Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Main Track

A Uniform Language to Explain Decision Trees

Marcelo Arenas[1,2] , **Pablo Barceló**[1] , **Diego Bustamante**[1] , **Jose Caraball**[1] , **Bernardo Subercaseaux**[3]

[1]Pontificia Universidad Católica de Chile
[2]RelationalAI
[3]Carnegie Mellon University

Abstract

The formal XAI community has studied a plethora of interpretability queries aiming to understand the classifications made by decision trees. However, a more uniform understanding of what questions we can hope to answer about these models, traditionally deemed to be easily interpretable, has remained elusive. In an initial attempt to understand uniform languages for interpretability, Arenas et al. (2021) proposed FOIL, a logic for explaining black-box ML models, and showed that it can express a variety of interpretability queries. However, we show that FOIL is limited in two important senses: (i) it is not expressive enough to capture some crucial queries, and (ii) its model-agnostic nature results in a high computational complexity for decision trees. In this paper, we carefully craft two fragments of first-order logic that allow for efficiently interpreting decision trees: Q-DT-FOIL and its optimization variant OPT-DT-FOIL. We show that our proposed logics can express not only a variety of interpretability queries considered by previous literature but also elegantly allows users to specify different objectives the sought explanations should optimize for. Using finite model-theoretic techniques, we show that the different ingredients of Q-DT-FOIL are necessary for its expressiveness, and yet that queries in Q-DT-FOIL can be evaluated with a polynomial number of queries to a SAT solver, as well as their optimization versions in OPT-DT-FOIL. Besides our theoretical results, we provide a SAT-based implementation of the evaluation for OPT-DT-FOIL that is performant on industry-size decision trees.

1 Introduction

Formal XAI. The increasing need to comprehend the decisions made by machine learning (ML) models has fostered a large body of research in *explainable AI* (XAI) methods (Molnar, 2022), leading to the introduction of numerous queries and scores that aim to explain the predictions produced by such models. Within the wide variety of methods and subareas in XAI, our work is part of the *formal XAI* approach (Marques-Silva and Ignatiev, 2022; Darwiche, 2023; Marques-Silva, 2023), which aims to ground the study of explainability in a mathematical framework. In this line, our work leverages ideas from finite model theory (Libkin, 2004) to study the complexity and expressiveness of a fragment of first order logic tailored to explain decision trees, as well as ideas from automated reasoning to produce efficient CNF encodings for evaluating these queries through SAT solvers.

Decision Trees and Explanations. Decision trees are a very popular choice of ML models for tabular data, and one of the standard arguments in favor of their use is their supposed *interpretability* (Gunning and Aha, 2019; Molnar, 2022; Lipton, 2016). However, the formal XAI community has shown that the interpretability of decision trees is nuanced, and that even for these apparently simple models, some kinds of explanations are easy to produce while others are computationally challenging (Audemard et al., 2022b; Barceló et al., 2020; Arenas et al., 2022; Izza, Ignatiev, and Marques-Silva, 2020, 2022). Let us immediately present some examples of queries (illustrated in Figure 1) that have been considered in the literature (Darwiche and Hirth, 2020; Barceló et al., 2020; Izza, Ignatiev, and Marques-Silva, 2022).

- *Minimal/Minimum Sufficient Reasons*: Given an input instance $\mathbf{e}$ and a decision tree $\mathcal{T}$, what is the *smallest* subset S of features in $\mathbf{e}$ such that the classification $\mathcal{T}(\mathbf{e})$ is preserved regardless of the values of the features outside S? The notion of *smallest* can be defined either in terms of set containment (*minimal*) or cardinality (*minimum*).
- *Minimum Change Required/Maximum Change Allowed*: Given an input instance $\mathbf{e}$ and a decision tree $\mathcal{T}$, what is the smallest set of features that must be changed in $\mathbf{e}$ to change the classification $\mathcal{T}(\mathbf{e})$? Conversely, what is the largest set of features that can be changed in $\mathbf{e}$ without changing the classification $\mathcal{T}(\mathbf{e})$?
- *Minimal Determinant Feature Set*: Given a decision tree $\mathcal{T}$, what is the smallest set of features that, when fixed, determines the classification of any input instance?

Motivation for Interpretability Languages. The variety of interpretability queries and scores that have been proposed in the literature can be seen as a call for *interpretability languages* in which such queries could be expressed in a uniform manner. We highlight two reasons for the development of interpretability languages:

- *No Silver-Bullet Principle:* The variety of interpretability queries seems to reflect the fact that no single kind of explanation is always the best. Moreover, it is often not a single query or score, but a combination of them, that provides the best explanation (Doshi-Velez and Kim, 2017; Marques-Silva and Ignatiev, 2023). In the same line, it has

(a) Minimum Sufficient Reason (b) Minimum Change Required (c) Maximum Change Allowed (d) Determinant Feature Set

Figure 1: Illustration of different explanations for decision trees of 500 leaves over the binarized MNIST dataset (Deng, 2012). The explanations are obtained using our implementation over the OPT-DT-FOIL queries described in Section 5. Figure 1a displays a minimum sufficient reason for an image classified as `1`, where the white pixels of the original image that are part of the explanation are highlighted in perfect white, and the black pixels that are part of the explanation are highlighted in red. Arguably, this explanation reveals that the model, trained to recognize digit `1`, has learned to detect a slanted vertical stripe of white pixels, surrounded by black pixels. Figure 1b shows that adding a single white pixel to the image of a `3` is enough to change its classification (cf. *one-pixel attacks* (Su, Vargas, and Sakurai, 2019)). Interestingly, Figure 1c shows that one can simultaneously flip all pixels on an image of digit `8` while retaining its classification, showing the model is somewhat invariant to the roles of white and black pixels in the original image. Finally, Figure 1d shows that a subset of the pixels, around the center of MNIST images (colored in yellow), is enough to determine the verdicts of a model trained to detect digit `1`. As an application, one could leverage this knowledge to reduce the dimensionality of the dataset by cropping the borders.

been shown that some widely used explainability scores, believed to be theoretically mature and robust, may behave counterintuitively in certain situations (Ignatiev, Narodytska, and Marques-Silva, 2019b; Ignatiev, 2020; Camburu et al., 2019; Slack et al., 2020; Kumar et al., 2020; Huang and Marques-Silva, 2023).

- *A Uniform Understanding of Interpretability:* As posed by (Barceló et al., 2020), the computational complexity of interpretability queries on a class of models (e.g., decision trees, neural networks) can be seen as a measure of their interpretability. However, existing analyses (see also (Alfano et al., 2024; Lin et al., 2024)) rely on the particular queries being chosen. In contrast, by analyzing the complexity of evaluating interpretability queries in a uniform language, we can obtain a more general understanding of the interpretability of a class of models.

A first step toward ML interpretability languages was carried out by Arenas et al. (2021), who designed a simple explainability language based on first-order logic, called FOIL (*first-order interpretability logic*), that was able to express some basic explainability queries. However, as noted by Arenas et al. (2021), the primary purpose of FOIL was not to serve as a practical explainability language but as a foundation upon which such languages could be constructed. To date, nevertheless, we have no complete understanding of why FOIL is not a good practical language for explainability, nor what needs to be added to it in order to make it a more effective tool for performing such tasks. To gain a deeper understanding of this issue, we introduce two desiderata that any language used for explainability queries should meet:

- *Rich expressive power:* The language should be able to express a broad range of explainability queries used in practice. Some desirable characteristics in terms of expressiveness are the combination of queries (e.g., is there a *minimum sufficient reason* common to two input instances $\mathbf{e}$ and $\mathbf{e}'$?), and the possibility of expressing *preferred* explanations that contain specific features of interest to the user (Audemard et al., 2022a; Alfano et al., 2024).
- *Efficiency:* The complexity of the language used to express explainability queries must be manageable. Note that this does not necessarily imply that the evaluation should take polynomial time; SAT solvers are a mature technology that allows to solve many NP-hard problems in practice, and has been effective in computing explanations for various ML models (Izza and Marques-Silva, 2021; Yu et al., 2020; Ignatiev and Silva, 2021). In this sense, a language whose evaluation requires a small number of calls to a SAT solver can still be practical, which is the case in our work.

Theoretical Contributions. We start by assessing the suitability of FOIL as an explainability language. Regarding its expressive power, we show that there are crucial explainability queries that cannot be expressed in this language, e.g., the query *minimum sufficient reason* (Shih, Choi, and Darwiche, 2018; Barceló et al., 2020), as described earlier, cannot be expressed in FOIL. Regarding computational complexity, we show that, under some widely believed complexity assumptions, queries expressed in FOIL cannot be evaluated with a polynomial number of calls to an NP oracle. Specifically, we show that the FOIL evaluation problem over decision trees is hard for each level of the polynomial hierarchy, which goes well beyond the problems that can be solved with a polynomial number of calls to an NP oracle.

Considering these limitations, we pursue progress along two key avenues: First, we define an extension of FOIL that can capture many of the explainability notions found in practice. Then, we seek a meaningful restriction of it that maintains expressive richness while remaining compatible with evaluation via SAT solver technology.

- Regarding the extension of FOIL, we enhance it with a simple predicate $\preceq$ that enables reasoning about the cardinalities of sets of features. This addition allows us to express minimum sufficient reason and other cardinality-based inquiries like *minimum change required*.

- The high complexity of FOIL + $\{\preceq\}$ motivates the design of DT-FOIL, a similar logic to FOIL + $\{\preceq\}$ that is tailored specifically for decision trees, which makes it evaluation tractable. In particular, by *guarding quantification*, formulas in DT-FOIL can be evaluated in polynomial time. Unfortunately, this guarded quantification prevents DT-FOIL from expressing queries relating to minimization, or more in general, that require unbounded quantification. From this observation, we study two possible ways forward: (i) Q-DT-FOIL, a simple extension of DT-FOIL that allows for unbounded quantification without alternations, and (ii) OPT-DT-FOIL, an *optimization* version of DT-FOIL. We show that the evaluation problem for Q-DT-FOIL lies in the *Boolean hierarchy* (BH), thus requiring a constant number of calls to an NP oracle (e.g., a SAT solver), and that the evaluation problem for OPT-DT-FOIL is in P^{NP}, thus requiring a polynomial number of calls to an NP oracle.

Implementation. We provide a partial implementation of the evaluation of Q-DT-FOIL and OPT-DT-FOIL queries over decision trees, leveraging modern SAT-solvers and automated reasoning techniques for obtaining efficient CNF encodings. The fragment of Q-DT-FOIL and OPT-DT-FOIL queries that we can evaluate includes all examples of queries presented in this paper. The reason for our implementation being limited is that, part of the theoretical evaluation algorithm in P^{NP} for certain queries relies on a polynomial-time subroutine stemming from finite model theory whose constant factor is prohibitively large. Nonetheless, for the subset of queries that we can evaluate, we show that our implementation runs in the order of magnitude of seconds over decision trees with thousands of nodes and hundreds of features, thus making it suitable for practical use (cf. (Izza, Ignatiev, and Marques-Silva, 2020; Gomes Mantovani et al., 2024)).

Additional Material. Complete proofs for the results in this paper can be found at `https://arxiv.org/abs/2310.11636`. The repository containing the implementation is available at `https://github.com/jtcaraball/goexpdt`, and the repository with the experiments from this paper is available at `https://github.com/jtcaraball/goexpdt-experiments`.

2 Background

Models and instances. We use an abstract notion of a model of dimension n, and define it as a Boolean function $\mathcal{M} : \{0,1\}^n \to \{0,1\}$.[1] We write $\dim(\mathcal{M})$ for the dimension of a model $\mathcal{M}$. A *partial instance* of dimension n is a tuple $\mathbf{e} \in \{0,1,\bot\}^n$, where $\bot$ is used to represent undefined features. We define $\mathbf{e}_\bot = \{i \in \{1,\dots,n\} \mid \mathbf{e}[i] = \bot\}$. An *instance* of dimension n is a tuple $\mathbf{e} \in \{0,1\}^n$, that is, a partial instance without undefined features.

[1]We focus on Boolean models, which is common in formal XAI research (Wäldchen et al., 2021; Audemard et al., 2022b; Cabodi et al., 2024).

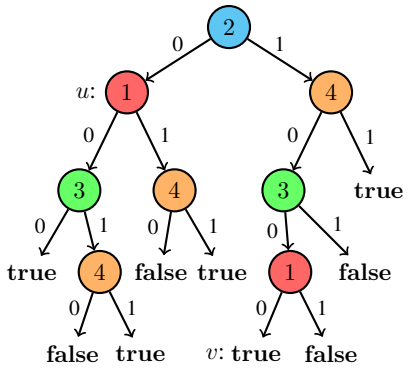

Figure 2: Example of a decision tree of dimension 4.

Given partial instances $\mathbf{e}_1$, $\mathbf{e}_2$ of dimension n, we say that $\mathbf{e}_1$ is *subsumed* by $\mathbf{e}_2$ if, and only if, $\mathbf{e}_1[i] = \mathbf{e}_2[i]$, for every $i \in \{1,\dots,n\}$ with $\mathbf{e}_1[i] \neq \bot$. That is, it is possible to obtain $\mathbf{e}_2$ from $\mathbf{e}_1$ by replacing some unknown values. For example, $(1,\bot)$ is subsumed by $(1,0)$, but it is not subsumed by $(0,0)$. A partial instance $\mathbf{e}$ can be seen as a compact representation of the set of instances $\mathbf{e}'$ such that $\mathbf{e}$ is subsumed by $\mathbf{e}'$, where such instances $\mathbf{e}'$ are called the *completions* of $\mathbf{e}$.

Decision trees. A *decision tree* over instances of dimension n is a rooted directed tree $\mathcal{T}$ with labels on edges and nodes such that: (i) each leaf is labeled with $\mathbf{true}$ or $\mathbf{false}$; (ii) each internal node (a node that is not a leaf) is labeled with a feature $i \in \{1,\dots,n\}$; (iii) each internal node has two outgoing edges, one labeled 0 and the other one labeled 1; and (iv) in every path from the root to a leaf, no two nodes on that path have the same label. Every instance $\mathbf{e} \in \{0,1\}^n$ defines a unique path $\pi_\mathbf{e} = u_1 \cdots u_k$ from the root u_1 to a leaf u_k of $\mathcal{T}$ such that: if the label of u_i is $j \in \{1,\dots,n\}$, where $i \in \{1,\dots,k-1\}$, then the edge from u_i to u_{i+1} is labeled with $\mathbf{e}[j]$. Further, the instance $\mathbf{e}$ is positive, denoted by $\mathcal{T}(\mathbf{e}) = 1$, if the label of u_k is $\mathbf{true}$; otherwise the instance $\mathbf{e}$ is negative, which is denoted by $\mathcal{T}(\mathbf{e}) = 0$. For example, for the decision tree $\mathcal{T}$ in Figure 2 and instances $\mathbf{e}_1 = (0,0,1,1)$ and $\mathbf{e}_2 = (0,1,1,0)$, it holds that $\mathcal{T}(\mathbf{e}_1) = 1$ and $\mathcal{T}(\mathbf{e}_2) = 0$.

2.1 First Order Interpretability Logic (FOIL)

Our work is inspired by the *first-order interpretability logic* (FOIL) (Arenas et al., 2021), which is a simple explainability language rooted in first-order logic. In particular, FOIL is nothing else than first-order logic over two relations on the set of partial instances of a given dimension: A unary relation POS which indicates the value of an instance in a model, and a binary relation $\subseteq$ that represents the subsumption relation among partial instances.

Given a vocabulary σ consisting of relations $R_1, \dots, R_\ell$, recall that a structure $\mathfrak{A}$ over σ consists of a domain, where quantifiers are instantiated, and an interpretation for each relation R_i. Moreover, given a first-order formula φ defined over the vocabulary σ, we write $\varphi(x_1,\dots,x_k)$ to indicate

that $\{x_1, \ldots, x_k\}$ is the set of free variables of φ. Finally, given a structure $\mathfrak{A}$ over the vocabulary σ and elements a_1, …, a_k in the domain of $\mathfrak{A}$, we use $\mathfrak{A} \models \varphi(a_1, \ldots, a_k)$ to indicate that formula φ is satisfied by $\mathfrak{A}$ when each variable x_i is replaced by element a_i ($1 \leq i \leq k$).

Consider a model $\mathcal{M}$ with $\dim(\mathcal{M}) = n$. The structure $\mathfrak{A}_\mathcal{M}$ representing $\mathcal{M}$ over the vocabulary formed by POS and $\subseteq$ is defined as follows. The domain of $\mathfrak{A}_\mathcal{M}$ is the set $\{0, 1, \bot\}^n$ of all partial instances of dimension n. An instance $\mathbf{e} \in \{0,1\}^n$ is in the interpretation of POS in $\mathfrak{A}_\mathcal{M}$ if and only if $\mathcal{M}(\mathbf{e}) = 1$, and no partial instance including undefined features is contained in the interpretation of POS. Moreover, a pair $(\mathbf{e}_1, \mathbf{e}_2)$ is in the interpretation of relation $\subseteq$ in $\mathfrak{A}_\mathcal{M}$ if and only if $\mathbf{e}_1$ is subsumed by $\mathbf{e}_2$. Finally, given a formula $\varphi(x_1, \ldots, x_k)$ in FOIL and partial instances $\mathbf{e}_1$, …, $\mathbf{e}_k$ of dimension n, model $\mathcal{M}$ is said to *satisfy* $\varphi(\mathbf{e}_1, \ldots, \mathbf{e}_k)$, denoted by $\mathcal{M} \models \varphi(\mathbf{e}_1, \ldots, \mathbf{e}_k)$, if $\mathfrak{A}_\mathcal{M} \models \varphi(\mathbf{e}_1, \ldots, \mathbf{e}_k)$.

Notice that for a decision tree $\mathcal{T}$, the structure $\mathfrak{A}_\mathcal{T}$ can be exponentially larger than $\mathcal{T}$. Hence, $\mathfrak{A}_\mathcal{T}$ is a theoretical construction needed to formally define the semantics of FOIL, but that should not be built when verifying in practice if a formula φ is satisfied by $\mathcal{T}$.

2.2 Expressing interpretability queries in FOIL

It will be instructive for the rest of our presentation, to see a few examples of how FOIL can be used to express some natural explainability queries on models. In these examples we make use of the following FOIL formula:

$$\textsc{Full}(x) \;=\; \forall y\,(x \subseteq y \;\rightarrow\; y \subseteq x).$$

Notice that if $\mathcal{M}$ is a model and $\mathbf{e}$ is a partial instance, then $\mathcal{M} \models \textsc{Full}(\mathbf{e})$ if and only if $\mathbf{e}$ is also an instance (i.e., it has no undefined features). We also use the formula

$$\textsc{AllPos}(x) \;=\; \forall y\,\big((x \subseteq y \wedge \textsc{Full}(y)) \;\rightarrow\; \textsc{Pos}(y)\big),$$

such that $\mathcal{M} \models \textsc{AllPos}(\mathbf{e})$ if and only if every completion $\mathbf{e}'$ of $\mathbf{e}$ is a positive instance of $\mathcal{M}$. Analogously, we define a formula $\textsc{AllNeg}(x)$.

A *sufficient reason* (SR) for an instance $\mathbf{e}$ over a model $\mathcal{M}$ is a partial instance $\mathbf{e}'$ such that $\mathbf{e}' \subseteq \mathbf{e}$ and each completion of $\mathbf{e}'$ takes the same value over $\mathcal{M}$ as $\mathbf{e}$. We can define SRs in FOIL as follows:

$$\textsc{SR}(x, y) = \textsc{Full}(x) \wedge y \subseteq x \wedge$$
$$(\textsc{Pos}(x) \rightarrow \textsc{AllPos}(y)) \wedge (\neg\textsc{Pos}(x) \rightarrow \textsc{AllNeg}(y)).$$

In fact, it is easy to see that $\mathcal{M} \models \textsc{SR}(\mathbf{e}, \mathbf{e}')$ if and only if $\mathbf{e}'$ is a SR for $\mathbf{e}$ over $\mathcal{M}$. Notice that $\mathbf{e}$ is always a SR for itself. However, we are typically interested in SRs that satisfy some optimality criterion. A common such criterion is that of being *minimal* (Shih, Choi, and Darwiche, 2018; Izza, Ignatiev, and Marques-Silva, 2020; Barceló et al., 2020). Formally, $\mathbf{e}'$ is a *minimal SR* for $\mathbf{e}$ over $\mathcal{M}$, if $\mathbf{e}'$ is a SR for $\mathbf{e}$ over $\mathcal{M}$ and there is no partial instance $\mathbf{e}''$ that is properly subsumed by $\mathbf{e}'$ that is also a SR for $\mathbf{e}$. Let us write $x \subset y$ for $x \subseteq y \wedge \neg(y \subseteq x)$. Then for

$$\textsc{MinimalSR}(x, y) = \textsc{SR}(x, y) \wedge \forall z\,(z \subset y \;\rightarrow\; \neg\textsc{SR}(x, z)),$$

we have that $\mathcal{M} \models \textsc{MinimalSR}(\mathbf{e}, \mathbf{e}')$ if and only if $\mathbf{e}'$ is a minimal SR for $\mathbf{e}$ over $\mathcal{M}$. Minimal SRs have also been called *prime implicants* or *abductive* explanations in the literature (Ignatiev, Narodytska, and Marques-Silva, 2019a; Marques-Silva, 2022).

A usual global interpretability question about an ML model is to decide which features are sufficient/relevant for the prediction (Huang et al., 2023; Darwiche and Hirth, 2020). In other words, which features determine the decisions made by the model. Such a notion can be defined in FOIL as follows:

$$\textsc{DFS}(x) = \forall y\big(\textsc{SUF}(x, y) \rightarrow (\textsc{AllPos}(y) \vee \textsc{AllNeg}(y))\big)$$

where $\textsc{SUF}(x, y)$ is a FOIL formula (SUF stands for *same undefined features*) such that $\mathcal{M} \models \textsc{SUF}(\mathbf{e}, \mathbf{e}')$ if and only if $\mathbf{e}_\bot = \mathbf{e}'_\bot$, i.e., the sets of undefined features in $\mathbf{e}$ and $\mathbf{e}'$ are the same (see the supplementary material for the definition of SUF). Then we have that $\mathcal{M} \models \textsc{DFS}(\mathbf{e})$ if and only if for every $\mathbf{e}'$ with $\mathbf{e}_\bot = \mathbf{e}'_\bot$, all completions of $\mathbf{e}'$ receive the same classification over $\mathcal{M}$. That is, the output of the model on each instance is invariant to the features that are undefined in $\mathbf{e}$. We call this a *Determinant Feature Set* (DFS).

As before, we can also express that $\mathbf{e}$ is *minimal* with respect to feature determinacy using the formula:

$$\textsc{MinimalDFS}(x) = \textsc{DFS}(x) \wedge \forall y\,(y \subset x \rightarrow \neg\textsc{DFS}(y)).$$

3 Limitations of FOIL

As we show in this section, FOIL fails to meet either of the two criteria we are looking for in a practical language that provides explanations about decision trees. The first issue is its limited expressivity: there are important notions of explanations that cannot be expressed in this language. The second issue is its high computational complexity: There are queries in FOIL that cannot be evaluated with a polynomial number of calls to an NP oracle.

3.1 Limited expressiveness

In some scenarios we want to express a stronger condition for SRs and DFSs: not only that they are minimal, but also that they are *minimum*. Formally, a SR $\mathbf{e}'$ for $\mathbf{e}$ over $\mathcal{M}$ is *minimum*, if there is no SR $\mathbf{e}''$ for $\mathbf{e}$ over $\mathcal{M}$ with $|\mathbf{e}''_\bot| < |\mathbf{e}'_\bot|$, i.e., $\mathbf{e}''$ has more undefined features than $\mathbf{e}'$. Analogously, we can define the notion of *minimum* DFS. One can observe that a DFS is minimum if and only if it is minimal (Shahaf Bassan, 2023). Therefore, the FOIL formula $\textsc{MinimalDFS}(x)$ presented earlier indicates that $\mathbf{e}$ is both the minimum and minimal DFS. This is however not the case for SRs; a sufficient reason can be minimal without being minimum. The following theorem shows that FOIL cannot express the query that verifies if a partial instance $\mathbf{e}'$ is a minimum SR for a given instance $\mathbf{e}$ over decision trees. Due to space constrains, see supplementary material for all proofs.

Theorem 1. *There is no formula* $\textsc{MinimumSR}(x, y)$ *in* FOIL *such that, for every decision tree* $\mathcal{T}$*, instance* $\mathbf{e}$ *and partial instance* $\mathbf{e}'$*, we have that* $\mathcal{T} \models \textsc{MinimumSR}(\mathbf{e}, \mathbf{e}') \Leftrightarrow$ $\mathbf{e}'$ *is a minimum SR for* $\mathbf{e}$ *over* $\mathcal{T}$*.*

3.2 High complexity

For each query $\varphi(x_1, \ldots, x_k)$ in FOIL, we define its associated problem $\textsc{Eval}(\varphi)$ as follows.

PROBLEM:	EVAL(φ)
INPUT:	A decision tree $\mathcal{T}$ and partial instances $\mathbf{e}_1, \ldots, \mathbf{e}_k$ of dimension n
OUTPUT:	YES, if $\mathcal{T} \models \varphi(\mathbf{e}_1, \ldots, \mathbf{e}_k)$, and NO otherwise

It is known that there exists a formula $\phi(x)$ in FOIL for which its evaluation problem over the class of decision trees is NP-hard (Arenas et al., 2021). We want to determine whether the language FOIL is appropriate for implementation using SAT encodings. Thus, it is natural to ask whether the evaluation problem for formulas in this logic can always be decided in polynomial time by using a NP oracle. However, we prove that this is not always the case. Although the evaluation of FOIL formulas is always in the polynomial hierarchy (PH), there exist formulas in FOIL for which their corresponding evaluation problems are hard for every level of PH. Based on widely held complexity assumptions, we can conclude that FOIL contains formulas whose evaluations cannot be decided in polynomial time by using a NP oracle.

Theorem 2. *The following statements hold:*

1. *Let ϕ be a* FOIL *formula. Then there exists $k \geq 0$ such that* EVAL(ϕ) *is in the Σ_k^{P} complexity class.*
2. *For every $k \geq 0$, there is an* FOIL*-formula ϕ_k such that* EVAL(ϕ_k) *is Σ_k^{P}-hard.*

4 A Better Logic to Explain Decision Trees

According to the previous section, we have two limitations regarding FOIL that we need to address in order to build a practical language to provide explanations about decision trees. This imposes two needs on us: on one hand, we must extend FOIL to increase its expressive power, and on the other hand, we must constrain the resulting language to ensure that its evaluation complexity is appropriate. In this section, we define the language DT-FOIL that takes into consideration both criteria and is specifically tailored for decision trees. We show that DT-FOIL is a natural language to express explainability notions for decision trees, which, however, lacks a general mechanism to express minimality conditions. Based on these observations, we present in the following section two extensions of DT-FOIL that are capable of expressing rich notions of explanation over decision trees, and for which the evaluation problem can be solved with a polynomial number of calls to an NP oracle.

4.1 The definition of DT-FOIL

FOIL cannot express properties such as minimum sufficient reason that involve comparing cardinalities of sets of features. As a first step, we solve this issue extending the vocabulary of FOIL with a simple binary relation $\preceq$ defined as:

$$\mathcal{M} \models \mathbf{e} \preceq \mathbf{e}' \iff |\mathbf{e}_\perp| \geq |\mathbf{e}'_\perp|.$$

As we will show later, the use of this predicate indeed allows us to express many notions of explanations. However, the inclusion of this predicate in FOIL can only add extra complexity. Therefore, our second step is to define the logic DT-FOIL, which is tailored for decision trees and can efficiently make use of this new extra power.

Atomic formulas. Predicates $\subseteq$ and $\preceq$, as well as predicates FULL and SUF used in Section 2, can be called *syntactic* in the sense that they refer to the values of the features of partial instances, and they do not make reference to classification models. It turns out that all the syntactic predicates needed in our logical formalism can be expressed as first-order queries over the predicates $\subseteq$ and $\preceq$. Moreover, such formulas can be evaluated in polynomial time:

Theorem 3. *Let ϕ be a first-order formula defined over the vocabulary $\{\subseteq, \preceq\}$. Then* EVAL(ϕ) *is in* P.

The *atomic formulas* of DT-FOIL are defined as the set of FOIL formulas over the vocabulary $\{\subseteq, \preceq\}$. Note that we could not have simply taken one of these predicates when defining atomic formulas, as we show in the supplementary material that they cannot be defined in terms of each other. The following is an example of a new atomic formula in DT-FOIL: $\textsc{Cons}(x, y) = \exists z (x \subseteq z \wedge y \subseteq z)$. This relation checks whether two partial instances $\mathbf{e}$ and $\mathbf{e}'$ are *consistent*, in the sense that features that are defined in both $\mathbf{e}$ and $\mathbf{e}'$ have the same value. We use this formula in the rest of this work.

The logic DT-FOIL. At this point, we depart from the model-agnostic approach of FOIL and introduce the concept of *guarded* quantification, which specifically applies to decision trees. This involves quantifying over the elements that define a decision tree, namely its nodes and leaves.

Given a decision tree $\mathcal{T}$ and a node u of $\mathcal{T}$, the instance $\mathbf{e}_u$ *represented* by u is defined as follows. If $\pi = u_1 \cdots u_k$ is the unique path that leads from the root of $\mathcal{T}$ to $u_k = u$, then: (i) for every $i \in \{1, \ldots, k-1\}$, if the label of node u_i is $j \in \{1, \ldots, n\}$, then $\mathbf{e}_u[j]$ is equal to the label of the edge in $\mathcal{T}$ from u_i to u_{i+1}; and (ii) for each $j \in \{1, \ldots, n\}$, $\mathbf{e}_u[j] = \perp$ if the label of u_i is different from j for every $i \in \{1, \ldots, k-1\}$. For example, for the decision tree $\mathcal{T}$ in Figure 2 and the nodes u, v shown in this figure, it holds that $\mathbf{e}_u = (\perp, 0, \perp, \perp)$ and $\mathbf{e}_v = (0, 1, 0, 0)$. Then we define predicates $\textsc{Node}(x)$ and $\textsc{PosLeaf}(x)$ as follows, given a decision tree $\mathcal{T}$ and a partial instance $\mathbf{e}$: (i) $\mathcal{T} \models \textsc{Node}(\mathbf{e})$ if and only if $\mathbf{e} = \mathbf{e}_u$ for some node $u \in \mathcal{T}$; (ii) $\mathcal{T} \models \textsc{PosLeaf}(\mathbf{e})$ if and only if $\mathbf{e} = \mathbf{e}_u$ for some leaf u of $\mathcal{T}$ with label $\mathbf{true}$. Then considering the vocabulary $\{\subseteq, \preceq, \textsc{Node}, \textsc{PosLeaf}\}$, the logic DT-FOIL is recursively defined as follows: (i) Atomic formulas are DT-FOIL formulas. (ii) DT-FOIL formulas are closed under Boolean combinations. (iii) If ϕ is a DT-FOIL formula, then $\exists x(\textsc{Node}(x) \wedge \phi)$, $\forall x(\textsc{Node}(x) \rightarrow \phi)$, $\exists x(\textsc{PosLeaf}(x) \wedge \phi)$ and $\forall x(\textsc{PosLeaf}(x) \rightarrow \phi)$ are DT-FOIL formulas.

The logic DT-FOIL is termed *guarded* due to the fact that every quantification is protected by a collection of nodes or leaves in the decision tree. As the decision tree comprises a linear number of nodes (in the size of the tree), and hence a linear number of leaves, it follows from Theorem 3 that every guarded formula can be evaluated within polynomial time.

Proposition 4. *Let ϕ be a* DT-FOIL *formula. Then* EVAL(ϕ) *is in* P.

4.2 On the expressiveness of DT-FOIL

The logic DT-FOIL allows to express in a simple way the basic notions of explanation that we study in this paper. In particular, the basic predicates that are needed to express such notions can be easily expressed using guarded quantification, considering the predicate CONS defined in Section 4.1:

$$\begin{aligned}
\textsc{Leaf}(x) &= \textsc{Node}(x) \wedge \forall y \big(\textsc{Node}(y) \rightarrow (x \subseteq y \rightarrow y \subseteq x)\big)\\
\textsc{AllPos}(x) &= \forall y \big(\textsc{Node}(y) \rightarrow \\
&\quad ((\textsc{Leaf}(y) \wedge \textsc{Cons}(x,y)) \rightarrow \textsc{PosLeaf}(y))\big)\\
\textsc{AllNeg}(x) &= \forall y \big(\textsc{Node}(y) \rightarrow \\
&\quad ((\textsc{Leaf}(y) \wedge \textsc{Cons}(x,y)) \rightarrow \textsc{NegLeaf}(y))\big)\\
\textsc{Pos}(x) &= \textsc{Full}(x) \wedge \textsc{AllPos}(x)
\end{aligned}$$

With these definitions, both predicates SR and DFS can be expressed as DT-FOIL formulas. In fact, with the new definitions of the predicates POS, ALLPOS and ALLNEG, we have that the formula in Section 2.2 defining SR is a DT-FOIL formula. For DFS the situation is a bit more complex. Observe first that we cannot use the definition of $\text{DFS}(x)$ provided in Section 2.2, as such a formula involves an unrestricted quantifier. Instead, we can use the following DT-FOIL formula $\text{DFS}(x)$:

$$\begin{aligned}
&\forall y \big[\textsc{Node}(y) \rightarrow (\textsc{AllPos}(y) \rightarrow \\
&\quad \forall z\,(\textsc{Node}(z) \rightarrow (\textsc{AllNeg}(z) \rightarrow \\
&\qquad \neg\exists w\,(\text{SUF}(x,w) \wedge \textsc{Cons}(w,y) \wedge \textsc{Cons}(w,z)))))\big].
\end{aligned} \tag{1}$$

Notice that this is a DT-FOIL formula since the formula $\neg\exists w\,(\text{SUF}(x,w) \wedge \textsc{Cons}(w,y) \wedge \textsc{Cons}(w,z))$ is atomic (that is, it is defined by using only the predicates $\subseteq$ and $\preceq$).

The logic DT-FOIL does not have an explicit mechanism to represent minimal notions of explanation. This can be proved by showing that the notion of minimum sufficient reason cannot be expressed in the logic, which is a simple corollary of Proposition 4 and the fact that the problem of verifying, given an instance $\mathbf{e}$, a partial instance $\mathbf{e}'$, and a decision tree $\mathcal{T}$, whether $\mathbf{e}'$ is a minimum sufficient reason for $\mathbf{e}$ over $\mathcal{T}$ is coNP-complete (Barceló et al., 2020).

Corollary 5. *Assuming* $\text{P} \neq \text{NP}$*, there is no formula* $\textsc{MinimumSR}(x,y)$ *in* DT-FOIL *such that, for every decision tree* $\mathcal{T}$*, instance* $\mathbf{e}$ *and partial instance* $\mathbf{e}'$*, it holds that* $\mathcal{T} \models \textsc{MinimumSR}(\mathbf{e},\mathbf{e}') \Leftrightarrow \mathbf{e}'$ *is a minimum* SR *for* $\mathbf{e}$ *over* $\mathcal{T}$*.*

This result motivates two extensions of DT-FOIL that will finally meet our desiderata for an interpretability logic, which are presented in the following section.

5 Making DT-FOIL Practical

As shown in the previous section, DT-FOIL can be evaluated in polynomial time and can express some natural explainability properties, but lacks the ability to express *optimality* properties. To remedy this, in this section we propose two extension of DT-FOIL. We start by proposing Q-DT-FOIL, a logic that is defined by allowing quantification without alternation over DT-FOIL. As we will show in this section, Q-DT-FOIL meets all the criteria stated in the introduction, except for the fact that the computation of an answer for a Q-DT-FOIL formula cannot be done with a polynomial number of calls to an NP oracle. Based on the findings, we then propose OPT-DT-FOIL, a logic that is defined by introducing a minimality operator over DT-FOIL. As we will shown in this section, OPT-DT-FOIL meets all the criteria for an appropriate interpretability logic.

5.1 The logic Q-DT-FOIL

The logic Q-DT-FOIL is recursively defined as follows: (i) each formula in DT-FOIL is a Q-DT-FOIL formula; (ii) Boolean combinations of Q-DT-FOIL formulas are Q-DT-FOIL formulas; and (iii) if ϕ is a DT-FOIL formula, then $\exists x_1 \cdots \exists x_\ell\, \phi$ and $\forall x_1 \cdots \forall x_\ell\, \phi$ are Q-DT-FOIL formulas. Both predicates SR and DFS can be expressed as Q-DT-FOIL formulas, as we shown in the previous section that they can be expressed as DT-FOIL formulas. More importantly, the form of quantification allowed in Q-DT-FOIL is enough to express optimality properties. As a first example of this, consider the following simple definition of the explainability queries studied in the paper, where $z \prec y$ is a shorthand for $z \preceq y \wedge \neg(y \preceq z)$:

$$\begin{aligned}
\textsc{MinimalSR}(x,y) &= \text{SR}(x,y) \wedge \forall z\big(z \subset y \rightarrow \neg\text{SR}(x,z)\big)\\
\textsc{MinimumSR}(x,y) &= \text{SR}(x,y) \wedge\ \forall z\big(z \prec y \rightarrow \neg\text{SR}(x,z)\big)\\
\textsc{MinimalDFS}(x) &= \text{DFS}(x) \wedge\ \forall y\big(y \subset x \rightarrow \neg\text{DFS}(y)\big)
\end{aligned}$$

As a second example of the expressiveness of Q-DT-FOIL, consider the notion of minimum change required (MCR) mentioned in the introduction. Given an instance $\mathbf{e}$ and a decision tree $\mathcal{T}$, MCR aims to find another instance $\mathbf{e}'$ such that $\mathcal{T}(\mathbf{e}) \neq \mathcal{T}(\mathbf{e}')$ and the number of features whose values need to be flipped in order to change the output of the decision tree is minimal, which is the same as saying that the Hamming distance between $\mathbf{e}$ and $\mathbf{e}'$ is minimal. It is possible to express MCR in Q-DT-FOIL as follows. In the supplementary material, we show that there exists a ternary atomic DT-FOIL formula LEH such that for every decision tree $\mathcal{T}$ of dimension n and every sequence of instances $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ of dimension n, it holds that: $\mathcal{T} \models \text{LEH}(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ if and only if the Hamming distance between $\mathbf{e}_1$ and $\mathbf{e}_2$ is less or equal than the Hamming distance between $\mathbf{e}_1$ and $\mathbf{e}_3$. By using LEH, we can express in Q-DT-FOIL the notion of minimum change required:

$$\begin{aligned}
&\textsc{MinimumCR}(x,y) = \textsc{Full}(x) \wedge \textsc{Full}(y) \wedge\\
&\quad \neg(\textsc{Pos}(x) \leftrightarrow \textsc{Pos}(y)) \wedge\\
&\quad \forall z\big[\big(\textsc{Full}(z) \wedge \neg(\textsc{Pos}(x) \leftrightarrow \textsc{Pos}(z))\big) \rightarrow \text{LEH}(x,y,z)\big].
\end{aligned}$$

The next necessary step in the study of Q-DT-FOIL is to establish the complexity of deciding whether a tuple of partial instances is an answer to a Q-DT-FOIL formula, and the complexity of computing such answers. To this aim, we first consider the evaluation problem $\textsc{Eval}(\phi)$ for a fixed Q-DT-FOIL formula $\phi(x_1, \ldots, x_m)$, which is defined exactly as in Section 3.2 for the case of FOIL. Next we provide a precise characterization of the complexity of the evaluation problem for Q-DT-FOIL. More specifically, we establish that this

problem can always be solved in the *Boolean Hierarchy over* NP (Wechsung, 1985; Cai et al., 1988), i.e., as a Boolean combination of NP problems. In this theorem, a level of the Boolean hierarchy is denoted as BH_k (the definition of this hierarchy can be found in the supplementary material).

Theorem 6. *(i) For each* Q-DT-FOIL *formula ϕ, there is $k \geq 1$ such that* $\textsc{Eval}(\phi)$ *is in* BH_k*; (ii) For every $k \geq 1$, there is a* Q-DT-FOIL *formula ϕ_k such that* $\textsc{Eval}(\phi_k)$ *is* BH_k*-hard.*

This result tells us that Q-DT-FOIL meets one of the fundamental criteria for an interpretability logic, namely that it can be verified whether a tuple is an answer to a Q-DT-FOIL formula with a polynomial number of calls to an NP oracle. Hence, the next step in the study of Q-DT-FOIL is to establish the complexity of computing such answers. For a fixed formula $\phi(x_1, \ldots, x_m)$ in Q-DT-FOIL, we define its corresponding computation problem $\textsc{Comp}(\phi)$ as the problem of computing, given decision tree $\mathcal{T}$ of dimension n, a sequence of partial instances $\mathbf{e}_1$, ..., $\mathbf{e}_m$ of dimension n such that $\mathcal{T} \models \phi(\mathbf{e}_1, \ldots, \mathbf{e}_m)$ (and answering no if such a sequence does not exist). Unfortunately, the following result tells us that this problem cannot be solved with a polynomial number of calls to an NP oracle, showing a limitation of Q-DT-FOIL when computing answers.

Theorem 7. *There exists a* Q-DT-FOIL *formula ϕ such that if* $\textsc{Comp}(\phi)$ *can be solved in* $\mathrm{FP}^{\mathrm{NP}}$*, then the polynomial hierarchy collapses to its second level, Σ_2^p.*

5.2 The logic OPT-DT-FOIL

Given what we have learned in the previous sections, our aim is to construct the right extension of DT-FOIL that meets all the criteria for an interpretability logic. This logic is OPT-DT-FOIL, which is studied in this section by defining its components, studying its expressiveness, and finally showing that the problem of computing an answer to an OPT-DT-FOIL formula can be solved with a polynomial number of calls to an NP oracle.

The definition of the logic. An atomic DT-FOIL formula $\rho(x, y, v_1, \ldots, v_\ell)$ represents a strict partial order if for every dimension n and assignment of partial instances of dimension n for the variables v_1, ..., v_ℓ, the resulting binary relation over the variables x and y is a strict partial order over the partial instances of dimension n. Formally, $\rho(x, y, v_1, \ldots, v_\ell)$ *represents a strict partial order* if for every decision tree $\mathcal{T}$:

$$\mathcal{T} \models \forall v_1 \cdots \forall v_\ell \left[\forall x \neg\rho(x, x, v_1, \ldots, v_\ell) \wedge \forall x \forall y \forall z \left((\rho(x, y, v_1, \ldots, v_\ell) \wedge \rho(y, z, v_1, \ldots, v_\ell)) \rightarrow \rho(x, z, v_1, \ldots, v_\ell)\right)\right].$$

Notice that variables v_1, ..., v_ℓ in the formula $\rho(x, y, v_1, \ldots, v_\ell)$ are considered as parameters that define a strict partial order. In fact, different assignments for these variables can give rise to different orders. Hence, we use notation $\rho[v_1, \ldots, v_\ell](x, y)$ to make explicit the distinction between the parameters v_1, ..., v_ℓ that define the order and the variables x, y that are instantiated with partial instances to be compared. For example, the strict partial order defined from the subsumption relation is defined by the formula $\rho_1(x, y) = x \subset y$. As a second example, consider the case where a certain feature must be disregarded when defining an order for partial instances. For instance, in many cases, it is not desirable to use the feature *gender* when comparing partial instances. Such an order can be defined as follows. With the appropriate values for variables v_1 and v_2, the following formula checks whether instance x has value $\perp$ in the i-th feature: $\mathrm{NF}(x, v_1, v_2) = \neg(v_1 \subseteq x) \wedge \neg(v_2 \subseteq x)$. For instance, if we are considering partial instances of dimension 5 and we need to check whether instance x has value $\perp$ in the first feature, then we can use the values $c_1 = (0, \perp, \perp, \perp, \perp)$ and $c_2 = (1, \perp, \perp, \perp, \perp)$ for the variables v_1 and v_2, respectively. Moreover, let $\mathrm{PR}(x, y) = x \subset y \wedge \neg\exists z\,(x \subset z \wedge z \subset y)$ be a formula that checks whether x is a predecessor of y under the order $\subset$. Then, with the appropriate values for the parameters v_1 and v_2, the following formula defines a strict partial order based on $\subset$ but that disregards the i-th feature when comparing partial instances:

$$\begin{aligned}\rho_2[v_1, v_2](x, y) = \exists x' \exists y' \big[&(\mathrm{NF}(x, v_1, v_2) \rightarrow x = x') \wedge \\ &(\neg\mathrm{NF}(x, v_1, v_2) \rightarrow (\mathrm{PR}(x', x) \wedge \mathrm{NF}(x', v_1, v_2))) \wedge \\ &(\neg\mathrm{NF}(y, v_1, v_2) \rightarrow (\mathrm{PR}(y', y) \wedge \mathrm{NF}(y', v_1, v_2))) \wedge \\ &(\mathrm{NF}(y, v_1, v_2) \rightarrow y = y') \wedge x' \subset y'\big]\end{aligned} \tag{2}$$

For instance, $\rho_2[c_1, c_2](x, y)$, for the constants c_1 and c_2 mentioned above, defines a strict partial order that disregards the first feature when comparing partial instances of dimension 5.

Atomic DT-FOIL formulas representing strict partial orders will be used in the definition of OPT-DT-FOIL. Hence, it is necessary to have an algorithm that verifies whether this condition is satisfied in order to have a decidable syntax for OPT-DT-FOIL. In what follows, we prove that such an algorithm exists.

Proposition 8. *The problem of verifying, given an atomic* DT-FOIL*-formula $\rho[v_1, \ldots, v_\ell](x, y)$, whether $\rho[v_1, \ldots, v_\ell](x, y)$ represents a strict partial order can be solved in double exponential time.*

Although the algorithm in Proposition 8 has high complexity, we are convinced that it can be used in practice, as we expect formulas representing strict partial orders to be small and to have a simple structure.

We need one additional piece of notation to define the syntax of OPT-DT-FOIL. Given a DT-FOIL-formula $\varphi(x, u_1, \ldots, u_k)$, we use notation $\varphi[u_1, \ldots, u_k](x)$ to indicate that x is a distinguished variable and $u_1, \ldots, u_k$ are parameters that define the possible values for x. In general, we use this syntax when x stores an explanation given an assignment for the variables u_1, ..., u_ℓ. For example, we use notation $\varphi[u](x) = \mathrm{SR}(u, x)$ to indicate that x is a sufficient reason given an assignment for the variable u (that is, x is a sufficient reason for u). We use this terminology to be consistent with the notation used for the formulas that represent strict partial orders, so that we have a consistent notation when defining OPT-DT-FOIL. Formally, given a DT-FOIL-formula $\varphi[u_1, \ldots, u_k](x)$ and an atomic DT-FOIL-formula $\rho[v_1, \ldots, v_\ell](y, z)$ that represents a strict partial order, an

OPT-DT-FOIL-formula is an expression of the following form:

$$\Psi[u_1, \ldots, u_k, v_1, \ldots, v_\ell](x) = \min[\varphi[u_1, \ldots, u_k](x), \rho[v_1, \ldots, v_\ell](y, z)].$$

Notice that x, $u_1, \ldots, u_k$, $v_1, \ldots, v_\ell$ are the free variables of this formula, while the variables y, z are quantified variables in it. In particular, x is a variable used to store a minimal explanation, $u_1, \ldots, u_k$ are the parameters that define the notion of explanation, and $v_1, \ldots, v_\ell$ are the parameters that define the strict partial order over which we are minimizing. The semantics of $\Psi[u_1, \ldots, u_k, v_1, \ldots, v_\ell](x)$ is defined by considering the following Q-DT-FOIL-formula:

$$\theta_{\min}(x, u_1, \ldots, u_k, v_1, \ldots, v_\ell) = \varphi(x, u_1, \ldots, u_k) \wedge \forall y \left(\varphi(y, u_1, \ldots, u_k) \rightarrow \neg\rho(y, x, v_1, \ldots, v_\ell)\right)$$

More precisely, given a decision tree $\mathcal{T}$ of dimension n and partial instances $\mathbf{e}, \mathbf{e}'_1, \ldots, \mathbf{e}'_k, \mathbf{e}''_1, \ldots, \mathbf{e}''_\ell$ of dimension n, we have that $\mathcal{T} \models \Psi[\mathbf{e}'_1, \ldots, \mathbf{e}'_k, \mathbf{e}''_1, \ldots, \mathbf{e}''_\ell](\mathbf{e})$ if and only if $\mathcal{T} \models \theta_{\min}(\mathbf{e}, \mathbf{e}'_1, \ldots, \mathbf{e}'_k, \mathbf{e}''_1, \ldots, \mathbf{e}''_\ell)$.

On the expressiveness of OPT-DT-FOIL. As is customary, a logic $\mathcal{L}_1$ is *contained* in a logic $\mathcal{L}_2$ if for every formula in $\mathcal{L}_1$, there exists an equivalent formula in $\mathcal{L}_2$. Moreover, $\mathcal{L}_1$ is *properly contained* in $\mathcal{L}_2$ if $\mathcal{L}_1$ is contained in $\mathcal{L}_2$ and $\mathcal{L}_2$ is not contained in $\mathcal{L}_1$. The following proposition shows that the expressive power of OPT-DT-FOIL lies between that of DT-FOIL and Q-DT-FOIL.

Proposition 9. *Assuming that the polynomial hierarchy does not collapse,* DT-FOIL *is strictly contained in* OPT-DT-FOIL*, and* OPT-DT-FOIL *is strictly contained in* Q-DT-FOIL*.*

The logic OPT-DT-FOIL allows to express in a simple way all notions of explanation that we study in this paper. For example, assuming that $\varphi[u](x) = \text{SR}(u, x)$, the following minimal OPT-DT-FOIL-formulas encode the notions of minimal and minimum sufficient reason:

$$\begin{aligned} \text{MINIMALSR}[u](x) &= \min[\varphi[u](x), y \subset z], \\ \text{MINIMUMSR}[u](x) &= \min[\varphi[u](x), y \preceq z \wedge \neg(z \preceq y)], \end{aligned}$$

while the minimal DT-FOIL-formula $\min[\varphi[u](x), \rho_2(y, z)]$ encodes the notion of minimal sufficient reason for the order $\rho_2(y, z)$ defined in eq. (2) that disregards a feature. As a second example, consider the notion of minimum change required and the predicate LEH defined in Section 5.1. Then letting $\varphi[u](x) = \text{FULL}(u) \wedge \text{FULL}(x) \wedge \neg(\text{POS}(u) \leftrightarrow \text{POS}(x))$ and $\rho_3[u](y, z) = \text{LEH}(u, y, z) \wedge \neg\text{LEH}(u, z, y)$, we can express the notion of minimum change required in OPT-DT-FOIL as follows:

$$\text{MINIMUMCR}[u](x) = \min[\varphi[u](x), \rho_3[u](y, z)].$$

The logic OPT-DT-FOIL can also be used to express notions of explanation that involve maximality conditions, just by reversing the order being considered. For example, consider the explainability query *maximum change allowed* (Alfano et al., 2024) that asks for the maximum number of changes that can be made to an instance without changing the output of the classification model. Considering $\varphi[u](x) = \text{FULL}(u) \wedge \text{FULL}(x) \wedge (\text{POS}(u) \leftrightarrow \text{POS}(x))$, and defining the reverse order $\rho_4[u](y, z) = \rho_3[u](z, y)$, we can express the notion of maximum change allowed in OPT-DT-FOIL as follows:

$$\text{MAXIMUMCA}[u](x) = \min[\varphi[u](x), \rho_4[u](y, z)].$$

An important feature of OPT-DT-FOIL is that it allows for the combination of notions of explanation. For example, given two instances u_1 and u_2, let $\text{CSR}[u_1, u_2](x) = \text{SR}(u_1, x) \wedge \text{SR}(u_2, x)$, so that this formula checks whether x is a common sufficient reason for the instances u_1 and u_2. Then it is possible to prove that the following OPT-DT-FOIL-formula computes a common minimal sufficient reason for two instances (if such a minimal sufficient reason exists):

$$\Psi_1[u_1, u_2](x) = \min[\text{CSR}[u_1, u_2](x), y \subset z].$$

Finally, another important feature of OPT-DT-FOIL is that it allows for the exploration of the space of explanations for a given classification. For example, assume that we already have a minimal sufficient reason x_1 for an instance u, which can be computed using the OPT-DT-FOIL-formula $\text{MINIMALSR}[u](x)$. Then we can compute a second minimal sufficient reason x_2 for u as follows. Let

$$\text{NSR}[u, x_1](x) = \text{SR}(u, x) \wedge \text{SR}(u, x_1) \wedge \neg(x_1 \subseteq x),$$

so this formula checks whether x is a sufficient reason for u which does not subsume sufficient reason x_1. Then a minimal sufficient reason for the instance u that is different from the minimal sufficient reason x_1 can be computed using the following OPT-DT-FOIL formula:

$$\Psi_1[u, x_1](x) = \min[\text{NSR}[u, x_1](x), y \subset z].$$

We can apply the same idea to other notions of explanation, such as the MINIMUMCR explainability query, in order to compute multiple explanations for the output of a classification model.

The computation problem. The computation problem for OPT-DT-FOIL has to be defined considering the different roles of the variables in an OPT-DT-FOIL formula $\Psi[u_1, \ldots, u_k, v_1, \ldots, v_\ell](x)$. In particular, the parameters $u_1, \ldots, u_k, v_1, \ldots, v_\ell$ should be given as input, while the value of x is the explanation to be computed. The following definition takes these considerations into account.

PROBLEM:	$\text{COMP}(\Psi)$
INPUT:	A decision tree $\mathcal{T}$ of dimension n and partial instances $\mathbf{e}'_1, \ldots, \mathbf{e}'_k, \mathbf{e}''_1, \ldots, \mathbf{e}''_\ell$ of dimension n
OUTPUT:	Partial instance $\mathbf{e}$ of dimension n such that $\mathcal{T} \models \Psi[\mathbf{e}'_1, \ldots, \mathbf{e}'_k, \mathbf{e}''_1, \ldots, \mathbf{e}''_\ell](\mathbf{e})$, and NO if no such a partial instance exists

We show that OPT-DT-FOIL meets our desiderata by proving that the computation problem for OPT-DT-FOIL can be solved with a polynomial number of calls to an NP oracle,

Theorem 10. *For every formula Ψ in* OPT-DT-FOIL*, the problem* $\text{COMP}(\Psi)$ *is in* FP^{NP}.

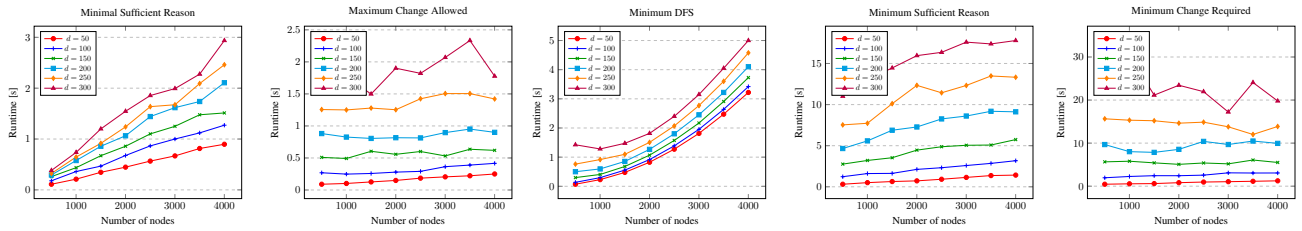

Figure 3: Empirical evaluation of the runtime for different OPT-DT-FOIL queries over random synthetic data.

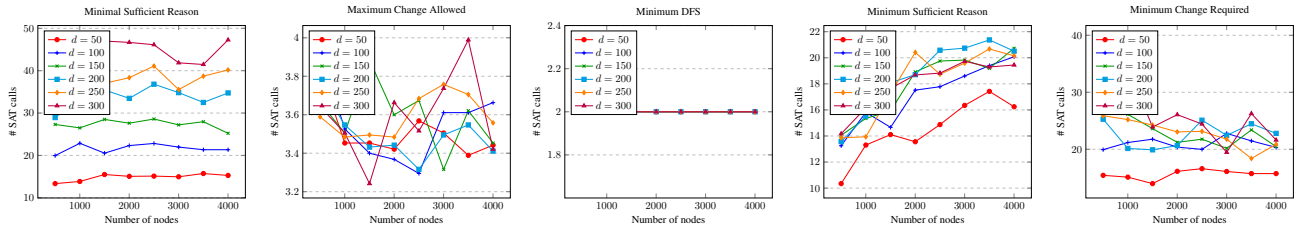

Figure 4: Empirical evaluation of the number of SAT calls for different OPT-DT-FOIL queries over random synthetic data.

6 Implementation and Challenges

Our implementation consists of three main components: (i) a prototype *simplifier* for Q-DT-FOIL/OPT-DT-FOIL formulas, (ii) an encoder translating DT-FOIL formulas into CNF formulas, and (iii) the algorithms to either compute answers for OPT-DT-FOIL queries, or decide the truth value of Q-DT-FOIL formulas. We will give a high-level explanation and also present some key experiments–the supplementary material contains additional details.

Simplifier. Logical connectives can significantly increase the size of our resulting CNF formulas; consider for example the formula: $\varphi(y) = \exists x\,[\neg(\neg(\neg(\neg(\neg(x \subseteq y))))) \vee (1,0,\bot) \subseteq (1,1,1)]$. It is clear that double-negations can be safely eliminated, and also that sub-expressions involving only constants can be pre-processed and also eliminated, thus resulting in the simplified formula $\varphi(y) = \exists x\,\neg(x \subseteq y)$.

Encoder. We use standard encoding techniques for SAT-solving, for which we refer the reader to the *Handbook of Satisfiability* (Biere et al., 2009, 2021). The basic variables of our propositional encoding are of the form $v_{x,i,s}$, indicating that the DT-FOIL variable x has value s in its i-th feature, with $i \in \{1,\dots,\dim(\mathcal{T})\}$, for an input decision tree $\mathcal{T}$, and $s \in \{0,1,\bot\}$. Then, the clauses (and further auxiliary variables) are mainly built on two layers of abstraction: a *circuit layer*, and a *first-order layer*. The circuit layer consists of individual ad-hoc encodings for each of the predicates and shorthands that appear frequently in queries, such as $\subseteq$, $\preceq$, LEH, DFS, CONS, FULL, ALLPOS and ALLNEG. The first-order layer consists of encoding the logical connectives $(\neg, \vee, \wedge)$ as well as the quantifiers (with the corresponding NODE and POSLEAF guards when appropriate).

For two interesting examples of encoding the circuits, let us consider $\preceq$ and ALLPOS. For ALLPOS, we use a *reachability* encoding, in which we create variables $r_{x,u}$ to represent that a node u of $\mathcal{T}$ is *reachable* by a partial instance y subsuming x. We start by enforcing that $r_{x,\text{root}(\mathcal{T})}$ is set to true, to then propagate the reachability from every node u to its children $u \to 0$ and $u \to 1$ depending on the value of $x[a]$ with a the label of u. Finally, by adding unit clauses stating that $(\neg r_{x,\ell})$ for every **false** leaf ℓ, we have encoded that no instance y subsuming x reaches a false leaf, and thus is a positive instance. For the case of $\preceq$, we can see it as a pseudo-Boolean constraint and implement it by leveraging the counting variables of the *sequential encoder* of Sinz (2005) (cf. (Biere et al., 2021)), thus amounting to a total of $O(\dim(\mathcal{T})^2)$ auxiliary variables and clauses.

For the first-order layer, we implement the Tseitin transformation (Tseitin, 1968) to more efficiently handle $\neg$ and $\vee$, while treating the guarded-$\forall$ as a conjunction over the $O(|\mathcal{T}|)$ partial instances for which either NODE$(\cdot)$ or POSLEAF$(\cdot)$ holds, which can be pre-computed from $\mathcal{T}$. An interesting problem that arises when handling negations or disjunctions is that of *consistency constraints*, e.g., for each $i \in \{1,\dots,\dim(\mathcal{T})\}$, the clause $(v_{x,i,\bot} \vee v_{x,i,0} \vee v_{x,i,1})$ should be true. To address this, we partition the clauses of our encoding into two sets: CONSISTENCYCLS (consistency clauses) and SEMANTICCLS (semantic clauses), so that logical connectives operate only over SEMANTICCLS, preserving the internal consistency of our variables, both the original and auxiliary ones.

Algorithms for OPT-DT-FOIL and Q-DT-FOIL. In a nutshell, to compute the answer for an OPT-DT-FOIL query

$$\Psi(x) = \min[\varphi[\mathbf{e}'_1,\dots,\mathbf{e}'_k](x), \rho[\mathbf{e}''_1,\dots,\mathbf{e}''_\ell](y,z)],$$

we first find, through a single SAT call, a partial instance $\mathbf{e}$ that satisfies $\varphi[\mathbf{e}'_1,\dots,\mathbf{e}'_k](\mathbf{e})$. Letting $\mathbf{e}^{(0)} := \mathbf{e}$ be the obtained partial instance, we will iteratively search for a *smaller* (according to ρ) partial instance $\mathbf{e}^{(i)}$ that satisfies $\varphi[\mathbf{e}'_1,\dots,\mathbf{e}'_k](\mathbf{e}^{(i)}) \wedge \rho[\mathbf{e}''_1,\dots,\mathbf{e}''_\ell](\mathbf{e}^{(i)},\mathbf{e}^{(i-1)})$. Whenever such a partial instance $\mathbf{e}^{(i)}$ is not found, we conclude that

$\mathbf{e}^{(i-1)}$ is the answer. The proof of Theorem 10 ensures the number of iterations of this process is polynomial in $\dim(\mathcal{T})$.

In the case of a Q-DT-FOIL query, which can be in turn a Boolean combination of smaller Q-DT-FOIL formulas, we recursively evaluate each of the sub-formulas and then combine the results according to the logical connectives. The base case of the recursion corresponds to DT-FOIL formulas, potentially with a single kind of quantifier, whose truth value requires solving a single CNF formula.

Results, challenges, and next steps. Our implementation can handle all queries considered in this paper and provides an elegant way to specify further queries. We have tested our implementation on a variety of decision trees and found that it scales up to thousands of nodes and hundreds of features, as shown in Figure 3. The difference in scaling behaviors across the different queries match theoretical expectations; for example, for minimal sufficient reasons, each SAT call is refining the answer by removing 1 additional feature, thus leading to a large number of calls, each of which is very cheap. In contrast, for computing minimum DFSs, those tend to have almost all features in the dataset, meaning that generally at most 1 feature can be ignored, which is always detected in 2 SAT calls. Similarly, for the maximum change allowed query, the initial solution changes many features, and that is iteratively reduced until the minimum is found using dozens of calls, whereas for the maximum change allowed, the initial solution usually changes a single feature of the input instance, and this is iteratively increased. Both of these cases are consistent with the theoretical expectation of low "robustness" on a decision tree trained on random data, where the classification of instances can change by changing a small number of features, and cannot be maintained by changing a large number of them. All experiments were run on a personal computer: AMD Ryzen 7 7800X3D CPU with 32GB of RAM, running on a Debian distribution. We have used the award-winning solver `Kissat` (Biere et al., 2020). Moreover, a comprehensive suite of tests validates our implementation, with over 2600 unit and integration tests. Nonetheless, our implementation cannot be considered complete for OPT-DT-FOIL or Q-DT-FOIL; a main challenge is that, even though atomic DT-FOIL queries can be evaluated in polynomial time (cf. Theorem 3), we do not know of an efficient propositional encoding for them, as the algorithm provided by the proof of Theorem 3 is only of theoretical interest. Naturally, some concrete atomic DT-FOIL queries can be encoded efficiently, as we have done for, e.g., LEH, CONS.

In terms of future work, two roads could make our implementation more suitable for practice: (i) extending our logic to handle multi-class trees with numerical features, and (ii) using incremental SAT-solving techniques (Nadel, Ryvchin, and Strichman, 2014) to speed up the minimization algorithm for OPT-DT-FOIL.

Acknowledgements

Bustamante is funded by ANID - Subdirección de Capital Humano (Magíster Nacional, 2023, folio 22231282). Arenas is funded by ANID - Millennium Science Initiative Program - Code ICN17002. Barceló is funded by ANID - Millennium Science Initiative Program - Code ICN17002 and by the National Center for Artificial Intelligence CENIA FB210017, Basal ANID. Part of this work was done when Arenas and Barceló were visiting the Simons Institute for the Theory of Computing. Subercaseaux is supported by the U.S. National Science Foundation under grant CCF-2229099.

References

Alfano, G.; Greco, S.; Mandaglio, D.; Parisi, F.; Shahbazian, R.; and Trubitsyna, I. 2024. Even-if explanations: Formal foundations, priorities and complexity.

Arenas, M.; Baez, D.; Barceló, P.; Pérez, J.; and Subercaseaux, B. 2021. Foundations of symbolic languages for model interpretability. In *NeurIPS 2021*, 11690–11701.

Arenas, M.; Barceló, P.; Romero Orth, M.; and Subercaseaux, B. 2022. On computing probabilistic explanations for decision trees. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 28695–28707. Curran Associates, Inc.

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

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.-M.; and Marquis, P. 2022b. On the explanatory power of Boolean decision trees. *Data Knowl. Eng.* 142(C).

Barceló, P.; Monet, M.; Pérez, J.; and Subercaseaux, B. 2020. Model interpretability through the lens of computational complexity. In *Advances in Neural Information Processing Systems*, volume 33, 15487–15498.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T. 2009. *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. NLD: IOS Press.

Biere, A.; Fazekas, K.; Fleury, M.; and Heisinger, M. 2020. Cadical, kissat, paracooba, plingeling and treengeling entering the SAT competition 2020. In Balyo, T.; Froleyks, N.; Heule, M.; Iser, M.; Järvisalo, M.; and Suda, M., eds., *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, 51–53. University of Helsinki.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2021. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Cabodi, G.; Camurati, P. E.; Marques-Silva, J.; Palena, M.; and Pasini, P. 2024. Optimizing binary decision diagrams for interpretable machine learning classification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 1–1.

Cai, J.; Gundermann, T.; Hartmanis, J.; Hemachandra, L. A.; Sewelson, V.; Wagner, K. W.; and Wechsung, G. 1988. The Boolean hierarchy I: structural properties. *SIAM J. Comput.* 17(6):1232–1252.

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

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

Darwiche, A. 2023. Logic for explainable AI.

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

Doshi-Velez, F., and Kim, B. 2017. Towards a rigorous science of interpretable machine learning.

Gomes Mantovani, R.; Horváth, T.; Rossi, A. L. D.; Cerri, R.; Barbon Junior, S.; Vanschoren, J.; and de Carvalho, A. C. P. L. F. 2024. Better trees: An empirical study on hyperparameter tuning of classification decision tree induction algorithms. *Data Mining and Knowledge Discovery*.

Gunning, D., and Aha, D. 2019. DARPA's Explainable Artificial Intelligence (XAI) Program. *AI Magazine* 40(2):44–58.

Huang, X., and Marques-Silva, J. 2023. The inadequacy of shapley values for explainability.

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

Ignatiev, A., and Silva, J. P. M. 2021. SAT-based rigorous explanations for decision lists. In Li, C., and Manyà, F., eds., *SAT*, volume 12831 of *LNCS*, 251–269. Springer.

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

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

Ignatiev, A. 2020. Towards trustable explainable AI. In Bessiere, C., ed., *IJCAI*, 5154–5158. ijcai.org.

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

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

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

Kumar, I. E.; Venkatasubramanian, S.; Scheidegger, C.; and Friedler, S. A. 2020. Problems with shapley-value-based explanations as feature importance measures. In *ICML*, 5491–5500.

Libkin, L. 2004. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer.

Lin, A. W.; Schrader, M.; Künnemann, M.; and Jaipuriyar, P. 2024. Complexity of formal explainability for sequential models.

Lipton, Z. C. 2016. The mythos of model interpretability. *CoRR* abs/1606.03490.

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

Marques-Silva, J., and Ignatiev, A. 2023. No silver bullet: interpretable ML models must be explained. *Frontiers in Artificial Intelligence* 6:1128212.

Marques-Silva, J. 2022. Logic-based explainability in machine learning. *CoRR* abs/2211.00541.

Marques-Silva, J. 2023. Logic-based explainability in machine learning.

Molnar, C. 2022. *Interpretable Machine Learning*. 2 edition.

Nadel, A.; Ryvchin, V.; and Strichman, O. 2014. Ultimately Incremental SAT. In Sinz, C., and Egly, U., eds., *Theory and Applications of Satisfiability Testing – SAT 2014*, 206–218. Cham: Springer International Publishing.

Shahaf Bassan, Guy Amir, G. K. 2023. Local vs. global interpretability: A computational perspective. *openreview.net*.

Shih, A.; Choi, A.; and Darwiche, A. 2018. A symbolic approach to explaining bayesian network classifiers. *arXiv preprint arXiv:1805.03364*.

Sinz, C. 2005. Towards an optimal CNF encoding of Boolean cardinality constraints. In van Beek, P., ed., *Principles and Practice of Constraint Programming - CP 2005*, 827–831. Berlin, Heidelberg: Springer Berlin Heidelberg.

Slack, D.; Hilgard, S.; Jia, E.; Singh, S.; and Lakkaraju, H. 2020. Fooling LIME and SHAP: adversarial attacks on post hoc explanation methods. In Markham, A. N.; Powles, J.; Walsh, T.; and Washington, A. L., eds., *AIES*, 180–186.

Su, J.; Vargas, D. V.; and Sakurai, K. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23(5):828–841.

Tseitin, G. S. 1968. On the complexity of derivation in propositional calculus. In *Studies in Mathematics and Mathematical Logic 2*, 115–125.

Wäldchen, S.; MacDonald, J.; Hauch, S.; and Kutyniok, G. 2021. The computational complexity of understanding binary classifier decisions. *J. Artif. Intell. Res.* 70:351–387.

Wechsung, G. 1985. On the Boolean closure of NP. In *Fundamentals of Computation Theory, FCT '85*, volume 199 of *Lecture Notes in Computer Science*, 485–493.

Yu, J.; Ignatiev, A.; Stuckey, P. J.; and Bodic, P. L. 2020. Computing optimal decision sets with SAT. In Simonis, H., ed., *CP*, volume 12333 of *LNCS*, 952–970. Springer.