Distance-Restricted Explanations: Theoretical Underpinnings & Efficient Implementation

Yacine Izza[1], **Xuanxiang Huang**[2], **Antonio Morgado**[3]
Jordi Planes[4], **Alexey Ignatiev**[5] and **Joao Marques-Silva**[6]
[1]CREATE, NUS, Singapore
[2]CNRS@CREATE, Singapore
[3]IST/INESC-ID, Portugal
[4]University of Lleida, Spain
[5]Monash University, Australia
[6]ICREA, University of Lleida, Spain
izza@comp.nus.edu.sg, xuanxiang.huang@cnrsatcreate.sg, ajrmorgado@inesc-id.pt,
jordi.planes@udl.cat, alexey.ignatiev@monash.edu, jpms@icrea.cat

Abstract

The uses of machine learning (ML) have snowballed in recent years. In many cases, ML models are highly complex, and their operation is beyond the understanding of human decision-makers. Nevertheless, some uses of ML models involve high-stakes and safety-critical applications. Explainable artificial intelligence (XAI) aims to help human decision-makers in understanding the operation of such complex ML models, thus eliciting trust in their operation. Unfortunately, the majority of past XAI work is based on informal approaches, that offer no guarantees of rigor. Unsurprisingly, there exists comprehensive experimental and theoretical evidence confirming that informal methods of XAI can provide human-decision makers with erroneous information. Logic-based XAI represents a rigorous approach to explainability; it is model-based and offers the strongest guarantees of rigor of computed explanations. However, a well-known drawback of logic-based XAI is the complexity of logic reasoning, especially for highly complex ML models. Recent work proposed distance-restricted explanations, i.e. explanations that are rigorous provided the distance to a given input is small enough. Distance-restricted explainability is tightly related with adversarial robustness, and it has been shown to scale for moderately complex ML models, but the number of inputs still represents a key limiting factor. This paper investigates novel algorithms for scaling up the performance of logic-based explainers when computing and enumerating ML model explanations with a large number of inputs.

1 Introduction

Recent years have witnessed remarkable progress in machine learning (ML). In some domains, systems of ML far exceed human-level capabilities. Motivated by an ever-increasing range of possible uses, the advances in ML are having a profound societal impact, and that is expected to continue in the foreseeable future. Nevertheless, trust in the operation of ML models is arguably the main obstacle to their widespread deployment. In application domains that directly affect human beings, the issue of trust is paramount. These domains include those deemed of high-risk or that are safety-critical. Given the complexity of the most widely used ML models (including highly complex neural networks (NNs)), one stepping stone for building trust in the operation of ML models is to be able to explain the operation of those models. This is the grand general objective of eXplainable Artificial Intelligence (XAI).

Most work on XAI is based on informal methods, that offer no guarantees of rigor (Ribeiro, Singh, and Guestrin, 2016; Lundberg and Lee, 2017; Ribeiro, Singh, and Guestrin, 2018). The alternative is formal XAI, where (local and/or global) explanations are represented by rigorous logic-based definitions, which are then computed using practically efficient automated reasoners (Marques-Silva and Ignatiev, 2022; Marques-Silva, 2022; Marques-Silva, 2024). However, formal XAI also exhibits a few crucial challenges, the most visible of which being the complexity of reasoning, among others (Marques-Silva, 2022). Despite the challenges, there has been observable progress in formal XAI since its inception in 2018/19 (Shih, Choi, and Darwiche, 2018; Ignatiev, Narodytska, and Marques-Silva, 2019a; Marques-Silva, 2022; Marques-Silva, 2024), including the ability for explaining complex tree ensembles (Izza and Marques-Silva, 2021; Ignatiev et al., 2022; Audemard et al., 2022b; Audemard et al., 2022a; Yu, Ignatiev, and Stuckey, 2023a; Izza, Meel, and Marques-Silva, 2023; Izza et al., 2024b;). However, and until recently, NNs represented a major challenge for formal XAI, with existing tools only capable of explaining very small NNs, i.e. with a few tens of activation units (Ignatiev, Narodytska, and Marques-Silva, 2019a). Recent work (Wu, Wu, and Barrett, 2022; Huang and Marques-Silva, 2023b; Wu, Wu, and Barrett, 2023) revisited a fundamental connection between explanations and adversarial examples. (This connection had been proved earlier (Ignatiev, Narodytska, and Marques-Silva, 2019b), but in the more generalized context of globally-defined explanations.) By introducing the concept of distance-restricted explanations (Huang and Marques-Silva, 2023b), one is able to compute those explanations using tools for finding adversarial examples. This result is significant given the remarkable progress observed in such tools in recent years (Brix et al., 2023). Using highly optimized tools for deciding the existence of adversarial examples, it is now possible to explain NNs with a few hundreds of neurons (Wu, Wu, and Barrett, 2023).

Despite this recent breakthrough, key challenges remain. The algorithms used for computing abductive explanations mimic the algorithms developed over the years for computing minimal unsatisfiable subsets (MUSes) of logic formulas, which are also referred to as minimal unsatisfiable cores (MUCs) in the case of constraint programming (Junker, 2004; Hemery et al., 2006). Well-known examples include the deletion-based algorithm (Chinneck and Dravnieks, 1991), the insertion-based algorithm (de Siqueira N. and Puget, 1988), the Quickxplain algorithm (Junker, 2004), but also dichotomic search (Hemery et al., 2006) and the progression algorithm (Marques-Silva, Janota, and Belov, 2013), among others (Marques-Silva and Lynce, 2011; Belov, Lynce, and Marques-Silva, 2012). More importantly, it is known that the same algorithms can also be used for computing contrastive explanations, which mimic minimal correction subsets (MCSes) of logic formulas, because in both cases one is computing a minimal set over a monotone predicate (Marques-Silva, Janota, and Mencía, 2017). Nevertheless, one fundamental limitation of such algorithms is that they operate in a mostly sequential fashion, allowing little to no flexibility for parallelization, i.e. constraints are excluded from a minimal set one at a time. This fundamental limitation becomes more acute in the case of abductive explanations when the number of features is very large. For complex NNs, each call for deciding the existence of an adversarial example can be time consuming. Running thousands of calls in sequence becomes a major performance bottleneck.

This paper details novel insights towards parallelizing algorithms for the computation of abductive explanations for ML models exhibiting a very large number of features. Furthermore, the paper identifies novel key properties of distance-restricted explanations, which allow not only the efficient computation of distance-restricted abductive explanations, but also the computation of distance-restricted contrastive explanations and their enumeration. Extended version of this paper including appendix is available in (Izza et al., 2024a).

2 Preliminaries

Classification problems. Classification problems are defined on a set of features $\mathcal{F} = \{1, \ldots, m\}$ and a set of classes $\mathcal{K} = \{c_1, \ldots, c_K\}$. Each feature i has a domain $\mathbb{D}_i$. Features can be ordinal or categorical. Ordinal features can be discrete or real-valued. Feature space is defined by the cartesian product of the features' domains: $\mathbb{F} = \mathbb{D}_1 \times \cdots \times \mathbb{D}_m$. A classifier computes a total function $\kappa : \mathbb{F} \to \mathcal{K}$. Throughout the paper, a classification problem $\mathcal{M}$ represents a tuple $\mathcal{M} = (\mathcal{F}, \mathbb{F}, \mathcal{K}, \kappa)$.

An instance (or a sample) is a pair $(\mathbf{v}, c)$, with $\mathbf{v} \in \mathbb{F}$ and $c \in \mathcal{K}$. An explanation problem $\mathcal{E}$ is a tuple $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$. The generic purpose of XAI is to find explanations for each given instance. Moreover, when reasoning in terms of robustness, we are also interested in the behavior of a classifier given some instances. Hence, we will also use explanation problems when addressing robustness.

Example 1. *Throughout the paper, we consider the following classification problem. The features are $\mathcal{F} = \{1, 2, 3\}$, all ordinal with domains $\mathbb{D}_1 = \mathbb{D}_2 = \mathbb{D}_3 = \mathbb{R}$. The set of classes is $\mathcal{K} = \{0, 1\}$. Finally, the classification function is $\kappa : \mathbb{F} \to \mathcal{K}$, defined as follows (with $\mathbf{x} = (x_1, x_2, x_3)$):*

$$\kappa(\mathbf{x}) = \begin{cases} 1 & \text{if } 0 < x_1 < 2 \wedge 4x_1 \geq (x_2 + x_3) \\ 0 & \text{otherwise} \end{cases}$$

Moreover, let the target instance be $(\mathbf{v}, c) = ((1, 1, 1), 1)$.

Norm l_p. The distance between two vectors $\mathbf{v}$ and $\mathbf{u}$ is denoted by $\|\mathbf{v} - \mathbf{u}\|$, and the actual definition depends on the norm being considered. Different norms l_p can be considered. For $p \geq 1$, the p-norm is defined as follows (Horn and Johnson, 2012):

$$\|\mathbf{x}\|_p \quad := \quad \left(\sum_{i=1}^{m} |x_i|^p\right)^{1/p} \tag{1}$$

Let $d_i = 1$ if $x_i \neq 0$, and let $d_i = 0$ otherwise. Then, for $p = 0$, we define the 0-norm, l_0, as follows (Robinson, 2003):

$$\|\mathbf{x}\|_0 \quad := \quad \sum_{i=1}^{m} d_i \tag{2}$$

In general, for $p \geq 1$, l_p denotes the Minkowski distance. Well-known special cases include the Manhattan distance l_1, the Euclidean distance l_2, and the Chebyshev distance $l_\infty = \lim_{p \to \infty} l_p$. l_0 denotes the Hamming distance. In the remainder of the paper, we use $p \in \mathbb{N}_0$ (but we also allow $p = \infty$ for the Chebyshev distance).

Adversarial examples. Let $\mathcal{M} = (\mathcal{F}, \mathbb{F}, \mathcal{K}, \kappa)$ be a classification problem. Let $(\mathbf{v}, c)$, with $\mathbf{v} \in \mathbb{F}$ and $c = \kappa(\mathbf{v})$, be a given instance. Finally, let $\epsilon > 0$ be a value of distance for norm l_p.

We say that there exists an adversarial example if the following logic statement holds true,

$$\exists(\mathbf{x} \in \mathbb{F}). (\|\mathbf{x} - \mathbf{v}\|_p \leq \epsilon) \wedge (\kappa(\mathbf{x}) \neq c) \tag{3}$$

(The logic statement above holds true if there exists a point $\mathbf{x}$ which is less than ϵ distance (using norm l_p) from $\mathbf{v}$, and such that the prediction changes.) If (3) is false, then the classifier is said to be ϵ-robust. If (3) is true, then any $\mathbf{x} \in \mathbb{F}$ for which the following predicate holds:[1]

$$\mathsf{AEx}(\mathbf{x}; \mathcal{E}, \epsilon, p) \;:=\; (\|\mathbf{x} - \mathbf{v}\|_p \leq \epsilon) \wedge (\kappa(\mathbf{x}) \neq c) \tag{4}$$

is referred to as an *adversarial example*.[2] Tools that decide the existence of adversarial examples will be referred to as *robustness tools*. (In the case of neural networks (NNs), the progress observed in robustness tools is documented by VNN COMP (Brix et al., 2023).)

Example 2. *For the classifier from Example 1, for distance l_1, and with $\epsilon = 1$, there exist adversarial examples by either setting $x_1 = 0$ or $x_1 = 2$.*

It may happen that we are only interested in inputs that respect some constraint, i.e. not all points of feature space are allowed or interesting. In such cases, we define adversarial examples subject to some constraint $\mathcal{C} : \mathbb{F} \to \{0, 1\}$, which are referred to as *constrained AExs*. In this case, the adver-

[1]Parameterizations are shown as predicate arguments positioned after ';'. These may be dropped for the sake of brevity.

[2]For regression problems (Wu, Wu, and Barrett, 2023) considers two distance-related parameters for characterizing adversarial examples: i) the distance to the point being considered, which corresponds to ϵ in the definitions above; and ii) the amount of change in the output that is deemed relevant, which corresponds to a maximum value $\delta > 0$ between the prediction of $\mathbf{v}$ and any other point restricted by the distance ϵ. Throughout this paper, we will only study classification problems; the case of regression problems could adopt the definitions from earlier work (Wu, Wu, and Barrett, 2023).

sarial examples must satisfy the following logic statement:

$$\mathcal{C}(\mathbf{x}) \wedge (\|\mathbf{x}-\mathbf{v}\|_p \leq \epsilon) \wedge (\kappa(\mathbf{x}) \neq c) \tag{5}$$

Clearly, the predicate AEx (see (4)) can be parameterized by the constraint being used.

Minimal hitting sets. Let $\mathcal{S}$ be a set and $\mathbb{B} \subseteq 2^{\mathcal{S}}$ be a set of subsets of $\mathcal{S}$. A hitting set (HS) $\mathcal{H} \subseteq \mathcal{S}$ of $\mathbb{B}$ is such that $\forall(\mathcal{P} \in \mathbb{B}).\mathcal{P} \cap \mathcal{H} \neq \emptyset$. A minimal hitting set (MHS) $\mathcal{Q} \subseteq \mathcal{S}$ is a hitting set of $\mathbb{B}$ such that no proper subset of $\mathcal{Q}$ is a hitting set of $\mathbb{B}$, i.e. $\forall(\mathcal{R} \subsetneq \mathcal{Q})\exists(\mathcal{P} \in \mathbb{B}).\mathcal{R} \cap \mathcal{P} = \emptyset$. A minimal hitting set is said to be subset-minimal or irreducible.

MUSes, MCSes, etc. The paper assumes basic knowledge of minimal unsatisfiable subsets (MUSes), minimal correction subsets (MCSes) and related concepts in the context of logic formulas. Examples of recent accounts of these concepts include (Marques-Silva, Janota, and Mencía, 2017; Marques-Silva and Mencía, 2020; Biere et al., 2021; Gupta, Genc, and O'Sullivan, 2021).

3 Logic-Based Explainability

In the context of explaining ML models, rigorous, model-based explanations have been studied since 2018 (Shih, Choi, and Darwiche, 2018; Ignatiev, Narodytska, and Marques-Silva, 2019a). We follow recent treatments of the subject (Marques-Silva and Ignatiev, 2022; Marques-Silva, 2022; Marques-Silva, 2024).A PI-explanation (which is also referred to as an abductive explanation (AXp) (Marques-Silva and Ignatiev, 2022)) is an irreducible subset of the features which, if fixed to the values specified by a given instance, are sufficient for the prediction.

Given an instance $(\mathbf{v}, c)$, a set of features $\mathcal{X} \subseteq \mathcal{F}$ is sufficient for the prediction if the following logic statement holds true,

$$\forall(\mathbf{x} \in \mathbb{F}).\left[\wedge_{i\in\mathcal{X}}(x_i = v_i)\right] \rightarrow (\kappa(\mathbf{x}) = c)) \tag{6}$$

If (6) holds, but $\mathcal{X}$ is not necessarily irreducible (i.e. it is not subset-minimal), then we say that $\mathcal{X}$ is a weak abductive explanation (WAXp). As a result, we associate a predicate WAXp with (6), such that $\mathsf{WAXp}(\mathcal{X};\mathcal{E})$ holds true iff (6) holds true. An AXp is a weak AXp that is subset-minimal. The predicate $\mathsf{AXp}(\mathcal{X};\epsilon,\mathcal{E})$ holds true iff set $\mathcal{X}$ is also an AXp. An AXp answers a *Why?* question, i.e. why is the prediction c (given the values assigned to the features).

A set of features $\mathcal{Y} \subseteq \mathcal{F}$ is sufficient for changing the prediction if the following logic statement holds true,

$$\exists(\mathbf{x} \in \mathbb{F}).\left[\wedge_{i\in\mathcal{F}\setminus\mathcal{Y}}(x_i = v_i)\right] \wedge (\kappa(\mathbf{x}) \neq c)) \tag{7}$$

If (7) holds, but $\mathcal{Y}$ is not necessarily irreducible, then we say that $\mathcal{Y}$ is a weak contrastive explanation (WCXp). As a result, we associate a predicate WCXp with (7), such that $\mathsf{WCXp}(\mathcal{Y};\mathcal{E})$ holds true iff (7) holds true. A CXp is a weak CXp that is also subset-minimal. The predicate $\mathsf{CXp}(\mathcal{Y};\mathcal{E})$ holds true iff set $\mathcal{Y}$ is a CXp. A CXp can be viewed as answering a *Why Not?* question (Miller, 2019; Ignatiev et al., 2020), i.e. why not some prediction other than c (given the values assigned to the features). Given an explanation problem, if at least one of the features in each CXp is not allowed to change value, then the prediction remains unchanged.

Example 3. *For the classifier of Example 1, and given the target instance, the AXp is $\{1, 2, 3\}$. Clearly, if we allow any feature to take any value, then we can change the prediction. Hence, the prediction does not change only if all features are fixed.*

Given the above, we define,

$$\mathbb{A}(\mathcal{E}) = \{\mathcal{X} \subseteq \mathcal{F} \,|\, \mathsf{AXp}(\mathcal{X};\mathcal{E})\} \tag{8}$$

$$\mathbb{C}(\mathcal{E}) = \{\mathcal{Y} \subseteq \mathcal{F} \,|\, \mathsf{CXp}(\mathcal{Y};\mathcal{E})\} \tag{9}$$

which capture, respectively, the set of all AXps and the set of all CXps given an explanation problem $\mathcal{E}$. We define related sets for weak AXps and CXps, as follows:

$$\mathbb{WA}(\mathcal{E}) = \{\mathcal{X} \subseteq \mathcal{F} \,|\, \mathsf{WAXp}(\mathcal{X};\mathcal{E})\} \tag{10}$$

$$\mathbb{WC}(\mathcal{E}) = \{\mathcal{Y} \subseteq \mathcal{F} \,|\, \mathsf{WCXp}(\mathcal{Y};\mathcal{E})\} \tag{11}$$

A simple observation is that WAXps are hitting sets of the sets of WCXps, and WCXps are hitting sets of the sets of WAXps. For example, if some WAXp $\mathcal{X}$ were not a hitting set of the set of WCXps, then there would exist some non-hit set $\mathcal{Y}$ that would enable changing the prediction whereas the disjoint set $\mathcal{X}$ would enable fixing the prediction; a contradiction. The previous observation can be substantially refined. The following result relating AXps and CXps is used extensively in devising explainability algorithms (Ignatiev et al., 2020).

Proposition 1 (MHS Duality between AXps and CXps). *Given an explanation problem $\mathcal{E}$,*

1. *A set $\mathcal{X} \subseteq \mathcal{F}$ is an AXp iff $\mathcal{X}$ a minimal hitting set of the CXps in $\mathbb{C}(\mathcal{E})$.*
2. *A set $\mathcal{Y} \subseteq \mathcal{F}$ is a CXp iff $\mathcal{Y}$ a minimal hitting set of the AXps in $\mathbb{A}(\mathcal{E})$.*

(Proposition 1 is a consequence of an earlier seminal result in model-based diagnosis (Reiter, 1987).) Proposition 1 is instrumental for enumerating abductive (but also contrastive) explanations (Ignatiev et al., 2020). In contrast with non-formal explainability, the navigation of the space of abductive (or contrastive) explanations, i.e. their enumeration, is a hallmark of formal XAI (Ignatiev et al., 2020; Audemard et al., 2021; Marques-Silva and Ignatiev, 2022). More importantly, enumeration of abductive and/or contrastive explanations found its application in computing formal *feature attribution* explanations (Yu, Ignatiev, and Stuckey, 2023b; Yu et al., 2023a; Biradar et al., 2024; Letoffe et al., 2024). More efficient algorithms for partial enumeration of explanations are described in (Yu et al., 2023a).

Progress in formal explainability is documented in recent overviews (Darwiche, 2023; Marques-Silva and Ignatiev, 2022, 2023; Marques-Silva, 2022; Marques-Silva, 2024), but also in recent publications (Arenas et al., 2022; Bassan and Katz, 2023; Bassan et al., 2023; Biradar et al., 2024; Carbonnel, Cooper, and Marques-Silva, 2023; Cooper and Marques-Silva, 2023; Darwiche and Hirth, 2023; Huang and Marques-Silva, 2023a; Huang et al., 2023; Huang, Izza, and Marques-Silva, 2023; Hurault and Marques-Silva, 2023; Izza et al., 2023, 2024b; Letoffe et al., 2024; Letoffe, Huang, and Marques-Silva, 2024; Liu and Lorini, 2023; Marques-Silva, 2023; Yu et al., 2023b).

4 Distance-Restricted Explainability

This section proposes a generalized definition of (W)AXps and (W)CXps, that take into account the l_p distance between $\mathbf{v}$ and the points that can be considered in terms of changing the prediction $c = \kappa(\mathbf{v})$.[3] The section starts by defining distance-restricted AXps/CXps, i.e. ∂AXpa/∂CXps, which take the l_p distance into account. Afterwards, the section proves a number of properties motivated by the proposed generalized definition of AXps & CXps, including that MHS duality between AXps and CXps extends to the distance-restricted definition of explanations.

Definitions. The standard definitions of AXps & CXps can be generalized to take a measure l_p of distance into account.

Definition 1 (Distance-restricted (W)AXp, ∂(W)AXp)**.** *For a norm l_p, a set of features $\mathcal{X} \subseteq \mathcal{F}$ is a distance-restricted weak abductive explanation (WAXp) for an instance $(\mathbf{v}, c)$, within distance $\epsilon > 0$ of $\mathbf{v}$, if the following predicate holds true,*

$$\partial\mathsf{WAXp}(\mathcal{X}; \mathcal{E}, \epsilon, p) \ := \ \forall(\mathbf{x} \in \mathbb{F}). \quad (12)$$
$$\left(\bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i) \wedge (\|\mathbf{x} - \mathbf{v}\|_p \leq \epsilon)\right) \rightarrow (\kappa(\mathbf{x}) = c)$$

If a (distance-restricted) weak AXp $\mathcal{X}$ is irreducible (i.e. it is subset-minimal), then $\mathcal{X}$ is a (distance-restricted) AXp, or ∂AXp.

Definition 2 (Distance-restricted (W)CXp, ∂(W)CXp)**.** *For a norm l_p, a set of features $\mathcal{Y} \subseteq \mathcal{F}$ is a weak contrastive explanation (WCXp) for an instance $(\mathbf{v}, c)$, within distance $\epsilon > 0$ of $\mathbf{v}$, if the following predicate holds true,*

$$\partial\mathsf{WCXp}(\mathcal{Y}; \mathcal{E}, \epsilon, p) \ := \ \exists(\mathbf{x} \in \mathbb{F}). \quad (13)$$
$$\left(\bigwedge\nolimits_{i \in \mathcal{F} \setminus \mathcal{Y}} (x_i = v_i) \wedge (\|\mathbf{x} - \mathbf{v}\|_p \leq \epsilon)\right) \wedge (\kappa(\mathbf{x}) \neq c)$$

If a (distance-restricted) weak CXp $\mathcal{Y}$ is irreducible, then $\mathcal{Y}$ is a (distance-restricted) CXp, or ∂CXp.

Furthermore, when referring to ∂AXps (resp. ∂CXps), the predicates ∂AXp (resp. ∂CXp) will be used.

Example 4. *For the classifier of Example 1, let the norm used be l_1, with distance value $\epsilon = 1$. From Example 2, we know that there exist adversarial examples, e.g. by setting $x_1 = 0$ or $x_1 = 2$. However, if we fix the value of x_1 to 1, then any assignment to x_2 and x_3 with $|x_2 - 1| + |x_3 - 1| \leq 1$, will not change the prediction. As a result, $\mathcal{X} = \{1\}$ is a distance-restricted AXp when $\epsilon = 1$. Moreover, by allowing only feature 1 to change value, we are able to change prediction, since we know there exists an adversarial example.*

The (distance-unrestricted) AXps (resp. CXps) studied in earlier work (Shih, Choi, and Darwiche, 2018; Ignatiev, Narodytska, and Marques-Silva, 2019a) represent a specific case of the distance-restricted AXps (resp. CXps) introduced in this section.

Remark 1. *Distance unrestricted AXps (resp. CXps) correspond to m-distance ∂AXps (resp. ∂CXps) for norm l_0, where m is the number of features.*

Properties. Distance-restricted explanations exhibit a number of relevant properties.

The following observation will prove useful in designing efficient algorithms (on the complexity of the oracle for adversarial examples) for finding distance-restricted AXps/CXps.

[3]Sections 4 and 5 substantially extend preliminary ideas contained in an earlier report (Huang and Marques-Silva, 2023b).

Proposition 2. *The predicates ∂WAXp and ∂WCXp are monotonically increasing with respect to set inclusion (i.e. they are both monotonic and up-closed).*

Proposition 2 mimics a similar earlier observation for WAXp and WCXp (e.g. see (Ignatiev et al., 2020)), and follows from monotonicity of entailment.

Moreover, it is apparent that ϵAXps and ϵCXps offer a rigorous definition of the concept of *local* explanations studied in non-formal XAI (Molnar, 2020).

Proposition 3. *Consider an explanation problem $\mathcal{E} = (\mathcal{M}, (\mathbf{v}, c))$ and some $\epsilon > 0$ for norm l_p. Let $\mathbf{x} \in \mathbb{F}$, with $\|\mathbf{x} - \mathbf{v}\|_p \leq \epsilon$, and let $\mathcal{D} = \{i \in \mathcal{F} \,|\, x_i \neq v_i\}$. Then,*

1. *If $\mathsf{AEx}(\mathbf{x}; \mathcal{E}, \epsilon, p)$ holds, then $\partial\mathsf{WCXp}(\mathcal{D}; \mathcal{E}, \epsilon, p)$ holds;*
2. *If $\partial\mathsf{WCXp}(\mathcal{D}; \mathcal{E}, \epsilon, p)$ holds, then $\exists(\mathbf{y} \in \mathbb{F}).||\mathbf{y} - \mathbf{v}||_p \leq ||\mathbf{x} - \mathbf{v}||_p \wedge \mathsf{AEx}(\mathbf{y}; \mathcal{E}, \epsilon, p)$.*

Proof sketch. We prove each claim separately.

1. This case is immediate. Given ϵ and $\mathcal{E}$, the existence of an adversarial example guarantees, by (13), the existence of a distance-restricted weak CXp.
2. By (13), if the features in $\mathcal{D}$ are allowed to change, given the distance restriction, then there exists at least one point $\mathbf{y}$ such that the prediction changes for $\mathbf{y}$, and the distance from $\mathbf{y}$ to $\mathbf{v}$ does not exceed that from $\mathbf{x}$ to $\mathbf{v}$. □

Given the definitions above, we generalize (8) and (9) to distance-restricted explanations, as follows:

$$\partial\mathbb{A}(\mathcal{E}, \epsilon; p) = \{\mathcal{X} \subseteq \mathcal{F} \,|\, \partial\mathsf{AXp}(\mathcal{X}; \mathcal{E}, \epsilon, p)\} \quad (14)$$
$$\partial\mathbb{C}(\mathcal{E}, \epsilon; p) = \{\mathcal{Y} \subseteq \mathcal{F} \,|\, \partial\mathsf{CXp}(\mathcal{Y}; \mathcal{E}, \epsilon, p)\} \quad (15)$$

As before, we define related sets for weak ∂AXps and ∂CXps, as follows:

$$\partial\mathbb{WA}(\mathcal{E}, \epsilon; p) = \{\mathcal{X} \subseteq \mathcal{F} \,|\, \partial\mathsf{WAXp}(\mathcal{X}; \mathcal{E}, \epsilon, p)\} \quad (16)$$
$$\partial\mathbb{WC}(\mathcal{E}, \epsilon; p) = \{\mathcal{Y} \subseteq \mathcal{F} \,|\, \partial\mathsf{WCXp}(\mathcal{Y}; \mathcal{E}, \epsilon, p)\} \quad (17)$$

In turn, this yields the following result regarding MHS duality between ∂(W)AXps and ∂(W)CXps.

Proposition 4. *Given an explanation problem $\mathcal{E}$, norm l_p, and a value of distance $\epsilon > 0$ then,*

1. *A set $\mathcal{X} \subseteq \mathcal{F}$ is a ∂AXp iff $\mathcal{X}$ is a MHS of the ∂CXps in $\partial\mathbb{C}(\mathcal{E}, \epsilon; p)$.*
2. *A set $\mathcal{Y} \subseteq \mathcal{F}$ is a ∂CXp iff $\mathcal{Y}$ is a MHS of the ∂AXps in $\partial\mathbb{A}(\mathcal{E}, \epsilon; p)$.*

MHS duality between $\partial\mathbb{A}(\mathcal{E}, \epsilon; p)$ and $\partial\mathbb{C}(\mathcal{E}, \epsilon; p)$ exhibits a special case when there are no adversarial examples, i.e. when the predicate $\mathsf{AEx}(\mathbf{x}; \mathcal{E}, \epsilon, p)$ (see (4)) does not hold for any $\mathbf{x} \in \mathbb{F}$.[4] In such a case, there is no ∂CXp and the unique ∂AXp is the empty set. As can be concluded, Proposition 4 holds even in such a situation.

Proposition 4 is instrumental for the enumeration of ∂AXps and ∂CXps, as shown in earlier work in the case of distance-unrestricted AXps/CXps (Ignatiev et al., 2020), since it enables adapting well-known algorithms for the enumeration of subset-minimal reasons of inconsistency (Liffiton et al., 2016).

Example 5. *For the running example, we have that $\partial\mathbb{A}(\mathcal{E}, 1; 1) = \partial\mathbb{C}(\mathcal{E}, 1; 1) = \{\{1\}\}$.*

[4]For example, when the given ϵ is so small that the ML model is constant within the considered ϵ-ball.

The definitions of distance-restricted AXps and CXps also reveal novel uses for abductive & contrastive explanations. For a given distance ϵ, a $\mathfrak{d}$AXp represents an irreducible sufficient reason for the ML model not to have an adversarial example.

The number of distance-restricted WAXps/WCXps is non-decreasing with the distance ϵ.

Proposition 5. *Let $\mathcal{E}$ represent some explanation problem. Let $0 < \epsilon < \eta$ and p some norm. For any $\mathcal{S} \subseteq \mathcal{F}$:*
1. *If $\mathfrak{d}\mathsf{WAXp}(\mathcal{S}; \mathcal{E}, \eta, p)$, then $\mathfrak{d}\mathsf{WAXp}(\mathcal{S}; \mathcal{E}, \epsilon, p)$;*
2. *If $\mathfrak{d}\mathsf{WCXp}(\mathcal{S}; \mathcal{E}, \epsilon, p)$, then $\mathfrak{d}\mathsf{WCXp}(\mathcal{S}; \mathcal{E}, \eta, p)$;*

A similar result does not hold for $\mathfrak{d}$AXps/$\mathfrak{d}$CXps, as the following examples illustrate:

Example 6. *Consider a classifier $\mathcal{M}$ defined on $\mathcal{F} = \{1, 2\}$ and $\mathcal{K} = \{0, 1\}$, with feature domains $\mathbb{D}_1 = \{0, 0.5, 1\}$ and $\mathbb{D}_2 = \{0, 0.5, 1\}$. Let $\kappa(x_1, x_2)$ be its classification function such that $\kappa(0.5, 0.5) = 0$, $\kappa(0, 1) = 0$, $\kappa(1, 0) = 0$. For any other point $\mathbf{x}$, we have $\kappa(\mathbf{x}) = 1$. Let $\mathcal{E} = (\mathcal{M}, ((1, 1), 1))$. Let us use l_∞, and suppose $\epsilon_1 = 0.5$ and $\epsilon_2 = 1$. For ϵ_1, there is one AEx $\{(0.5, 0.5)\}$, from which we deduce $\mathfrak{d}\mathbb{C}(\mathcal{E}, \epsilon_1; \infty) = \{\{1, 2\}\}$. For ϵ_2, there are three AEx $\{(0.5, 0.5), (1, 0), (0, 1)\}$, but then we can deduce $\mathfrak{d}\mathbb{C}(\mathcal{E}, \epsilon_2; \infty) = \{\{1\}, \{2\}\}$. In this case, $\{1, 2\} \notin \mathfrak{d}\mathbb{C}(\mathcal{E}, \epsilon_2; \infty)$.*

Example 7. *Consider a classifier $\mathcal{M}$ defined on $\mathcal{F} = \{1, 2, 3\}$ and $\mathcal{K} = \{0, 1\}$, with feature domains $\mathbb{D}_1 = \mathbb{D}_3 = \{-0.5, 0, 0.5, 1\}$, and $\mathbb{D}_2 = \{0, 0.5, 1\}$. Let $\kappa(x_1, x_2, x_3)$ be its classification function such that $\kappa(0.5, 0.5, 1) = 0$, $\kappa(1, 0.5, 0.5) = 0$, $\kappa(-0.5, 1, 1) = 0$ and $\kappa(1, 1, -0.5) = 0$. For any other point $\mathbf{x}$, we have $\kappa(\mathbf{x}) = 1$. Let $\mathcal{E} = (\mathcal{M}, ((1, 1, 1), 1))$. Let us use l_1, and suppose $\epsilon_1 = 1$ and $\epsilon_2 = 1.5$. For $\epsilon_1 = 1$, there are two AEx $\{(0.5, 0.5, 1), (1, 0.5, 0.5)\}$, from which we deduce $\mathfrak{d}\mathbb{C}(\mathcal{E}, \epsilon_1; 1) = \{\{1, 2\}, \{2, 3\}\}$. For $\epsilon_2 = 1.5$, there are four AEx $\{(0.5, 0.5, 1), (1, 0.5, 0.5), (-0.5, 1, 1), (1, 1, -0.5)\}$, but then we can deduce $\mathfrak{d}\mathbb{C}(\mathcal{E}, \epsilon_2; 1) = \{\{1\}, \{3\}\}$. By MHS, we have $\mathfrak{d}\mathbb{A}(\mathcal{E}, \epsilon_1; 1) = \{\{2\}, \{1, 3\}\}$, but $\mathfrak{d}\mathbb{A}(\mathcal{E}, \epsilon_2; 1) = \{\{1, 3\}\}$. Clearly, $\{2\} \notin \mathfrak{d}\mathbb{A}(\mathcal{E}, \epsilon_2; 1)$.*

Related work on distance-restricted explanations. Distance-restricted explanations have been studied in recent works (Wu, Wu, and Barrett, 2022; Huang and Marques-Silva, 2023b; Wu, Wu, and Barrett, 2023). The tight relationship between distance-restricted AXps and distance-unrestricted AXps is first discussed in (Huang and Marques-Silva, 2023b); other works (Wu, Wu, and Barrett, 2022; Wu, Wu, and Barrett, 2023) do not address this relationship. Furthermore, some properties of distance-restricted explanations can be related with those studied in earlier work on computing explanations subject to additional constraints (Cooper and Marques-Silva, 2021; Cooper and Marques-Silva, 2023) or subject to background knowledge (Gorji and Rubin, 2022; Yu et al., 2023b), including hitting-set duality.

5 Explanations Using Adversarial Examples

This section shows that the formal framework that has been developed in the case of computing and enumerating AXps & CXps can be adapted to the case of computing and enumerating $\mathfrak{d}$AXps and $\mathfrak{d}$CXps.

5.1 Computation of $\mathfrak{d}$AXps & $\mathfrak{d}$CXps

As noted earlier in the paper, existing algorithms for computing AXps mimic those for computing MUSes of logic formulas. The same observation can be made in the case of distance-restricted AXps, i.e. any algorithm developed for computing one MUS of a logic formula can be used for computing one $\mathfrak{d}$AXp. Furthermore, because the predicates WAXp, WCXp, $\mathfrak{d}\mathsf{WAXp}$ and $\mathfrak{d}\mathsf{WCXp}$ are monotone, the algorithms used for computing $\mathfrak{d}$AXps can be used for computing $\mathfrak{d}$CXps. As a result, throughout this section, we will only detail the computation of $\mathfrak{d}$AXps.

By double-negating (12), it is plain to conclude that a set of features $\mathcal{X}$ is a $\mathfrak{d}$WAXp if the following statement does *not* hold:

$$\exists(\mathbf{x} \in \mathbb{F}). \left(\bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i) \land (||\mathbf{x} - \mathbf{v}||_p \leq \epsilon)\right) \land (\kappa(\mathbf{x}) \neq c) \tag{18}$$

or, alternatively, that the following logic formula is *not* satisfiable:

$$\left(\bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i) \land (||\mathbf{x} - \mathbf{v}||_p \leq \epsilon)\right) \land (\kappa(\mathbf{x}) \neq c) \tag{19}$$

which corresponds to deciding for the non-existence of a constrained AEx when the set of constraints requires the features in $\mathcal{X}$ to take the values dictated by $\mathbf{v}$.

The observations above provide all the insights that justify the algorithms for distance-restricted explanations proposed in recent work (Wu, Wu, and Barrett, 2022; Huang and Marques-Silva, 2023b; Wu, Wu, and Barrett, 2023). Moreover, the same observations and the monotonicity of $\mathfrak{d}\mathsf{WAXp}$, allow mimicking the algorithms developed for MUS extraction (and also for the case of computing one AXp) to the case of computing one $\mathfrak{d}$AXp, provided the oracle used decides the non-existence of an adversarial example.

For the two algorithms described in this section, calls to the robustness oracle will be denoted by $\mathsf{FindAdvEx}$. Furthermore, we will require that the robustness oracle allows some features to be fixed, i.e. the robustness oracle decides the existence of constrained adversarial examples. As a result, the call to $\mathsf{FindAdvEx}$ uses as arguments the distance ϵ and the set of fixed features, and it is parameterized by the explanation problem $\mathcal{E}$ and the norm p used.

As outlined earlier, in cases where no AEx exists in the vicinity of an ϵ-ball, $\mathfrak{d}$AXp = $\emptyset$. Clearly, the algorithms discussed above for finding one $\mathfrak{d}$AXp will return $\emptyset$. However, adapting these algorithms for finding one $\mathfrak{d}$CXp (see (Ignatiev et al., 2020) for the case of AXp/CXp) would require an initial oracle call to check whether there exists at least one $\mathfrak{d}\mathsf{WCXp}$ (i.e. $\mathsf{FindAdvEx}(\epsilon, \emptyset; \mathcal{M}, (\mathbf{v}, c), p) = \textbf{true}$), otherwise the algorithm prints that there is no $\mathfrak{d}$CXp and terminates.

Deletion-based algorithm. Algorithm 1 summarizes the simplest algorithm for the computation of a single $\mathfrak{d}$AXp. (The same algorithm is often used for MUS extraction, but also for computing distance-unrestricted AXps/CXps. Since $\mathfrak{d}$AXps and $\mathfrak{d}$CXps are also examples of MSMP (minimal sets over a monotone predicate Marques-Silva, Janota, and Belov (2013); Marques-Silva, Janota, and Mencía (2017)), then the same algorithm can also be used for computing one $\mathfrak{d}$CXp.) As can be observed, given some l_p distance $\epsilon > 0$, the algorithm iteratively picks a feature to be allowed to be

Algorithm 1 Deletion algorithm to find ∂AXp

Input: Arguments: ϵ; Parameters: $\mathcal{E}, p$
Output: One ∂AXp $\mathcal{S}$

function FindAXpDel($\epsilon; \mathcal{E}, p$)
  $\mathcal{S} \leftarrow \mathcal{F}$ ▷ Initially, no feature is allowed to change
  for $i \in \mathcal{F}$ **do** ▷ Invariant: ∂WAXp($\mathcal{S}$)
    $\mathcal{S} \leftarrow \mathcal{S} \setminus \{i\}$
    hasAE ← FindAdvEx($\epsilon, \mathcal{S}; \mathcal{E}, p$)
    if hasAE **then**
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{i\}$
  return $\mathcal{S}$ ▷ ∂WAXp($\mathcal{S}$) ∧ minimal($\mathcal{S}$) → ∂AXp($\mathcal{S}$)

Algorithm 2 Dichotomic search algorithm to find ∂AXp

Input: Arguments: ϵ; Parameters: $\mathcal{E}, p$
Output: One ∂AXp $\mathcal{S}$

function FindAXpDichotomic($\epsilon; \mathcal{E}, p$)
  $(\mathcal{S}, \mathcal{W}) \leftarrow (\emptyset, \mathcal{F})$ ▷ Precondition: ∂WAXp($\mathcal{S} \cup \mathcal{W}$)
  while $\mathcal{W} \neq \emptyset$ **do** ▷ Invariant: $\exists(\mathcal{X} \in \partial\mathbb{A}).\mathcal{S} \subseteq \mathcal{X}$
    $(i, j) \leftarrow (0, |\mathcal{W}|)$
    while $i < j$ **do** ▷ Invariant: ∂WAXp($\mathcal{S} \cup \mathcal{W}_{1..j}$)
      $t \leftarrow \lfloor (i+j)/2 \rfloor$
      if FindAdvEx($\epsilon, \mathcal{S} \cup \mathcal{W}_{1..t}; \mathcal{E}, p$) **then**
        $i \leftarrow t + 1$ ▷ Fix more features
      else
        $j \leftarrow t$ ▷ Free more features
    $(\mathcal{S}, \mathcal{W}) \leftarrow (\mathcal{S} \cup \mathcal{W}_{j..j}, \mathcal{W}_{1..j-1})$
  return $\mathcal{S}$ ▷ $\exists(\mathcal{X} \in \partial\mathbb{A}).(\mathcal{S} \subseteq \mathcal{X}) \wedge (\mathcal{W} = \emptyset) \rightarrow$ ∂AXp($\mathcal{S}$)

unconstrained, starting by fixing all features to the values dictated by $\mathbf{v}$. If no adversarial example is identified, then the feature is left unconstrained; otherwise, it becomes fixed again. It is plain that the algorithm requires $\Theta(|\mathcal{F}|)$ calls to the robustness oracle. Furthermore, the robustness oracle must enable the fixing of some or all of the features. In the context of distance-restricted explanations, variants of the deletion algorithm were studied in recent work (Wu, Wu, and Barrett, 2022; Huang and Marques-Silva, 2023b; Wu, Wu, and Barrett, 2023).

It is convenient to introduce the concept of *transition feature* $i \in \mathcal{F}$. Given some set $\mathcal{S}$ of fixed features, $i \in \mathcal{F}$ is a transition feature if (a) when i is not fixed (i.e. $i \notin \mathcal{S}$), then an adversarial example exists; and (b) when i is fixed (i.e. $i \in \mathcal{S}$), then no adversarial example exists. The point is that i must be included in $\mathcal{S}$ for $\mathcal{S}$ to represent a ∂WAXp. Accordingly, the deletion algorithm iteratively decides whether a feature is a transition feature, given the features already in $\mathcal{S}$.

Dichotomic search algorithm. Aiming to reduce the overall running time of computing one ∂AXp, this paper seeks mechanisms to avoid the $\Theta(|\mathcal{F}|)$ *sequential* calls to the robustness oracle. For that, it will be convenient to study another (less used) algorithm, one that implements dichotomic (or binary) search (Hemery et al., 2006). The dichotomic search algorithm is shown in Algorithm 2.[5] At each iteration of the outer loop, the algorithm uses binary search in an internal loop to find a *transition feature*, i.e. fixing the features in $\mathcal{S} \cup \mathcal{W}_{1..j}$ does not yield an adversarial example, but fixing only the features in $\mathcal{S} \cup \mathcal{W}_{1..i-1}$ exhibits an adversarial example, if $i \neq 0$. (Upon termination of the inner loop, if $i = 0$, then $\mathcal{W} = \emptyset$.) Moreover, the features in $\mathcal{W}_{j+1..|\mathcal{W}|}$ can be safely discarded. As a result, it is the case that the inner loop of the algorithm maintains the following two invariants: (i) ∂WAXp($\mathcal{S} \cup \mathcal{W}_{1..j}$); and (ii) $(i = 0) \vee \neg$∂WAXp($\mathcal{S} \cup \mathcal{W}_{1..i-1}$). (The pseudo-code only shows the first invariant.) Clearly, the updates to i and j in the inner loop maintain the invariants. Moreover, it is easy to see that the features in $\mathcal{S}$ denote a subset of a ∂AXp, since we only add to $\mathcal{S}$ transition features; this represents the invariant of the outer loop. When there are no more features to analyze, then $\mathcal{S}$ will denote a ∂AXp. If k_M is the size of the largest ∂AXp, then the number of calls to the robustness oracle is $\mathcal{O}(k_M \log m)$. If the largest ∂AXp is significantly smaller than $\mathcal{F}$, then one can expect dichotomic search to improve the performance with respect to the deletion algorithm.

It will be helpful to view the dichotomic search algorithm as a procedure for analyzing chunks of features. A similar observation can be made with respect to the QuickXplain (Junker, 2004) and the Progression (Marques-Silva, Janota, and Belov, 2013) algorithms. We will later see that parallelization can be elicited by analyzing different chunks of features in parallel.

Other algorithms. As noted earlier, several other algorithms have been devised over the years. These include the well-known QuickXplain (Junker, 2004) algorithm, the insertion algorithm (de Siqueira N. and Puget, 1988), and the progression algorithm (Marques-Silva, Janota, and Belov, 2013), among others (Belov, Lynce, and Marques-Silva, 2012). Nevertheless, and despite some past attempts at parallelizing some of these algorithms (Belov, Manthey, and Marques-Silva, 2013), performance gains have been modest. Later in the paper, we revisit the dichotomic search algorithm, and introduce the SwiftXplain algorithm in Section 6.

In the case of MCSes or CXps, there are more efficient algorithms (in the number of oracle calls) that can be used (Marques-Silva and Mencía, 2020). One example is the *clause D* (CLD) algorithm (Marques-Silva et al., 2013), where elements that can be dropped from the minimal set are iteratively identified, and several can be removed in each oracle call. In contrast with algorithms for MUSes and AXps, an MCS (or CXp) can be decided with a single call using the so-called clause D (or disjunction clause). If the elements represented in the clause D represent a minimal set, then no additional elements can be found, and so the algorithm terminates by reporting the minimal set (Marques-Silva et al., 2013). As shown later in the paper, we can use parallelization to emulate the CLD algorithm in the case of computing one ∂AXp, (Evidently, the same observation also holds true for AXps and MUSes.)[6]

[5] With a slight abuse of notation, set $\mathcal{W}$ is assumed to be ordered, such that $\mathcal{W}_{a..b}$ denotes picking the elements (i.e. features) ordered from index a up to b. It is also assumed that $\mathcal{W}_{a..b}$, with $a = 0 \vee a > b$ represents an empty set.

[6] It should be underscored that the CLD algorithm has *never* been used in the past for finding MUSes, AXps, or ∂AXps. However, in the case parallelization is used, the CLD algorithm can be emulated, as shown later in the paper.

5.2 Additional Problems & Main Limitations

Enumeration of explanations. The standard solution for the enumeration of explanations is based on the MARCO algorithm (Liffiton et al., 2016; Marques-Silva and Mencía, 2020). This is the case with distance-unrestricted explanations (Ignatiev et al., 2020). Moreover, by changing the oracles used, and in light of duality between distance-restricted explanations, either the MARCO algorithm, or one of its variants, can also be used for enumerating distance-restricted explanations. Observe that enumeration is limited by the number of explanations and the time for computing each ∂AXp or ∂CXp.

Main limitations. The algorithms outlined in this section, or the examples used in recent work (Huang and Marques-Silva, 2023b; Wu, Wu, and Barrett, 2022, 2023), link the performance of computing explanations to the ability of deciding the existence of adversarial examples. More efficient tools for deciding the existence of adversarial examples (e.g. from Brix et al. (2023)) will result in more efficient algorithms for computing distance-restricted explanations. Nevertheless, one bottleneck of the algorithms discussed in this section is that the number of calls to an oracle deciding the existence of an adversarial example grows with the number of features. For complex ML models with a large number of features, the overall running time can become prohibitive. The next section outlines novel insights on how to reduce the overall running time by exploiting opportunities to parallelize calls to the robustness oracle.

6 SwiftXplain

Remarks about parallelization of finding one ∂AXp. As noted earlier in the paper, a limitation of algorithms for computing AXps, ∂AXps, or MUSes is that one element is identified at each step, and so it is unlikely that one can improve substantially over algorithms requiring $\mathcal{O}(m)$ calls to a robustness oracle.

Nevertheless, as one can conclude from inspection of dichotomic search algorithm (see Section 5), in the worst case, the main loop runs $\mathcal{O}(k_M)$ times the second loop and, each time, it calls the robustness oracle. In the case of the dichotomic search algorithm, the second loop requires $\mathcal{O}(\log m)$ calls to the robustness oracle. If we were able to reduce the number of robustness calls of the second loop to one, then the overall number of calls to a robustness oracle would be $\mathcal{O}(k_M)$ with a constant of 2.

Suppose that we could run in parallel as many calls to the robustness oracle as we deemed necessary. Then, we could run $\mathcal{O}(m)$ calls to decide which feature should be deemed a transition feature, as follows. For processor r, we would test the existence of an adversarial example by fixing the features in $C_r \triangleq \mathcal{S} \cup \mathcal{W}_{1..r}$. If for the value r, C_r does not yield an adversarial example, and C_{r-1} does yield such an adversarial example, then r represents the transition feature.

Given the algorithm outlined above, and assuming the existence of arbitrary processors to run the robustness tool in parallel, then we would be able to find a ∂AXp with $\mathcal{O}(k_M)$ calls to a robustness oracle. If $k_M \ll m$, then this would reduce (substantially) the number of sequential calls to the robustness oracle.[7] Furthermore, in practice, the number of features can be much larger than the number of available processors, and so we need to devise mechanisms to parallelize the internal loop of the dichotomic search algorithm, without using too many processors.

We can view the dichotomic search algorithm as a procedure for analyzing (possibly in parallel) different *chunks* of features, with the purpose of finding an element that *must* be included in some minimal set. Perhaps unsurprisingly, both the QuickXplain and the Progression algorithm can also be viewed as procedures for analyzing different chunks of features.

Moreover, and as hinted in Section 5, in the case of ∂AXps (and also AXps and MUSes) we can emulate the CLD algorithm with many parallel oracle calls. As before, we start by assuming that we have enough available processors. Suppose that $\mathcal{F}$ is partitioned into three sets. $\mathcal{S}$ denotes the elements that are known to be included in some minimal set, $\mathcal{N}$ denotes the elements that have been discarded from being included in some minimal set (i.e. these elements will no longer be used), and $\mathcal{W}$ denotes the elements we are unsure about. Let $q = |\mathcal{W}|$. Create q sets of fixed features: $\mathcal{S} \cup \mathcal{W} \setminus \{i\}$, for each $i \in \mathcal{W}$. In addition, run in parallel each of possible sets of fixed features, using q processors. If $\mathcal{S} \cup \mathcal{W}$ represents a minimal set, e.g. a ∂AXp or an AXp or an MUS, then all of the q calls will return an indication that some adversarial example was found. Thus, with a single step (involving q parallel oracle calls) we are able to decide that the target set $\mathcal{S} \cup \mathcal{W}$ represents a minimal set. In this case, all the features in $\mathcal{W}$ are added to $\mathcal{S}$ and the algorithm terminates by returning $\mathcal{S}$. Furthermore, if for some feature i, the oracle call decides that no adversarial example exists, then feature i can be dropped from $\mathcal{W}$ and added to $\mathcal{N}$. If no adversarial example is returned for multiple features, only one of the features can be dropped. In practice, implementing the CLD algorithm in parallel may require multiple sequential calls, concretely when the number of available processors is insufficient.

It should be noted that the parallelization of dichotomic search and of CLD serve different purposes, and so one must be able to pick the right parallel oracle calls to make. At each step, one can either mimic the dichotomic search algorithm to drop multiple features, or mimic the CLD algorithm. For example, one may start by running several steps of the (parallel) dichotomic search algorithm, and at some point switch to the (parallel) clause D algorithm. The rest of this section addresses how this can be done in practice.

A parallel algorithm for finding one ∂AXp. Algorithm 3 outlines the SwiftXplain algorithm, which runs in parallel on multi-core CPU or GPU. The procedure takes as input the explanation problem $\mathcal{E}$, an l_p distance $\epsilon > 0$, a threshold $\delta \in [0, 1]$ used for activating the optional feature disjunction check, and the number q of available processors; and returns a ∂AXp $\mathcal{S} \subseteq \mathcal{F}$. Intuitively, the algorithm implements a parallel dichotomic search by splitting the set of features to analyze into a collection of chunks and instruments decision

[7] Unfortunately, AXps can represent a significant percentage (often up to 65-75%) of the set of features. As a result, the dichotomic search algorithm is unlikely to outperform (in terms of robustness oracle calls) the basic deletion-based algotihm. However, as shown in Section 7, the sizes of ∂AXps can represent a small percentage (in some cases down to 15%) of the number of features; hence dichotomic search can become more effective.

Algorithm 3 SwiftXplain algorithm to find one ∂AXp

Input: Arguments: ϵ, q, δ; Parameters: $\mathcal{E}$, p
Output: One ∂AXp $\mathcal{S}$

function $\mathsf{SwiftXplain}(\epsilon, q, \delta; \mathcal{E}, p)$
    $(\mathcal{W}, \mathcal{S}) \leftarrow (\mathcal{F}, \emptyset)$ ▷ Precond: $\partial\mathsf{WAXp}(\mathcal{F}) \wedge (q \geq 2)$
    while $\mathcal{W} \neq \emptyset$ **do** ▷ $\mathcal{S} \subseteq \mathcal{X} \in \partial\mathbb{A}$
        if $|\mathcal{W}| < \delta \times |\mathcal{F}|$ **then** ▷ Run FD check
            $(\mathcal{W}, \mathcal{S}) \leftarrow \mathsf{FeatDisjunct}(\epsilon, q, \mathcal{W}, \mathcal{S}; \mathcal{E}, p)$
            continue
        $(\ell, u) \leftarrow (0, |\mathcal{W}|)$
        while $\ell + 1 < u$ **do** ▷ Inv.: $\partial\mathsf{WAXp}(\mathcal{S} \cup \mathcal{W}_{1..u})$
            $\omega \leftarrow \min(q, u - \ell)$ ▷ # parallel calls
            $\sigma \leftarrow \lfloor (u - \ell)/\omega \rfloor$ ▷ σ: chunk size
            $\mathcal{D} \leftarrow \{\ell + \iota \times \sigma \mid \iota \in \{1, \ldots, \omega\}\}$
            for $i \in \mathcal{D}$ **do in parallel**
                $\mathsf{AE}_i \leftarrow \mathsf{FindAdvEx}(\epsilon, \mathcal{S} \cup \mathcal{W}_{1..i}; \mathcal{E}, p)$
            $u \leftarrow \min(\{i \in \mathcal{D} \mid \mathsf{AE}_i = \textbf{false}\} \cup \{u\})$
            $\ell \leftarrow \max(\{i \in \mathcal{D} \mid i < u\} \cup \{\ell\})$
        if $u = 1 \wedge \neg\mathsf{FindAdvEx}(\epsilon, \mathcal{S}; \mathcal{E}, p)$ **then**
            return $\mathcal{S}$
        $(\mathcal{W}, \mathcal{S}) \leftarrow (\mathcal{W}_{1..u-1}, \mathcal{S} \cup \mathcal{W}_{u..u})$
    return $\mathcal{S}$

oracle calls checking the existence of adversarial examples, done in parallel on those chunks. Upon completion of such a parallel oracle call, the algorithm proceeds by zooming into a chunk that is deemed to contain a transition feature.

The algorithm starts by initializing the operational set of features $\mathcal{W}$ to contain all the features of $\mathcal{F}$ and a subset-minimal ∂AXp $\mathcal{S}$ to extract as $\emptyset$. (Note that one can potentially impose a heuristic feature order on $\mathcal{F}$ aiming to quickly remove irrelevant features.) In each iteration of the outer loop, the lower and upper bounds ℓ and u on feature indices are set, respectively, to 1 and $|\mathcal{W}|$.

Each transition element is determined in parallel by the inner loop of the algorithm, which implements dichotomic search and iterates until $\ell + 1 = u$. An iteration of this loop splits the set of features $\mathcal{W}$ into ω chunks determined by the splitting indices kept in $\mathcal{D} \subseteq \mathcal{W}$. (Note that the value of ω equals either the number of available CPUs q or the number of remaining features in $\mathcal{W}$, depending on which of these values is smaller.) Given the largest feature index $i \in \mathcal{D}$ in each such chunk, the iteration tests whether an adversarial example can be found while fixing the features $\mathcal{S} \cup \mathcal{W}_{1..i}$. The test is applied in parallel for all the splitting indices $i \in \mathcal{D}$ employing ω CPUs.

The aim of the algorithm is to determine the first case when an oracle call reports that no adversarial example exists, i.e. that $\mathsf{AE}_t = \textbf{false}$ s.t. $\mathsf{AE}_{\geq t} = \textbf{false}$ and $\mathsf{AE}_{<t} = \textbf{true}$. Importantly, as soon as such case t is determined, all the parallel jobs are terminated. (Note that in practice terminating the jobs after i s.t. $\mathsf{AE}_i = \textbf{false}$ and before i s.t. $\mathsf{AE}_i = \textbf{true}$ helps to save a significant amount of time spent on the parallel oracle calls.) The inner loop proceeds by zooming into the t'th chunk of features by updating the values of the lower and upper bounds ℓ and u as it is deemed to contain a transition feature. If all the oracle calls unanimously decide that an adversarial example exists (or does not exist), the algorithm

Algorithm 4 Parallelized feature disjunction check

Input: Arguments: ϵ, $\mathcal{W}$, $\mathcal{S}$; Parameters: $\mathcal{E}$, p

procedure $\mathsf{FeatDisjunct}(\epsilon, q, \mathcal{W}, \mathcal{S}; \mathcal{E}, p)$
    $\mathcal{T} \leftarrow \mathcal{W}_{\max(|\mathcal{W}|-q+1,1)..|\mathcal{W}|}$
    for $i \in \mathcal{T}$ **do in parallel** ▷ Traversing target set $\mathcal{T}$
        $\mathsf{AE}_i \leftarrow \mathsf{FindAdvEx}(\epsilon, \mathcal{S} \cup \mathcal{W} \setminus \{i\}; \mathcal{E}, p)$
    if $\bigwedge_{i \in \mathcal{T}} \mathsf{AE}_i = \textbf{true}$ **then** ▷ Fix more features
        return $(\mathcal{W} \setminus \mathcal{T}, \mathcal{S} \cup \mathcal{T})$
    else ▷ Free one feature
        $j \leftarrow \mathsf{PickRandom}(\{i \in \mathcal{T} \mid \mathsf{AE}_i = \textbf{false}\})$
        return $(\mathcal{W} \setminus \{j\}, \mathcal{S})$

proceeds by zooming into the corresponding *boundary* chunk of features. Note that if the value of the upper bound u is updated from $|\mathcal{W}|$ all the way down to 1, which happens if all the parallel oracle calls report no adversarial example, the algorithm needs to check whether set $\mathcal{S}$ is sufficient for the given prediction. If it is, the algorithm terminates by reporting $\mathcal{S}$. Otherwise, it collects a newly determined transition feature and proceeds by updating $\mathcal{W}$ and $\mathcal{S}$.

Finally, we integrate an analogue of the CLD procedure (Marques-Silva et al., 2013) widely used in the computation of minimal correction subsets (MCS) of an unsatisfiable CNF formula. The analogue is referred to as feature disjunction check (see FeatDisjunct in Algorithm 4) and used as an optional optimization step in Algorithm 3 at the beginning of the main (outer) loop. We implement a heuristic order over $\mathcal{F}$ and activate featDisjunct (given some threshold δ) at the last iterations of SwiftXplain, where it is likely to conclude that all the features in a selected subset $\mathcal{T} \subseteq \mathcal{W}$ of size $\min(q, |\mathcal{W}|)$ are relevant for the explanation and can be safely moved to $\mathcal{S}$ at once; otherwise, one can randomly pick a single feature in $\mathcal{T}$ among those verified as irrelevant features, i.e. removing the feature does not yield an adversarial example, and make it free. Finally, we observe that after running featDisjunct, the algorithm does not invoke dichotomic search in the subsequent iterations.

Related work. Previous work on ∂AXps considered sequential algorithms (Wu, Wu, and Barrett, 2022; Huang and Marques-Silva, 2023b; Wu, Wu, and Barrett, 2023). The same holds true for earlier work on computing AXps. In the more general setting of model-based diagnosis (MBD), existing work on parallelization considers solely local optimizations to sequential algorithms (Cardoso and Abreu, 2013; Jannach, Schmitz, and Shchekotykhin, 2015; Jannach, Schmitz, and Shchekotykhin, 2016; Shchekotykhin, Jannach, and Schmitz, 2018; Silva et al., 2020; Le et al., 2023). For finding one MUS, (Belov, Manthey, and Marques-Silva, 2013) proposes the parallelization of the deletion-based algorithm. This work is the closest to the ideas developed in this paper, with significant differences, including the parallelization of the dichotomic search algorithm, and the parallelization of the CLD algorithm.

7 Experimental Evidence

We assess our approach to computing ∂AXp for *deep* NNs on well known image data and compare with the SOTA deletion algorithm. Results on comparison with dichotomic search

Model	Deletion							SwiftXplain						
	avgC	nCalls	Len	Mn	Mx	avg	TO	avgC	nCalls	Len	FD%	Mn	Mx	avg
gtsrb-dense	0.06	1024	448	52.0	76.3	63.1	0	0.23	54	447	77.4	10.8	14.0	12.2
gtsrb-convSmall	0.06	1024	309	59.2	82.6	65.1	0	0.22	74	313	39.7	15.1	19.5	16.2
gtsrb-conv	—	—	—	—	—	—	100	96.49	45	174	33.2	3858.7	6427.7	4449.4
mnist-denseSmall	0.28	784	177	190.9	420.3	220.4	0	0.77	111	180	15.5	77.6	104.4	85.1
mnist-dense	0.19	784	231	138.1	179.9	150.6	0	0.75	183	229	11.5	130.1	145.5	136.8
mnist-convSmall	—	—	—	—	—	—	100	98.56	52	116	21.3	4115.2	6858.3	5132.8

Table 1: Detailed performance evaluation of computing ∂AXp for DNNs with SwiftXplain and comparison with the baseline linear search (deletion) algorithm. The table shows results for 2 image data, i.e. *MNIST* (image size: 28×28) and *GSTRB* (image size: 32×32), and for each row (DNN) a batch of 25 input images, randomly selected, were tested. Columns **avgC** and **nCalls** report, resp. the average time and average number of instrumented (AEx robustness) oracle calls. Column **avg** (resp. **Mn** and **Mx**) reports the average (resp. min and max) time in seconds to deliver a ∂AXp, and **TO** is the percentage of timeout tests. Lastly, column **Len** reports the average explanation length and **FD%** is the average percentage of successful FD calls to augment $\mathcal{S}$ with q features in one iteration.

can be found in (Izza et al., 2024a).

Experimental Setup. All experiments were carried out on a high-performance computer cluster with machines equipped with AMD EPYC 7713 processors. Each instance test is provided with 2 and 60 cores, resp., when running the sequential algorithms deletion (Algorithm 1), dichotomic (Algorithm 2), and our parallel SwiftXplain algorithm, namely 1 core for 1 oracle used and 1 additional core to run the main script. Furthermore, the memory limit was set to 16GB, and the time limit to 14400 seconds (i.e. 4 hours).

Prototype Implementation. The proposed approach was prototyped as a set of Python scripts[8], and PyTorch library (Paszke et al., 2019) was used to train and handle the learned DNNs. An unified Python interface for robustness oracles is implemented and it enables us to use any DNN reasoner of the VNN-COM (Brix et al., 2023). MN-BaB (Ferrari et al., 2022), which is a complete neural network verifier, is used to instrument AEx checking on CPU mode. (Note that one can run MN-BaB on (multi-)GPU with CUDA library to speed up the oracle calls, but this clearly would not change the results of our evaluation.) Moreover, Gurobi (Gurobi Optimization, LLC, 2023) is applied for empowering MN-BaB resolution. Other tools like Marabou (Katz et al., 2019) were also tested, but were unable to scale for large benchmarks, thus we focus solely on the results obtained with MN-BaB. Besides, we implemented the *pixel sensitivity* ranking heuristic, as described in (Wu, Wu, and Barrett, 2023), and LIME (Ribeiro, Singh, and Guestrin, 2016) for the traversal order of features in all algorithms outlined above. Note that preliminary results shown that traversal heuristic based on sensitivity score is faster to compute and often provide smaller explanations[9]. Therefore, presented results in this section are showing experiments applying pixel sensitivity heuristic. It is worth noting that the implemented deletion algorithm in our prototype corresponds to VeriX (Wu, Wu, and Barrett, 2023) (i.e. outlined procedure in Algorithm 1 augmented with pixel sensitivity ranking heuristic), but instead of using Marabou it calls MN-BaB. Additionally, when discussing the results we do not refer (explicitly) to VeriX but to the original algorithm (Chinneck and Dravnieks, 1991).

Image Recognition Benchmarks. The experiments focus on two well-known image datasets, that have been studied in (Wu, Wu, and Barrett, 2023). Namely, we evaluate the widely used *MNIST* (Deng, 2012; Paszke et al., 2019) dataset, which features hand-written digits from 0 to 9. Also, we consider the image dataset *GSTRB* (Stallkamp et al., 2012) of traffic signs, and we select a collection of training data that represents the top 10 classes of the entire data. For each data we train 3 DNNs, namely a fully-connected NN, a convolutional NN and a large convolutional NN. Trained DNNs contain up to ∼10,000 neurons for *MNIST* and ∼95,000 neurons for *GSTRB*. We considered different ϵ values for each model and image dataset, following the experimental protocol of (Wu, Wu, and Barrett, 2023), e.g. for MNIST benchmark ϵ varies from 0.003 to 0.15. Besides, the parameter δ in SwiftXplain is set to 0.75 for all benchmarks. Details about the training parameters and ϵ-ball ℓ_∞ distance are included in the appendix (Izza et al., 2024a).

Results. Table 1 summarizes the results comparing SwiftXplain and the baseline linear (deletion-based) method on fully-connected (dense) and convolutional NNs trained with the above image datasets. As can be observed from Table 1, SwiftXplain significantly outperforms the deletion algorithm on all tested benchmarks. More specifically, SwiftXplain is on average up to 4 times faster than deletion (e.g. 4.5 and 4 times faster on *gtsrb-dense* and *gtsrb-convSmall*), with a few exceptions on smaller image inputs but still takes the advantage on all computed ∂AXp's (i.e. 2.6 and 1.1 faster on *mnist-denseSmall* and *mnist-dense*). Moreover, the linear approach fails to produce a ∂AXp on larger (convolutional) models for all tested image samples within 4-hours time limit, whilst SwiftXplain successfully finds an explanation on all tests with an average runtime of 4449.4 sec (∼74 min) and 5132.8 sec (∼85 min), resp., on convolutional *gtsrb* and *mnist* NNs. Interestingly, the largest runtimes reported for SwiftXplain are smaller than or close to the smallest runtimes of the deletion algorithm. This observation does not come as a surprise if we recall that the number of robustness oracle calls is drastically reduced in SwiftXplain when compared to the linear approach, which is always exactly the number of data features. (Note that in case of dichotomic search, as shown in our extended experiments, this number is always larger

[8] Source code and benchmarks are available from: `https://github.com/izzayacine/SwiftXPlain`.

[9] This remark joins the observation of (Wu, Wu, and Barrett, 2023) of the empirical assessment of explanations size without and with sensitivity heuristic traversal order.

than the image size.) For instance in *GTSRB*, SwiftXplain makes on average between 13.8% to 22.7% (q parallel) oracle calls of the total number of calls instrumented by the deletion algorithm (i.e. 77.3% to 86.2% less). Focusing solely on the performance of SwiftXplain, we observe that activation of the feature disjunction (FD) technique is more effective when the feature set size $|\mathcal{W}|$ left to inspect is smaller than the average size of ∂AXp. Also, we observe that deactivating FD in the first iterations of SwiftXplain enables us to drop up to 50% of (chunks of) features with a few iterations in the inner loop. Furthermore, one can see from Table 1 that the average success of FD to capture q transition features in one iteration varies from 33.2% to 77.5% for *GTSRB* and 11.5% to 21.3% for *MNIST*. Also, the larger ∂AXp is, the more likely it is that FD detects chunks of $|\mathcal{T}|$ transition features to augment $\mathcal{S}$ with. Finally, we note that when comparing sensitivity and LIME feature traversal heuristics, the former one improves the effectiveness of FD, and thus helps us reducing the total number of (parallel) calls.

Discussion of performances and potential improvements. To sum up, these observations let us conclude that Algorithm 4 and Algorithm 3 complement each other, such that when the input data or a target ∂AXp is expected to be large (which can be heuristically measured based on feature/pixel importance score) then FD serves to augment $\mathcal{S}$ by means of fewer iterations; conversely for smaller ϵ distance or when a smaller ∂AXp is expected, Algorithm 3 allows us to eliminate more features with fewer iterations. Moreover, we underline that additional results on dichotomic algorithm (see appendix in (Izza et al., 2024a)) confirm the performance gains of SwiftXplain.

Independently from the outperformance of our approach against the SOTA methods demonstrated in this evaluation, we observed some (technical) limitations that we are willing to investigate in the future. First, the prototype does not yet enable multi-core (CPU/GPU) use in the oracle; second, it does not support the incremental mode when calling iteratively the oracle, which would likely improve the runtimes since incremental resolution has been shown beneficial in other settings like SAT oracles; third, we are interested in analyzing the evolution of the runtime curve when increasing the number of processors to hundreds or thousand CPUs/GPUs on data with not much larger in number of features; last but not least, a parallel portfolio of AEx oracles can be implemented such that for each test we instrument concurrently calls to different AEx tools (e.g. trop three solvers of the last VNN-COM Brix et al. (2023)), and terminate the jobs after the first outcome.

8 Conclusions

Formal explainability offers the strongest guarantees of rigor for trustworthy AI, but its limitations include the complexity of reasoning. This is demonstrated for example when explaining medium- to large-scale NNs. Recent work proposed distance-restricted explanations, which can be computed by resorting to practically efficient tools for deciding adversarial robustness (Wu, Wu, and Barrett, 2022; Huang and Marques-Silva, 2023b; Wu, Wu, and Barrett, 2023). The limitations of past work on computing distance-restricted explanations include understanding and proving some of its key properties, but also addressing the performance bottleneck represented by large numbers of features in complex ML models.

This paper provides a detailed formalization of distance-restricted explanations, showing the existence of key connections with past work on formal explainability. Furthermore, the paper proposes novel algorithms for computing distance-restricted explanations for ML models containing a large number of features. Preliminary experimental results confirm that significant performance gains can be achieved by parallelizing the calls to the adversarial robustness oracle. The obtained results also show indirect promise for improving the performance of computing MUSes of large logic formulas. Future work will seek better heuristics for scaling distance-restricted explainability for highly complex ML models, by also enabling the answering of a number of relevant explainability queries (Huang et al., 2021; Audemard et al., 2021; Huang, Izza, and Marques-Silva, 2023).

Acknowledgments

This work was supported in part by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, by the starting funds of ICREA, and by the project PID2022-139835NB-C22 funded by MCIN/AEI/10.13039/501100011033/FEDER, UE. The computational work was performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

Finally, we thank the reviewers for their overall constructive feedback. However, we must underline that one reviewer and the meta-reviewer considered this work to be highly incremental. Although we firmly disagree with such assessment, we also believe that this opinion should be disclosed, for future appraisal.

References

Arenas, M.; Barceló, P.; Romero, M.; and Subercaseaux, B. 2022. On computing probabilistic explanations for decision trees. In *NeurIPS*.

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

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.; and Marquis, P. 2022a. On preferred abductive explanations for decision trees and random forests. In *IJCAI*, 643–650.

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.; and Marquis, P. 2022b. Trading complexity for sparsity in random forest explanations. In *AAAI*, 5461–5469.

Bassan, S., and Katz, G. 2023. Towards formal XAI: formally approximate minimal explanations of neural networks. In *TACAS*, 187–207.

Bassan, S.; Amir, G.; Corsi, D.; Refaeli, I.; and Katz, G. 2023. Formally explaining neural networks within reactive systems. 1–13.

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

Belov, A.; Manthey, N.; and Marques-Silva, J. 2013. Parallel MUS extraction. In *SAT*, 133–149.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability - Second Edition*.

Biradar, G.; Izza, Y.; Lobo, E. A.; Viswanathan, V.; and Zick, Y. 2024. Axiomatic aggregations of abductive explanations. In *AAAI*, 11096–11104.

Brix, C.; Müller, M. N.; Bak, S.; Johnson, T. T.; and Liu, C. 2023. First three years of the international verification of neural networks competition (VNN-COMP). *Int. J. Softw. Tools Technol. Transf.* 25(3):329–339.

Carbonnel, C.; Cooper, M. C.; and Marques-Silva, J. 2023. Tractable explaining of multivariate decision trees. In *KR*, 127–135.

Cardoso, N., and Abreu, R. 2013. A distributed approach to diagnosis candidate generation. In *EPIA*, 175–186.

Chinneck, J. W., and Dravnieks, E. W. 1991. Locating minimal infeasible constraint sets in linear programs. *INFORMS J. Comput.* 3(2):157–168.

Cooper, M. C., and Marques-Silva, J. 2021. On the tractability of explaining decisions of classifiers. In *CP*, 21:1–21:18.

Cooper, M. C., and Marques-Silva, J. 2023. Tractability of explaining classifier decisions. *Artif. Intell.* 316:103841.

Darwiche, A., and Hirth, A. 2023. On the (complete) reasons behind decisions. *J. Log. Lang. Inf.* 32(1):63–88.

Darwiche, A. 2023. Logic for explainable AI. In *LICS*, 1–11.

de Siqueira N., J. L., and Puget, J.-F. 1988. Explanation-based generalisation of failures. In *ECAI*, 339–344.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29(6):141–142.

Ferrari, C.; Müller, M. N.; Jovanovic, N.; and Vechev, M. T. 2022. Complete verification via multi-neuron relaxation guided branch-and-bound. In *ICLR*.

Gorji, N., and Rubin, S. 2022. Sufficient reasons for classifier decisions in the presence of domain constraints. In *AAAI*, 5660–5667.

Gupta, S. D.; Genc, B.; and O'Sullivan, B. 2021. Explanation in constraint satisfaction: A survey. In *IJCAI*, 4400–4407.

Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual.

Hemery, F.; Lecoutre, C.; Sais, L.; and Boussemart, F. 2006. Extracting MUCs from constraint networks. In *ECAI*, 113–117.

Horn, R. A., and Johnson, C. R. 2012. *Matrix Analysis, 2nd Ed*. Cambridge University Press.

Huang, X., and Marques-Silva, J. 2023a. From decision trees to explained decision sets. In *ECAI*, volume 372, 1100–1108.

Huang, X., and Marques-Silva, J. 2023b. The inadequacy of Shapley values for explainability. *CoRR* abs/2302.08160.

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

Huang, X.; Cooper, M. C.; Morgado, A.; Planes, J.; and Marques-Silva, J. 2023. Feature necessity & relevancy in ML classifier explanations. In *TACAS*, 167–186.

Huang, X.; Izza, Y.; and Marques-Silva, J. 2023. Solving explainability queries with quantification: The case of feature relevancy. In *AAAI*, 3996–4006.

Hurault, A., and Marques-Silva, J. 2023. Certified logic-based explainable AI - the case of monotonic classifiers. In *TAP*, volume 14066, 51–67.

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

Ignatiev, A.; Izza, Y.; Stuckey, P. J.; and Marques-Silva, J. 2022. Using MaxSAT for efficient explanations of tree ensembles. In *AAAI*, 3776–3785.

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

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

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

Izza, Y.; Huang, X.; Ignatiev, A.; Narodytska, N.; Cooper, M. C.; and Marques-Silva, J. 2023. On computing probabilistic abductive explanations. *Int. J. Approx. Reason.* 159:108939.

Izza, Y.; Huang, X.; Morgado, A.; Planes, J.; Ignatiev, A.; and Marques-Silva, J. 2024a. Distance-restricted explanations: Theoretical underpinnings & efficient implementation. *CoRR* abs/2405.08297.

Izza, Y.; Ignatiev, A.; Stuckey, P. J.; and Marques-Silva, J. 2024b. Delivering inflated explanations. In *AAAI*, 12744–12753.

Izza, Y.; Meel, K. S.; and Marques-Silva, J. 2023. Locally-minimal probabilistic explanations. *CoRR* abs/2312.11831.

Jannach, D.; Schmitz, T.; and Shchekotykhin, K. M. 2015. Parallelized hitting set computation for model-based diagnosis. In *AAAI*, 1503–1510.

Jannach, D.; Schmitz, T.; and Shchekotykhin, K. M. 2016. Parallel model-based diagnosis on multi-core computers. *J. Artif. Intell. Res.* 55:835–887.

Junker, U. 2004. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In *AAAI*, 167–172.

Katz, G.; Huang, D. A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljić, A.; et al. 2019. The marabou framework for verification and analysis of deep neural networks. In *CAV*, 443–452.

Le, V.; Silva, C. V.; Felfernig, A.; Benavides, D.; Galindo, J. A.; and Tran, T. N. T. 2023. FASTDIAGP: an algorithm for parallelized direct diagnosis. In *AAAI*, 6442–6449.

Letoffe, O.; Huang, X.; Asher, N.; and Marques-Silva, J. 2024. From SHAP scores to feature importance scores. *CoRR* abs/2405.11766.

Letoffe, O.; Huang, X.; and Marques-Silva, J. 2024. On correcting SHAP scores. *CoRR* abs/2405.00076.

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

Liu, X., and Lorini, E. 2023. A unified logical framework for explanations in classifier systems. *J. Log. Comput.* 33(2):485–515.

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

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

Marques-Silva, J., and Ignatiev, A. 2023. No silver bullet: interpretable ML models must be explained. *Frontiers Artif. Intell.* 6.

Marques-Silva, J., and Lynce, I. 2011. On improving MUS extraction algorithms. In *SAT*, 159–173.

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

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

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

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

Marques-Silva, J. 2022. Logic-based explainability in machine learning. In *Reasoning Web*, 24–104.

Marques-Silva, J. 2023. Disproving XAI myths with formal methods - initial results. In *ICECCS*, 12–21.

Marques-Silva, J. 2024. Logic-based explainability: Past, present & future.

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

Molnar, C. 2020. *Interpretable machine learning*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 8024–8035.

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

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

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

Robinson, D. J. 2003. *An introduction to abstract algebra*. Walter de Gruyter.

Shchekotykhin, K. M.; Jannach, D.; and Schmitz, T. 2018. Parallel model-based diagnosis. In Hamadi, Y., and Sais, L., eds., *Handbook of Parallel Constraint Reasoning*. 547–580.

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

Silva, C. V.; Felfernig, A.; Galindo, J. A.; Atas, M.; and Benavides, D. 2020. A parallelized variant of Junker's QuickXPlain algorithm. In *ISMIS*, 457–468.

Stallkamp, J.; Schlipsing, M.; Salmen, J.; and Igel, C. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 32:323–332.

Wu, M.; Wu, H.; and Barrett, C. W. 2022. VeriX: Towards verified explainability of deep neural networks. *CoRR* abs/2212.01051.

Wu, M.; Wu, H.; and Barrett, C. W. 2023. Verix: Towards verified explainability of deep neural networks. In *NeurIPS*.

Yu, J.; Farr, G.; Ignatiev, A.; and Stuckey, P. J. 2023a. Anytime approximate formal feature attribution. *CoRR* abs/2312.06973.

Yu, J.; Ignatiev, A.; Stuckey, P. J.; Narodytska, N.; and 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*, 4123–4131.

Yu, J.; Ignatiev, A.; and Stuckey, P. J. 2023a. From formal boosted tree explanations to interpretable rule sets. In *CP*, volume 280, 38:1–38:21.

Yu, J.; Ignatiev, A.; and Stuckey, P. J. 2023b. On formal feature attribution and its approximation. *CoRR* abs/2307.03380.