. Since there are no points classifying out of $\mathcal{T}$, the predicate is true and the feature is removed from $\mathcal{W}$. In iteration 2, the predicate is called with feature $\{P\}$, and observing the monotonic classifier we can see the range of Score is $[3 - 10]$, that classify in $\{A, B, C, D\}$. Since there are points classifying out of $\mathcal{T}$, the predicate is false and the feature is not removed from $\mathcal{W}$. In iteration 3, the predicate is called with feature $\{E\}$, and observing the monotonic classifier we can see the range of Score is $[4 - 10]$, that classify as $\{A, B, C, D\}$. Since there are points classifying out of $\mathcal{T}$, the predicate is false and the feature is not removed from $\mathcal{W}$. The algorithm terminates with $\mathcal{W} = \{E, P\}$ as the reported AXp.

In the example execution for computing an CXp (in Table C.19(b)), the predicate $\mathbb{P}_{\text{cxp}}$ checks if there exists a point in the feature space that classifies different than $\mathcal{T} = \{A, B\}$. The algorithm starts with the seed $\mathcal{R} = \mathcal{S} = \{Q, E, P\}$. In iteration 1, the predicate is called with features $\{E, P\}$, and observing the monotonic classifier we can see the range of Score is $[1 - 10]$, that classify as $\{A, B, C, D, E\}$. Since there are points classifying out of $\mathcal{T}$, the predicate is true and the feature is removed from $\mathcal{W}$. In iteration 2, the predicate is called with feature $\{P\}$, and observing the monotonic classifier we can see the range of Score is $[5 - 10]$, that classify in $\{A, B, C\}$. Since there are points classifying out of $\mathcal{T}$, the predicate is true and the feature is removed from $\mathcal{W}$. In iteration 3, the predicate is called with no feature, and observing the monotonic classifier we can see the range of Score is $[8 - 10]$, that classify in $\{A, B\}$. Since there are no points classifying out of $\mathcal{T}$, the predicate is false and the feature is not removed from $\mathcal{W}$. The algorithm terminates with $\mathcal{W} = \{P\}$ as the reported CXp.

We present additional detail for computing explanations in general framework. For the monotonic classifiers, we outline polynomial-time algorithms. We will briefly summarize the algorithm proposed in [54].

Suppose we split $\mathcal{F}$ into a set $\mathcal{S}$ of fixed features and a set $\mathcal{U}$ of non-fixed features. For each feature in $\mathcal{U}$ we know its lower bound and its upper bound. Hence, due to monotonicity, the smallest possible predicted class c_m is obtained when the features in $\mathcal{U}$ are assigned to their lower bound, and the features in $\mathcal{S}$ are assigned to the values dictated by $\mathbf{z}$. Similarly, the largest possible predicted class c_M is obtained when the features in $\mathcal{U}$ are assigned to their upper bound, and the features in $\mathcal{S}$ are assigned to the values dictated by $\mathbf{z}$. Moreover, if range of values between c_m and c_M intersects classes not in $\mathcal{T}$, then the set of fixed features is not sufficient for the prediction; otherwise the set of fixed features is sufficient for the prediction. Clearly, assuming the classifier computes predictions in polynomial time on the size of its representation, then the algorithm for deciding $\mathbb{P}_{\text{axp}}$ runs in polynomial time.

A generalization that can be made is with regards to querying the participation of features in explanations. Namely the queries of necessity, relevancy and irrelevancy [25].

Example 38 (Necessary, Relevant and Irrelevant Features - Monotonic Classifiers)**.** *Consider the monotonic classifier in Example 32, and the instance* $(\mathbf{z}, \mathcal{T})$ *with* $\mathbf{z} = (10, 10, \mathfrak{u}, 10)$*,* $\mathcal{T} = \{A, B\}$ *with missing inputs from Example 35. The set of specified features is* $\mathcal{S} = \{Q, E, P\}$*, and the set of explanations is* $\mathbb{A}_{\mathfrak{u}} = \{\{E, P\}\}$ *(Example 17).*

Feature Q does not belong to the explanations, and so it is an irrelevant feature. On the other hand, both features E and P, belong to the only explanation, and so are both necessary and relevant features.

In the following we present some example executions of checking the consistency of features for the running examples. Similar to previous example executions, for demonstration purposes, we assume a SAT solver biased towards unsatisfying variables.

Example 39 ((Ir)Relevancy - Monotonic Classifiers)**.** *Consider the monotonic classifier in Example 32, and the instance in Example 35, for which we wish to compute the (ir)relevancy of feature E (the exam part). An example execution of Algorithm 3 is as in Table C.20.*

In the first iteration $\mathcal{H}$ contains only (s_E) since E is the feature to check for relevancy. The formula is satisfiable and the SAT solver returns the assignment unsatisfying all selector variables except for s_E. The algorithm computes as candidate WeakAXp the set $\mathcal{P}$ containing only feature E, and calls the predicate $\mathbb{P}_{axp}(\{E\})$. The predicate asks for the inexistance of a point in the feature space that sets the grade of the exam to 10 and yet its

Iter.	outc, s	$\mathcal{P}$	$\neg\mathbb{P}_{axp}(\mathcal{P})$	$\neg\mathbb{P}_{axp}(\mathcal{P}\setminus\{t\})$	Add to $\mathcal{H}$
					(s_E)
1	$\top, \{s_Q = s_P = 0, s_E = 1\}$	$\{E\}$	$\top$		$(s_Q \vee s_P)$
2	$\top, \{s_P = 0, s_Q = s_E = 1\}$	$\{Q, E\}$	$\top$		(s_P)
3	$\top, \{s_Q = 0, s_P = s_E = 1\}$	$\{P, E\}$	$\bot$	$\top$	

Table C.20: Execution of Algorithm 3 for deciding the (ir)relevancy of feature E of the monotonic classifier in Example 32, with instance to explain $((10, 10, \mathfrak{u}, 10), \{A, B\})$.

classification not in A or B. For example, students with grades as $(0, 10, 0, 0)$ *have a assessment that respects the exam part to be 10, and yet their Score is of 4 which means a classification of D (not in* $\mathcal{T}$*). Thus the predicate is false, and its negation is true. This means the candidate is not a WeakAXp and the algorithm adds to* $\mathcal{H}$ *the clause* $(s_Q \vee s_P)$*.*

In the second iteration, formula $\mathcal{H}$ *is satisfiable and the SAT solver returns an assignment that satisfies both variables* s_Q *and* s_E *(for example). The candidate WeakAXp corresponds to the set* $\{Q, E\}$*, and the algorithm asks if* $\{Q, E\}$ *is a WeakAXp, that is, it calls the predicate* $\mathbb{P}_{axp}(\{Q, E\})$*. Similar than before, students with grades* $(10, 10, 0, 0)$ *have an assessment that respects both the quiz with value 10 and exam value 10, yet their Score is of 5 which means a classification of C not in* $\mathcal{T}$*. Thus,* $\mathbb{P}_{axp}(\{Q, E\})$ *is false and its negation is true. This means* $\{Q, E\}$ *is not a WeakAXp and the algorithm adds to* $\mathcal{H}$ *the clause* (s_P)*.*

Finally, in the third iteration, formula $\mathcal{H}$ *is again satisfiable, and this time the SAT solver returns an assignment that satisfies both variables* s_P *and* s_E *(for example). The candidate WeakAXp corresponds to the set* $\{P, E\}$*, and the algorithm asks if* $\{P, E\}$ *is a WeakAXp, that is, it calls the predicate* $\mathbb{P}_{axp}(\{P, E\})$*. Unlike the previous iterations, setting the exam grade to 10 and the project grade to 10 will force the student to have a Score of at least 7, which means the final grade of the student will be either A or B, all of which are in* $\mathcal{T}$*. Thus,* $\mathbb{P}_{axp}(\{P, E\})$ *is true and* $\{P, E\}$ *is a WeakAXp. The algorithm proceeds by checking if removing the exam component, the candidate is still a WeakAXp, that is, it calls* $\mathbb{P}_{axp}(\{P\})$*. This time, students with grades* $(0, 0, 0, 10)$ *have an assessment that respects the project with value 10, yet their Score is of 3 which means a final grade of E not in* $\mathcal{T}$*.* $\mathbb{P}_{axp}(\{Q, E\})$ *is then false, and feature E is necessary for* $\{P, E\}$ *to be a WeakAXp. Meaning* *there is an AXp in $\{P, E\}$ that contains E. The algorithm terminates with E as a relevant feature.*

References

[1] Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M.I., Rudin, C., 2017a. Learning certifiably optimal rule lists, in: SIGKDD, ACM. pp. 35–44.

[2] Angelino, E., Larus-Stone, N., Alabi, D., Seltzer, M.I., Rudin, C., 2017b. Learning certifiably optimal rule lists for categorical data. J. Mach. Learn. Res. 18, 234:1–234:78.

[3] Arenas, M., Barceló, P., Romero, M., Subercaseaux, B., 2022. On computing probabilistic explanations for decision trees, in: NeurIPS.

[4] Arora, S., Barak, B., 2009. Computational Complexity - A Modern Approach. Cambridge University Press.

[5] Audemard, G., Bellart, S., Bounia, L., Koriche, F., Lagniez, J., Marquis, P., 2021. On the computational intelligibility of boolean classifiers, in: KR, pp. 74–86.

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

[7] Audemard, G., Koriche, F., Marquis, P., 2020. On tractable XAI queries based on compiled representations, in: KR, pp. 838–849.

[8] Bertsimas, D., Dunn, J., 2017. Optimal classification trees. Mach. Learn. 106, 1039–1082.

[9] Biere, A., Heule, M., van Maaren, H., Walsh, T. (Eds.), 2021. Handbook of Satisfiability, IOS Press.

[10] Biradar, G., Izza, Y., Lobo, E., Viswanathan, V., Zick, Y., 2024. Axiomatic aggregations of abductive explanations, in: AAAI, pp. 11096–11104.

[11] Boumazouza, R., Alili, F.C., Mazure, B., Tabia, K., 2021. ASTERYX: A model-agnostic sat-based approach for symbolic and score-based explanations, in: CIKM, pp. 120–129.

[12] Breiman, L., 2001. Statistical modeling: The two cultures. Statistical science 16, 199–231.

[13] Carbonnel, C., Cooper, M.C., Marques-Silva, J., 2023. Tractable explaining of multivariate decision trees, in: KR, pp. 127–135.

[14] Clark, P., Niblett, T., 1989. The CN2 induction algorithm. Mach. Learn. 3, 261–283.

[15] Cooper, M.C., Marques-Silva, J., 2021. On the tractability of explaining decisions of classifiers, in: Michel, L.D. (Ed.), CP, pp. 21:1–21:18.

[16] Cooper, M.C., Marques-Silva, J., 2023. Tractability of explaining classifier decisions. Artif. Intell. 316, 103841. URL: https://doi.org/10.1016/j.artint.2022.103841, doi:10.1016/j.artint.2022.103841.

[17] EU, 2016. General Data Protection Regulation. https://eur-lex.europa.eu/eli/reg/2016/679/oj. Accessed: 2021-12-01.

[18] EU, 2021. Artificial Intelligence Act. tiny.cc/wy8juz. Accessed: 2021-12-01.

[19] Fard, M.M., Canini, K.R., Cotter, A., Pfeifer, J., Gupta, M.R., 2016. Fast and flexible monotonic functions with ensembles of lattices, in: NeurIPS, pp. 2919–2927.

[20] Friedrich, G., Gottlob, G., Nejdl, W., 1990. Hypothesis classification, abductive diagnosis and therapy, in: ESE, pp. 69–78.

[21] Gorji, N., Rubin, S., 2022. Sufficient reasons for classifier decisions in the presence of domain constraints, in: AAAI, pp. 5660–5667.

[22] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D., 2019. A survey of methods for explaining black box models. ACM Comput. Surv. 51, 93:1–93:42.

[23] Gunning, D., Aha, D.W., 2019. Darpa's explainable artificial intelligence (XAI) program. AI Mag. 40, 44–58. doi:10.1609/aimag.v40i2.2850.

[24] Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., Yang, G., 2019. XAI - explainable artificial intelligence. Sci. Robotics 4. doi:10.1126/scirobotics.aay7120.

[25] Huang, X., Cooper, M.C., Morgado, A., Planes, J., Marques-Silva, J., 2023a. Feature necessity & relevancy in ml classifier explanations, in: ETAPS (Ed.), Proc. in TACAS23.

[26] Huang, X., Izza, Y., Ignatiev, A., Cooper, M.C., Asher, N., Marques-Silva, J., 2022. Tractable explanations for d-DNNF classifiers, in: AAAI, pp. 5719–5728.

[27] Huang, X., Izza, Y., Ignatiev, A., Marques-Silva, J., 2021a. On efficiently explaining graph-based classifiers, in: KR, pp. 356–367.

[28] Huang, X., Izza, Y., Ignatiev, A., Marques-Silva, J., 2021b. On efficiently explaining graph-based classifiers. CoRR abs/2106.01350. doi:10.48550/arXiv.2106.01350.

[29] Huang, X., Izza, Y., Marques-Silva, J., 2023b. Solving explainability queries with quantification: The case of feature relevancy, in: AAAI.

[30] Huang, X., Marques-Silva, J., 2023. From robustness to explainability and back again. arXiv doi:10.48550/arXiv.2306.03048.

[31] Huang, X., Marques-Silva, J., 2023. Inadequacy of Shapley values for explainability. CoRR abs/2302.08160. doi:10.48550/arXiv.2302.08160.

[32] Ignatiev, A., 2020. Towards trustable explainable AI, in: IJCAI, pp. 5154–5158.

[33] Ignatiev, A., Izza, Y., Stuckey, P.J., Marques-Silva, J., 2022. Using MaxSAT for efficient explanations of tree ensembles, in: AAAI, pp. 3776–3785.

[34] Ignatiev, A., Marques-Silva, J., 2021. SAT-based rigorous explanations for decision lists, in: SAT, pp. 251–269.

[35] Ignatiev, A., Narodytska, N., Asher, N., Marques-Silva, J., 2020. From contrastive to abductive explanations and back again, in: AIxIA, pp. 335–355.

[36] Ignatiev, A., Narodytska, N., Marques-Silva, J., 2019a. Abduction-based explanations for machine learning models, in: AAAI, pp. 1511–1519.

[37] Ignatiev, A., Narodytska, N., Marques-Silva, J., 2019b. On relating explanations and adversarial examples, in: NeurIPS, pp. 15857–15867.

[38] Ignatiev, A., Narodytska, N., Marques-Silva, J., 2019c. On validating, repairing and refining heuristic ML explanations. CoRR abs/1907.02509. doi:10.48550/arXiv.1907.02509.

[39] Izza, Y., Huang, X., Ignatiev, A., Narodytska, N., Cooper, M., Marques-Silva, J., 2023. On computing probabilistic abductive explanations. International Journal of Approximate Reasoning 159, 108939.

[40] Izza, Y., Huang, X., Ignatiev, A., Narodytska, N., Cooper, M.C., Marques-Silva, J., 2022a. On computing probabilistic abductive explanations. CoRR abs/2212.05990. doi:10.48550/arXiv.2212.05990.

[41] Izza, Y., Huang, X., Morgado, A., Planes, J., Ignatiev, A., Marques-Silva, J., 2024. Distance-restricted explanations: Theoretical underpinnings & efficient implementation. CoRR abs/2405.08297. URL: https://doi.org/10.48550/arXiv.2405.08297, arXiv:2405.08297.

[42] Izza, Y., Ignatiev, A., Marques-Silva, J., 2020. On explaining decision trees. CoRR abs/2010.11034. doi:10.48550/arXiv.2010.11034.

[43] Izza, Y., Ignatiev, A., Marques-Silva, J., 2022b. On tackling explanation redundancy in decision trees. J. Artif. Intell. Res. 75, 261–321.

[44] Izza, Y., Marques-Silva, J., 2021. On explaining random forests with SAT, in: IJCAI, pp. 2584–2591.

[45] Lakkaraju, H., Bach, S.H., Leskovec, J., 2016. Interpretable decision sets: A joint framework for description and prediction, pp. 1675–1684.

[46] Lelis, V.M., Guzmán, E., Belmonte, M., 2020. Non-invasive meningitis diagnosis using decision trees. IEEE Access 8, 18394–18407. URL: https://doi.org/10.1109/ACCESS.2020.2966397, doi:10.1109/ACCESS.2020.2966397.

[47] Liffiton, M.H., Previti, A., Malik, A., 2016. Fast, flexible mus enumeration. Constraints , 223–250.

[48] Liu, X., Han, X., Zhang, N., Liu, Q., 2020. Certified monotonic neural networks, in: NeurIPS.

[49] Lundberg, S.M., Lee, S.I., 2017. A unified approach to interpreting model predictions, in: NeurIPS'17, p. 47684777.

[50] Marques-Silva, J., 2022. Logic-based explainability in machine learning, in: Reasoning Web, pp. 24–104.

[51] Marques-Silva, J., 2022. Logic-based explainability in machine learning. CoRR abs/2211.00541. doi:10.48550/arXiv.2211.00541.

[52] Marques-Silva, J., 2023. Disproving xai myths with formal methods–initial results, in: ICECCS (Ed.), Proc. in International Conference on Engineering of Complex Computer Systems.

[53] Marques-Silva, J., Gerspacher, T., Cooper, M.C., Ignatiev, A., Narodytska, N., 2020. Explaining naive bayes and other linear classifiers with polynomial time and delay, in: NeurIPS.

[54] Marques-Silva, J., Gerspacher, T., Cooper, M.C., Ignatiev, A., Narodytska, N., 2021. Explanations for monotonic classifiers, in: ICML, pp. 7469–7479.

[55] Marques-Silva, J., Ignatiev, A., 2022. Delivering trustworthy AI through formal XAI, in: AAAI, pp. 12342–12350.

[56] Marques-Silva, J., Ignatiev, A., 2023. No silver bullet: interpretable ml models must be explained. Frontiers in Artificial Intelligence 6. doi:10.3389/frai.2023.1128212.

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

[58] Mitchell, T.M., 1997. Machine learning, International Edition. McGraw-Hill Series in Computer Science, McGraw-Hill.

[59] Molnar, C., 2020. Interpretable machine learning. Lulu. com.

[60] Montavon, G., Samek, W., Müller, K., 2018. Methods for interpreting and understanding deep neural networks. Digit. Signal Process. 73, 1–15.

[61] Narodytska, N., Shrotri, A.A., Meel, K.S., Ignatiev, A., Marques-Silva, J., 2019. Assessing heuristic machine learning explanations with model counting, in: SAT, pp. 267–278.

[62] Reiter, R., 1987. A theory of diagnosis from first principles. Artif. Intell. 32, 57–95. doi:10.1016/0004-3702(87)90062-2.

[63] Ribeiro, M.T., Singh, S., Guestrin, C., 2016. "why should i trust you?" explaining the predictions of any classifier, in: SIGKDD, pp. 1135–1144.

[64] Ribeiro, M.T., Singh, S., Guestrin, C., 2018. Anchors: High-precision model-agnostic explanations, in: AAAI.

[65] Rivest, R.L., 1987. Learning decision lists. Machine learning 2, 229–246.

[66] Rudin, C., 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence 1, 206–215.

[67] Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., Zhong, C., 2022. Interpretable machine learning: Fundamental principles and 10 grand challenges. Statistics Surveys 16, 1–85.

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

[69] Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K.R., 2019. Explainable AI: interpreting, explaining and visualizing deep learning. volume 11700. Springer Nature.

[70] Samek, W., Müller, K.R., 2019. Towards explainable artificial intelligence. Explainable AI: interpreting, explaining and visualizing deep learning , 5–22.

[71] Selman, B., Levesque, H.J., 1990. Abductive and default reasoning: A computational core, in: AAAI, pp. 343–348.

[72] Shapley, L.S., 1953. A VALUE FOR n-PERSON GAMES. Princeton University Press. pp. 307–318.

[73] Shih, A., Choi, A., Darwiche, A., 2018. A symbolic approach to explaining bayesian network classifiers, in: IJCAI, pp. 5103–5111.

[74] Sivaraman, A., Farnadi, G., Millstein, T.D., den Broeck, G.V., 2020. Counterexample-guided learning of monotonic neural networks, in: NeurIPS.

[75] Tanner, L., Schreiber, M., Low, J.G.H., Ong, A., Tolfvenstam, T., Lai, Y.L., Ng, L.C., Leo, Y.S., Puong, L.T., Vasudevan, S.G., Simmons, C.P., Hibberd, M.L., Ooi, E.E., 2008. Decision tree algorithms predict the diagnosis and outcome of dengue fever in the early phase of illness. PLoS neglected tropical diseases 2, e196.

[76] Wäldchen, S., MacDonald, J., Hauch, S., Kutyniok, G., 2021. The computational complexity of understanding binary classifier decisions. J. Artif. Intell. Res. 70, 351–387.

[77] Wu, M., Wu, H., Barrett, C.W., 2023. VeriX: Towards verified explainability of deep neural networks, in: NeurIPS.

[78] You, S., Ding, D., Canini, K.R., Pfeifer, J., Gupta, M.R., 2017. Deep lattice networks and partial monotonic functions, in: NeurIPS, pp. 2981–2989.

[79] Yu, J., Ignatiev, A., Stuckey, P.J., 2023a. On formal feature attribution and its approximation. CoRR abs/2307.03380. URL: https://doi.org/10.48550/arXiv.2307.03380, arXiv:2307.03380.

[80] Yu, J., Ignatiev, A., Stuckey, P.J., Narodytska, N., Marques-Silva, J., 2022. Eliminating the impossible, whatever remains must be true. CoRR abs/2206.09551. doi:10.48550/arXiv.2206.09551.

[81] Yu, J., Ignatiev, A., Stuckey, P.J., Narodytska, N., Marques-Silva, J., 2023b. Eliminating the impossible, whatever remains must be true: on extracting and applying background knowledge in the context of formal explanations, in: AAAI, pp. 4123–4131.