REASONX: Declarative Reasoning on Explanations

Neurosymbolic Artificial Intelligence
XX(X):1–60

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Laura State[1,2,3], Salvatore Ruggieri[1] and Franco Turini[1]

Abstract

Explaining opaque Machine Learning (ML) models has become an increasingly important challenge. However, current eXplanation in AI (XAI) methods suffer several shortcomings, including insufficient abstraction, limited user interactivity, and inadequate integration of symbolic knowledge. We propose REASONX, an explanation tool based on expressions (or, queries) in a closed algebra of operators over theories of linear constraints. REASONX provides declarative and interactive explanations for decision trees, which may represent the ML models under analysis or serve as global or local surrogate models for any black-box predictor. Users can express background or common sense knowledge as linear constraints. This allows for reasoning at multiple levels of abstraction, ranging from fully specified examples to under-specified or partially constrained ones. REASONX leverages Mixed-Integer Linear Programming (MILP) to reason over the features of factual and contrastive instances. We present here the architecture of REASONX, which consists of a Python layer, closer to the user, and a Constraint Logic Programming (CLP) layer, which implements a meta-interpreter of the query algebra. The capabilities of REASONX are demonstrated through qualitative examples, and compared to other XAI tools through quantitative experiments.

Keywords

eXplainable AI, Constraint Logic Programming, Decision Trees, Black-box Machine Learning Models

[1] University of Pisa, IT
[2] Scuola Normale Superiore, IT
[3] Alexander von Humboldt Institute for Internet and Society, GER

Corresponding author:
Laura State, Alexander von Humboldt Institute for Internet and Society, Französische Str. 9, 10117 Berlin, GER.
Email: laura.state@hiig.de

Prepared using sagej.cls *[Version: 2017/01/17 v1.20]*

Introduction

The acceptance and trust of Artificial Intelligence (AI) applications is hampered by the opacity and complexity of Machine Learning (ML) models, possibly resulting in biased or even socially discriminatory decision-making (Ntoutsi et al., 2020). As a possible solution, the field of eXplainable Artificial Intelligence (XAI) provides methods to understand how a ML model reaches a decision (Guidotti et al., 2019b; Minh et al., 2022; Mersha et al., 2024). However, most existing approaches focus on descriptive explanations and rarely support declarative reasoning over the decision rationale. By reasoning, we mean the possibility for the user to define any number of conditions over explanations (both factual and contrastive ones), which would codify both background knowledge and what-if analyses (Beckh et al., 2023), and then query for explanations at the symbolic and intensional level. Answers could be expressed in the same language as the user queries, thus making the query language closed, i.e., (part of) the answer to a query can be used in following queries. This would support the interactivity of the explanation process – a requirement of XAI (Miller, 2019).

In this paper, we define a closed algebra of operators on sets of linear constraints. Expressions over such an algebra represent user queries that allow for computing factual and contrastive explanations. On top of the algebra, we design REASONX (*reason to explain*), an explanation tool that consists of two layers. The first layer is in Python, closer to the data analyst user, where decision trees (DTs) and user queries are parsed and translated. The second layer is in Constraint Logic Programming (CLP) over the reals (Jaffar and Maher, 1994), where embedding of decision trees, and user constraints coding background knowledge are reasoned about. DTs can be themselves the ML models to reason about, or they can be surrogate models of other black-box ML models at the global level or at the local level, i.e., in the neighborhood of an instance to explain. The path from the root to a leaf naturally encodes a linear constraints explaining the decision logic of the DT (or of the black-box model it approximates). Thus, linear constraints over the reals appear a natural knowledge representation mechanism for symbolic reasoning. The expressions of the algebra used at the Python layer are evaluated through a CLP meta-interpreter – a powerful reasoning capability of logic programming (Brogi et al., 1993). The evaluation of expressions produces linear constraints, which are passed back to the Python layer. Since we rely on an intensional symbolic representation of conditions, REASONX offers the unique property of reasoning on under-specified instances, where features may be left unspecified or partly bounded. Moreover, the algebra allows for defining any number of instances and models, hence allowing for reasoning over explanations for the same instance and different models, e.g., models at different points of time.

Consider a simple example illustrating the need for a high-level abstraction mechanism. Figure 1 shows a simple DT describing the state of a cup of water ("liquid" or "not liquid"), based on room temperature T_1 and an additional temperature T_2 contributed by a heater (assuming temperatures are measured in Celsius). Traditional XAI tools typically explain predictions only for fully specified instances. E.g., for $T_1 = -10, T_2 = 20$, the prediction of the decision tree is "liquid", and the corresponding

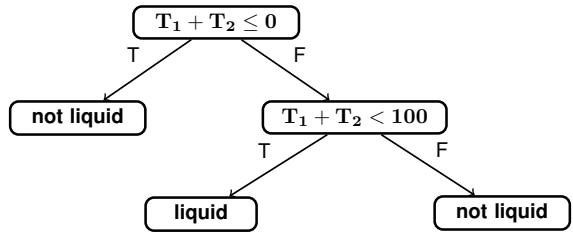

Figure 1. A simple decision tree illustrating the state of a cup of water in a room at temperature T_1, which is warmed by a heater contributing an additional temperature T_2.

factual explanation is the rule $0 < T_1 + T_2 < 100 \rightarrow$ "liquid". At a more abstract (intensional) level, we would be interested in explaining a partially specified instance such as $T_1 = -10$. In such a case, the prediction "liquid" holds for values of T_2 such that $10 < T_2 < 110$, based on the same factual rule, and the prediction "not liquid" holds for $T_2 \leq 10$ or $T_2 \geq 100$. REASONX enables reasoning at precisely this intensional level.

The contributions of this paper are the following:

- We introduce an algebra of operators over sets of linear constraints. The algebra is expressive enough to model complex explanation problems over DTs as expressions (or *queries*[*]). The algebra is closed, namely the operators can be freely composed.
- We develop the REASONX tool, consisting of two layers. The CLP layer implements an interpreter of the above algebra through meta-programming. The Python layer offers a user-friendly interface specifically designed for ML developers, who can easily integrate models, instances, background knowledge to (silently) generate queries over the algebra and decode back the answers.
- We demonstrate the capabilities of REASONX through qualitative demonstrations, and a quantitative comparison against other XAI tools.

The code of REASONX together with the data used in the demonstrations is publicly available at:

https://github.com/lstate/reasonx

This paper is a substantial extension of two previous works: an introductory paper (State et al., 2023b), targeting an interdisciplinary audience, and a technical paper (State et al., 2023a), providing the preliminary architecture of REASONX. This extension covers the formal definition of the query algebra and its operators, and the definition of explanation problems as queries; an in-depth presentation of the architecture

[*]This naming is inspired by the relational algebra used in database theory (Garcia-Molina et al., 2008).

of REASONX, and the demonstrators of its capabilities; as well as the comparison with other XAI tools.

The paper is structured as follows: we discuss the background on XAI topics and related work in Section XAI and Related Work. To keep the paper self-contained, a brief introduction on CLP and meta-reasoning is provided in the Supplemental material. In Section A Query Language over Linear Constraints for XAI, we introduce the algebra of operators, and its capabilities to model explanation problems. The implementation details on REASONX are presented in Section REASONX: Reason to explain. We demonstrate REASONX via some examples in Section Demonstrations, present its evaluation in Section Quantitative Evaluation, and close with limitations and future work in Section Limitations and Future Work. Finally, we close with Section Concluding Remarks. The Supplemental material reports further (experimental) details.

XAI and Related Work

XAI aims at coupling effective ML models with explanations of how they work (Guidotti et al., 2019b; Minh et al., 2022; Molnar, 2019; Mersha et al., 2024). This can be achieved through novel *explainable-by-design* ML models, or through *post-hoc* explanations of non-interpretable ML models, commonly referred to as *black-box* models. Post-hoc explanation methods can be categorized with respect to two aspects (Guidotti et al., 2019b). One contrasts model-specific *vs.* model-agnostic approaches, depending on whether the explanation method exploits knowledge about the internals of the black-box or not. The other aspect contrasts local *vs.* global approaches, depending on whether an explanation refers to a specific instance or to the black-box as a whole.

In this work, we focus on post-hoc explanations of classification models. Our approach is model-agnostic, as we will reason about a surrogate model of the black-box in the form of a decision tree. The surrogate model can approximate the black-box either globally or locally to a specific instance to explain. In the special case that the black-box is a decision tree itself, i.e., it is too large to be directly understood, our approach is model-specific.

Table 1 compares REASONX with related explainability methods. It shows that our tool uniquely combines reasoning over constraints, rules and contrastive examples, over several instances and models, as well as with under-specified instances. We review related work in this section.

Neuro-symbolic XAI

Calegari et al. (2020) survey how symbolic methods based on some formal logics can be integrated with sub-symbolic models for (neuro-symbolic) explanation purposes. Sabbatini (2025) provides an overview of symbolic knowledge extraction methods from black-boxes. Existing approaches either aim at blending symbolic and sub-symbolic methods (*integration*), or treat them as distinct parts that remain separated but work together (*combination*). The central building blocks of these approaches are decision rules, decision trees, and knowledge graphs. Decision rules have been shown to be effective for both developers (Piorkowski et al., 2023) and domain users (Ming et al., 2019; van der Waa et al., 2021).

Method (*)	Type		Data			Output			CON	DIV	USI	MM	INT
			tab.	text	im.	FR	CR	CE					
REASONX	MA/MS	g/l	✓	✗	✗	✓	✓	✓	linear	✓	✓	✓	✓
LORE	MA	l	✓	✗	✗	✓	✓	✗	✗	n.a.	n.a.	✗	✗
GLocalX	MA	g	✓	✗	✗	rule sets		✗	✗	n.a.	n.a.	✗	✗
ANCHORS	MA	l	✓	✓	✓	✓	✗	✗	n.a.	n.a.	n.a.	✗	✗
Wachter	MA	l	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗
DiCE	MA	l	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
DACE	MS	l	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗
Cui	MS	l	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Russell	MS	l	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
Ustun	MS	l	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗
MACE	MA	l	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
Karimi	MS	l	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗
Bertossi	MA	l	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗
Takemura	MS	g	✓	✗	✗	rule sets		✗	✓	n.a.	n.a.	✗	✗
LIMEtree	MA	l	✓	✓	✓	✓	✓	✗	✓	n.a.	n.a.	✗	✓

Table 1. Comparing REASONX against related explainability methods. Explanation type according to the introduced taxonomy (MA = model-agnostic, MS = model-specific, g = global, l = local). The output can be factual decision rules (FR), contrastive decision rules (CR) or contrastive examples (CE). CON = possibility to enforce constraints on CR/CE, DIV = possibility of optimizing for diversity on a set of CE. USI = under-specified information for a CE. n.a. = not applicable, tab. = tabular, im. = image. MM = explanations over time and multiple models, INT = interactive explanations.
(*) For space reasons, corresponding papers are listed next: LORE (Guidotti et al., 2019a), GlocalX (Setzu et al., 2021), ANCHORS (Ribeiro et al., 2018), Wachter (Wachter et al., 2017), DiCE (Mothilal et al., 2020), DACE (Kanamori et al., 2020), Cui (Cui et al., 2015), Russell (Russell, 2019), Ustun (Ustun et al., 2019), MACE (Karimi et al., 2020), Karimi (Karimi et al., 2021), Bertossi (Bertossi, 2023), Takemura (Takemura and Inoue, 2024), LIMEtree (Sokol and Flach, 2020a).

REASONX relies on a novel usage of constraint logic programming – a form of rule-based symbolic logic reasoning – serving as an explanation tool that adopts the *combination* approach. Our post-hoc explanation method can be categorized as a "symbolic [neuro]" approach (type 2 in Bhuyan et al. (2024)), where the black-box model is the neuro subroutine called by the symbolic explanation method locally to an instance to explain or globally. The approach is complementary to other neuro-symbolic ones, e.g., "neuro[symbolic]" (type 6) that embeds a symbolic thinking engine directly into a neural engine. Such forms of integration aim at making AI models explainable-by-design.

Other explanation methods exist that borrow concepts from logics but do not allow users to reason over those concepts. The closest ones to our approach are those using (propositional) logic rules as forms of model-agnostic explanations both locally (Ming et al., 2019; Guidotti et al., 2019a; Ribeiro et al., 2018) and globally (Setzu et al., 2021). Another prominent example is TREPAN (Craven and Shavlik, 1995), which computes a decision tree with m-of-n split binary conditions. Further related work in this category is presented by a series of papers by Sokol et al. (Sokol, 2021; Sokol and Flach, 2020a, 2018), based on local surrogate regression trees. Furthermore, Answer Set Programming (ASP), another powerful logic programming extension, has been adopted for local and

global explanations of tree ensembles (Takemura and Inoue, 2024), and for contrastive explanations from the angle of actual causality (Bertossi, 2023).

Model-specific logic approaches to XAI offer formal guarantees of rigor, giving rise to the sub-field of formal XAI (Marques-Silva, 2022). In the Supplemental material, we compare sufficient reasons from the formal XAI literature to our notion of factual explanations. Nevertheless, being model-agnostic, our approach does not belong to formal XAI. Recently, epistemic logics have been used to model the justification (i.e., a "proof") for a belief embedded in an AI model, e.g., for the statements made by Large Language Models (LLMs) (Šekrst, 2025). The justification logic (Artemov and Fitting, 2019), for instance, has been used to provide personalized explanations (Luo et al., 2023) in a multi-agent conversation framework.

Decision Trees and Factual Explanations

Decision trees (DTs) (Rokach and Maimon, 2005; Costa and Pedreira, 2023) are classification models that predict or describe a nominal feature (*the class*) on the basis of predictive features, which can be nominal, ordinal or continuous. A decision tree is a tree data structure comprised of internal nodes and leaves. Each internal node points to a number of child nodes, the degree of the node, based on the possible outcomes of a split condition at the node. The split conditions are defined over the predictive features. Here, we will consider binary split conditions in the form of linear inequality constraints. Leaf nodes terminate the tree and are associated with an estimated probability distribution over class labels. The class label with the highest estimated probability is predicted by the DT (*Bayes rule*) for instances satisfying all split conditions from the root node to the leaf. The highest estimated probability is called the *confidence* of the prediction.

DTs are inherently transparent and intelligible (Rudin, 2019), unless their size is large. A path from the root note to a leaf corresponds to a conjunction of split conditions that describe the decision rule for class prediction at the leaf. The decision rule of the path followed by an input instance is an explanation for the prediction of the DT over such an instance. This rule-based definition of *(factual) explanations* is shared with most rule-based explainers (Ming et al., 2019; Guidotti et al., 2019a; Ribeiro et al., 2018; Setzu et al., 2021; Craven and Shavlik, 1995). *Sufficient reasons* (Izza et al., 2022) (or subset-minimal abductive explanations) from the sub-field of formal XAI consist of a minimal subset of feature values of the instance to explain, such that the prediction of the DT is the same as for the instance regardless of the values of the remaining features. See Amgoud et al. (2023) for a discussion of the relation to the rule-based definition.

A trending research topic in XAI consists of the design of DT learning algorithms that can reproduce the behavior of complex and opaque models, e.g., of a tree ensemble (Weinberg and Last, 2019; Vidal and Schiffer, 2020; Bonsignori et al., 2021; Dudyrev and Kuznetsov, 2021; Ferreira et al., 2022). Such algorithms can be used for training surrogate models, either globally or locally to an instance to explain. Alternative approaches rely on the synthetic generation of a plausible dataset (such as in the neighborhood of the instance to explain) for training surrogate DTs (Guidotti et al., 2019a, 2024; Barbosa et al., 2024). Our research is orthogonal to these topics. We assume

a given DT to reason about, independently of whether it is the black-box itself or a global or local surrogate model of a black-box. Nevertheless, high *fidelity* of a surrogate DT w.r.t. the black-box is a key property that we need to assume in order to support the reasoning over DTs as an approximation of the reasoning over the black-box.

In our approach, we reason over a DT by encoding it as a set of linear constraints. This problem, known as *embedding* (Bonfietti et al., 2015), requires to satisfy $c(x, y) \Leftrightarrow f(x) = y$, where $f(x)$ is the predicted class by the DT, x is the vector of predictive features, and c is a logic expression. In our approach, c is a disjunction of linear constraints, one for each path from the root to a leaf. This rule-based encoding takes space in $O(n \log n)$ where n is the number of nodes in the DT. Other DT encodings, such as Table and MDD from Bonfietti et al. (2015), require a discretization of continuous features, thus losing the expressive power of reasoning on linear constraints.

Contrastive Explanations

Factual explanations, or simply explanations, provide answers to "Why did the input data lead to the provided prediction?". Contrastive explanations (CEs)[†] refer to data instances similar to the one being explained but with a different predicted outcome. They answer "What should the input data look like, in order to obtain a different output?" which can be easily translated to "How should the current input change to obtain a different output?", or to answer *what-if* questions, such as "*What* happens to the output *if* the input changes that way?".

In recent years, contrastive explanations have received considerable attention, as demonstrated by an increasing number of surveys, e.g., (Karimi et al., 2023; Guidotti, 2024; Keane et al., 2021; Stepin et al., 2021). CEs can be represented not only as data instances but also, e.g., as decision rules (Guidotti et al., 2019a), images (Chang et al., 2019) or texts (Wu et al., 2021), and are considered *local*, *model-agnostic* explanations. Contrastive explanations have been introduced to XAI by Wachter et al. (2017). The authors propose to generate them through the following optimization problem:

$$\arg\min_{x_{ce}} \max_{\lambda} \lambda(f(x_{ce}) - y_{ce})^2 + d(x_f, x_{ce}) \tag{1}$$

where $f()$ is the ML model, x_{ce} refers to the contrastive instance, x_f the original instance, y_{ce} to the prediction to change to, λ denotes a tuning parameter, and $d(.,.)$ a distance measure. Intuitively, a CE is the more "realistic" the closer it is to the original instance, understood as the change of as few feature values as possible while $f(x_{ce})$ must align with y_{ce}. The optimization function (Equation 1) can be augmented by constraints, e.g., to make the changes actionable (Karimi et al., 2023) or for computing a diverse set of CEs (Mothilal et al., 2020; Laugel et al., 2023). Such an extension can be understood as

[†]Contrastive explanations are closely related to the concept of counterfactuals as understood in the statistical causality literature. However, it is not the same, and to avoid confusion, we use – in contrast to other literature in XAI – the term contrastive explanation (later also "contrastive example/rule/instance") instead of counterfactuals. For further discussion on this, see e.g., Miller (2019).

adding *background knowledge* to the explanation. Solutions to the optimization problem can be found by reducing it to satisfiability (SAT) or mixed integer linear programming (MILP) problems (Ustun et al., 2019; Russell, 2019; Kanamori et al., 2020). Contrastive explanations (or CXp's) in the sense of Izza et al. (2022); Audemard et al. (2023) consist of a minimal subset of features of the instance to explain, such that the prediction of the DT is changed for some value of such features, all the other feature values of the instance being unchanged.

REASONX is the first approach adopting CLP for generating CEs. We consider a rule-based definition of CEs, guided by a DT and user-defined constraints linking the instance to explain to its CEs.

Group Explanations and Under-specified Instances

Local explanations focus on predictions for single instances. Global explanations focus on the decision logic of a black-box for the whole population. *Group explanations* research considers sub-populations of similar instances with the same predictions, sometimes called cohorts or profiles, that share similar (contrastive) explanations. Existing approaches consist of clustering factual explanations (Setzu et al., 2021) and CEs (Warren et al., 2024). Other works extend the optimization problem of Equation 1 (Carrizosa et al., 2024); focus on explanations that refer to an ensemble of decisions (Artelt et al., 2022); or connect group contrastive explanations to actionable recourse (Rawal and Lakkaraju, 2020).

The literature for local and group explanations deals with an *extensional* format of instances, i.e., instances are points in a dimensional space, and sub-populations are simply sets of instances. Instead, the symbolic approach of REASONX deals with instances in an *intensional* format as linear constraints (Cook, 2009). Fully specified instances are defined by setting equality constraints for all features, e.g., `age = 20`. In addition, *under-specified instances* can be defined, e.g., bounding the feature values, e.g., `age >= 20`, or even setting no constraint on that feature. An under-specified instance allows for computing (contrastive) explanations of a specific sub-population. This feature distinguishes our approach.

Background Knowledge

Exploiting background knowledge in computing an explanation has the potential to significantly improve its quality (Beckh et al., 2023; State, 2021). Background (or prior) knowledge is any information that is relevant in the decision context, but that does not (explicitly) emerge through the data. Not only do simple facts count as such, but we would also consider a natural law as such knowledge, or a specific restriction or requirement a (lay) user has, related to her living reality (e.g., a minimum credit amount in a credit application scenario). Background knowledge integration also draws a connection between our work and the field of *actionable recourse* (Karimi et al., 2023): this research field proposes explanations not only to explain the model outcome to the affected individual, but also to recommend specific actions so that an

undesirable outcome can be changed. Integrated knowledge then helps to better tailor the recommendation to the needs of the user.

REASONX incorporates knowledge in the form of linear constraints over the reals. Some examples of knowledge integration from the literature include: a local explanation tool for medical data that incorporates an ontology to generate a meaningful local neighborhood for explanation generation (Panigutti et al., 2020); a discrimination discovery approach (Ruggieri et al., 2010), where knowledge in the form of association rules is used to detect cases of indirect discrimination; and, in the sub-field of formal XAI, computational complexity investigations of (subset-minimal) abductive and contrastive explanations under domain theories (Audemard et al., 2024) and of constrained classifiers (Cooper and Marques-Silva, 2023).

Interactivity

Although being acknowledged as a key property of explanation tools (Miller, 2019; Weld and Bansal, 2019; Lakkaraju et al., 2022), few XAI methods incorporate interactivity. Sokol and Flach (2020b) outline the usefulness of interactivity prominently in one of their articles: "Truly interactive explanations allow the user to tweak, tune and personalise them (i.e., their content) via an interaction, hence the explainee is given an opportunity to guide them in a direction that helps to answer selected questions". We also point to Lakkaraju et al. (2022), presenting an interview study with practitioners and revealing that interactivity for explanations is preferred over static explanations. A notable tool that provides interactive explanations is the glass-box tool (Sokol and Flach, 2018), which can be queried by voice or chat and provides explanations in natural language. Recent advances of LLMs are promising to transform explanations into natural, human-readable narratives (Zytek et al., 2024; Mavrepis et al., 2024).

REASONX natively offers an interactive interface, where explanations can be refined by asserting and retracting constraints, by adding and removing instances and CEs, by projecting to features of interest, and by optimizing tailored distance functions. Since the output of a query to REASONX consists of linear constraints, part of an answer can be readily embedded in subsequent queries by the user. This enables a two-way flow of information, commonly referred to as *interaction*, between the system and the user.

Reasoning over Time and Multiple Models

In operational deployment of ML models, it is common that the model changes over time. A change can be induced, for example, by new incoming data that lead to a retraining. Explanation methods that contrast different models are rare. Malandri et al. (2024) present (global) model-contrastive factual explanations of symbolic models, such as decision rules or trees, by computing conditions satisfied by one model and not satisfied by another one. A specific issue is that of contrastive examples that are invalidated by model changes (Barocas et al., 2020). An invalidation can be critical if the CE is intended for the user to seek recourse, i.e., to change the unfavorable model decision by actions proposed by the CE. Ferrario and Loi (2022) call such cases "unfortunate counterfactual events", and suggest a data augmentation strategy that relies on CE generated previously.

A related case is to compare explanations of the same instance classified by different ML models *at the same time*. For example, model developers are interested in reasoning on explanations of the same instance with respect to two different but similarly performing models. The instance may have received the same predictions and the same explanations, but different CEs. Or it may have received the same predictions and the same CEs, but different explanations. Pawelczyk et al. (2020) discuss the generation of contrastive explanations for recommendations under predictive multiplicity. This phenomenon refers to different models that give very similar results in performance for a specific prediction problem.

REASONX allows to declare instances and CEs over different models, and to relate them via constraints. This is a natural by-product of the query language REASONX is built on, which is able to cover the case of the same instance and two (or more) different models, either referring to different training times or to different black-box models.

A Query Language over Linear Constraints for XAI

We define an algebra of operators over sets of linear constraints. Expressions over the operators define a language of queries. We claim that such a language is expressive enough to model several explanation problems over decision trees as queries.

Linear Constraints and Decision Trees

A primitive linear constraint is an expression $a_1 \cdot x_1 + \ldots a_n \cdot x_n \simeq b$, where $\simeq$ is in $\{<, \leq, =, \geq, >\}$, $a_1, \ldots, a_n$ are constants in $\mathbb{R}$ and $x_1, \ldots, x_n$ are variables. We will use the inner product form by rewriting it as $\mathbf{a}^T\mathbf{x} \simeq b$, where $\mathbf{x} = x_1, \ldots, x_n$ is the vector of variables and $\mathbf{a} = a_1, \ldots, a_n$ the vector of constants. A linear constraint, denoted by c, is a sequence of primitive linear constraints, whose interpretation is their conjunction. A path from the root node to a leaf in a decision tree with linear split conditions trivially maps to a linear constraint.

We write $\models \psi$, to denote the validity of the first order formula ψ in the domain of the reals. A constraint c is satisfiable if $\models \exists\, c$, namely if there exists an assignment of all of its variables to real values that evaluates to true in the domain of reals – also called a solution of c. The solutions of a linear constraint c are a polyhedron in the space of the variables $\mathbf{x}$. The projection of c over a subset $\mathbf{w}$ of variables is the linear constraint equivalent to existentially quantifying over all variables except $\mathbf{w}$, or in formula $\exists_{-\mathbf{w}}\, c$. The linear projection can be computed by the Fourier-Motzkin projection method (Schrijver, 1987).

We assume any number of decision tree identifiers $DT_1, DT_2, \ldots$, each decision tree is defined over the features $\mathbf{x} = x_1, \ldots, x_n$. Also, we assume any number of data instance identifiers (or, simply, *instances*) $I_1, I_2, \ldots$. For each instance I_i, and feature x_j, the variable identifier $I_i.x_j$ is given and called an *instance feature*. In other words, instance features are variables whose identifiers are prefixed by the instance identifier and suffixed by the feature names. We consider sets of linear constraints over variables including at least the instance features set $\mathcal{F} = \{I_i.x_j \mid i \geq 1, j \in [1, n]\}$.

We call a set of linear constraints a *theory*. Each constraint in a theory is interpreted as a conjunction of linear inequalities over instance features. The theory as a whole is interpreted as the disjunction of the constraints in the theory.

An Algebra of Operators

We reason over theories by a closed algebra of operators, i.e., each operator takes theories as input and returns a theory. The syntax of expressions E in the algebra is shown next:

$$E \quad ::= \quad inst(I_i, DT_k, l, pr) \mid \{U\} \mid cross(E+) \mid sat(E) \mid \pi_{\mathbf{w}}(E) \mid inf(E, f(\mathbf{w})) \mid relax(E)$$

where l is a constant that denotes a class label, pr is a constant that denotes a probability in $[0, 1]$, $\mathbf{w} \subseteq \mathcal{F}$ is a set of instance features, and $f()$ a linear function.

The semantics $Th(E)$ of an expression E is a theory, defined as follows:

- $Th(inst(I_i, DT_k, l, pr))$ is the set $\{\wedge_{\mathbf{a}^T\mathbf{x} \simeq b \in P}\ \mathbf{a}^T I_i.\mathbf{x} \simeq b \mid P \in DT_k(l, pr)\}$ where $DT_k(l, pr)$ are the paths in DT_k leading to a leaf with class label l predicted with confidence at least pr;
- $Th(\{U\})$ is a singleton linear constraint $\{\wedge_{c \in U} c\}$ where U is a sequence of (user-provided) linear constraints;
- $Th(cross(E_1, \ldots, E_k)) = \{c_1 \wedge \ldots \wedge c_k \mid c_1 \in Th(E_1), \ldots, c_k \in Th(E_k)\}$ is the cross-product of constraints in the theories $Th(E_1)$, ..., $Th(E_k)$;
- $Th(sat(E)) = \{c \in Th(E) \mid \models \exists\ c\}$ is the set of satisfiable constraints c in $Th(E)$;
- $Th(\pi_{\mathbf{w}}(E)) = \{\exists_{-\mathbf{w}}\ c \mid c \in Th(E)\}$ is the set of projected constraints c in $Th(E)$;
- $Th(relax(E)) = \{c^{=} \mid c \in Th(E)\}$ where $c^{=}$ is obtained by replacing strict inequalities in c with inequalities: $<$ is replaced by $\leq$, and $>$ is replaced by $\geq$;
- $Th(inf(E, f(\mathbf{w}))) = \{c \wedge (f(\mathbf{w}) = (inf f(\mathbf{w}) \text{ s.t. } c)) \mid c \in Th(E)\}$ is the set of constraints in $Th(E)$ extended with the minimization of $f()$.

Closedness of the algebra is immediate for all operators, with the exception of inf. In such a case, if the problem $\inf f(\mathbf{w})$ s.t. c is unbounded or infeasible, then a linear constraint equivalent to $f(\mathbf{w}) = (\inf f(\mathbf{w}) \text{ s.t. } c)$ is any unsatisfiable one, e.g., $0 = 1$. If the problem has a solution v, then $f(\mathbf{w}) = v$ is a linear inequality since $f()$ is linear.

Explanations as Queries

We claim that the algebra above is expressive enough to model several explanation problems. We substantiate such a claim with expressions for factual, contrastive, and minimal contrastive explanations. See Table 2 for a summary of example expressions. For ease of presentation, we assume here that the features are continuous. In the next section, we will discuss how REASONX deals with discrete features.

Factual	
Base expression	$e_1 = sat(cross(inst(I_1, DT_1, 0, 0.95), \{U_1\}))$
Factual constraints	$c \in Th(inst(I_1, DT_1, 0, 0.95))$
Factual rule	$c \rightarrow l = 0\ [p]$
Constrastive	
Base expression	$e_2 = sat(cross(inst(I_1, DT_1, 0, 0.95), inst(I_2, DT_1, 1, 0.95), \{U_2\}))$
Constrastive constraint	$c \in Th(e_3)$, where $e_3 = \pi_{I_2.\mathbf{x}}(e_2)$
Contrastive example	any (ground) solution of c
Contrastive rule	$c \rightarrow l = 1\ [p]$
Minimal contrastive	
Base expression	$e_4 = inf(e_2, f(I_1.\mathbf{x}, I_2.\mathbf{x}))$
Minimal contrastive constraint	$c \in Th(e_5)$, where $e_5 = \pi_{I_2.\mathbf{x}}(e_4)$
Minimal contrastive example	any (ground) solution of c
Minimal contrastive rule	$c \rightarrow l = 1\ [p]$

Table 2. Example expressions for factual, contrastive and minimal contrastive explanations.

Factual explanations. Consider the following expression:

$$e_1 = sat(cross(inst(I_1, DT_1, 0, 0.95), \{U_1\})) \tag{2}$$

where U_1 is $I_1.x_1 = v_1, \ldots, I_1.x_n = v_n$, i.e., it fixes the values of the features of the instance I_1 to constants $v_1, \ldots, v_n$. Then $c \in Th(e_1)$ iff c consists of the constraints appearing in a path of DT_1 predicting the class label 0 with confidence $p \geq 0.95$, and such that the constraints in the path are consistent with the values v_i's for the features x_i's. In other words, e_1 tests for a factual explanation for the instance $I_1.\mathbf{x} = \mathbf{v}$, where $\mathbf{v} = v_1, \ldots, v_n$. If $Th(e_1)$ is not empty, such a factual explanation exists. In such a case, the sub-expression $inst(I_1, DT_1, 0, 0.95)$ states why DT_1 classifies I_1 with label $l = 0$ and confidence $p \geq 0.95$. We call a constraint $c \in Th(inst(I_1, DT_1, 0, 0.95))$ that contributes to $Th(e_1)$ a *factual constraint*, and the rule $c \rightarrow l = 0\ [p]$ a *factual (explanation) rule*.

Example. Consider a decision tree DT_1 with a single split condition, $x_1 + x_2 < 5$ that perfectly separates two classes. Instances that match this condition belong to class $l = 0$ with confidence 1.0, while instances that do not match this condition belong to class $l = 1$ with confidence 1.0. We set U_1 to $I_1.x_1 = 2, I_1.x_2 = 2$, and we obtain from the tree $Th(inst(I_1, DT_1, 0, 0.95)) = \{I_1.x_1 + I_1.x_2 < 5\}$. The expression e_1 evaluates to:

$$Th(e_1) = \{I_1.x_1 + I_1.x_2 < 5 \wedge I_1.x_1 = 2 \wedge I_1.x_2 = 2\} = \{I_1.x_1 = 2 \wedge I_1.x_2 = 2\}$$

which is non-empty. The factual rule $I_1.x_1 + I_1.x_2 < 5 \rightarrow l = 0\ [1.0]$ then describes why DT_1 classifies the instance I_1 with label 0 and confidence 1.0.

Contrastive explanations. Consider the following expression:

$$e_2 = sat(cross(inst(I_1, DT_1, 0, 0.95), inst(I_2, DT_1, 1, 0.95), \{U_2\}))$$

where U_2 extends U_1 by further constraints relating the features of I_1 and I_2. Here, it is assumed that the instance I_2 is classified by DT_1 with class label 1, hence it is a

contrastive instance w.r.t. I_1. Then $Th(e_2)$ is the set of satisfiable constraints specifying: for I_1 a path in DT_1 leading to class label 0 (factual constraints for I_1); for I_2 a path in DT_1 leading to class label 1 (factual constraints for I_2); the satisfiability of the constraints between I_1 and I_2 stated in U_2. By projecting over $I_2.\mathbf{x}$, that is:

$$e_3 = \pi_{I_2.\mathbf{x}}(e_2)$$

we obtain constraints over the variables of the contrastive instance I_2. We call a constraint $c \in Th(e_3)$ a *contrastive constraint*, and any solution of c a *contrastive example*. Intuitively, contrastive examples are the extensional counterpart of the intensional contrastive constraints. Moreover, we call the rule $c \rightarrow l = 1\ [p]$, where p is the confidence[‡] for I_2, a *contrastive (explanation) rule*. Notice that this refers to I_2 contrasted to I_1. One can still apply the definition of factual explanation for I_2, resulting in the factual constraints in $Th(inst(I_2, DT_1, 1, 0.95))$. Notice that more than one instance per class label can be defined. In general, an instance is contrastive w.r.t. any other instance with a different class label. This extends the 1-to-1 settings of one factual and one contrastive rule/example to a m-of-n setting of m factual and n contrastive rules/examples. E.g., one can look for a contrastive instance common to two given instances. Such a m-of-n setting in XAI is novel.

Example. We revisit the previous example and keep the same decision tree DT_1 and the same instance I_1. Additionally, we define the instance I_2 such that $Th(inst(I_2, DT_1, 1, 0.95)) = \{I_2.x_1 + I_2.x_2 \geq 5\}$. Moreover, U_2 is defined as U_1 plus $I_2.x_1 = I_1.x_1$, namely we add immutability to feature x_1. Then, the previously introduced expressions evaluate to the following:

$$\begin{aligned} Th(e_2) &= \{I_2.x_1 + I_2.x_2 \geq 5 \wedge I_1.x_1 = 2 \wedge I_1.x_2 = 2 \wedge I_2.x_1 = I_1.x_1\} \\ Th(e_3) &= \{I_2.x_1 = 2 \wedge I_2.x_2 \geq 3\} \end{aligned}$$

The contrastive rule $I_2.x_1 = 2 \wedge I_2.x_2 \geq 3 \rightarrow l = 1\ [1.0]$ describes then a condition under which DT_1 would classify the instance I_2 with label 1 (and confidence 1.0) and for which the immutability constraint holds. Notice that $I_2.x_1 + I_2.x_2 \geq 5 \rightarrow l = 1\ [1.0]$ is a factual rule[§] for the instance I_2.

Minimal contrastive explanations. Consider the following expression:

$$e_4 = inf(e_2, f(I_1.\mathbf{x}, I_2.\mathbf{x}))$$

[‡]Actually p is the confidence of the prediction $l = 1$ for I_2 at a path of DT_1. We are thus extending such a confidence under the assumption that the additional constraints of e_2, e.g., user constraints relating I_1 and I_2, do not alter the overall class distribution. This is the best one can do without further information. For instance, assuming the availability of a representative dataset (e.g., the training set), we could estimate the actual confidence of the decision tree under the additional constraints.

[§]Recall Footnote ‡. The confidence of the factual rule is 1.0. For any additional constraint, such as those in the contrastive rule, the confidence will remain 1.0. This is a special case. For confidence values strictly lower than 1.0, the confidence after adding further constraints may differ, possibly considerably. E.g., because the added constraints precisely restrict to the solutions where the prediction is wrong.

where $f()$ is a linear (or linearizable) distance function between the instances I_1 and I_2. Then $Th(e_4)$ includes the constraints that, in addition to $Th(e_2)$, minimize the distance between I_1 and the contrastive instance I_2. Given:

$$e_5 = \pi_{I_2.\mathbf{x}}(e_4)$$

we call a constraint $c \in Th(e_5)$ a *minimal contrastive constraint*, and the rule $c \rightarrow l = 1\ [p]$, where p is the confidence for I_2, a *minimal contrastive (explanation) rule*.

Example We refer to the previous example, and keep the same decision tree DT_1, instances I_1 and I_2 and constraints U_2. Further, we define $f(I_1.\mathbf{x}, I_2.\mathbf{x}) = \sum_i |I_2.x_i - I_1.x_i|$, i.e., we use the L_1 norm as the distance function. The minimal value of $f(I_1.\mathbf{x}, I_2.\mathbf{x})$ occurs when $f(I_1.\mathbf{x}, I_2.\mathbf{x}) = |I_2.x_2 - 2| + |2 - 2| = I_2.x_2 - 2 = 1$. The expressions e_4 and e_5 evaluate to:

$$\begin{aligned} Th(e_4) &= \{I_1.x_1 = 2 \wedge I_1.x_2 = 2 \wedge I_2.x_1 = 2 \wedge I_2.x_2 = 3\} \\ Th(e_5) &= \{I_2.x_1 = 2 \wedge I_2.x_2 = 3\} \end{aligned}$$

Hence, $I_2.x_1 = 2 \wedge I_2.x_2 = 3 \rightarrow l = 1\ [1.0]$ is a minimal contrastive explanation.

Explanations for underspecified instances. Reconsider the previous examples under the revised assumption that U_1 consists of $I_1.x_1 = 2$ only, i.e., we want to reason not on a fully specified instance, but on a set of instances (those for which $x_1 = 2$). It turns out that:

$$Th(e_1) = \{I_1.x_1 = 2 \wedge I_1.x_2 < 3\}$$

and $Th(e_3)$ is unchanged. Hence, the contrastive rule holds also in such a case. This means that it is contrastive for all the instances satisfying the constraints stated for I_1. Regarding the minimal contrastive rule, now $|I_2.x_2 - I_1.x_2| = I_2.x_2 - I_1.x_2 > 0$ since $I_1.x_2 < 3$ (see $Th(e_1)$) and $I_2.x_2 \geq 3$ (see $Th(e_3)$). The inf operator then imposes $I_2.x_2 = I_1.x_2$. This leads to an unsatisfiable theory, as it includes both $I_2.x_2 = I_1.x_2$, $I_1.x_2 < 3$, and $I_2.x_2 \geq 3$. The issue is that the minimum of $I_2.x_2 - I_1.x_2$, under the constraint that it is greater than 0, does not exist. In practice, inf implements an infimum, but its solution is no longer consistent with the other constraints. We could stop here, accepting no answer to the query. Considering the uncertainty behind decision node splits, we propose instead to relax strict inequalities to inequalities:

$$e'_4 = inf(relax(e_2), f(I_1.\mathbf{x}, I_2.\mathbf{x})) \qquad e'_5 = \pi_{I_2.\mathbf{x}}(e'_4)$$

We then have:

$$\begin{aligned} Th(e'_4) &= \{I_1.x_1 = 2 \wedge I_1.x_2 = 3 \wedge I_2.x_1 = 2 \wedge I_2.x_2 = 3\} \\ Th(e'_5) &= \{I_2.x_1 = 2 \wedge I_2.x_2 = 3\} \end{aligned}$$

The minimal contrastive constraint $e'_5 = e_5$ is unchanged, but the constraint e'_4 differs from the previous one e_4. The theory of e'_4 contains and therefore constraints both instances I_1 and I_2 in such a way as to minimize their distance. Since I_1 is now underspecified, i.e., a set of instances, this means the minimization will look for the subset of such instances for which the distance is minimal. In this example, such a subset turns out to be a singleton.

On the $relax$ operator. The need to relax strict inequalities occurs in general only in the case of expressions using the inf operator, since it is actually implemented as an infimum. By applying the $relax$ operator, we basically admit that instances at the boundary of a split condition in a decision tree can be directed to both branches. This is a reasonable assumption, since the boundary of a split condition is the region with the highest predictive uncertainty. As a further extension, REASONX offers a parameter ϵ to relax strict inequalities with a margin of uncertainty ϵ, e.g., $\mathbf{a}^T\mathbf{x} < b$ is relaxed to $\mathbf{a}^T\mathbf{x} \leq b + \epsilon$. Again, such a relaxation is reasonable for small ϵ's, especially when there is aleatoric uncertainty close to the boundary (see Hüllermeier and Waegeman (2021)). In summary, the usage of $relax$ allows to compute additional minimal contrastive explanation (w.r.t. not using it) to account for uncertainty in model decision boundaries.

Explanations over different models: contrastive explanations. We consider contrastive explanations of a same instance over two different decision trees. These trees can represent either: two models built on different training sets (e.g., trained at different points of time); or, two different models trained on the same training set at the same point of time. The expressions:

$$\begin{aligned} e_6 = sat(cross(&inst(I_1, DT_1, 0, 0.95), inst(I_1, DT_2, 0, 0.95),\\ &inst(I_2, DT_1, 1, 0.95), inst(I_2, DT_2, 1, 0.95), \{U_3\})) \end{aligned}$$
$$e_7 = \pi_{I_2.\mathbf{x}}(e_6)$$

look for conditions for which I_1 is an instance with the same class label in DT_1 and DT_2, and I_2 is a contrastive instance for both DT_1 and DT_2. Such conditions regard contrastive explanations that remain unchanged between DT_1 and DT_2.

Example Let us consider DT_1 fixed as in the previous examples, and DT_2 be defined with a single split condition $x_1 + x_2 < 6$. All instances matching this condition belong to class $l = 0$. Let U_3 include $I_1.x_1 = 2, I_1.x_2 = 2$ plus $I_2.x_1 = I_1.x_1$. The expressions e_6 and e_7 evaluate to:

$$\begin{aligned} Th(e_6) &= \{I_2.x_1 + I_2.x_2 \geq 6 \wedge I_1.x_1 = 2 \wedge I_1.x_2 = 2 \wedge I_2.x_1 = I_1.x_1\}\\ Th(e_7) &= \{I_2.x_1 = 2 \wedge I_2.x_2 \geq 4\} \end{aligned}$$

Comparing $Th(e_7)$ to $Th(e_3)$, we notice that the minimal contrastive examples for I_1 that are also minimal contrastive examples for DT_2 are those for which $I_2.x_1 = 2 \wedge I_2.x_2 \geq 4$. The ones that are not anymore contrastive in DT_2 are those for which $I_2.x_1 = 2 \wedge 4 > I_2.x_2 \wedge I_2.x_2 \geq 3$.

REASONX: Reason to explain

REASONX is a tool for reasoning over explanations and is based on the query language over constraints introduced in the previous section. Here, we present the architecture of REASONX, the translation of user inputs into queries and of answer constraints into explanations, and the execution of the above queries through a CLP($\mathbb{R}$) meta-interpreter.

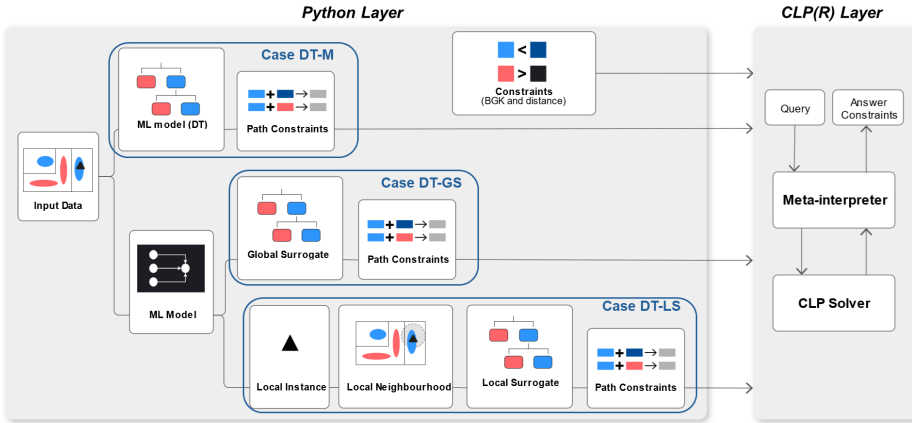

Figure 2. Workflow of data, models and explanations for REASONX. To be read from left to right. The three paths show the cases **(DT-M)**, **(DT-GS)** and **(DT-LS)** for the base model: base model is a DT model, or a global, or a local surrogate. The explanations are created from the answer constraints produced by a meta-interpreter of the query language over queries generated from user inputs (background and distance), and embeddings of the base model (path constraints).

REASONX *architecture*

Figure 2 shows the workflow from data, models, and user constraints to queries, and back from answer constraints to explanations. There are two main layers in REASONX. The core layer is written in CLP($\mathbb{R}$) as provided by the SWI Prolog system (Wielemaker et al., 2012). It implements a meta-interpreter for computing the theory $Th(e)$ of an expression e over the language of queries. On top of it, there is a Python layer, which is constructed such that a user – here, a developer – can easily name instances, link them to decision trees, and state user constraints. The Python layer takes care of the meta-data (name and type of features) and of the ML model. It is designed for an easy integration with the `pandas` and `scikit-learn` standard libraries for data storage and ML model construction.

REASONX is strongly guided by a decision tree, called the *base model*. Such a decision tree can be: **(DT-M)** the model to be explained and reasoned about; **(DT-GS)** a global surrogate model of a ML model, such as a neural network or an ensemble; or **(DT-LS)** a local surrogate model trained to mimic a ML model in the neighborhood instances of the instance to explain. Decision trees are directly interpretable only if their size/depth is limited. Large decision trees are hard to reason about, especially in a contrastive explanation scenario. Case **(DT-M)** can be used in such contexts. In cases **(DT-GS)** and **(DT-LS)**, the surrogate model is assumed to have high fidelity in reproducing the decisions of the original ML model. This is reasonable for local models, i.e., in case **(DT-LS)**, since learning an interpretable model over a neighborhood has been a common

strategy in perturbation-based explainability methods such as LIME (Ribeiro et al., 2016) and LORE (Guidotti et al., 2019a). Explanations produced by REASONX are global in cases **(DT-M)** and **(DT-GS)**, local in case **(DT-LS)**, model-specific in case **(DT-M)**, and model-agnostic in cases **(DT-GS)** and **(DT-LS)**.

Python Layer and Query Generation

Consider the Python layer. Data is stored in tabular form as `pandas` data frames. One or more decision trees are obtained by one of the cases **(DT-M)**, **(DT-GS)**, and **(DT-LS)** discussed above. The user can define one or more instances, with the expected class label, and associate each instance to a specific decision tree. Moreover, the user can directly state linear constraints over the features of one or more instances. In addition, the user can fully or partially specify the value of some of the features of an instance. The Python layer transforms decision trees, user constraints, and additional (called, *implicit*) constraints into CLP($\mathbb{R}$) facts through an embedding procedure described later. The implicit constraints are used to encode the nominal features, which are not directly accounted for in the query language, in a way that is transparent to the user. Then, the Python layer submits one or more expressions over the query language to the CLP($\mathbb{R}$) layer, parses back the answer constraints, and returns, or simply displays, the results to the user.

Let $I_1, \ldots, I_m$ be the instances declared, and $DT_1, \ldots, DT_m$ the linked decision trees. Moreover, let Φ be the user constraints and Ψ the implicit constraints. The following query expressions are general versions of the ones presented in Section Explanations as Queries:

- $S = sat(cross(inst(I_1, DT_1, l_1, pr_1), \ldots, inst(I_m, DT_m, l_m, pr_m), \{\Phi, \Psi\}))$ tests satisfiability of the constraints generated from decision trees, stated by the users, and imposed by the data types;

- $F_i = inst(I_i, DT_i, l_i, pr_i)$ is used to generate factual rules of the form $c \rightarrow l = l_i\ [p]$ from a path in DT_i such that the confidence p of the prediction at the path is such that $p \geq pr_i$. The factual rules are produced only for c's that appear as $c \wedge c' \in Th(S)$ for some c'. This condition states that a constraint c from a path of a decision tree must be consistent with the other constraints of the query S;

- $CE_i = \pi_{I_i.\mathbf{x}}(S)$ generates contrastive constraints $c \in Th(CE_i)$, and contrastive rules of the form $c \rightarrow l = l_i\ [p]$. Here, contrastive refers to the other instances I_j with the class label $l_j \neq l_i$. Also, p is the confidence of the prediction of DT_i for the path that contributes to c;

- $M = inf(relax(S), f(I_h.\mathbf{x}, I_k.\mathbf{x}))$ and $MCE_i = \pi_{I_i.\mathbf{x}}(M)$ generalize S and CE_i to the case of minimal solutions. In particular, minimal contrastive rules are of the form $c \rightarrow l = l_i\ [p]$, where $c \in Th(MCE_i)$ is a minimal contrastive constraint;

- $P = \pi_{\mathbf{w}}(M)$ or $P = \pi_{\mathbf{w}}(S)$ are the most general expressions generated[¶], including the projection of the answer constraints over a set of features $\mathbf{w} \subseteq \mathcal{F}$ required by the user.

The Python layer submits one of the two queries $P = \pi_{\mathbf{w}}(M)$ or $P = \pi_{\mathbf{w}}(S)$ to the CLP($\mathbb{R}$) layer, depending on whether the user requires minimization or not. If the user does not require projection, then $\mathbf{w} = \mathcal{F}$ and P boils down to M or to S respectively. The result of the submitted query, and of the factual rules from F_i are returned by the CLP($\mathbb{R}$) layer. For efficiency reasons, the actual implementation does not submit separate queries for the F_i's and CE_i's, but keeps track of factual and contrastive rules while executing the submitted query. Finally, the user can obtain a contrastive rule CE_i by querying $P = \pi_{\mathbf{w}}(S)$ with $\mathbf{w} = I_i.\mathbf{x}$.

From Python to CLP: Embeddings

Let us discuss here how base models, instances, implicit and user constraints are mapped from Python to CLP($\mathbb{R}$) terms and facts.

First, we point out that REASONX is agnostic about the learning algorithm used to produce the base models. We make the following assumptions. Predictive features can be nominal, ordinal, or continuous. Ordinal features are coded as consecutive integer values. Nominal features can be one-hot encoded or not. If not, we force one-hot encoding before submitting the queries to CLP($\mathbb{R}$), and silently decode back from the answer constraints to nominal features before returning the results to the user. In particular, a nominal feature x_i is one-hot encoded into features $x_i^{v_1}, \ldots, x_i^{v_k}$ with $v_1, \ldots, v_k$ being the distinct values in the domain of x_i. It is important to emphasize that the choice of encoding methods for ordinal and nominal features is non-trivial, as it can significantly impact the performance and fairness of machine learning models (Mougan et al., 2023), as well as the quality of the explanations generated.

We assume that binary split conditions from a parent node to a child node in a decision tree are of the form $\mathbf{a}^T\mathbf{x} \simeq b$, where $\mathbf{x}$ is the vector of all features x_i's. The following common split conditions are covered by such an assumption:

- axis-parallel splits based on a single continuous or ordinal feature, i.e., $x_i \leq b$ or $x_i > b$;
- linear splits over features,i.e., $\mathbf{a}^T\mathbf{x} \leq b$ or $\mathbf{a}^T\mathbf{x} > b$;
- (in)equality splits for nominal features, i.e., $x_i = v$ or $x_i \neq v$; in terms of one-hot encoding, they respectively translate into $x_i^v = 1$ or $x_i^v = 0$.

Axis parallel and equality splits are used in CART (Breiman et al., 1984) and C4.5 (Quinlan, 1993). Linear splits are used in oblique (Murthy et al., 1994; Lee and

[¶]Notice that $\pi_{\mathbf{w}}(S)$ is not a special case of $\pi_{\mathbf{w}}(M)$, e.g., by setting $f()$ to a constant. First, *relax* is not applied in $\pi_{\mathbf{w}}(S)$, while it is in $\pi_{\mathbf{w}}(M)$. Second, the actual implementation of *inf* grounds integer variables, as it will be discussed later on, which is not necessarily the case for $\pi_{\mathbf{w}}(S)$.

Jaakkola, 2020) and optimal decision trees (Bertsimas and Dunn, 2017). Linear model trees combine axis parallel splits at nodes and linear splits at leaves (Frank et al., 1998).

Base models. The base models are encoded into a set of Prolog facts, one for each path in the decision tree from the root node to a leaf node:

$$\texttt{path(}d\texttt{,[}\mathbf{x}\texttt{],}\ \texttt{[}\mathbf{a}_1^T\mathbf{x} \simeq b_1\texttt{,}\ \ldots\texttt{,}\ \mathbf{a}_k^T\mathbf{x} \simeq b_k\texttt{],}\ l\texttt{,}\ p\texttt{).}$$

where d is an id of the decision tree, $\texttt{[}\mathbf{x}\texttt{]}$ is a list of CLP($\mathbb{R}$) variables, one for each feature, l the class predicted at the leaf, p the confidence of the prediction, and $\texttt{[}\mathbf{a}_1^T\mathbf{x} \simeq b_1\texttt{,}\ \ldots\texttt{,}\ \mathbf{a}_k^T\mathbf{x} \simeq b_k\texttt{]}$ is the list of the split conditions from the root to the leaf.

Instances. Each declared instance is encoded by a list of CLP($\mathbb{R}$) variables, one for each feature. The mapping between the feature name and the variable is positional. All the instances are collectively represented by a list of lists of variables.

Implicit constraints (Ψ). A number of constraints on the features $\mathbf{x}$ of each instance naturally derive from the data type of the features. We call them implicit constraints, because the system can generate them from meta-data about features:

- for continuous features: $x_i \in \mathbb{R}$;
- for ordinal features: $x_i \in \mathbb{Z}$ and $m_i \leq x_i \leq M_i$ where $dom(x_i) = \{m_i, \ldots, M_i\}$ is the domain of x_i;
- for one-hot encoded nominal features: $x_i^{v_1}, \ldots, x_i^{v_k} \in \mathbb{Z}$ and $\wedge_{j=1}^{k}(0 \leq x_i^{v_j} \leq 1)$ and $\sum_{j=1}^{k} x_i^{v_j} = 1$;

We denote by Ψ the conjunction of all implicit constraints.

User constraints (Φ). A number of user constraints or *background knowledge*, loosely categorized as in Karimi et al. (2023), and extended based on work by Karimi et al. (2021); Ustun et al. (2019); Mothilal et al. (2020); Mahajan et al. (2019) can be readily expressed in REASONX. We provide examples and use F to refer to a factual instance and CE to refer to a corresponding contrastive instance. *Feasibility* constraints concern the possibility of feature changes between the factual and contrastive instance, and how changes depend on previous values or other features. A feature that is unconstrained in feasibility is *actionable without any condition*. Among the feasibility constraints, we can distinguish:

- *Immutable features:* a feature cannot or must not change between the factual and contrastive instances, e.g., the birthplace: $CE.birthplace = F.birthplace$.
- *Mutable but not actionable features:* the change of a feature from the factual to the contrastive instance is only a result of changes in the features it depends upon. An example is the change of unit scale, e.g., from Euro to US Dollars, which can be encoded as $CE.€ = 1.16 \cdot CE.\$$ – assuming a conversion rate of 1.16.
- *Actionable but constrained features:* a feature can be changed only under some conditions, e.g., age can only increase : $CE.age \geq F.age$.

Another class regards *consistency constraints*, which aim at bounding the domain values of a feature, e.g., $0 \leq CE.age \leq 120$ limits the age feature in the range $[0, 120]$.

Encoding distance functions. The assumption that instances have a specific predicted class label (see operator $inst$), allows us to remove the first term from Equation 1, hence reducing the problem of computing minimal contrastive explanations to minimize the distance function between the factual and contrastive instances. REASONX offers two distance functions that can be used with the inf operator: L_1 combined with a matching norm for the nominal variables (referred to as as L_1 norm for the remainder of this paper) and L_∞, both over normalized features. The L_1 norm penalizes the average change over all features, while the L_∞ norm penalizes the maximum changes over all features. See also Wachter et al. (2017); Karimi et al. (2020) for a discussion of how the different norms affect the generated contrastive explanations. In the Supplemental material, we show how to linearize such norms, i.e., how to express them using linear constraints only, possibly introducing slack variables and adding further implicit constraints to Ψ.

Regarding the diversity of the contrastive explanations, we combine the approach of Lampridis et al. (2023) and the diversity evaluation of Mothilal et al. (2020). Assume that we have a (factual) instance I_f and a large pool of contrastive instances. From this pool, our objective is to select a diverse subset of $|S|$ instances. REASONX encodes the following distance:

$$f(x_f, S) = \frac{\lambda}{|S|} \sum_{i \in S} dist(I_f.\mathbf{x}, I_i.\mathbf{x}) - \frac{1}{|S|^2} \sum_{i \in S} \sum_{j \in S} dist(I_i.\mathbf{x}, I_j.\mathbf{x}) \tag{3}$$

where S is a set of indices of contrastive instances w.r.t. I_f, and λ is a tuning parameter. The first term measures the distance between the factual and the contrastive instances (*proximity*), the second term the distance within the set of contrastive instances (*diversity*). Thus, minimization of the function optimizes the trade-off between proximity and diversity.

CLP Layer and the Meta-interpreter

The core engine of REASONX is a CLP($\mathbb{R}$) meta-interpreter of expressions over the query language presented in Section A Query Language over Linear Constraints for XAI.

Meta-reasoning is a powerful technique that allows a logic program to manipulate programs encoded as terms. In CLP, meta-reasoning is extended by encoding also constraints as terms (Wielemaker et al., 2012). For example, consider Listing 1.

```
satisfiable(P) :-
    copy_term(P, CopyP),
    tell_cs(CopyP).
tell_cs([]).
tell_cs([C|Cs]) :-
    {C},
    tell_cs(Cs).
```

Listing 1: Definition of predicates `satisfiable` and `tell_cs`.

The query `tell_cs([X >= 0, Y = 1 - X])` asserts the linear constraints in the list, adding them to the constraint store. Linear constraints can be the result of some manipulation beforehand. For example, the query `satisfiable([X = 0, Y >= 1 - X])` first makes a copy of the list by renaming all variables into fresh ones, and then it asserts the copy to the constraint store. In this way, the assertion succeeds if and only if the constraints are satisfiable, but without altering the original constraints, e.g., without fixing `X` to 0.

Here, we discuss a simplified version of the REASONX interpreter using the predicate `solve(`E`, `$Vars$`, `C`)`, where E is the expression to interpret, $Vars$ is the list of lists of variables (one list per instance), and C is the returned answer constraint consisting of $Th(E)$. We proceed with the interpretation of each operator.

($\{U\}$) The interpretation of the $\{U\}$ operator is split into two base operators: `userc` for the user constraints and `typec` for the implicit constraints that cover the data types. Listing 2 shows that the former simply retrieves the list of constraints Ψ embedded in the `user_constraints` predicate by the Python layer. The latter appends the lists of constraints for nominal variables and ordinal variables. Details on the called predicates are omitted.

```
solve(userc, Vars, Cons) :-
    user_constraints(Vars, Cons).

solve(typec, Vars, Cons) :-
    cat_constraints(Vars, CCat),
    ord_constraints(Vars, COrd),
    append(CCat, COrd, Cons).
```

Listing 2: Interpretation of the $\{U\}$ operator.

(inst) The operator $inst(I, DT, l, pr)$ is interpreted in Listing 3. The position N of I in the list of all instances is decoded through the `data_instance` predicate asserted by the Python layer. The interpreter accesses the variables V of I (predicate `nth0`), and matches them with the embedding of a path in the DT decision tree that ends in a leaf with class label l and with predicted accuracy P. Finally, only paths for which P is at least the required probability pr are considered.

```
solve(inst(I, DT, L, Pr), Vars, Cons) :-
    data_instance(N, I),
    nth0(N, Vars, V),
    path(DT, V, Cons, L, P),
    P >= Pr.
```

Listing 3: Interpretation of the $inst$ operator.

($cross$) The interpretation of the $cross(L)$ operator is stated by recursively solving the tail list of L and the head of L, and then calculating the cross-product of the constraints in the returned results.

```
solve(cross([]), _, []).
solve(cross([T|Ts]), Vars, Cons) :-
    solve(cross(Ts), Vars, TsCons),
    solve(T, Vars, TCons),
```

```
5     append(TCons, TsCons, Cons).
```

Listing 4: Interpretation of the *cross* operator.

(sat) The $sat(T)$ operator is interpreted as shown in Listing 5. After solving the sub-expression T, the resulting constraints are checked for satisfiability through the `satisfiable` predicate. Such a predicate takes also as input the list of variables in the domain of the integers. By using the meta-programming capabilities of CLP($\mathbb{R}$), after making a fresh copy of its inputs through the built-in predicate `copy_term`, the satisiability is checked by: (1) asserting the copy of the constraints using `tell_cs` (see Listing 1); and (2) checking that among the solutions there is one assigning integer values to integer variables using the `bb_inf` predicate. In this last call, the argument `0` regards the (constant) function to minimize, and the unnamed variable "_" concerns the returned minimum value (not used).

```
1 solve(sat(T), Vars, Cons :-
2     solve(T, Vars, Cons),
3     int_vars(Vars, IntVars),
4     satisfiable(Cons, IntVars).
5
6 satisfiable(Cons, Ints) :-
7   copy_term(Cons-Ints, CopyCons-CopyInts),
8   tell_cs(CopyCons),
9   bb_inf(CopyInts, 0, _).
```

Listing 5: Interpretation of the sat operator.

(π) The interpretation of the $\pi_{\mathbf{w}}(T)$ operator is provided in Listing 6. After solving T, the projection of the solution over the features $\mathbf{w}$ is implemented by the meta-predicate `project` designed by Benoy et al. (2005).

```
1 solve(project(T, PVars), Vars, PCons) :-
2     solve(T, Vars, Cons),
3     project(PVars, Cons, PCons).
```

Listing 6: Interpretation of the π operator.

(inf) Finally, let us consider *inf(T, F)* in Listing 7. After solving T into constraints $TCons$, the interpreter linearizes the expression F regarding the distance function into a linear expression $LinF$, possibly generating additional constraints $ConF$. Satisfiability of the union of $TCons$ and $ConF$ is checked with an extended version of `satisfiable`. Such a variant also computes the infinum value of $LinF$ taking into account the integer variables $IntVars$. The infinum value Min is returned by `satisfiable` together with integer assignments for the integer variables. Such assignments are turned into equality constraints, and together with the key equality $LinF = Min$ from the definition of the min operator (see Section An Algebra of Operators) appended to the other constraints.

```
1 solve(inf(T, F), Vars, Cons)  :-
2     solve(T, Vars, TCons),
3     exp_eval(F, Vars, LinF, ConF),
4     append(ConF, Cons, ConsMin),
```

```
    int_vars(Vars, IntVars),
    satisfiable(ConsMin, IntVars, LinF, Min, IntValues),
    eq_con(IntVars, IntValues, ConsEq),
    append([LinF=Min|ConF], Cons, Cons1),
    append(CEq, Cons1, Cons).
```

Listing 7: Interpretation of the *min* operator.

Integer-linear constraints. The variant of `satisfiable` used in the interpretation of *inf* is based on the MILP built-in predicate `bb_inf(`C`,` $IntVars$`,` $LinF$`,` Min`,` $IntValues$`)`. The extension to MILP problems is required because (one-hot encoded) nominal and ordinal features lie in the domain of the integers. We discuss here the implications of such an extension on the query language of Section A Query Language over Linear Constraints for XAI.

For the given integer variables $IntVars$, a call to the `bb_inf` predicate returns only one answer, i.e., without backtracking (for efficiency reasons, since there may be an exponential number of solutions), with ground values $IntValues$. Such values have the property that the linear function $LinF$ reaches its minimum over C at the minimum of $C \wedge (IntVars = IntValues)$. The latter turns out to be a linear constraint. Instead, the minimization of C in presence of integer variables is an NP-hard problem that cannot always be expressed as a linear constraint (Karp, 1972). The implication for REASONX is that, when integer variables are present (namely, in presence of ordinal or nominal features), the returned answer constraint is a correct but not complete characterization of the space of solutions to the query. For instance, consider the query:

$$min(\{0 \leq x \leq 1, 0 \leq y \leq 1, x - y \leq z, y - x \leq z\}, z)$$

with the assumption that $x, y \in \mathbb{Z}$. This query is intended to constrain z to $|x - y|$, the absolute value of the difference between x and y. Such constraints naturally arise in the modeling of L_1 and L_∞ distances. The variables x and y are set to the ground values returned by `bb_inf`, namely $x = 0, y = 0$, and the minimum is reached for $z = 0$. Hence, the interpreter returns $x = 0, y = 0, z = 0$. This is a correct answer constraint, but it does not cover all possible solutions. For example, $x = 1, y = 1, z = 1$ is another solution which is never explored.

Demonstrations

Here, we present several use-cases to illustrate the functionalities of REASONX. The majority of these demonstrations is based on case **(DT-M)**, i.e., under the assumption that the base model is also the ML model. For an overview of the corresponding experimental settings, see Table 11 in the Supplemental material, which also reports additional examples.

Synthetic Dataset

We define a small synthetic dataset, consisting of two features (`feature1` and `feature2`) and a binary class with values `0` and `1`, each class having $1,000$ data

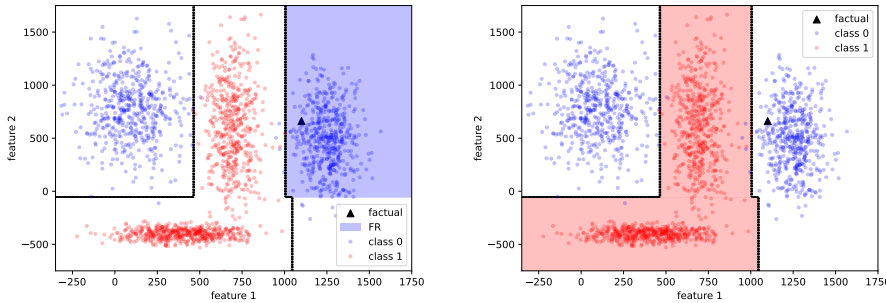

Figure 3. Left: factual region (FR) as provided by REASONX. Right: contrastive regions (CR) as provided by REASONX. Grey lines refer to the decision boundary of the base DT.

instances. These are sampled from different bivariate normal distributions that are almost separable by an axis-parallel decision tree. We choose a data instance from class `0`, called the *factual instance* ("factual" in the figures), to illustrate some of the functionalities of REASONX at the Python layer. To initialize REASONX, the following code is needed.

```
USER:    r = reasonx.ReasonX(pred_atts, target, df_code)
         r.model(clf1)
```

The constructor takes as input the list of features and the class name, and an object (`df_code`) that decodes categorical features. The base DT `clf1` is set via the `model` method.

As a first operation, we set a factual instance of interest, named `F`, using the method `instance`, and query REASONX through the `solveopt` method.

```
USER:    r.instance('F', features=[1100, 661], label=0)
         r.solveopt()
```

`F` is fully specified as all feature values are set. The query generated and solved by the CLP layer is of the form e_1 as in (2). There is a single contraint in $Th(e_1)$. The output of REASONX shows such a constraint (called *answer constraint*) and the respective factual rule $c \rightarrow l = 0$ with confidence 1.0.

```
REASONX: Answer constraint:
         F.feature1=1100.0,F.feature2=661.0
         Rule satisfied by F:
         IF F.feature2> -55.5,F.feature1>1004.0 THEN class 0 [1.0]
```

To help intuition, we also plot the results. The left-hand side c of the factual rule is shown as factual region in Figure 3 (left).

Next, we ask for the contrastive constraints, i.e., feasible regions of contrastive instances. We code the problem by declaring an additional instance with the name `CE`, with class label `1`, and now requiring a minimum confidence of 0.8 for its factual rule (or contrastive rule in relation to the first instance).

```
USER:    r.instance('CE', label=1, minconf=0.8)
         r.solveopt()
```

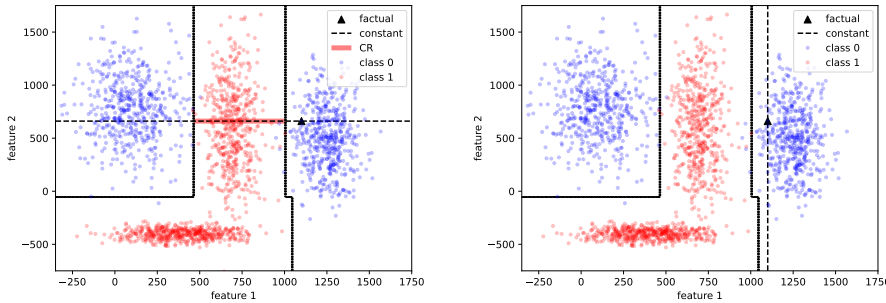

Figure 4. Left: contrastive region (CR) as provided by REASONX, and given a constraint on `feature2`. Right: given the constraint on `feature1`, no solution exists. The dashed line refers to the enforced constraint. Grey lines refer to the decision boundary of the DT.

```
REASONX: Answer constraint:
         F.feature1=1100.0,F.feature2=661.0,CE.feature2<= -55.5,CE.feature1<=1043.5
         Rule satisfied by F:
         IF F.feature2> -55.5,F.feature1>1004.0 THEN class 0 [1.0]
         Rule satisfied by CE:
         IF CE.feature2<= -55.5,CE.feature1<=1043.5 THEN class 1 [0.9974]
--
         Answer constraint:
         F.feature1=1100.0,F.feature2=661.0,CE.feature2> -55.5,CE.feature1>466.0,
         CE.feature1<=1004.0
         Rule satisfied by F:
         IF F.feature2> -55.5,F.feature1>1004.0 THEN class 0 [1.0]
         Rule satisfied by CE:
         IF CE.feature2> -55.5,CE.feature1>466.0,CE.feature1<=1004.0
         THEN class 1 [0.9939]
```

Figure 3 (right) shows the two contrastive regions denoted by the left-hand sides of the two rules for `CE`.

Notice that the instances `F` and `CE` are not related, apart from being predicted by the DT with different classes. This means that while we interpret instance `CE` as contrastive to instance `F`, it could be also interpreted as a different factual instance. We then use a constraint to ensure that `feature2` stays constant between the factual instance and the contrastive one (see Figure 4 (left)), or to ensure that instead `feature1` stays constant (Figure 4 (right)). While the first constraint leads to an admissible contrastive region as a line (`CE.feature2 = 661.0, CE.feature1 > 466.0, CE.feature1 <= 1004.0`), the second constraint leads to no solution. The following code shows these cases. Before the second constraint is asserted, the first constraint needs to be retracted (see second `USER` code box, line 1). Retraction facilitates interactivity: asserted constrains accumulate one after the other, until they are retracted[‖].

[‖] `r.retract(last=True)` retracts the last asserted constraint, and `r.reset(keep_model=True)` retracts all instances and constraints, but it keeps the DTs.

```
USER:    r.constraint('CE.feature2 = F.feature2')
         r.solveopt()
```

```
REASONX: Answer constraint:
         F.feature1=1100.0,F.feature2=661.0,CE.feature2=661.0,CE.feature1>466.0,
         CE.feature1<=1004.0
         Rule satisfied by F:
         IF F.feature2> -55.5,F.feature1>1004.0 THEN class 0 [1.0]
         Rule satisfied by CE:
         IF CE.feature2> -55.5,CE.feature1>466.0,CE.feature1<=1004.0
         THEN class 1 [0.9939]
```

```
USER:    r.retract('CE.feature2 = F.feature2')
         r.constraint('CE.feature1 = F.feature1')
         r.solveopt()
```

```
REASONX: No answer.
```

Now, we extend the example by asking for the *closest* contrastive points lying over the identity line (`CE.feature2 = CE.feature1`). The minimization of the L_1 norm between `F` and `CE` is passed as an argument to `solveopt`. From now on, we omit the output of the rules (this can be set with the method `verbosity()`).

```
USER:    r.retract(last=True)
         r.constraint('CE.feature2 = CE.feature1')
         r.solveopt(minimize='l1norm(F, CE)', project=['CE'])
```

```
REASONX: Answer constraint:
         F.feature1=1100.0,F.feature2=661.0,CE.feature1= -55.5,CE.feature2= -55.5
         Min value: 0.894
--
         Answer constraint:
         F.feature1=1100.0,F.feature2=661.0,CE.feature1=1004.0,CE.feature2=1004.0
         Min value: 0.185
```

Figure 5 (left) reports the two solutions returned by REASONX – shown as red dots. Both of them stay in the identity line. They are the closest instances for each of the contrastive regions characterized by leaves of the DT. In fact, REASONX solves the optimization problem for each of those independently and provides the user therefore not with the global optimum, but with two local optima. This leaves more flexibility to the user. Further processing of the results, e.g., filtering for the global optimum, can be implemented. The data point `CE.feature1 = 1004.0, CE.feature2 = 1004.0` denoted by *CE 2* in Figure 5 (left) is the global optimum. The distance between such pairs is the one reported in the output of REASONX.

Let us now allow `feature2` of the factual instance to be under-specified, by setting it to a range instead of a fixed value. We reset all constraints, and assert the value for `feature1` and bounds for `feature2`. In the `solveopt` we also add another parameter to project the answer constraint over the `CE` instance only. Figure 5 (right) shows the region of the two answer constraints: a single point and a line. Each point in the line represent a closest point to some of the possible values for `F`.

```
USER:    r.reset(keep_model=True)
         r.instance('F', label=0)
```

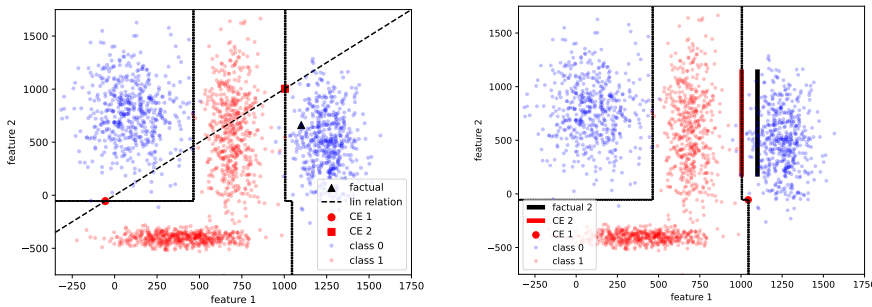

Figure 5. Left: minimal CEs provided by REASONX, under the constraint denoted by the identity line. Right: minimal CEs provided by REASONX for under-specified instances. Grey lines refer to the decision boundary of the DT.

Description	Output
$\mathcal{R}_{\mathcal{F},0}$	`IF F0.capitalgain<=5095.5, F0.education<=12.5, F0.age>29.5 THEN <=50K [0.8095]`
$\mathcal{R}_{\mathcal{F},1}$	`IF F1.capitalgain<=5119.0, F1.education<=12.5, F1.age>33.5 THEN <=50K [0.7907]`

Table 3. Factual rules $\mathcal{R}_{\mathcal{F},i}$ for a same instance and different models over time DT_i with $i = 0, 1$.

```
        r.constraint('F.feature1 = 1100, F.feature2 >= 161,
        F.feature2 <= 1161')
        r.instance('CE', label=1, minconf=0.8)
        r.solveopt(minimize='l1norm(F, CE)', project=['CE'])
```

```
REASONX: Answer constraint:
         CE.feature1=1043.5,CE.feature2= -55.5
         Min value: 0.115
--
         Answer constraint:
         CE.feature1=1003.99999,CE.feature2>=161.0,CE.feature2<=1161.0
         Min value: 0.051
```

Reasoning over Time

REASONX can be run on several models and instances. Here, we assume the following case: a model is used over a long period of time, and it undergoes re-training due to new, incoming data instances. The re-training induces small model changes.

We simulate this case by splitting the Adult Income dataset into two datasets of the same size, and training the same type of model under the same parameter settings on these two splits. Let DT_0 and DT_1 be the resulting decision trees.

Independent explanations. First, we define two instances `F0` and `F1`, assigning the same data values to them, but a different model – DT_0 and DT_1 respectively. We

Index	Output (rules or contrastive examples)	
(1)	$\mathcal{R}_{\mathcal{C},0}$	IF CE0.capitalgain<=5095.5,CE0.education>12.5,CE0.age>30.5 THEN >50K [0.5087]
(2)	$\mathcal{R}_{\mathcal{C},0}$	IF CE0.capitalgain>5095.5,CE0.capitalgain<=6457.5 THEN >50K [0.9286]
(3)	$\mathcal{R}_{\mathcal{C},0}$	IF CE0.capitalgain>7055.5,CE0.age>20.0 THEN >50K [0.9873]
(4)	CE_0 (0.267)	CE0.race=AsianPacIslander, CE0.sex=Male,CE0.workclass=Private, CE0.education=13.0,CE0.age=40.0,CE0.capitalgain=0.0, CE0.capitalloss=0.0,CE0.hoursperweek=40.0
(5)	CE_0 (0.051)	CE0.race=AsianPacIslander, CE0.sex=Male,CE0.workclass=Private, CE0.education=9.0,CE0.age=40.0,CE0.capitalgain=5095.51, CE0.capitalloss=0.0,CE0.hoursperweek=40.0
(6)	CE_0 (0.071)	CE0.race=AsianPacIslander, CE0.sex=Male,CE0.workclass=Private, CE0.education=9.0,CE0.age=40.0,CE0.capitalgain=7055.51, CE0.capitalloss=0.0,CE0.hoursperweek=40.0
(7)	$\mathcal{R}_{\mathcal{C},1}$	IF CE1.capitalgain<=5119.0,CE1.education>12.5,CE1.age>29.5 THEN >50K [0.5125]
(8)	$\mathcal{R}_{\mathcal{C},1}$	IF CE1.capitalgain>5119.0,CE1.capitalgain<=5316.5 THEN >50K [1.0000]
(9)	$\mathcal{R}_{\mathcal{C},1}$	IF CE1.capitalgain>7073.5,CE1.age>20.5 THEN >50K [0.9904]
(10)	CE_1 (0.267)	CE1.race=AsianPacIslander,CE1.sex=Male,CE1.workclass=Private, CE1.education=13.0,CE1.age=40.0,CE1.capitalgain=0.0, CE1.capitalloss=0.0,CE1.hoursperweek=40.0
(11)	CE_1 (0.051)	CE1.race=AsianPacIslander,CE1.sex=Male,CE1.workclass=Private, CE1.education=9.0,CE1.age=40.0,CE1.capitalgain=5119.01, CE1.capitalloss=0.0,CE1.hoursperweek=40.0
(12)	CE_1 (0.071)	CE1.race=AsianPacIslander,CE1.sex=Male,CE1.workclass=Private, CE1.education=9.0,CE1.age=40.0,CE1.capitalgain=7073.51, CE1.capitalloss=0.0,CE1.hoursperweek=40.0
(13)	$\mathcal{R}_{\mathcal{C},0,1}$	IF CE.capitalgain>5095.5,CE.capitalgain<=5119.0,CE.education>12.5, CE.age>29.5 THEN >50K [0.5125]
(14)	$CE_{0,1}$ (0.318)	CE.race=AsianPacIslander,CE.sex=Male,CE.workclass=Private, CE.education=13.0,CE.age=40.0,CE.capitalgain=5095.51, CE.capitalloss=0.0,CE.hoursperweek=40.0

Table 4. Contrastive rules and examples for a same instance and different models. Rules $\mathcal{R}_{\mathcal{C},0}$ and examples CE_0 in (1) - (6) refer to DT_0. Rules $\mathcal{R}_{\mathcal{C},1}$ and examples CE_1 in $\mathcal{R}_{\mathcal{C},1}$ in (7) - (12) refer to DT_1. Rule $\mathcal{R}_{\mathcal{C},0,1}$ and example $CE_{0,1}$ in (13) and (14) refer to additional conditions that are the result of intersecting explanations from DT_0 and DT_1. Numerical values in round brackets indicate the distance between factual instance and contrastive examples.

independently query for factual and contrastive rules, and for the minimal CEs under the L_1 norm. Results are displayed in Table 3 for the factual rules and in Table 4 for the contrastive rules and examples. In both cases, three contrastive rules and three minimal contrastive examples are generated. The contrastive examples (4) and (10) are the same.

Intersecting explanations. Now, we use the reasoning capabilities of REASONX to better understand how these two sets of explanations relate to each other. In order to do so, we use the output produced for instance `F0` (the answer constraints) as an additional input before querying for contrastive rules and instances of instance `F1`. This is achieved using the `r.constraint()` method to assert the answer constraints. As a result, we obtain four rules that are not only contrastive to $F1$ but also to $F0$ and thus in their intersection. Three out of these four rules can be identified as rule (1), (8) and (9) in Table 4. There is also a novel rule, displayed under index (13). This rule is the intersection of rule (2) and (7). Similarly, we obtain the contrastive examples (4/10), (11) and (12)

Description	Output
$\mathcal{R}_{\mathcal{F},0}$	`IF F0.capitalgain>7055.5,F0.age>20.0 THEN >50K [0.9882]`
$\mathcal{R}_{\mathcal{F},1}$	`IF F1.education<=12.5,F1.capitalgain>7073.5 THEN >50K [0.9940]`

Table 5. Factual rules $\mathcal{R}_{\mathcal{F},i}$ for a same instance and different models types DT_i with $i = 0, 1$.

Index	Output (rules)	
(1)	$\mathcal{R}_{\mathcal{C},0}$	`IF CE0.capitalgain<=5119.0,CE0.education<=12.5,CE0.age<=30.5 THEN <=50K [0.9653]`
(2)	$\mathcal{R}_{\mathcal{C},0}$	`IF CE0.capitalgain<=5119.0,CE0.education<=12.5,CE0.age>30.5 THEN <=50K [0.8032]`
(3)	$\mathcal{R}_{\mathcal{C},0}$	`IF CE0.capitalgain<=5119.0,CE0.education>12.5,CE0.age<=28.5 THEN <=50K [0.9022]`
(4)	$\mathcal{R}_{\mathcal{C},0}$	`IF CE0.capitalgain<=5119.0,CE0.education>12.5,CE0.age>28.5 THEN <=50K [0.5068]`
(5)	$\mathcal{R}_{\mathcal{C},0}$	`IF CE0.capitalgain>5316.5,CE0.capitalgain<=7055.5 THEN <=50K [0.7400]`
(6)	$\mathcal{R}_{\mathcal{C},0}$	`IF CE0.capitalgain>7055.5,CE0.age<=20.0 THEN <=50K [0.8000]`
(7)	$\mathcal{R}_{\mathcal{C},1}$	`IF CE1.education<=12.5,CE1.capitalgain<=4243.5, CE1.capitalloss<=1805.0 THEN <=50K [0.9760]`
(8)	$\mathcal{R}_{\mathcal{C},1}$	`IF CE1.education>12.5,CE1.sex_Female<=0.5,CE1.age<=29.5 THEN <=50K [0.9474]`
(9)	$\mathcal{R}_{\mathcal{C},1}$	`IF CE1.education>12.5,CE1.sex_Female>0.5, CE1.capitalgain<=4718.5 THEN <=50K [0.9395]`
(10)	$\mathcal{R}_{\mathcal{C},0,1}$	`IF CE.capitalgain<=4243.5,CE.education<=12.5,CE.age<=30.5, CE.capitalloss<=1805.0 THEN <= 50K [0.9653]`
(11)	$\mathcal{R}_{\mathcal{C},0,1}$	`IF CE.capitalgain<=5119.0,CE.education>12.5,CE.age<=28.5, CE.sex_Female<=0.5 THEN <= 50K [0.9022]`
(12)	$\mathcal{R}_{\mathcal{C},0,1}$	`IF CE.capitalgain<=4718.5,CE.education>12.5,CE.age<=28.5, CE.sex_Female>0.5 THEN <= 50K [0.9022]`

Table 6. Contrastive rules for a same instance and different model types. Rules $\mathcal{R}_{\mathcal{C},0}$ in (1) - (6) refer to DT_0. Rules $\mathcal{R}_{\mathcal{C},1}$ in (7) - (9) refer to DT_1. Rules $\mathcal{R}_{\mathcal{C},0,1}$ in (10) - (12) refer to additional conditions that are the result of intersecting explanations from DT_0 and DT_1.

in Table 4, and a new example (14). With this approach, we obtained the *intersection* of the previously (independently) derived contrastive rules, meaning that the obtained contrastive rules are valid in *both* models. Similarly, the computed contrastive examples are valid in both models. If we remind ourselves of the time dimension, it means that the change in the model over time became indeed visible in its explanation and that some of the contrastive rules and instances derived at the first time point (model DT_0) are no longer valid at the second time point (model DT_1) as they are not contained in the intersection of both.

Reasoning over Different Model Types

In this section, we show how REASONX can be used to reason over different model types. As an example, we compare a DT model (case **(DT-M)**) with a ML model that is explained via a global surrogate model, i.e., case **(DT-GS)**. A simple comparison between a random forest (RF), a multi-layer perceptron (MLP), and an XGBoost (XGB) model shows that the latter slightly outperforms the former models (an accuracy of 0.851 against 0.814 for the MLP and 0.808 for the RF, in comparison 0.802 for the DT). We therefore

compare the DT with the XGB model. The fidelity, i.e., the ratio of correct prediction of the global surrogate DT w.r.t. prediction of the XGB model is $fidelity = 0.919$.

Independent explanations. We compare the factual decision rules behind a the same data instance. The results are depicted in Table 5. The reasons behind the classification of the data instance, i.e., the features appearing on the left-hand side of the rules, are different – even if both models correctly classify the instance as class `>50K`. Both models use the feature `capitalgain`, but while the DT (model 0) relies on the feature `age`, the XGB model (model 1) uses the feature `education`.

We expect to see these differences also in the contrastive decision rules. These are displayed in Table 6. Three aspects immediately stand out. *First*, the DT offers more contrastive rules than the XGB model. *Second*, the confidence values of the DT are on average lower than for the XGB (0.786 versus 0.954, on average). *Third*, the feature `sex`, a sensitive attribute, appears only in the contrastive rules of the XGB.

Intersecting explanations. We compute the *intersection* between the contrastive rules of both models, using the same approach as in Section Reasoning over Time. However, we restrict the rules to those that have a minimum confidence value of 0.9, i.e., we intersect only rules with the index (1), (3), and (7) - (9) from Table 6. This restriction is a conscious choice and can be easily enforced by declaring the `minconf` when initializing the contrastive instances using `r.instance(...)`. As a result, we obtain rule (10) as the intersection between (1) and (7), rule (11) as the intersection between (3) and (8), and rule (12) as the intersection between (3) and (9).

Recalling that we compare two different ML models trained on the same dataset and that have a small gap in performance, we observe that these two models, according to REASONX, draw on different reasons for the same classification. Further, only one of the two models makes use of a sensitive attribute. This is an important observation: while model performance matters, it cannot be the only determining reason to choose a specific model for an application. Rather, it is important *why* a model produced a specific output. Understanding that these reasons can be very different for similarly well performing models trained on the same dataset is thereby an integral part.

In general, more reasoning operations using REASONX are possible, such as reasoning over the produced contrastive explanations (similar to the demonstration in Section Reasoning over Time), or reasoning over more than two ML models.

Diversity Optimization

In this experiment, we select three contrastive explanations from an admissible set of contrastive explanations, using the optimization function discussed in Section From Python to CLP: Embeddings, paragraph Encoding distance functions. We run this experiment on the Adult Income dataset, case **(DT-M)** and for a decision tree with a depth of $depth = 5$. This parameter is changed to obtain a higher number of contrastive examples. Furthermore, we set $\lambda = 0.5$ and repeat the optimization for 50 instances.

Results are compared to the classical approach of selecting contrastive instances, i.e., optimizing only by proximity. In both cases, we compute the value of the optimization

function $f(x_f, S)$ and of the proximity and diversity term separately. We plot the results in Figure 6 (see the Supplemental material in Diversity Optimization). The classical approach leads to a distribution of values of the optimization function that has a sharp peak and is centered around zero. When considering the diversity of generated contrastive explanations in the optimization, the distribution of the function changes: values are much more diverse, there is no sharp peak. To confirm that these values indeed belong to different distributions, we calculated the Kolmogorov-Smirnow test (p-value ≈ 0).

Detecting Biases

Here, we demonstrate how the generation of contrastive examples and the use of constraints in REASONX can support the detection of societal biases. This case is close to the definition of *explicit bias* as in Goethals et al. (2024) and may be useful in the context of *direct discrimination* and *prima facie* discrimination. We investigated a DT and an XGB model, and possible discrimination based on the sensitive attributes `age`, `race`, `sex`, and pairwise combinations.

In summary, we find that in the case of a DT, REASONX does not highlight potential biases. For the XGB model, the influences of `age` and `sex` on the predicted outcome may be relevant. Details on the experiment, more information on bias detection through XAI and definitions, as well as details on the experimental results can be found in the Supplemental material in Detecting Biases.

Quantitative Evaluation

In this section, we present the quantitative evaluation of REASONX. We start by evaluating REASONX only. Then, we compare (minimal) contrastive examples of REASONX against those produced by DiCE (Mothilal et al., 2020), and factual rules of REASONX against those produced by ANCHORS (Ribeiro et al., 2018). The chosen approaches approximate two aspects of REASONX separately so that a fair comparison is possible on these aspects. In addition, we compare the runtime of the three approaches. Information on experimental details and datasets can be found in Section Experiments in the Supplemental material.

The metrics used in this section are standard metrics (Vilone and Longo, 2021; Pawelczyk et al., 2021; Guidotti et al., 2019a), adapted to the characteristics of REASONX as a tool based on CLP. Details on metrics are discussed in Section Metrics in the Supplemental material.

Also, the Supplemental material contains experiments on the parameters of REASONX in Section Parameter Testing.

Results

REASONX *only.* Results for REASONX are shown in Table 7. Results depend on the dataset and case. Lengths of factual and contrastive rules are in most cases not the same but of similar size. In most cases, the number of solutions of contrastive rules and constraints (for both norms) is the same, i.e., an example could be identified on all

	case	$acc/f\,(^\star)$	l_F	l_C	N_C	S/N		N_{CE}	d_{CE}	dim_{CE}
ADULT	(DT-M)	0.802	2.988	2.008	2.049	0.82	L_1	2.049	0.063	p.
							L_∞	2.049	0.062	h.-d.
	(DT-GS)	0.851/0.919	2.978	2.832	4.674	0.92	L_1	4.674	0.449	p.
							L_∞	4.674	0.378	h.-d.
	(DT-LS)	0.851/0.840	1.944	2.564	2.371	0.89	L_1	2.371	0.244	p.
							L_∞	2.371	0.201	h.-d.
SGC	(DT-M)	0.737	3.0	3.0	3.208	0.77	L_1	3.208	0.762	p.
							L_∞	2.922	0.515	h.-d.
	(DT-GS)	0.757/0.767	2.986	2.739	3.217	0.767	L_1	3.217	0.687	p.
							L_∞	2.565	0.514	h.-d.
	(DT-LS)	0.757/0.797	2.947	2.961	3.092	0.76	L_1	3.092	0.819	p.
							L_∞	2.632	0.515	h.-d.
GMSC	(DT-M)	0.936	3.0	2.0	1.0	0.95	L_1	1.0	0.020	p.
							L_∞	1.0	0.015	h.-d.
	(DT-GS)	0.936/0.984	2.041	2.666	3.020	0.98	L_1	3.020	0.022	p.
							L_∞	3.020	0.019	h.-d.
	(DT-LS)	0.936/0.880	2.500	2.587	4.294	0.34	L_1	4.235	0.266	p.
							L_∞	4.294	0.228	h.-d.
DCCC	(DT-M)	0.780	3.0	3.0	1.081	0.74	L_1	0.865	0.150	p./h.-d.
							L_∞	1.081	0.148	h.-d.
	(DT-GS)	0.778/0.926	3.0	3.0	1.064	0.94	L_1	0.819	0.148	p./h.-d.
							L_∞	1.064	0.145	p.
	(DT-LS)	0.778/0.940	2.411	2.603	2.011	0.90	L_1	1.533	0.495	p./h.-d.
							L_∞	2.011	0.400	h.-d.
ACA	(DT-M)	0.845	3.0	3.0	4.506	0.83	L_1	4.506	0.942	p.
							L_∞	4.506	0.897	h.-d.
	(DT-GS)	0.850/0.857	2.796	2.796	3.389	0.857	L_1	3.389	0.768	p.
							L_∞	3.389	0.705	h.-d.
	(DT-LS)	0.850/0.990	2.013	2.497	2.425	0.80	L_1	2.25	1.037	p.
							L_∞	2.25	0.862	h.-d.

Table 7. Evaluation of REASONX. SGC = South German Credit, GMSC = Give Me Some Credit, DCCC = Default Credit Card Clients, ACA = Australian Credit Approval. The CE obtained after the MILP optimization can be a point (abbreviated p.) or a higher-dimensional object such as a line (abbreviated h.-d.). (*) No fidelity in case **(DT-M)**.

admissible paths of the DT. Also, the values of the L_1 norm are always a bit larger than values for the L_∞ norm. Excluding the DCCC datasets, for the L_1 norm, only points (zero-dimensional) are obtained while the L_∞ norm returns higher-dimensional objects (solutions to linear constraints are polyhedra, in general). This is due to the latter penalizing only the maximum change over all features while the first penalizes all changes.

Constrative examples. A comparison between REASONX and DiCE is provided in Table 8. Distances of REASONX are smaller than those obtained with DiCE, i.e., the contrastive examples are closer to the original data instance. An exception is case **(DT-LS)** under the immutability constraint on `capitalgain`. When constraints are enforced, less solutions are found by REASONX. Such a behavior is expected as adding constraints cuts down on the admissible paths in the (constant) base model. The change in the distance between factual and contrastive depends then on the CE that remain on the admissible paths. In DiCE, the number of contrastive examples is a parameter. No general statement can be made about the change in the distance.

Setting/Approach	acc/f	N_{CE} L_1	N_{CE} L_{inf}	d_{CE} ($w_{div}=0$) L_1	d_{CE} ($w_{div}=0$) L_{inf}
No constraints					
DiCE	0.785 (*)	2		1.131	0.775
		3		1.097	0.754
		4		1.170	0.784
		5		1.107	0.767
REASONX (case DT-GS)	0.851/0.919	4.674	4.674	0.449	0.378
REASONX (case DT-LS)	0.851/0.840	2.371	2.371	0.244	0.201
Immutability on `capitalgain`					
DiCE	0.785 (*)	2		1.128	0.778
		3		1.167	0.808
		4		1.125	0.777
		5		1.113	0.778
REASONX (case DT-GS)	0.851/0.919	1.935	1.935	0.581	0.516
REASONX (case DT-LS)	0.851/0.840	0.011	0.011	1.467	1.0
Immutability on `race` and `sex`					
DiCE	0.785 (*)	2		0.976	0.680
		3		0.994	0.683
		4		0.983	0.687
		5		0.974	0.668
REASONX (case DT-GS)	0.851/0.919	3.674	3.674	0.217	0.202
REASONX (case DT-LS)	0.851/0.840	2.371	2.371	0.244	0.201

Table 8. Contrastive examples/constraints: comparison between DiCE and REASONX. (*) No fidelity for DiCE.

Approach	acc/f	D	p_0 or MC_F (**)	p_c	c or S/N (**)	l_F
ANCHORS	0.850 (*)	n.a.	0.9	0.910	0.433	2.06
			0.95	0.944	0.352	3.04
			0.99	0.966	0.273	4.55
REASONX (case DT-GS)	0.851/0.891	2	0.9	n.a.	0.69	2.0
			0.95	n.a.	0.69	2.0
			0.99	n.a.	0	n.a.
REASONX (case DT-LS)	0.851/0.839	2	0.9	n.a.	0.87	1.011
			0.95	n.a.	0.86	1.0
			0.99	n.a.	0.23	1.0
REASONX (case DT-GS)	0.851/0.919	3	0.9	n.a.	0.81	2.975
			0.95	n.a.	0.7	2.971
			0.99	n.a.	0.02	2.0
REASONX (case DT-LS)	0.851/0.840	3	0.9	n.a.	0.78	1.936
			0.95	n.a.	0.72	1.931
			0.99	n.a.	0.39	1.923
REASONX (case DT-GS)	0.851/0.929	4	0.9	n.a.	0.79	3.127
			0.95	n.a.	0.79	3.127
			0.99	n.a.	0.02	3.0
REASONX (case DT-LS)	0.851/0.846	4	0.9	n.a.	0.66	2.136
			0.95	n.a.	0.66	2.136
			0.99	n.a.	0.59	2.068

Table 9. Factual rules: comparison between REASONX and ANCHORS. (*) ANCHORS has no fidelity. (**) The first metric (p_0 or c) is reported for ANCHORS, the second (MC_F or S/M) is a non-equivalent but similar metric for REASONX.

Factual rules. Results on factual rules produced by REASONX and ANCHORS are displayed in Table 9. For ANCHORS, the higher the precision the lower the coverage

Approach	Setting	Constraints	Time (in s)
REASONX	initialization (*)	n.a.	0.129
	case DT-M, $D = 3$, $MC_F = 0$		
	initialization, query	no constraints	1.102
	initialization, query	immutability `capitalgain`	0.685
	case DT-GS, $D = 3$, $MC_F = 0$		
	initialization, query	no constraints	0.825
	initialization, query	immutability `capitalgain`	0.574
	case DT-LS, $D = 3$, $MC_F = 0$		
	initialization, query	no constraints	1.375
	initialization, query	immutability `capitalgain`	0.891
ANCHORS	$p_0 = 0.9$	n.a.	0.558
	$p_0 = 0.95$	n.a.	0.942
	$p_0 = 0.99$	n.a.	0.974
DiCE	$N_{CE} = 2$	no constraints	0.445
	$N_{CE} = 2$	immutability `capitalgain`	0.382
	$N_{CE} = 3$	no constraints	0.169
	$N_{CE} = 3$	immutability `capitalgain`	0.117
	$N_{CE} = 4$	no constraints	0.212
	$N_{CE} = 4$	immutability `capitalgain`	0.158
	$N_{CE} = 5$	no constraints	0.247
	$N_{CE} = 5$	immutability `capitalgain`	0.142

Table 10. Runtime for REASONX, ANCHORS and DiCE. (*) relies on case **(DT-M)**.

of the extracted rule. Also, the rule length increases with precision. In REASONX, the rule length correlates with the depth of the decision tree. For a fixed depth D, the higher the minimum confidence MC_F the smaller the ratio S/N. This is expected as a higher minimum confidence restricts the number of admissible paths in the base model. The average rule length is less than or equal to the tree depth and decreases (slightly) with higher minimum confidence.

Runtime. We report runtimes of REASONX, ANCHORS and DiCE Table 10. Runtimes of REASONX are higher compared to the runtimes of ANCHORS and DiCE, excluding high precision values of ANCHORS and case **(DT-GS)** of REASONX. This is expected, given its higher complexity and expressivity of REASONX and the overhead due to communication between Python and CLP($\mathbb{R}$). For ANCHORS, runtimes increase with an increased precision parameter p_0. For DiCE, while no systematic change can be observed when the parameter N_{CE} is increased, runtimes decrease if an immutability constraint is added. The same decrease in runtime with an additional constraint can be observed for REASONX. In addition, runtimes for REASONX are highest for case **(DT-LS)**. This is expected as case **(DT-LS)** is the most complex one: one base model has to be learned per instance.

Completeness, Coverage, and Constraint Validity

The following statements regarding the completeness, coverage and validity of REASONX hold, assuming that its base model was learned successfully, i.e., that accuracy (case **(DT-M)**) or fidelity (case **(DT-GS)** and **(DT-LS)**) are sufficiently high. This is guaranteeing a certain level of correctness between the base model's prediction and the original class (case **(DT-M)**) or the predicted class (case **(DT-GS)** and **(DT-LS)**): *first*, under

no additional constraints, for every data instance, REASONX can find the factual, and one or more contrastive rules. It can also find one or more contrastive data instances. If no factual rule is returned, this is because the prediction of the base model does not agree with the data label (case **(DT-M)**) or the prediction of the approximated ML model (case **(DT-GS)** and **(DT-LS)**), or because the parameter of the minimum confidence value is too high. However, this is an intentional design choice and REASONX can still retrieve the results by a change in the query – to explain the incorrect prediction of the base model, or to provide an explanation with a lower confidence value. Additionally, for the returned contrastive rule and the contrastive constraints/examples, an additional check of the prediction w.r.t. the black-box ML model in cases **(DT-GS)** and **(DT-LS)** is needed to ensure that they are indeed located on the other side of the decision boundary. Such a check is currently not implemented, relying on the assumption of very high fidelity of the surrogate DT to the black-box model. A possible future implementation consists of relying on a dataset of instances (possibly, a synthetic one) to estimate the precision of the contrastive rule. This is a non-trivial problem when the instance to explain is under-specified. *Second*, the average length and the number of the obtained (factual or contrastive) rules depend on the depth and the width of the base model (see also Section Quantitative Evaluation), and the distance between the original and the contrastive instances may not be minimal w.r.t. the original decision boundary, as the base model only approximates the boundary. *Third*, all data instances in the input space of the base model are covered by factual rules which have no overlap – only one rule is the factual rule. Other XAI approaches based on rules do not have this property, therefore (Lakkaraju et al., 2016; Vilone and Longo, 2021) introduce a metric called "fraction overlap". *Fourth*, if constraints are enforced in REASONX, the rules – if they exist – align with the constraints. This is not always the case for XAI approaches based on rules, thus, (Pawelczyk et al., 2021) introduce a metric called "constraint violation" specifically for contrastive explanations.

Limitations and Future Work

Let us discuss here the limitations and future work of our approach regarding its evaluation, technical aspects, the user-perspective, and social assumptions.

Evaluation. From the theoretical side, REASONX advances other methods in qualitative terms. However, further work is needed to verify this also from a practical side: how the tool is perceived by expert or lay users, how easy it is to encode and reason about domain knowledge as constraints, and how the approach scales in presence of noise and non-separable classes. This validation can be conducted through user studies (see Rong et al. (2024) for a survey specifically on XAI) and through real-world case studies. Regarding the application setting of this paper, namely the credit domain, some reference literature on credit scoring and ML/XAI and can be found at Bücker et al. (2022); Demajo et al. (2020); Shi et al. (2022). We further re-emphasize that the cases presented in this paper are constructed and do not represent real-world conditions.

Technical. We demonstrated the capabilities of REASONX on tabular data, and for binary decision problems. While an extension to multi-class problems is straightforward, the extension to unstructured data, such as text and images, needs to be strictly formalized. In the domain of images, a solution can be the integration of concepts, as shown, e.g., by Donadello and Dragoni (2020). Moreover, the extension of the approach to other constraint (logic programming) schemes beyond linear constraints over the reals is a relevant direction, including, for instance, finite domains, non-linear constraints, or probabilistic/fuzzy logic programming. The literature on such extensions is large (Morettin et al., 2021; Körner et al., 2022). To adapt our framework to such extensions, the key requirement is to preserve the closedness property of the extended language, ensuring that the optimization and reasoning capabilities remain sound and tractable. Moreover, it will be worth investigating how domain knowledge expressed in other formalisms (e.g., knowledge graphs) can be encoded and reasoned about as constraints.

User-centered. We aim at extending REASONX along the perspective of the Human-XAI interaction (Chromik and Butz, 2021). First, by improving the Python layer through the integration of a plotting functionality (to produce similar plots as in Section Synthetic Dataset), and additional syntactic sugar functionalities, e.g., to express a class label of the form "not class X" for the generation of contrastive rules. Second, by adding a natural language layer on top of the Python layer for an easier communication with users (see also next section). Third, by experimenting with the quality of explanations containing constraints (Byrne, 2023), and with the known implications of XAI methods, such as automation bias (Vered et al., 2023).

Social. Two basic assumptions of REASONX must be taken care of in socially-sensitive applications, where contrastive rules might be used as recommendations to change the decision (a.k.a. for *algorithmic recourse*). First, we assume that the ML model is *stable over time*, which implies that a contrastive rule may change the model's decision in future applications. While this might hold in a toy setting, it likely does not in practice and can create some wrong promises (Barocas et al., 2020; State, 2021). Using REASONX to retrospectively reason over time is a first step in the correct direction, but we cannot know how a ML model will evolve in the future. Second, by querying the minimal contrastive rules, there is the implicit assumption that the closer the result to the instance under analysis, the easier the proposed change. This does not necessarily hold in practice, as pointed out already (Karimi et al., 2023). While an extension of REASONX to account for a variety of norms in the optimization, i.e., beyond the L_1 norm and the L_∞ norm, is possible and part of future work, we acknowledge the importance of the choice of a suitable distance function, depending on the context of the application.

Concluding Remarks

We introduced REASONX (*reason to explain*), a declarative explanation tool grounded on constraint logic programming. REASONX explains decision tree models and, via surrogate trees, any tabular ML model at both the local and global level. The

tool generates explanations in a declarative and computationally principled manner, integrating user requirements and symbolic knowledge encoded in the form of linear constraints. Moreover, REASONX supports higher-level reasoning capabilities, such as reasoning over time and across model types, and can operate on data profiles rather than individual instances in under-specified information settings. REASONX produces interpretable explanations in the form of decision rules or contrastive examples, built upon a well-defined algebra of operators, and supports interactive exploration to foster enhanced communication with the user. This design makes REASONX not only a versatile and practically useful tool but also one that explicitly centers the user and her needs.

Finally, we present a fictional dialog between REASONX and a user, designed to illustrate how the system could evolve through the integration of large language models (LLMs) to bridge natural and formal languages – an emerging direction within the rapidly growing field of "LLMs for XAI" (e.g., Zytek et al. (2024); Mavrepis et al. (2024); Xu et al. (2023)). Although REASONX currently lacks natural language processing capabilities, the explanatory content presented in the dialog faithfully reflects the types of computations that the tool can already perform. Determining how to best adapt and fine-tune LLMs to improve natural language explanation quality, however, remains open for future research (Lukassen et al., 2026). The dialog is between REASONX and a developer as a user of the tool. The questions aim to provide a first understanding of the model, which is again situated in the credit domain.

USER: I have two ML models that perform similarly well on my dataset. For a single data instance, what is its classification and the intersection of the factual rules between these two models?
REASONX: Both models assign the same label to the instance (credit denial). There is no intersection between the rules.
USER: Is there a sensitive attribute appearing in the factual rules?
REASONX: No, no sensitive attribute appears in the factual rules.
USER: Is there an intersection between the contrastive rules of the two models?
REASONX: Yes, two of the constrative rules intersect. The intersection is given by the following new rule: if there was no lease on the car and if the credit requested was between $8,000$ EUR and $10,000$ EUR, the credit would have been granted in both models.
USER: Is there also a contrastive instance that is valid in both models?
REASONX: Yes, there is one contrastive instance. The changed features compared to the factual data instance are: a credit amount of $8,500$ EUR and no lease of a car.

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation program under Marie Sklodowska-Curie Actions (g.a. number 860630) for the project "NoBIAS - Artificial Intelligence without Bias", and by PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI", funded by the European Commission under the NextGeneration EU programme. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the EU. Neither the EU nor the granting authority can be held responsible for them.

References

Amgoud, L., Muller, P., and Trenquier, H. (2023). Leveraging argumentation for generating robust sample-based explanations. In *IJCAI*, pages 3104–3111. ijcai.org.

Artelt, A., Vrachimis, S. G., Eliades, D. G., Polycarpou, M. M., and Hammer, B. (2022). One explanation to rule them all - ensemble consistent explanations. *CoRR*, abs/2205.08974.

Artemov, S. and Fitting, M. (2019). *Justification Logic: Reasoning with Reasons*. Cambridge University Press.

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

Audemard, G., Lagniez, J., Marquis, P., and Szczepanski, N. (2023). On contrastive explanations for tree-based classifiers. In *ECAI*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 117–124. IOS Press.

Audemard, G., Lagniez, J., Marquis, P., and Szczepanski, N. (2024). Deriving provably correct explanations for decision trees: The impact of domain theories. In *IJCAI*, pages 3688–3696. ijcai.org.

Bagnara, R., Hill, P. M., Ricci, E., and Zaffanella, E. (2005). Not necessarily closed convex polyhedra and the double description method. *Formal Aspects Comput.*, 17(2):222–257.

Barbosa, P., Savisaar, R., and Fonseca, A. (2024). Semantically rich local dataset generation for explainable AI in genomics. In *GECCO*. ACM.

Barocas, S., Selbst, A. D., and Raghavan, M. (2020). The hidden assumptions behind counterfactual explanations and principal reasons. In *FAT**, pages 80–89. ACM.

Beckh, K., Müller, S., Jakobs, M., Toborek, V., Tan, H., Fischer, R., Welke, P., Houben, S., and von Rüden, L. (2023). Harnessing prior knowledge for explainable machine learning: An overview. In *SaTML*, pages 450–463. IEEE.

Benoy, F., King, A., and Mesnard, F. (2005). Computing convex hulls with a linear solver. *Theory Pract. Log. Program.*, 5(1-2):259–271.

Bertossi, L. E. (2023). Declarative approaches to counterfactual explanations for classification. *Theory Pract. Log. Program.*, 23(3):559–593.

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

Bhuyan, B. P., Ramdane-Cherif, A., Tomar, R., and Singh, T. P. (2024). Neuro-symbolic artificial intelligence: a survey. *Neural Comput. Appl.*, 36(21):12809–12844.

Bonfietti, A., Lombardi, M., and Milano, M. (2015). Embedding decision trees and random forests in constraint programming. In *CPAIOR*, volume 9075 of *Lecture Notes in Computer Science*, pages 74–90. Springer.

Bonsignori, V., Guidotti, R., and Monreale, A. (2021). Deriving a single interpretable model by merging tree-based classifiers. In *DS*, volume 12986 of *Lecture Notes in Computer Science*, pages 347–357. Springer.

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

Brogi, A., Mancarella, P., Pedreschi, D., and Turini, F. (1993). Theory construction in computational logic. In Jacquet, J., editor, *Constructing Logic Programs*, pages 241–250. Wiley.

Bücker, M., Szepannek, G., Gosiewska, A., and Biecek, P. (2022). Transparency, auditability, and explainability of machine learning models in credit scoring. *J. Oper. Res. Soc.*, 73(1):70–90.

Byrne, R. M. J. (2023). Good explanations in explainable artificial intelligence (XAI): evidence from human explanatory reasoning. In *IJCAI*, pages 6536–6544. ijcai.org.

Calegari, R., Ciatto, G., and Omicini, A. (2020). On the integration of symbolic and sub-symbolic techniques for XAI: A survey. *Intelligenza Artificiale*, 14(1):7–32.

Carrizosa, E., Ramírez-Ayerbe, J., and Morales, D. R. (2024). Generating collective counterfactual explanations in score-based classification via mathematical optimization. *Expert Syst. Appl.*, 238(Part E):121954.

Chang, C., Creager, E., Goldenberg, A., and Duvenaud, D. (2019). Explaining image classifiers by counterfactual generation. In *ICLR (Poster)*. OpenReview.net.

Chromik, M. and Butz, A. (2021). Human-xai interaction: A review and design principles for explanation user interfaces. In *INTERACT (2)*, volume 12933 of *Lecture Notes in Computer Science*, pages 619–640. Springer.

Cook, R. T. (2009). Intensional definition. In *A Dictionary of Philosophical Logic*, page 155. Edinburgh University Press.

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

Costa, V. G. and Pedreira, C. E. (2023). Recent advances in decision trees: an updated survey. *Artif. Intell. Rev.*, 56(5):4765–4800.

Craven, M. W. and Shavlik, J. W. (1995). Extracting tree-structured representations of trained networks. In *NIPS*, pages 24–30. MIT Press.

Cui, Z., Chen, W., He, Y., and Chen, Y. (2015). Optimal action extraction for random forests and boosted trees. In *KDD*, pages 179–188. ACM.

Demajo, L. M., Vella, V., and Dingli, A. (2020). Explainable AI for Interpretable Credit Scoring. In *Computer Science & Information Technology (CS & IT)*, pages 185–203. AIRCC Publishing Corporation.

Donadello, I. and Dragoni, M. (2020). SeXAI: Introducing concepts into black boxes for explainable Artificial Intelligence. In *XAI.it@AI*IA*, volume 2742 of *CEUR Workshop Proceedings*, pages 41–54. CEUR-WS.org.

Dudyrev, E. and Kuznetsov, S. O. (2021). Summation of decision trees. In *FCA4AI@IJCAI*, volume 2972 of *CEUR Workshop Proceedings*, pages 99–104. CEUR-WS.org.

European Union Agency for Fundamental Rights (2018). *Handbook on European non-discrimination law*.

Ferrario, A. and Loi, M. (2022). The robustness of counterfactual explanations over time. *IEEE Access*, 10:82736–82750.

Ferreira, C. A., Cantão, A. H., and Baranauskas, J. A. (2022). Decision tree induction through meta-learning. In *AIAI (2)*, volume 647 of *IFIP Advances in Information and Communication Technology*, pages 101–111. Springer.

Frank, E., Wang, Y., Inglis, S., Holmes, G., and Witten, I. H. (1998). Using model trees for classification. *Mach. Learn.*, 32(1):63–76.

Garcia-Molina, H., Ullman, J. D., and Widom, J. (2008). *Database Systems: The Complete Book*. Pearson College, 2 edition.

Goethals, S., Martens, D., and Calders, T. (2024). PreCoF: counterfactual explanations for fairness. *Mach. Learn.*, 113(5):3111–3142.

Grömping, U. (2019). South german credit data: Correcting a widely used data set. Reports in Mathematics, Physics and Chemistry, Department II, Beuth University of Applied Sciences Berlin. http://www1.beuth-hochschule.de/FB_II/reports/Report-2019-004.pdf.

Guidotti, R. (2024). Counterfactual explanations and how to find them: literature review and benchmarking. *Data Min. Knowl. Discov.*, 38(5):2770–2824.

Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., and Turini, F. (2019a). Factual and counterfactual explanations for black box decision making. *IEEE Intell. Syst.*, 34(6):14–23.

Guidotti, R., Monreale, A., Ruggieri, S., Naretto, F., Turini, F., Pedreschi, D., and Giannotti, F. (2024). Stable and actionable explanations of black-box models through factual and counterfactual rules. *Data Min. Knowl. Discov.*, 38(5):2825–2862.

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

Hüllermeier, E. and Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach. Learn.*, 110(3):457–506.

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

Izza, Y., Ignatiev, A., Stuckey, P. J., and Marques-Silva, J. (2024). Delivering inflated explanations. In *AAAI*, pages 12744–12753. AAAI Press.

Jaffar, J. and Maher, M. J. (1994). Constraint logic programming: A survey. *J. Log. Program.*, 19/20:503–581.

Jaffar, J., Maher, M. J., Marriott, K., and Stuckey, P. J. (1998). The semantics of constraint logic programs. *J. Log. Program.*, 37(1-3):1–46.

Kaggle (2025a). Default of Credit Card Clients. Accessed 2025-12.

Kaggle (2025b). Give Me Some Credit. Accessed 2025-12.

Kanamori, K., Takagi, T., Kobayashi, K., and Arimura, H. (2020). DACE: distribution-aware counterfactual explanation by mixed-integer linear optimization. In *IJCAI*, pages 2855–2862. ijcai.org.

Karimi, A., Barthe, G., Balle, B., and Valera, I. (2020). Model-agnostic counterfactual explanations for consequential decisions. In *AISTATS*, volume 108 of *Proceedings of Machine Learning Research*, pages 895–905. PMLR.

Karimi, A., Barthe, G., Schölkopf, B., and Valera, I. (2023). A survey of algorithmic recourse: Contrastive explanations and consequential recommendations. *ACM Comput. Surv.*, 55(5):95:1–95:29.

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

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York.

Keane, M. T., Kenny, E. M., Delaney, E., and Smyth, B. (2021). If only we had better counterfactual explanations: Five key deficits to rectify in the evaluation of counterfactual XAI techniques. In *IJCAI*, pages 4466–4474. ijcai.org.

Körner, P., Leuschel, M., Barbosa, J., Costa, V. S., Dahl, V., Hermenegildo, M. V., Morales, J. F., Wielemaker, J., Diaz, D., and Abreu, S. (2022). Fifty years of Prolog and beyond. *Theory Pract. Log. Program.*, 22(6):776–858.

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

Lakkaraju, H., Slack, D., Chen, Y., Tan, C., and Singh, S. (2022). Rethinking explainability as a dialogue: A practitioner's perspective. *CoRR*, abs/2202.01875.

Lampridis, O., State, L., Guidotti, R., and Ruggieri, S. (2023). Explaining short text classification with diverse synthetic exemplars and counter-exemplars. *Mach. Learn.*, 112(11):4289–4322.

Laugel, T., Jeyasothy, A., Lesot, M., Marsala, C., and Detyniecki, M. (2023). Achieving diversity in counterfactual explanations: a review and discussion. In *FAccT*, pages 1859–1869. ACM.

Lee, G. and Jaakkola, T. S. (2020). Oblique decision trees from derivatives of ReLU networks. In *ICLR*. OpenReview.net.

Lukassen, F., Herrmann, J., Weisser, C., Säfken, B., and Kneib, T. (2026). From XAI to stories: A factorial study of LLM-generated explanation quality. *CoRR*, abs/2601.02224.

Luo, J., Studer, T., and Dastani, M. (2023). Providing personalized explanations: A conversational approach. In *CLAR*, volume 14156 of *Lecture Notes in Computer Science*, pages 121–137. Springer.

Mahajan, D., Tan, C., and Sharma, A. (2019). Preserving causal constraints in counterfactual explanations for machine learning classifiers. In *CausalML: Machine Learning and Causal Inference for Improved Decision Making Workshop, NeurIPS 2019*.

Malandri, L., Mercorio, F., Mezzanzanica, M., and Seveso, A. (2024). Model-contrastive explanations through symbolic reasoning. *Decis. Support Syst.*, 176:114040.

Maréchal, A. and Périn, M. (2017). Efficient elimination of redundancies in polyhedra by raytracing. In *VMCAI*, volume 10145 of *Lecture Notes in Computer Science*, pages 367–385. Springer.

Marques-Silva, J. (2022). Logic-based explainability in machine learning. In *RW*, volume 13759 of *Lecture Notes in Computer Science*, pages 24–104. Springer.

Mavrepis, P., Makridis, G., Fatouros, G., Koukos, V., Separdani, M. M., and Kyriazis, D. (2024). XAI for all: Can large language models simplify explainable AI? *CoRR*, abs/2401.13110.

Mersha, M., Lam, K. N., Wood, J., AlShami, A. K., and Kalita, J. (2024). Explainable artificial intelligence: A survey of needs, techniques, applications, and future direction. *Neurocomputing*, 599:128111.

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

Ming, Y., Qu, H., and Bertini, E. (2019). Rulematrix: Visualizing and understanding classifiers with rules. *IEEE Trans. Vis. Comput. Graph.*, 25(1):342–352.

Minh, D., Wang, H. X., Li, Y. F., and Nguyen, T. N. (2022). Explainable artificial intelligence: A comprehensive review. *Artif. Intell. Rev.*, 55(5):3503–3568.

Molnar, C. (2019). *Interpretable machine learning*. Lulu. com.

Morettin, P., Martires, P. Z. D., Kolb, S., and Passerini, A. (2021). Hybrid probabilistic inference with logical and algebraic constraints: a survey. In *IJCAI*, pages 4533–4542. ijcai.org.

Mothilal, R. K., Sharma, A., and Tan, C. (2020). Explaining machine learning classifiers through diverse counterfactual explanations. In *FAT**, pages 607–617. ACM.

Mougan, C., Álvarez, J. M., Ruggieri, S., and Staab, S. (2023). Fairness implications of encoding protected categorical attributes. In *AIES*, pages 454–465. ACM.

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

Ntoutsi, E., Fafalios, P., Gadiraju, U., Iosifidis, V., Nejdl, W., Vidal, M., Ruggieri, S., Turini, F., Papadopoulos, S., Krasanakis, E., Kompatsiaris, I., Kinder-Kurlanda, K., Wagner, C., Karimi, F., Fernández, M., Alani, H., Berendt, B., Kruegel, T., Heinze, C., Broelemann, K., Kasneci, G., Tiropanis, T., and Staab, S. (2020). Bias in data-driven artificial intelligence systems - an introductory survey. *WIREs Data Mining Knowl. Discov.*, 10(3).

Oates, T. and Jensen, D. D. (1997). The effects of training set size on decision tree complexity. In *ICML*, pages 254–262. Morgan Kaufmann.

Panigutti, C., Perotti, A., and Pedreschi, D. (2020). Doctor XAI: an ontology-based approach to black-box sequential data classification explanations. In *FAT**, pages 629–639. ACM.

Pawelczyk, M., Bielawski, S., van den Heuvel, J., Richter, T., and Kasneci, G. (2021). CARLA: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms. In *NeurIPS Datasets and Benchmarks*.

Pawelczyk, M., Broelemann, K., and Kasneci, G. (2020). On counterfactual explanations under predictive multiplicity. In *UAI*, volume 124 of *Proceedings of Machine Learning Research*, pages 809–818. AUAI Press.

Piorkowski, D., Vejsbjerg, I., Cornec, O., Daly, E. M., and Alkan, Ö. (2023). AIMEE: an exploratory study of how rules support AI developers to explain and edit models. *Proc. ACM Hum. Comput. Interact.*, 7(CSCW2):1–25.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Ramachandranpillai, R., Baeza-Yates, R., and Heintz, F. (2025). FairXAI - A taxonomy and framework for fairness and explainability synergy in machine learning. *IEEE Trans. Neural Networks Learn. Syst.*, 36(6):9819–9836.

Rawal, K. and Lakkaraju, H. (2020). Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. In *NeurIPS*.

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

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

Rokach, L. and Maimon, O. (2005). Decision trees. In Maimon, O. and Rokach, L., editors, *The Data Mining and Knowledge Discovery Handbook*, pages 165–192. Springer.

Rong, Y., Leemann, T., Nguyen, T., Fiedler, L., Qian, P., Unhelkar, V. V., Seidel, T., Kasneci, G., and Kasneci, E. (2024). Towards human-centered explainable AI: A survey of user studies for model explanations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(4):2104–2122.

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

Ruggieri, S., Pedreschi, D., and Turini, F. (2010). Data mining for discrimination discovery. *ACM Trans. Knowl. Discov. Data*, 4(2):9:1–9:40.

Russell, C. (2019). Efficient search for diverse coherent explanations. In *FAT*, pages 20–28. ACM.

Sabbatini, F. (2025). Four decades of symbolic knowledge extraction from sub-symbolic predictors. a survey. *ACM Comput. Surv.*, 58(3):61.

Schrijver, A. (1987). *Theory of Linear and Integer Programming*. John Wiley and Sons, New York, NY.

Setzu, M., Guidotti, R., Monreale, A., Turini, F., Pedreschi, D., and Giannotti, F. (2021). GLocalX - from local to global explanations of black box AI models. *Artif. Intell.*, 294:103457.

Shi, S., Tse, R., Luo, W., D'Addona, S., and Pau, G. (2022). Machine learning-driven credit risk: A systemic review. *Neural Comput. Appl.*, 34(17):14327–14339.

Sokol, K. (2021). *Towards Intelligible and Robust Surrogate Explainers: A Decision Tree Perspective*. PhD thesis, School of Computer Science, Electrical and Electronic Engineering, and Engineering Maths, University of Bristol.

Sokol, K. and Flach, P. A. (2018). Glass-box: Explaining AI decisions with counterfactual statements through conversation with a voice-enabled virtual assistant. In *IJCAI*, pages 5868–5870. ijcai.org.

Sokol, K. and Flach, P. A. (2020a). LIMEtree: Interactively customisable explanations based on local surrogate multi-output regression trees. *CoRR*, abs/2005.01427.

Sokol, K. and Flach, P. A. (2020b). One explanation does not fit all. *Künstliche Intell.*, 34(2):235–250.

State, L. (2021). Logic programming for XAI: A technical perspective. In *ICLP Workshops*, volume 2970 of *CEUR Workshop Proceedings*. CEUR-WS.org.

State, L., Ruggieri, S., and Turini, F. (2023a). Declarative reasoning on explanations using constraint logic programming. In *JELIA*, volume 14281 of *Lecture Notes in Computer Science*, pages 132–141. Springer.

State, L., Ruggieri, S., and Turini, F. (2023b). Reason to explain: Interactive contrastive explanations (REASONX). In *xAI (1)*, volume 1901 of *Communications in Computer and Information Science*, pages 421–437. Springer.

Stepin, I., Alonso, J. M., Catalá, A., and Pereira-Fariña, M. (2021). A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, 9:11974–12001.

Takemura, A. and Inoue, K. (2024). Generating global and local explanations for tree-ensemble learning methods by answer set programming. *Theory Pract. Log. Program.*, 24(5):973–1010.

The European Union (2012). Charter of fundamental rights of the european union. OJ C C326/391. 2012/C 326/02.

Tschantz, M. C. (2022). What is proxy discrimination? In *FAccT*, pages 1993–2003. ACM.

UCI Machine Learning Repository (2025a). Adult Dataset. Accessed 2025-12.

UCI Machine Learning Repository (2025b). Australian Credit Approval. Accessed 2025-12.

UCI Machine Learning Repository (2025c). Default of Credit Card Clients. Accessed 2025-12.

UCI Machine Learning Repository (2025d). South German Credit. Accessed 2025-12.

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

van der Waa, J., Nieuwburg, E., Cremers, A. H. M., and Neerincx, M. A. (2021). Evaluating XAI: A comparison of rule-based and example-based explanations. *Artif. Intell.*, 291:103404.

Vered, M., Livni, T., Howe, P. D. L., Miller, T., and Sonenberg, L. (2023). The effects of explanations on automation bias. *Artif. Intell.*, 322:103952.

Vidal, T. and Schiffer, M. (2020). Born-again tree ensembles. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9743–9753. PMLR.

Vilone, G. and Longo, L. (2021). A quantitative evaluation of global, rule-based explanations of post-hoc, model agnostic methods. *Frontiers Artif. Intell.*, 4:717899.

Wachter, S. et al. (2017). Counterfactual explanations without opening the black box. *Harv. JL & Tech.*, 31:841.

Warren, G., Delaney, E., Guéret, C., and Keane, M. T. (2024). Explaining multiple instances counterfactually:user tests of group-counterfactuals for XAI. In *ICCBR*, volume 14775 of *Lecture Notes in Computer Science*, pages 206–222. Springer.

Weinberg, A. I. and Last, M. (2019). Selecting a representative decision tree from an ensemble of decision-tree models for fast big data classification. *J. Big Data*, 6:23.

Weld, D. S. and Bansal, G. (2019). The challenge of crafting intelligible intelligence. *Commun. ACM*, 62(6):70–79.

Wielemaker, J., Schrijvers, T., Triska, M., and Lager, T. (2012). SWI-Prolog. *Theory Pract. Log. Program.*, 12(1-2):67–96.

Wu, T., Ribeiro, M. T., Heer, J., and Weld, D. S. (2021). Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *ACL/IJCNLP (1)*, pages 6707–6723. Association for Computational Linguistics.

Xu, Y., Collenette, J., Dennis, L. A., and Dixon, C. (2023). Dialogue explanations for rule-based AI systems. In *EXTRAAMAS*, volume 14127 of *Lecture Notes in Computer Science*, pages 59–77. Springer.

Zytek, A., Pidò, S., and Veeramachaneni, K. (2024). LLMs for XAI: future directions for explaining explanations. *CoRR*, abs/2405.06064.

Šekrst, K. (2025). Unjustified untrue "beliefs": AI hallucinations and justification logics. In Świętorzecka, K., Grgić, F., and Brozek, A., editors, *Logic, Knowledge, and Tradition. Essays in Honor of Srecko Kovac*.

Supplemental material

Constraint Logic Programming and Meta-reasoning

The CLP scheme defines a family of languages that is parametric in a constraint domain (Jaffar and Maher, 1994; Körner et al., 2022). We consider the constraint domain over the reals CLP($\mathbb{R}$), and adhere to the SWI Prolog implementation (Wielemaker et al., 2012). We assume that the underlying constraint solver is ideal, that is, without rounding errors.

Syntax A term is an expression over a finite set of function symbols (lower case initial) and variables (upper case initial) and constants (0-ary function symbols). A list $[t_1, \ldots, t_n]$ is a term built by appending to the empty list $[]$ the terms t_i's using the list constructor $[h \mid t]$ where the element h is prefixed to the list t. An atom $p(t_1, \ldots, t_k)$ consists of a predicate p of arity k and terms $t_1, \ldots, t_k$, with $k \geq 0$. A CLP($\mathbb{R}$) program is a finite set of clauses of the form:

$$A \texttt{ :- } \{c\}, B_1, \ldots, B_n \texttt{ .}$$

with $n \geq 0$, where A (the head) is an atom, c a (possibly empty) linear constraint, and $B_1, \ldots, B_n$ (the body) a sequence of atoms. If $n = 0$, the clause is called a fact, and if c is empty, it is written as A . (without the :- operator). A query $\{c\}, B_1, \ldots, B_n$ consists of a linear constraint and a sequence of atoms.

Semantics The operational semantics of CLP($\mathbb{R}$) is an extension of the SLD-resolution of logic programming. We assume the (Prolog) leftmost selection rule (Jaffar et al., 1998). A state $\langle Q, cs \rangle$ consists of a query Q and a set of linear constraints cs, called the constraint store. In initial states, the constraint store is empty. A derivation is a sequence of states starting from initial ones through the following transitions:

- For constraints: $\langle (\{c\}, C_1, \ldots, C_m), cs \rangle \rightarrow \langle (C_1, \ldots, C_m), cs \cup \{c\} \rangle$ if adding the constraint c to cs yields a satisfiable constraint store, namely $\models \exists (cs \wedge c)$. If satisfiability does not occur, the derivation fails.

- For atoms: $\langle (C_1, \ldots, C_m), cs \rangle \rightarrow \langle (\{c\}, B_1, \ldots, B_n, C_2, \ldots, C_m)\theta, cs\theta \rangle$ if the atom C_1 unifies with the head of a (renamed apart) clause A :- $\{c\}, B_1, \ldots, B_n$. with the most general unifier θ. If no unification is possible, the derivation fails.

In the rewriting of atoms, more than one clause may unify, making the derivation non-deterministic: the transition diagram is a tree, which is explored depth-first, trying clauses in the order of how they appear in the program. Backtracking occurs if a failed transition is reached or if more answers are required.

A state for which no further transition is possible is of the form $\langle \emptyset, cs \rangle$, and it is called a success state. The *answer constraint* returned is the projection of cs over the variables $\mathbf{x}$ of the initial query, i.e., $\exists_{-\mathbf{x}} cs$ where $-\mathbf{x}$ denotes the variables of cs that are not in $\mathbf{x}$. Constraint stores and answer constraints are kept in a minimal representation, e.g., by removing redundant inequalities (Bagnara et al., 2005; Maréchal and Périn, 2017).

MILP optimization On top of the constraint store, CLP($\mathbb{R}$) implementations offer mixed integer linear programming (MILP) optimization functionalities. Common built-in predicates include the calculation of the supremum and the infimum of an expression w.r.t. the solutions of the constraint store. An example is the query `{X >= 1.5}, inf(X, R)`, which returns the infimum of the expression `X` in `R`, therefore `R=1.5`. We will use the predicate `bb_inf` which is similar to `inf`, but additionally accepts a list of variables in the domain of integers. For example, the query `{X >= 1.5}, bb_inf([X], X, R)`, i.e., the query previously discussed updated with `bb_inf`, now returns `R=2`.

Contrasting to Sufficient Reasons

We contrast here our notion factual constraints (see e.g., Table 2) with the one of *sufficient reasons* from Izza et al. (2022). We argue that while sufficient reasons are meaningful in the context of fully specified instances, they are not equipped to handle under-specified instances.

Fully-specified instances Sufficient reasons are defined for fully-specified instances, and they consists of a *minimal* subset of feature values of the instance to explain, such that the prediction of the DT is the same as for the instance whatever the values of the remaining features are. The general assumption here is that succinctness of explanations is a desired property, for which the decision path in a decision tree cannot lead to interpretability. However, the minimality requirement introduces notable challenges. In particular, multiple minimal subsets may exist for a single instance, leading to ambiguity in the explanation (Audemard et al., 2022). Furthermore, computing sufficient reasons is generally intractable (Cooper and Marques-Silva, 2023), limiting their practical applicability to sub-classes of decision trees. Our definition of a factual constraint reflects the intuitive decision path followed by the instance. It consists of the conjunction of split conditions (linear constraints) in the path from the root node to a leaf determined by the values of the instance. While these constraints are outputed in non-redundant form (e.g., split conditions $x \geq 0$ and $x \geq 5$ lead to factual constraint $x \geq 5$), an answer constraint is not necessarily minimal in the sense of sufficient reasons. This is because it is derived solely from the observed decision path, without considering alternative paths in the decision tree that might yield the same prediction.

Under-specified instances The notion of sufficient reasons is not defined for under-specified instances, as there is no guarantee that all possible groundings of such instances will lead to the same prediction. Consider for example the decision tree with a single split: the predicted class is $l = 1$ if $x_2 \geq 1$, and $l = 0$ otherwise. Now take the under-specified instance I_1 such that only $I_1.x_1 = 2$ is known. If we further specify $I_1.x_1 = 2, I_1.x_2 = 0$, the decision tree outputs $l = 0$. Instead, by considering $I_1.x_1 = 2, I_1.x_2 = 1$, the output is $l = 1$. This shows that multiple predictions are possible depending on how the unspecified features are instantiated. Consequently, the concept of sufficient reasons cannot be straightforwardly extended to under-specified instances, as the decision tree may yield different outcomes for different completions of the same partial input. A preliminary study on how to generalize abductive explanations is due to Izza et al. (2024), who introduce the notion of inflated explanations. Here, however, the input is still a fully specified instance. REASONX is instead able to deal with the example above by returning two answers. The first one has answer constraint $I_1.x_1 = 2, I_1.x_2 \geq 1$ and factual rule $I_1.x_2 \geq 1 \rightarrow l = 1$. The second one has answer constraint $I_1.x_1 = 2, I_1.x_2 < 1$ and factual rule $I_1.x_2 < 1 \rightarrow l = 0$.

Demonstration	Dataset	Splits	Instances	Case	Norm	ϵ	D
Synthetic dataset case	synthetic	train/test	training	(DT-M)	L_1	0	3
Adult income case (*)	ADULT	train/test	training	all	L_1	0	3
Reason over time	ADULT	50 : 50/train/test	test	(DT-M)	L_1	0.01	3
Reason over models	ADULT	train/test/expl	test/expl	(DT-M), (DT-GS)	L_1	0.01	3
Diversity optimization	ADULT	train/test/expl	test/expl	(DT-M), (DT-GS)	L_1	0.01	3
Detecting bias	ADULT	train/test	test	(DT-M)	n.a.	n.a.	5

Table 11. Overview of experimental settings for the demonstrations. For each demonstration, columns list the dataset, how it was split into sets, from which set the data instances were taken, which case of REASONX was used, the norm, the value of the parameter ϵ, and the depth of the base model D. (*) case **(DT-GS)** and **(DT-LS)** are only shown on the level of Python. n.a. = not applicable, because minimal contrastive rules were not used.

Qualitative Evaluation

The experimental settings of qualitative evaluation are reported in Table 11.

Adult Income Dataset

This example focuses on explanations for a hypothetical automated decision-making (ADM) system that assesses credit applications, for example, provided by a bank. The main purpose of the explanation is for the decision maker to make sense of the decision and for the applicant to receive information on possible opportunities for action. Secondary purposes are legal compliance of the ADM, as well as increasing the trust of the bank's stakeholders in internal decisions.

We run the example on a processed version of the Adult Income dataset (UCI Machine Learning Repository, 2025a), which has a binary response variable – whether the income of a person is $\leq$ 50K USD, or $>$ 50K USD. It contains twelve features, and we restrict ourselves to a subset: three nominal features (`workclass`, `race`, `sex`), one ordinal (`education`) and four continuous (`age`, `capitalgain`, `capitalloss`, `hoursperweek`). Nominal features are transparently one-hot encoded by the REASONX utility functions.

We split the dataset into 70% training and 30% test set. A DT is built on the training set. It has an accuracy over the test set of 83.9%. In the following, an instance is chosen from the test set with predicted income $\leq$ 50K USD.

After initialization, we turn towards a first question of the user of REASONX, in this scenario the applicant (who corresponds to the factual instance): *Why was my application rejected?* It can be answered by computing the rule behind the decision. This requires naming an instance (`F` in the code below, for "factual") and passing the feature values (`InstFeatures`) and the decision of the base model (`InstDecision`).

```
USER:    r.instance('F', features=InstFeatures, label=InstDecision)
         r.solveopt()
```

```
REASONX: Rule satisfied by F:
         IF F.capitalgain<=5119.0,F.education<=12.5,F.age<=30.5
         THEN <=50K [0.9652]
```

Listing 8: User query and answer as provided by REASONX.

The rule as returned by REASONX explains the classification of the factual instance. It refers to a specific region in the data input space as characterized by the base decision tree. A second answer can be given by comparing the factual instance against a contrastive rule, using the differences as an explanation. This requires naming a second instance (`CE` in the code below, for "contrastive example"), and possibly a minimum confidence of the rule leading to the contrastive decision. We obtain two rules by running the following query:

```
USER:    r.instance('CE', label=1-InstDecision, minconf=0.8)
         r.solveopt()
```

```
REASONX: Rule satisfied by CE:
         IF CE.capitalgain>5119.0,CE.capitalgain<=5316.5
         THEN >50K [1.0],
         Rule satisfied by CE:
         IF CE.capitalgain>7055.5,CE.age>20.0
         THEN >50K [0.9882]
```

Listing 9: User query and answer as provided by REASONX.

By comparing this output with the answer to the previous question, the user can understand the factual decision of the ADM better. This is especially relevant to answer a second question: *What are my options to change the outcome of the ADM, and receive the credit?* For example, comparing the first contrastive with the factual rule, an increase of the feature `capitalgain` will lead to a change in the predicted outcome of the ADM. Similarly, an increase in `capitalgain`, and a change in `age` (from 19 to 20 or higher) will alter it.

In the next step, we add some background knowledge to the explanations. We apply an immutability constraint on the feature `age`:

```
USER:    r.constraint('CE.age = F.age')
         r.solveopt()
```

```
REASONX: Rule satisfied by CE:
         IF CE.capitalgain>5119.0,CE.capitalgain<=5316.5
         THEN >50K [1.0]
```

Listing 10: User query and answer as provided by REASONX.

As expected, the solution has changed: by adding the above-stated constraint, the admissible region for CEs becomes smaller, and only one contrastive rule remains. Last, we ask for the CE that is *closest* to the factual instance:

```
USER:    r.solveopt(minimize='l1norm(F, CE)', project=['CE'])
```

```
REASONX: Answer constraint:
         CE.race=Black, CE.sex=Male,
         CE.workclass=Private,
         CE.education=10.0,
         CE.age=19.0,
         CE.capitalgain=5119.01,
         CE.capitalloss=0.0,
         CE.hoursperweek=40.0
```

Listing 11: User query and answer as provided by REASONX.

REASONX returned the *minimal* contrastive constraint CE. This corresponds to the most common notion of a CE in the literature. However, REASONX also took care of the specified background knowledge - the CE is returned under the above specified constraint on the feature `age`. This is a similar case as described in Section Synthetic Dataset and Figure 5 (left). We extend the example to account for under-specified information. e.g., it can be interesting to ask for the minimal CE in the case the feature `age` is not fixed, but restricted to 19 years or lower.

```
USER:    r.retract('F.age=19.0')
         r.constraint('F.age<=19.0')
         r.solveopt(minimize='l1norm(F, CE)', project=['CE', 'F.age'])
```

```
REASONX: Answer constraint:
         CE.race=Black, CE.sex=Male,
         CE.workclass=Private,
         CE.education=10.0,
         CE.age=F.age,
         CE.capitalgain=5119.01,
         CE.capitalloss=0.0,
         CE.hoursperweek=40.0
```

Listing 12: User query and answer as provided by REASONX.

The returned solution of REASONX is similar to the previous one, but we observe that also in the CE, the feature `age` is not fixed to a single value but is provided as an equality region.

Since queries work iteratively, the flow of the above corresponds exactly to how an *interaction* with REASONX could look like, forming what we call an *explanation dialogue*. Repeated specification of background knowledge and querying are a central part of this dialog between the user and REASONX, allowing to build individual, personalized explanations. A variant of this example for the cases **(DT-GS)** and **(DT-LS)** is discussed in the next section.

Global and Local Surrogate Models We extend the example from the previous section to cases **(DT-GS)** and **(DT-LS)**. We focus on the lines of code that are relevant to change and use the example of a neural network (NN) as a ML model. In cases **(DT-GS)** and **(DT-LS)** REASONX produces explanations via a global or local surrogate model, respectively. By this, any type of ML model on tabular data can be explained.

Global surrogate An extension of case **(DT-M)** to this is straightforward: what is needed for case **(DT-GS)** is to learn the base model not using the original class labels but the ML model's predicted class labels. After this, REASONX can be initialized and queried. Thus, the crucial lines are only in the Python code. They read as follows:

```
1 mlp = MLPClassifier(random_state=0)
2 mlp.fit(X1, y1)
3 mlp_label = mlp.predict(X1)
4 clf1 = DecisionTreeClassifier(max_depth=3)
5 clf1.fit(X1, mlp_label)
6 r = reasonx.ReasonX(pred_atts, target, df_code)
7 r.model(clf1)
```

Listing 13: Initialization of REASONX in case **(DT-GS)**.

In line 1-2 the NN is initialized and trained, in line 3 class labels are predicted. In line 4-5 the base model is initialized and trained with these labels, and in line 6-7 REASONX is initialized. An additional calculation of the fidelity of the surrogate model may be appropriate at this stage.

Local surrogate In case **(DT-LS)** such an extension means that *after* the instance to explain is chosen, a local neighborhood has to be created. Here, we rely on a random neighborhood generation. After all instances of that neighborhood are classified by the ML model, the base model can be learned using these neighborhood instances and their labels as predicted by the ML model. REASONX has to be instantiated with this base model, and the produced explanations are only valid for the specific instance. The important code lines are the following:

```
1 neigh = neighborhood_generation(...)
2 mlp = MLPClassifier(random_state=0)
3 mlp.fit(X1, y1)
4 neigh_mlp_label = mlp.predict(neigh)
5 clf1 = DecisionTreeClassifier(max_depth=3)
6 clf1.fit(neigh, neigh_mlp_label)
7 r = reasonx.ReasonX(pred_atts, target, df_code)
8 r.model(clf1)
```

Listing 14: Initialization of REASONX in case **(DT-LS)**.

The local neighborhood is generated in line 1, around the specified data instance and according to some neighborhood generation algorithm[**]. In line 2-3, the NN is initialized and trained, in line 4 class labels of the neighborhood are predicted. In line 5-6 the base model is initialized and trained with the neighborhood and its previously predicted labels, and in line 7-8 REASONX is initialized. Again, a fidelity check as described above, now on the data instances of the generated neighborhood, may be needed.

Diversity Optimization

In Figure 6, we compare the result of the diversity optimization (optimizing proximity *and* diversity) with the classical optimization (optimizing only proximity). The figure is commented in the main text in Section Diversity Optimization.

Detecting Biases

Explanations help users understand the reasoning behind a machine learning model's outcome. Here, we discuss how such explanations can support the detection of biases – for an in-depth discussion, cf. Ramachandranpillai et al. (2025). An important question towards this aim is: *Is a sensitive attribute (e.g., gender, race, age (The European Union, 2012)) determining the decision outcome?*[††] Computing the CE of a data instance and noting any change in the sensitive attribute is one way to answer the question. Below, we describe the experimental setup to obtain such an answer via REASONX. Our approach is close to the definition of "explicit bias" in Goethals et al. (2024): the authors define bias based on changes of the sensitive attribute between the data instance and CE. However, they focus on negative decisions and distinguish between groups when aggregating results which is different to us.

Furthermore, our approach is closely related to the legal definition of *direct discrimination* as well as to the concept of *prima facie* discrimination. While the first refers to differential

[**] While we have already integrated a simple neighborhood generation, relying on a more stable implementation such as the method used in LORE (Guidotti et al., 2019a) is a natural extension of REASONX.
[††] In this work, we use the original variable names from the Adult Income dataset, i.e., `sex` instead of `gender`.

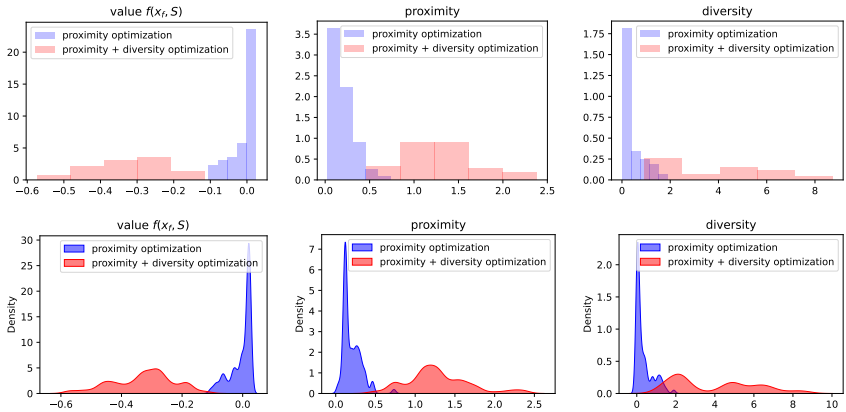

Figure 6. Diversity optimization: top row shows histograms, bottom row the density functions. The left column shows the value of the optimization function $f(x_f, S)$, the middle column the value of only the proximity term and the right column the values of only the diversity term. Color codes indicate the type of optimization.

treatment on the basis of a sensitive attribute (European Union Agency for Fundamental Rights, 2018), the second builds on the information asymmetry between those deploying the AI system and affected users: in cases of algorithmic discrimination, the party accused typically holds the majority of relevant information. A claim can therefore be established via a presumption of discrimination. The alleged defendant then has to prove that no discrimination took place or that it was justified (European Union Agency for Fundamental Rights, 2018).

Our approach is not taking into account *proxy variables* (Tschantz, 2022), nor does it refer to *indirect discrimination* (European Union Agency for Fundamental Rights, 2018).

Experimental setup. Our experiment starts from the following simplistic assumption: if the decision outcome can be altered by **only** changing a sensitive attribute[‡‡], or a set of such attributes, we take this as a *prima facie* evidence of discriminatory behavior of the ML model that must be further investigated. Here, we focus on those features that are protected by the EU law (The European Union, 2012). This assumption translates into the following experiment, testing the previous statement for one ($S = 1$) or two ($S = 2$) sensitive attributes. We first initialize REASONX with the ML model that we want to test, a data instance, and a corresponding contrastive instance. In a second step, we test whether a query for a contrastive decision rule succeeds, and under the following constraints settings: asserting a set of equality constraints to keep all features between the factual and contrastive data instance fixed and excluding one or two sensitive features. This means, we assert the constraint $F.x_i = CE.x_i$ for all x_i's excluding the features `age`, `sex` or `race`, or pairwise combinations of these three features. The features are the sensitive features identified from the processed version of the Adult Income dataset that we used already in the previous experiments. We repeat the experiment for $N - 1$ additional data instances, and compute the ratio N_s/N of the

[‡‡] Alternative names are: protected variable/feature or identity category.

model	age	race	sex	age, race	age, sex	race,sex
DT	0.01	0.00	0.00	0.01	0.01	0.00
XGB	0.12	0.00	0.11	0.12	0.26	0.11

Table 12. Fraction N_s/N of cases admitting a solution that change sensitive attributes only.

Name	N	Class ratio	$n_{nominal}$	$n_{ordinal}$	$n_{continuous}$
ADULT	$48,842$	$37,155 : 11,687, (\leq 50k :> 50k)$	3	1	4
SGC	$1,000$	$700 : 300, (1 : 0)$	0	17	3
GMSC	$150,000$	$139,974 : 10,024, (0 : 1)$	0	0	10
DCCC	$30,000$	$23,364 : 6,636, (0 : 1)$	3	0	8
ACA	690	$383 : 307, (0, 1)$	8	0	6

Table 13. Dataset summary statistics.

number N_s of successful queries (with at least one answer constraint) over the total number N of data instances. Further, after running the experiments for a DT, i.e., case **(DT-M)** and for $N = 100$, we repeat it on the XGB model ($accuracy = 0.851$, $fidelity = 0.919$ for the global surrogate) that we already used in Section Reasoning over Different Model Types of the main paper.

Results. Results of this experiment are displayed in Table 12. What we can infer is that while in the case of the DT, the sensitive attributes does not seem to have a significant influence on the outcome, both the attribute `age` and `sex` seem to have a non-negligible influence on the outcome for the XGB model. Changing only the attribute `age` alters the outcome in 12% of the cases while for the attribute `sex`, the outcome is altered in 11% of the cases. Further, if we allow both attributes to change, this ratio increases to 26%. In summary, we find different results: while in case of the DT, REASONX does not flag potential biases, it is the opposite for the XGB model explained by a global surrogate. Therefore, the XGB model should be further investigated.

Quantitative Evaluation

Dataset Information

In this work, we use the Adult Income (`ADULT`) dataset (UCI Machine Learning Repository, 2025a), the South German Credit (`SGC`) dataset (UCI Machine Learning Repository, 2025d; Grömping, 2019), the Give Me Some Credit (`GMSC`) dataset (Kaggle, 2025b), the Default of Credit Cards Clients (`DCCC`) dataset (UCI Machine Learning Repository, 2025c; Kaggle, 2025a) and the Australian Credit Approval (`ACA`) dataset (UCI Machine Learning Repository, 2025b). We display key dataset statistics in Table 13.

The preprocessing of the datasets contains several steps in accordance with the capacities of REASONX. All steps are reported in the notebooks of the released repository.

Metrics

No evaluation approach for explanation tools utilizing CLP exists so far. We therefore use some standard metrics (Vilone and Longo, 2021; Pawelczyk et al., 2021; Guidotti et al., 2019a), adapted to REASONX. We list them in Table 14. To assess the quality of the base model, we compute

Description		Notation
Metrics		
Base model	accuracy	acc
	fidelity*	f
	output ratio	S/N
Rules ($\mathbb{R}_F$ and $\mathbb{R}_C$)	rule length	l_F, l_C
	size of $\mathbb{R}_C$	N_C
Minimal contrastive	number	N_{CE}
examples/constraints (CEs)	distance	d_{CE}
	dimensionality	dim_{CE}
Parameters		
	tree depth	D
	minimum confidence factual	MC_F
	minimum confidence contrastive	MC_{CE}

Table 14. Notation for evaluation metrics and parameters of REASONX. (*) The fidelity metric applies only if there is a surrogate model, i.e., in cases **(DT-GS)** and **(DT-LS)**.

the accuracy acc, the fidelity f, and the output ratio S/N on a test set. Accuracy refers to the ratio of correctly predicted class labels of the ML model w.r.t. the original labels while fidelity refers to the ratio of correctly predicted class labels of the surrogate model w.r.t. the predicted class of the ML model (applicable only in case **(DT-GS)** and **(DT-LS)**). S/N refers to the ratio between the number of data instances for which an output could be produced, indicated by S and measured over the number of tested data instances, indicated by N. This ratio can be smaller than one if the prediction of the base model does not agree with the original label (case **(DT-M)**) or the prediction of the ML model (case **(DT-GS)** and **(DT-LS)**), or if the minimum confidence value of the factual is too high. To evaluate the decision rules, we compute their length l_F, l_C, i.e. the number of premises in the rules. Further, we count the number of contrastives rules in N_C. Contrastive examples are evaluated by three different metrics: the number of contrastive examples N_{CE}, the distance between the original and contrastive instance d_{CE} and their dimensionality dim_{CE}. These metrics are calculated w.r.t. the distance function used. The dimension dim_{CE} is obtained by parsing the results returned from Prolog for inequality operators. If such an operator is found, the dimension of the CE must be higher than zero (different to a point).

Experiments

Here, we provide the details of the experiments. For all experiments, we set $\epsilon = 0.01$ (default).

REASONX *only.* We compute all evaluation metrics for case **(DT-M)**, **(DT-GS)** and **(DT-LS)** and on five datasets in the credit domain (UCI Machine Learning Repository, 2025a,d; Kaggle, 2025b; UCI Machine Learning Repository, 2025c; Kaggle, 2025a; UCI Machine Learning Repository, 2025b), and under no user constraints. The datasets are split into training and test set with a ratio of $7:3$, the test set further divided into an explanation training set and an explanation test set with a ratio of $7:3$ for case **(DT-GS)**. In case **(DT-LS)**, we apply a split with the same conditions on the generated neighborhood. Results are averaged over $N = 100$ data instances from the test set, for $D = 3$, and for the XGB model in case **(DT-GS)** and **(DT-LS)**.

Contrastive examples. We evaluate REASONX against DiCE (Mothilal et al., 2020). Besides the case with no user constraints, we test two cases with different immutability constraints. Since the number of contrastive examples N_{CE} is a parameter in DiCE and not a metric to be measured, we run DiCE under different settings, $N_{CE} \in \{2, 3, 4, 5\}$. Further, DiCE allows to specify parameters that determine the weight of the proximity and the diversity term in the optimization. To ensure a fair comparison, we set the weight of the diversity term $w_{div} = 0$. Other parameters of DiCE are left to their default values. We transform results of DiCE using the preprocessing of REASONX and compute both norms as in REASONX. Evaluations are done on the ADULT dataset, for the XGB model, and with $D = 3$ for REASONX.

Factual rules. Here, we evaluate REASONX against ANCHORS (Ribeiro et al., 2018). ANCHORS has a parameter for the precision p_0. We vary this parameter, $p_0 \in \{0.9, 0.95, 0.99\}$. We obtain the actual precision p_c and coverage of the produced rules directly from the ANCHORS package. Other parameters for ANCHORS are left to their default values. To ensure a fair comparison to REASONX, we run our tool for different tree depths D (implying different rule lengths) and for different values of the minimum confidence of the factual rules MC_F. Results are computed under no user constraints, on the ADULT dataset, and for the XGB model. Further, they are averaged over $N = 100$ data instances.

Runtime. We evaluate the runtimes of REASONX, ANCHORS and DiCE ($w_{div} = 0$) for different parameters and under different constraint settings. Runtimes are computed as the difference between the complete explanation pipeline minus the steps before explainers are initialized (preprocessing/model learning). Evaluations are based on ADULT, computed for one data instance and averaged over 100 runs. In cases **(DT-GS)** and **(DT-LS)**, we rely on the XGB model. The runtime was computed on an Ubuntu 20.04 LTS system with $15.3GiB$ memory and $1.80GHz \times 8$ Intel Core i7.

Parameter Testing

Experiments We perform an evaluation of how the parameters of REASONX (D and MC_{CE}) influence the output of REASONX. To evaluate the dependency on the depth of the decision tree, we test for $D = 2...11$. To evaluate the dependency on the minimum confidence value of CE, we test for $MC_{CE} \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99\}$, with fixed $MC_F = 0.8$, $D = 3$. Both experiments were run on all cases, on ADULT, and for $N = 10$ instances. For cases **(DT-GS)** and **(DT-LS)**, we rely on a XGB model.

Results Results are displayed in Figure 7, 8 and 9 for cases **(DT-M)**, **(DT-GS)** and **(DT-LS)** respectively, and when varying the base model depth D. We observe the following changes when the base model depth is increasing: accuracy (or fidelity) is increasing and (in cases **(DT-M)** and **(DT-LS)**) reaching a plateau – a behavior well-known in the literature (Oates and Jensen, 1997). Rule length is also increasing. In cases **(DT-M)** and **(DT-GS)**, on average, the factual rules are longer than the contrastive rules while the factual rules are significantly shorter than the contrastive rules in case **(DT-LS)**. This last observation must be due to the quality of the local surrogate tree. Further, the number of solutions (admissible and contrastive, for both norms) increases exponentially. The distance increases and the increase is higher under the L_1 norm compared to the L_∞ norm. Finally, it is not always possible to find a solution for the tested data instances (ratio S/N), but no specific pattern can be inferred.

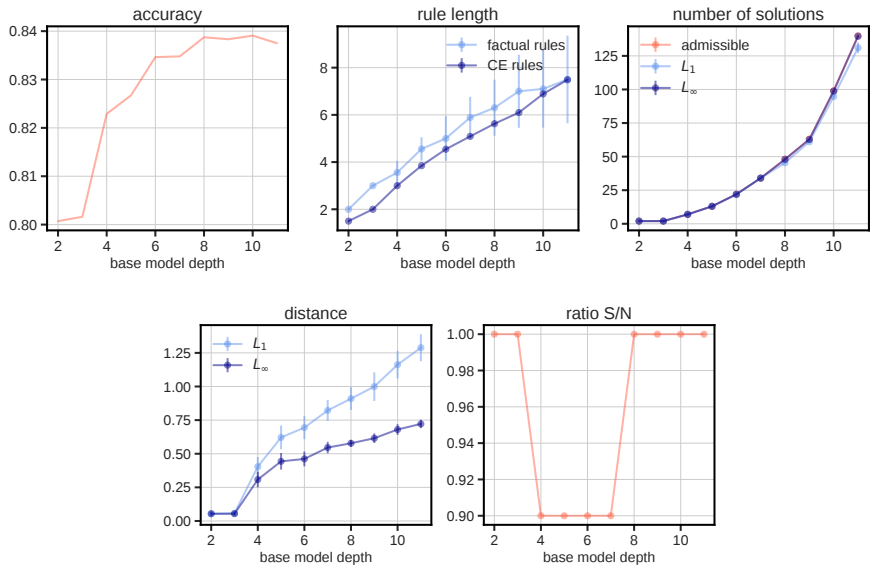

Figure 7. Parameter testing: metrics depending on depth of the base model, case **(DT-M)**.

For the minimum confidence value MC_{CE}, results are displayed in Figure 10, 11 and 12 for cases **(DT-M)**, **(DT-GS)** and **(DT-LS)** respectively. We observe the following results when increasing MC_{CE}: the number of solutions decreases for both norms. The distance also decreases but the behavior depends on the case. Distances computed w.r.t. L_1 norm are larger than w.r.t. the L_∞ norm.

Distance functions

REASONX offers two distance functions. The first one is a combination of the L_1 norm for the ordinal and continuous features and a simple matching distance for the nominal features, and it can be written as follows:

$$L_1(F, CE) = \min \sum_{i \text{ nominal}} \mathbb{1}(CE.x_i \neq F.x_i) + \sum_{i \text{ ord., cont.}} |CE.x_i - F.x_i|$$

The second one is the L_∞ norm, where nominal features are compared by simple matching. It can be written as follows:

$$L_\infty(F, CE) = \max\{ \max_{i \text{ ord., cont.}} |CE.x_i - F.x_i|, \max_{i \text{ nominal}} \mathbb{1}(CE.x_i \neq F.x_i) \}$$

For both distance functions, continuous and ordinal features are min-max normalized. Thus each feature in isolation has a maximum distance of 1, and then it contributes the same to the norm. The

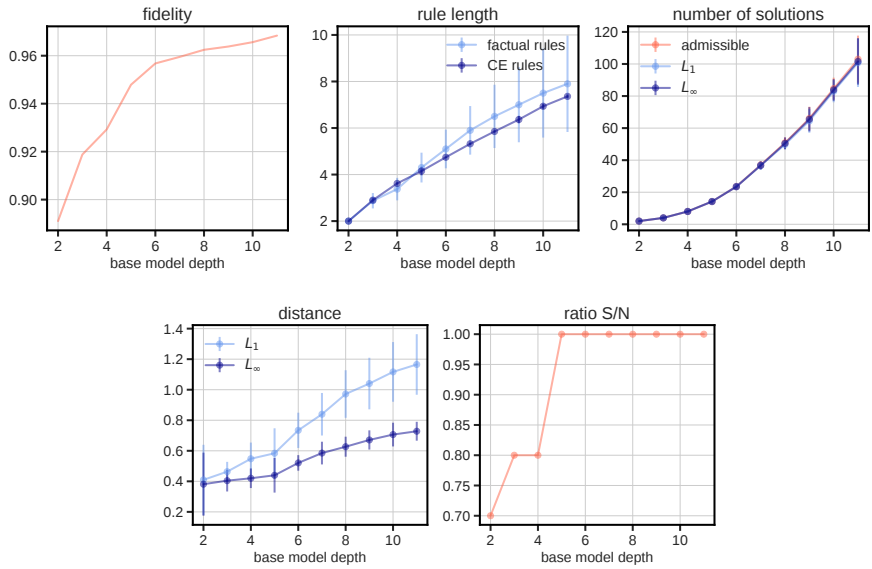

Figure 8. Parameter testing: metrics depending on depth of the base model, case **(DT-GS)**.

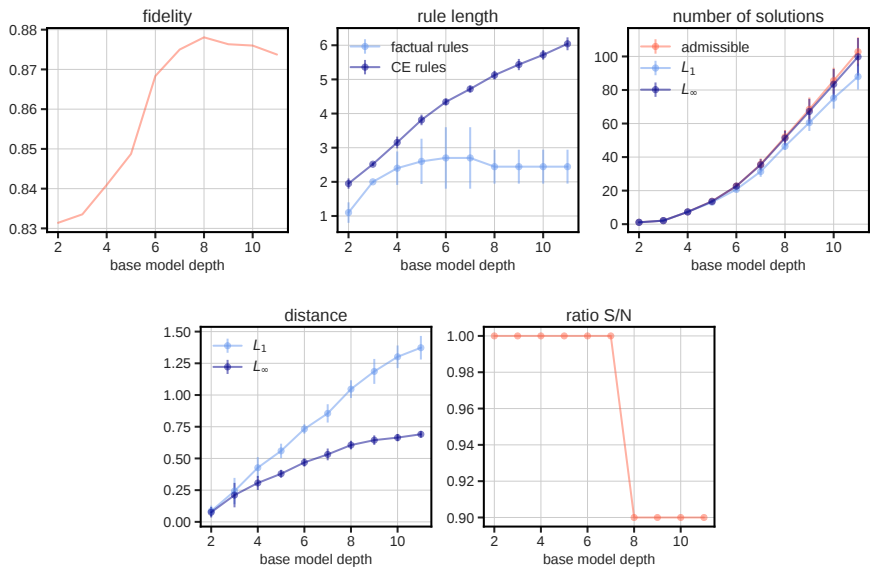

Figure 9. Parameter testing: metrics depending on depth of the base model, case **(DT-LS)**.

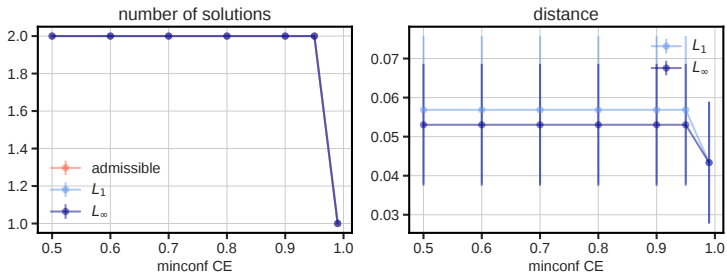

Figure 10. Parameter testing: metrics depending on minconf value of CE, case **(DT-M)**.

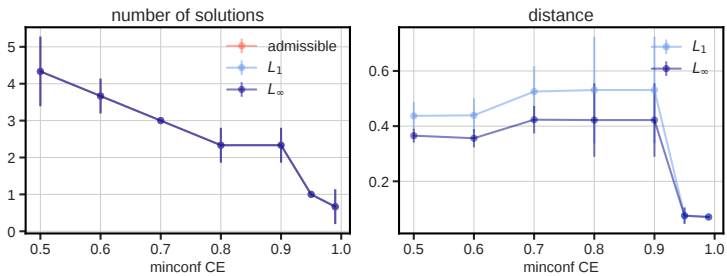

Figure 11. Parameter testing: metrics depending on minconf value of CE, case **(DT-GS)**.

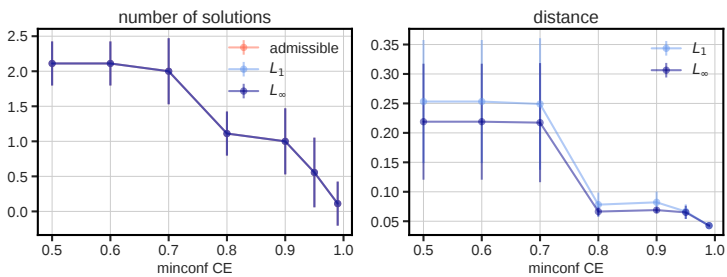

Figure 12. Parameter testing: metrics depending on minconf value of CE, case **(DT-LS)**.

L_∞ norm penalizes maximum changes over all features while the L_1 norm penalizes the average change over all features. See also Karimi et al. (2020); Wachter et al. (2017) for a discussion of how the different norms influence the outcome.

In order to solve the MILP problem of minimizing distance, we need to linearize each norm. This leads to the introduction of additional constraints and slack variables. The L_1 norm can be minimized by introducing the slack variables t_i's as follows:

$$\min \sum_i t_i \quad s.t.$$
$$CE.x_i - F.x_i \leq t_i \quad \forall i$$
$$F.x_i - CE.x_i \leq t_i \quad \forall i$$

Similarly, for the minimization of L_∞ norm, we introduce the slack variable s and define:

$$\min s \quad s.t.$$
$$CE.x_i - F.x_i \leq s \quad \forall i$$
$$F.x_i - CE.x_i \leq s \quad \forall i$$

Listing 15 report the Prolog code for the L_1 and the L_∞ norm respectively. Both predicates define a relation between the two instances (`Inst1`, `Inst2`) for which the distance is calculated, the list of the constraints that must be enforced, following the linerization above (`Cs`) and the expression to be minimized (`Norm`). The predicate `norm_weights` refers to the feature-specific weights, computed at the Python layer, such that the maximum normalized value of a features is always one. For continuous and ordinal features, this is the range of values. For nominal features (which are one-hot encoded), it is 0.5 for L_1 (because if the nominal values differ, then *two* one-hot encoded values differ) and 1 for L_∞. In both listings, line 4 considers the base case, and lines 5 - 6 the iteration over the features of the instances.

```
1  l1_con(Inst1, Inst2, Cs, Norm) :-
2      norm_weights(W),
3      l1_con(W, Inst1, Inst2, Cs, Norm).
4  l1_con([], [], [], [], 0).
5  l1_con([W|Ws], [X|Xs], [Y|Ys], [S >= X-Y, S >= Y-X|Cs], W*S+Sum) :-
6      l1_con(Ws, Xs, Ys, Cs, Sum).
7
8  linf_con(Inst1, Inst2, Cs, Norm) :-
9      norm_weights(W),
10     linf_con(W, Inst1, Inst2, Cs, Norm).
11 linf_con([], [], [], [], _).
12 linf_con([W|Ws], [X|Xs], [Y|Ys], [S >= W*(X-Y), S >= W*(Y-X)|Cs], S) :-
13     linf_con(Ws, Xs, Ys, Cs, S).
```

Listing 15: Definition of `l1_con` and of `linf_con`.