Computing Coverage-Based Prime Implicant Explanations for Tree-Based Models

Gilles Audemard[1] , **Sylvie Coste-Marquis**[1] , **Pierre Marquis**[1,2] , **Mehdi Sabiri**[1] , **Nicolas Szczepanski**[1]

[1]Univ. Artois, CNRS, CRIL, Lens, France
[2]Institut Universitaire de France, France
name@cril.fr

Abstract

Coverage-based prime implicant explanations are formal explanations offering a number of valuable assets, especially in terms of faithfulness and generality. Unfortunately, deriving a coverage-based prime implicant explanation for an instance is computationally hard in the general case (the problem of identifying such an explanation being at the second level of the polynomial hierarchy). In this paper, we focus on the computation of a coverage-based prime implicant explanation for an instance given a tree-based model. We show that the specific nature of the domain theory linking the Boolean conditions in such models makes the problem computationally easier. We present a greedy algorithm to derive coverage-based prime implicant explanations when dealing with a tree-based model. We also present an empirical evaluation showing that this algorithm is efficient enough to be used in many practical cases.

1 Introduction

During the last decade, the rise of ML-driven AI techniques has led to outstanding advancements across numerous fields. However, the most accurate ML models are black boxes: their decision-making processes are opaque [Molnar, 2019]. To deal with the demand for more transparency in AI systems, eXplainable AI (XAI) methods have been developed in recent years [Gunning, 2019]. In particular, many notions of explanations suited to various ML models and inputs (tabular data, images, texts, etc.) have been pointed out so far (see [Hsieh *et al.*, 2024; Bodria *et al.*, 2023; Arrieta *et al.*, 2020; Guidotti *et al.*, 2019; Molnar, 2019] for surveys).

In the following, we focus on *coverage-based prime implicant explanations* [Cooper and Amgoud, 2024; Cooper and Amgoud, 2023] in the specific case of tree-based models f used for binary classification. Coverage-based prime implicant explanations are abductive explanations that are local and post-hoc explanations. As such, they are about the prediction made by an already trained ML model f for a given input instance $\boldsymbol{x}$. They aim to answer the "why?" question [Miller, 2019; Ignatiev *et al.*, 2020] by identifying a subset t of the characteristics of $\boldsymbol{x}$ such that every instance sharing those characteristics is ensured to be classified by the ML model f in the same way as $\boldsymbol{x}$.

While no consensus exists about what a "good" explanation should be [Miller, 2019; Molnar, 2019], many criteria can be considered for evaluating the quality of an explanation (see e.g., [Nauta *et al.*, 2023]). Coverage-based prime implicant explanations [Cooper and Amgoud, 2024; Cooper and Amgoud, 2023] are faithful explanations that take advantage of a domain theory Σ^f not only for ensuring that explanations comply with it, but also for the sake of generality: only explanations that are as general as possible given Σ^f, i.e., that are implied by other explanations when Σ^f is assumed, are selected. [Cooper and Amgoud, 2024; Cooper and Amgoud, 2023] have shown that ignoring some consequences of Σ^f (especially, dependency constraints among features) may lead to the generation of superfluous abductive explanations and to exponentially many redundant abductive explanations. Coverage-based prime implicant explanations do not suffer from these impediments.

However, there is a price to be paid, and this price is computational: though deciding whether a given term is an abductive explanation is coNP-complete [Cooper and Marques-Silva, 2023] in general, deciding whether it is a coverage-based prime implicant explanation is Π_2^p-complete in general [Cooper and Amgoud, 2024; Cooper and Amgoud, 2023]. While an algorithm for computing a coverage-based prime implicant explanation for a given instance has been presented in the appendix of [Cooper and Amgoud, 2024], as far as we know, this algorithm has not been implemented so far. As a consequence, given the high complexity of the problem, the question of the practical efficiency of the computation of such explanations remains open.

In this paper, we *present a greedy algorithm for computing coverage-based prime implicant explanations when dealing with a tree-based model* f (a decision tree [Breiman *et al.*, 1984], a random forest [Breiman, 2001] or a boosted tree [Freund and Schapire, 1997]). We show that the specific nature of the domain theory linking the Boolean conditions in such models makes the problem computationally easier (only polynomially many calls to an NP oracle are needed). We also present an empirical evaluation of this algorithm and a comparison with our implementation of the algorithm given in [Cooper and Amgoud, 2024]. This evaluation shows that

the computation of coverage-based prime implicant explanations for tree-based models is practical enough to deal with "mildly complex" tree-based models (i.e., based on hundreds of Boolean conditions). As expected, experiments also show that our algorithm is much more efficient than the general-purpose algorithm for coverage-based prime implicant explanations presented in [Cooper and Amgoud, 2024] when applied to such tree-based models.

The rest of the paper is organized as follows. Formal preliminaries are given in Section 2. Our algorithm for computing coverage-based prime implicant explanations for tree-based models is presented in Section 3. Experiments are reported in Section 4. The proofs of the results presented in the paper, as well as additional empirical results and the code used in our experiments are furnished as a supplementary material available from [Audemard *et al.*, 2026].

2 Formal Preliminaries

Classification and tree-based models. Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be a finite set of attributes, where each attribute is Boolean, categorical, or numerical. An *instance* $\boldsymbol{x}$ over $\mathcal{A}$ is a vector from $D_1 \times \ldots \times D_n$. Its set of *characteristics* indicates for each $A_i \in \mathcal{A}$ the value $\boldsymbol{x}$ takes in D_i. $\boldsymbol{X}$ is the set of all instances. A *classifier* f over $\mathcal{A}$ is a mapping from $\boldsymbol{X}$ to a finite set $\mathcal{L}$ of classes. A *binary classifier* f over $\mathcal{A}$ is a mapping from $\boldsymbol{X}$ to $\mathcal{L} = \{0, 1\}$. An instance $\boldsymbol{x} \in \boldsymbol{X}$ is *positive* when $f(\boldsymbol{x}) = 1$ and it is *negative* when $f(\boldsymbol{x}) = 0$.

A *decision tree* (resp. *regression tree*) over $\mathcal{A}$ [Breiman *et al.*, 1984; Quinlan, 1986] is a binary tree T, each of whose internal nodes (aka decision nodes) is labelled with a Boolean condition over $A_i \in \mathcal{A}$, and each leaf is labelled by an element of $\mathcal{L}$ (resp. a decimal floating-point number).

Decision trees (DTs), sets of decision trees (alias random forests (RFs)), and sets of regression trees (alias boosted trees (BTs)) can be used for solving binary classification problems.

The value $T(\boldsymbol{x})$ of T on an input instance $\boldsymbol{x} \in \boldsymbol{X}$ is given by the label of the leaf reached from the root. When T is a decision tree, this value is 1 when the leaf reached is a 1-leaf (i.e., a leaf labelled by 1) and it is 0 when the leaf reached is a 0-leaf (i.e., a leaf labelled by 0). The unique path characterising the leaf that is met is defined as follows: at each decision node go to the left (resp. right) child if the Boolean condition labelling the node is evaluated to 0 (resp. 1) for $\boldsymbol{x}$.

A *random forest* over $\mathcal{A}$ [Breiman, 2001] is an ensemble $RF = \{T_1, \cdots, T_m\}$, where each T_i $(i \in [m])$ is a decision tree over $\mathcal{A}$, and such that the value $RF(\boldsymbol{x})$ is given by

$$RF(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \frac{1}{m}\sum_{i=1}^{m} T_i(\boldsymbol{x}) > \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

A *boosted tree* over $\mathcal{A}$ [Freund and Schapire, 1997; Friedman, 2001; Schapire and Freund, 2014] is an ensemble $BT = \{T_1, \cdots, T_m\}$, where each T_i $(i \in [m])$ is a regression tree over $\mathcal{A}$, and such that the value $BT(\boldsymbol{x})$ is given by

$$BT(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{m} T_i(\boldsymbol{x}) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{B} = \{B_1, \ldots, B_m\}$ be the set of Boolean conditions occurring in a tree-based model f (a decision tree, a random forest or a boosted tree over $\mathcal{A}$). Those conditions can also be viewed as Boolean attributes. Therefore f can also be considered as a Boolean function over $\mathcal{B}$. The Boolean attributes in $\mathcal{B}$ are not always pairwise independent because several conditions can be generated from a single numerical or categorical attribute from $\mathcal{A}$. A domain theory Σ^f is a propositional formula over $\mathcal{B}$ indicating how the corresponding attributes occurring in f are logically connected. Accordingly, beyond $\boldsymbol{X}$, another set of instances can be considered given f: the subset $\boldsymbol{X}_{\mathcal{B}}$ of $\{0,1\}^m$ containing Boolean vectors that correspond to the truth assignments over $\mathcal{B}$ satisfying Σ. Every instance $\boldsymbol{x} \in \boldsymbol{X}$ corresponds to a unique instance $\boldsymbol{x}_{\mathcal{B}} \in \boldsymbol{X}_{\mathcal{B}}$ and this instance $\boldsymbol{x}_{\mathcal{B}}$ is such that $f(\boldsymbol{x}_{\mathcal{B}}) = f(\boldsymbol{x})$. Unlike $\boldsymbol{X}$ that is (in general) infinite, $\boldsymbol{X}_{\mathcal{B}}$ is a finite set.

Considering instances over Boolean attributes $\mathcal{B} = \{B_1, \ldots, B_m\}$ that are connected by a domain theory Σ^f, tree-based models f can thus be viewed as Boolean functions mapping the truth assignments $\boldsymbol{x}$ from $\boldsymbol{X}_{\mathcal{B}}$, i.e., the truth assignments over $\mathcal{B}$ that satisfy Σ (aka the *models* of Σ), to a Boolean value that is 1 when $\boldsymbol{x}$ is a positive instance and 0 when it is a negative instance. Furthermore, every instance $\boldsymbol{x} = (x_1, \ldots, x_m)$ from $\boldsymbol{X}_{\mathcal{B}}$ can be viewed and be represented as a canonical term $t_{\boldsymbol{x}}$ (i.e., a conjunctively-interpreted set of literals over $\mathcal{B}$ in which every variable occurs once) such that $t_{\boldsymbol{x}} = \bigwedge_{i=1}^{m} x_i^*$ where $x_i^* = x_i$ $(i \in [m])$ when the ith coordinate of $\boldsymbol{x}$ is 1 and $x_i^* = \overline{x_i}$ when the ith coordinate of $\boldsymbol{x}$ is 0.

Example 1. *As a matter of illustration, Figure 1 presents a very simple random forest $RF = \{T_1, T_2, T_3\}$ over $\mathcal{A} = \{A_1, A_2\}$, where A_1 is a numerical attribute (say, the annual income in \$k of a loan applicant) and A_2 is a Boolean attribute (indicating whether the applicant holds a permanent position). The instance $\boldsymbol{x} = (33, 1)$ from $\boldsymbol{X}$ is such that $T(\boldsymbol{x}) = 1$.*

$$\mathcal{B} = \{\underbrace{A_1 > 20}_{B_1}, \underbrace{A_1 > 30}_{B_2}, \underbrace{A_2 = 1}_{B_3}\}$$

contains three Boolean conditions. The two conditions $B_1 = (A_1 > 20)$ and $B_2 = (A_1 > 30)$ about A_1 are obviously not independent: they are logically connected by a domain theory Σ^f equivalent to $B_2 \Rightarrow B_1$. $\boldsymbol{x} = (23, 1)$ (and more generally, every instance from $\boldsymbol{X}$ for which B_1, B_2, and B_3 hold) corresponds to the instance $\boldsymbol{x}_{\mathcal{B}} = (1, 1, 1)$ from $\boldsymbol{X}_{\mathcal{B}}$. It is easy to check that f is equivalent to the CNF *formula $B_1 \vee B_2 \vee B_3$.*

Abductive explanations. Abductive explanations are local explanations that aim to explain why the instance at hand has been classified as such by the ML model f. Formally, an abductive explanation for $\boldsymbol{x} \in \boldsymbol{X}_{\mathcal{B}}$ given f is a subset t of $t_{\boldsymbol{x}}$ such that every instance $\boldsymbol{x}' \in \boldsymbol{X}_{\mathcal{B}}$ such that $t \subseteq t_{\boldsymbol{x}'}$ satisfies $f(\boldsymbol{x}') = f(\boldsymbol{x})$. Such explanations t offer correctness guarantees in the sense that any instance $\boldsymbol{x}'$ covered by t is ensured to be classified by f in the same way as $\boldsymbol{x}$.

Let Σ^f, α, β be three formulae over $\mathcal{B}$. We note $\alpha \models_{\Sigma^f} \beta$ precisely when $\Sigma^f \wedge \alpha \models \beta$ and $\alpha \equiv_{\Sigma^f} \beta$ precisely when $\alpha \models_{\Sigma^f} \beta$ and $\beta \models_{\Sigma^f} \alpha$. When $\alpha \models_{\Sigma^f} \beta$ holds, we say that β is at least as general as α modulo Σ^f. When $\alpha \equiv_{\Sigma^f} \beta$ holds, α and β are equivalent modulo Σ^f.

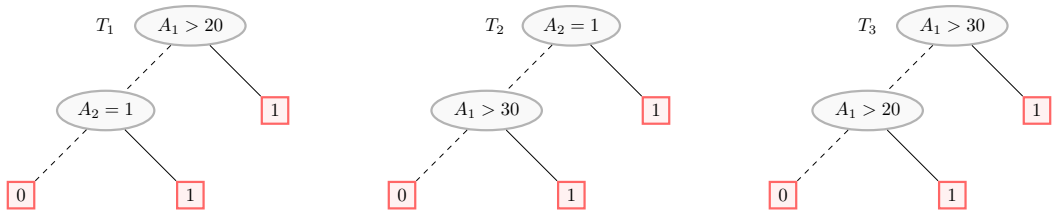

Figure 1: A simple random forest.

To clarify the landscape of abductive explanations, Table 1 summarizes the hierarchy of the core concepts used in this paper. Starting from base explanations (wAXp), domain constraints are added (wAXpc), minimality is enforced (AXp/AXpc), and finally, coverage is maximized modulo the domain theory Σ^f (CPI-Xp/mCPI-Xp).

Definition 1. *Let f be a Boolean function over a set $\mathcal{B}$ of Boolean conditions, $\boldsymbol{x}$ an instance from $\boldsymbol{X}_{\mathcal{B}}$. Let $f^* = f$ if $f(\boldsymbol{x}) = 1$ and $f^* = \neg f$ if $f(\boldsymbol{x}) = 0$. Let t be a term over $\mathcal{B}$:*

- *t is an* abductive explanation (wAXp) *for $\boldsymbol{x}$ given f iff $t \subseteq t_{\boldsymbol{x}}$ and $t \models f^*$;*
- *t is a* subset-minimal abductive explanation (AXp) *for $\boldsymbol{x}$ given f iff t is an abductive explanation for $\boldsymbol{x}$ given f and no proper subset of t is an abductive explanation for $\boldsymbol{x}$ given f;*
- *Let Σ^f be a formula over $\mathcal{B}$ (a domain theory, representing constraints over the Boolean conditions occurring in f) such that $t_{\boldsymbol{x}} \models \Sigma^f$ holds:*
  - *t is an* abductive explanation (wAXpc) *for $\boldsymbol{x}$ given f and Σ^f iff $t \subseteq t_{\boldsymbol{x}}$ and $t \models_{\Sigma^f} f^*$;*
  - *t is a* subset-minimal abductive explanation (AXpc) *for $\boldsymbol{x}$ given f and Σ^f iff t is an abductive explanation for $\boldsymbol{x}$ given f and Σ^f and no proper subset of t is an abductive explanation for $\boldsymbol{x}$ given f and Σ^f;*
  - *t is a* coverage-based prime implicant explanation (CPI-Xp) *for $\boldsymbol{x}$ given f and Σ^f iff $t \subseteq t_{\boldsymbol{x}}$, $t \models_{\Sigma^f} f^*$ and if there exists $t' \subseteq t_{\boldsymbol{x}}$ such that $t' \models_{\Sigma^f} f^*$ and $t \models_{\Sigma^f} t'$, then $t \equiv_{\Sigma^f} t'$;*
  - *t is a* minimal coverage-based prime implicant explanation (mCPI-Xp) *for $\boldsymbol{x}$ given f and Σ^f iff t is a coverage-based prime implicant explanation for $\boldsymbol{x}$ given f and Σ^f and no proper subset of t is a coverage-based prime implicant explanation for $\boldsymbol{x}$ given f and Σ^f.*

Acronym	Concept Definition
wAXp	Abductive explanation (implicant)
AXp	Subset-minimal abductive explanation (prime implicant)
wAXpc	Abductive explanation modulo Σ^f
AXpc	Subset-minimal abductive explanation modulo Σ^f
CPI-Xp	Coverage-based prime implicant expl. (modulo Σ^f)
mCPI-Xp	Minimal coverage-based prime implicant expl.

Table 1: Hierarchy of abductive explanations. The addition of 'c' denotes the integration of the domain theory Σ^f.

Example 2 (Example 1 cont'ed)**.** *Let us recall that $\mathcal{B} = \{B_1, B_2, B_3\}$, $f \equiv B_1 \vee B_2 \vee B_3$ and that $\Sigma^f = B_2 \Rightarrow B_1$.*

- *Consider first the instance $\boldsymbol{x} = (1, 1, 1)$. We have $f(\boldsymbol{x}) = 1$. $t_{\boldsymbol{x}} = \{B_1, B_2, B_3\}$ is a wAXp (and a wAXpc) for $\boldsymbol{x}$ given f, but not an AXp. $\{B_1\}$, $\{B_2\}$, and $\{B_3\}$ are the AXps for $\boldsymbol{x}$ given f. They are also the AXpcs for $\boldsymbol{x}$ given f and Σ^f. Among them $\{B_1\}$ and $\{B_3\}$ are the CPI-Xps and the mCPI-Xps for $\boldsymbol{x}$ given f and Σ^f.*
- *Consider now the instance $\boldsymbol{x}' = (0, 0, 0)$. We have $f(\boldsymbol{x}') = 0$. $t_{\boldsymbol{x}'} = \{\overline{B_1}, \overline{B_2}, \overline{B_3}\}$ is the unique AXp for $\boldsymbol{x}'$ given f, but it is not an AXpc for $\boldsymbol{x}'$ given f and Σ^f. $\{\overline{B_1}, \overline{B_3}\}$ is the unique AXpc for $\boldsymbol{x}'$ given f and Σ^f. It is also the unique CPI-Xp and mCPI-Xp for $\boldsymbol{x}'$ given f and Σ^f.*

mCPI-Xps of minimal size can be arbitrarily larger than AXps (or AXpcs) of minimal size. Unlike what happens with AXps, the generality criterion may conflict with the interpretability one. Indeed, minimal-size wAXps are AXps, thus are as general as possible while being more interpretable than other wAXps. Contrastingly, CPI-Xps (and even mCPI-Xps) are (by definition) more general than other wAXps but they can be less interpretable (they may contain many more characteristics).

3 Computing CPI-Xps

In order to derive CPI-Xps suited to tree-based models, we present a greedy algorithm **CPI-Xp-TBM** (see Algorithm 1). This algorithm leverages the specific nature of the domain theory Σ^f linking the Boolean conditions in tree-based models to make the search for CPI-Xps more efficient.

CPI-Xp-TBM takes as input an instance $\boldsymbol{x} \in \boldsymbol{X}_{\mathcal{B}}$. The goal is to explain the classification of $\boldsymbol{x}$ achieved by the binary classifier f over $\mathcal{B}$. Here f is a tree-based model (a decision tree, a random forest, or a boosted tree). Another part of the input is the *domain theory* Σ^f associated with f:

$$\Sigma^f = \bigwedge_{A_i \in \mathcal{A}} \Sigma^f_{A_i},$$

is the conjunction of the domain theories $\Sigma^f_{A_i}$ associated with the primitive attributes from $\mathcal{A}$. Each $\Sigma^f_{A_i}$ ($A_i \in \mathcal{A}$) is defined as follows. When A_i is a numerical attribute such that the set of conditions about A_i occurring in f is $\{A_i > v_i^1, \ldots, A_i > v_i^{c(i)}\}$ where $v_i^{c(i)} > \ldots > v_i^1$ (i.e., there are $c(i)$ thresholds considered for A_i), $\Sigma^f_{A_i}$ is given by the following CNF formula based on the Boolean variables

$B_i^1, \ldots, B_i^{c(i)}$ from $\mathcal{B}$, where each B_i^j ($j \in [c(i)]$) stands for $A_i > v_i^j$:

$$\Sigma^f_{A_i} = \bigwedge_{j=2}^{c(i)} (\neg B_i^j \vee B_i^{j-1}).$$

When A_i is a categorical attribute such that the set of conditions about A_i occurring in f is $\{A_i = v_i^1, \ldots, A_i = v_i^{c(i)}\}$, $\Sigma^f_{A_i}$ is given by the following CNF formula based on the Boolean variables $B_i^1, \ldots, B_i^{c(i)}$ from $\mathcal{B}$, where each B_i^j ($j \in [c(i)]$) stands for $A_i = v_i^j$:

$$\Sigma^f_{A_i} = \bigwedge_{j,k \in [c(i)] | j \neq k} (\neg B_i^j \vee \neg B_i^k).$$

Thus, when A_i is a numerical attribute, $\Sigma^f_{A_i}$ is a *chain of implications* that simply reflects the linear ordering over the threshold values v_i^j for A_i that have been found by the learning algorithm used to generate f. When A_i is a categorical attribute, $\Sigma^f_{A_i}$ is a *mutual exclusion constraint* that states that the values found for A_i by the learning algorithm used to generate f are mutually exclusive. Accordingly, the (running) domain of each categorical attribute A_i is considered as open: it is not assumed that the sole possible values for A_i are in $\{v_i^1, \ldots, v_i^{c(i)}\}$ ($v_i^1, \ldots, v_i^{c(i)}$ are those values found when f was learned but this does not preclude the existence of other possible values for A_i, especially because such values do not necessarily occur in the training set used to learn f). Finally, the Boolean attributes A_i from $\mathcal{A}$ do not give rise to any specific domain theory: $\Sigma^f_{A_i}$ is the empty CNF formula, represented by the Boolean constant always true $\top$.

It can be easily checked that the domain theory Σ^f associated with a tree-based model f is *decomposable*: whenever A_i and A_j are two distinct attributes from $\mathcal{A}$, then $\{B_i^1, \ldots, B_i^{c(i)}\} \cap \{B_j^1, \ldots, B_j^{c(j)}\} = \emptyset$.

In order to compute a CPI-Xp for $\boldsymbol{x} \in \boldsymbol{X}_{\mathcal{B}}$ given f and Σ^f, a space of explanation candidates is explored:

Definition 2. *An* explanation candidate t *for* $\boldsymbol{x} \in \boldsymbol{X}_{\mathcal{B}}$ *is a subset of literals from* $t_{\boldsymbol{x}}$.

Given that $\boldsymbol{x}$ satisfies Σ^f, every explanation candidate t for $\boldsymbol{x}$ is consistent with Σ^f.

Explanation candidates (when viewed up to equivalence modulo Σ^f) are ordered by the generality relation $\models_\Sigma$ that is a (partial) ordering, i.e., a reflexive, transitive, and antisymmetric relation. If t and t' are two explanation candidates for $\boldsymbol{x}$ ordered by the strict part of this ordering, i.e., such that $t \models_{\Sigma^f} t'$ and $t' \not\models_{\Sigma^f} t$ hold, we say that t' is a (strict) *generalisation* of t given Σ^f.

When looking for a CPI-Xp for $\boldsymbol{x}$ given f and Σ^f, the exploration of the space of explanation candidates can be guided by $\models_{\Sigma^f}$. To define the exploration strategy, the following notion of A_i-part of an explanation candidate will be useful:

Definition 3. *The* A_i-part *of an explanation candidate* t *for* $\boldsymbol{x} \in \boldsymbol{X}_{\mathcal{B}}$ *is the subset* t_{A_i} *of literals* ℓ *from* t *that are about* $A_i \in \mathcal{A}$*, i.e., the variable of* ℓ *corresponds to a condition* $A_i > v_i^j$ *or* $A_i = v_i^j$ *(*$j \in [c(i)]$*).*

Algorithm 1 CPI-Xp-TBM

Require: $\boldsymbol{x}$ an instance from $\boldsymbol{X}_{\mathcal{B}}$ satisfying Σ, f a tree-based classifier over $\mathcal{B}$, Σ^f a formula over $\mathcal{B}$ (the corresponding domain theory).
Ensure: a CPI-Xp cx for $\boldsymbol{x}$ given f and Σ^f.

```
f* ← (f(x) = 1) ? f : f̄
t ← t_x
cx ← ∅
for each A_i ∈ A do
    t' ← t_{A_i}
    while true do
        if t' = ∅ then
            exit-while
        else
            found ← false
            gs ← msg(t', Σ^f)
            for each g ∈ gs do
                gs ← gs \ {g}
                if imp(cx ∪ g ∪ ⋃_{j>i} t_{A_j}, f*, Σ^f) then
                    t' ← g
                    found ← true
                    exit-for
            if not found then
                cx ← cx ∪ t'
                exit-while
return cx
```

Because the domain theories $\Sigma^f_{A_i}$ ($A_i \in \mathcal{A}$) are based on Boolean variables that are pairwise disjoint, it is easy to show that the generality relation $\models_{\Sigma^f}$ over explanation candidates can be decomposed over the primitive attributes:

Proposition 1. *Let* t, t' *be two explanation candidates for* $\boldsymbol{x} \in \boldsymbol{X}_{\mathcal{B}}$*. We have* $t \models_{\Sigma^f} t'$ *if and only if* $\forall A_i \in \mathcal{A}$, $t_{A_i} \models_{\Sigma^f} t'_{A_i}$ *if and only if* $\forall A_i \in \mathcal{A}$, $t_{A_i} \models_{\Sigma^f_{A_i}} t'_{A_i}$.

CPI-Xp-TBM is an iterative algorithm: each primitive attribute A_i from $\mathcal{A}$ is considered in sequence, and the A_i-part of the explanation candidate $t_{\boldsymbol{x}}$ is generalised as much as possible in a greedy fashion, i.e., while the resulting explanation is a wAXpc for $\boldsymbol{x}$ given f and Σ^f.

CPI-Xp-TBM takes advantage of two main functions: $\mathbf{msg}$ and $\mathbf{imp}$. $\mathbf{imp}$ is used to achieve implicant tests and the goal of $\mathbf{msg}$ is to guide the search for a CPI-Xp for $\boldsymbol{x}$ given f and Σ^f by deriving at each step the most specific generalisations of the current explanation candidate:

Definition 4. *Given* Σ^f*, a* most specific generalisation *of an explanation candidate* t *for* $\boldsymbol{x}$ *is an explanation candidate* t' *for* $\boldsymbol{x}$ *such that* $t \models_{\Sigma^f} t'$, $t' \not\models_{\Sigma^f} t$ *and for every explanation candidate* t'' *for* $\boldsymbol{x}$ *such that* $t \models_{\Sigma^f} t''$ *and* $t'' \not\models_{\Sigma^f} t$*, if* $t'' \models_{\Sigma^f} t'$*, then* $t' \models_{\Sigma^f} t''$.

In the set $\mathbf{msg}(t, \Sigma^f)$ of the most specific generalisations of t given Σ^f, only one representative per equivalence class modulo Σ^f must be kept. This set is empty precisely when $t = \emptyset$ because of the specific nature of Σ^f when f is a tree ensemble.

Due to the decomposability of Σ^f, a most specific generalisation $\mathbf{msg}(t, \Sigma^f)$ of an explanation candidate can be char-

acterised by computing a most specific generalisation of one of its parts:

Proposition 2. *Let $A_i \in \mathcal{A}$ be such that $\exists t'_{A_i} \in \mathbf{msg}(t_{A_i}, \Sigma^f)$. Then $\bigcup_{A_j \in \mathcal{A} | j \neq i} t_{A_j} \cup t'_{A_i}$ is a most specific generalisation of the explanation candidate t for $\boldsymbol{x}$ given Σ^f.*

The decomposability of Σ^f and the specific nature of each $\Sigma^f_{A_i}$ ($A_i \in \mathcal{A}$) ensures that $\mathbf{msg}(t_{A_i}, \Sigma^f)$ can be computed in time polynomial in the size of t_{A_i} plus the size of $\Sigma^f_{A_i}$:

Proposition 3. *Let t be an explanation candidate for $\boldsymbol{x} \in \boldsymbol{X}_{\mathcal{B}}$ and let $A_i \in \mathcal{A}$.*

- *When A_i is numerical, $\Sigma^f_{A_i} = \bigwedge_{j=2}^{c(i)}(\neg B_i^j \vee B_i^{j-1})$ has the form of a chain of implications. Let $M = max(\{j \in [c(i)] \mid B_i^j \in t\})$ if t_{A_i} contains a positive literal and let $m = min(\{j \in [c(i)] \mid \overline{B_i^j} \in t\})$ if t_{A_i} contains a negative literal. Let $G^+ = \{B_i^{M-1}\}$ if $M \neq 1$ and $G^+ = \emptyset$ if $M = 1$, and let $G^- = \{\overline{B_i^{m+1}}\}$ if $m \neq c(i)$ and $G^- = \emptyset$ if $m = c(i)$.*
  - *If t_{A_i} contains a positive literal and a negative literal, then $\mathbf{msg}(t_{A_i}, \Sigma^f) = \{(t_{A_i} \setminus \{B_i^M\}) \cup G^+, (t_{A_i} \setminus \{\overline{B_i^m}\}) \cup G^-\}$.*
  - *If t_{A_i} contains a positive literal and no negative literals, then $\mathbf{msg}(t_{A_i}, \Sigma^f) = \{(t_{A_i} \setminus \{B_i^M\}) \cup G^+\}$.*
  - *If t_{A_i} contains no positive literals and a negative literal, then $\mathbf{msg}(t_{A_i}, \Sigma^f) = \{(t_{A_i} \setminus \{\overline{B_i^m}\}) \cup G^-\}$.*
- *When A_i is categorical, $\Sigma^f_{A_i} = \bigwedge_{j,k \in [c(i)] | j \neq k}(\neg B_i^j \vee \neg B_i^k)$ has the form of a mutual exclusion constraint.*
  - *If t_{A_i} contains a positive literal B_i^j, then $\mathbf{msg}(t_{A_i}, \Sigma^f) = \{\{\overline{B_i^k} \mid k \in [c(i)] \setminus \{j\}\}\}$.*
  - *If t_{A_i} does not contain any positive literal then $\mathbf{msg}(t_{A_i}, \Sigma^f) = \{t_{A_i} \setminus \{\overline{B_i^j}\} \mid \overline{B_i^j} \in t_{A_i}\}$.*

A second important component of **CPI-Xp-TBM** is the implicant test **imp**. Indeed, when such a test fails, i.e., when an explanation candidate t is not an abductive explanation for $\boldsymbol{x}$ given f and Σ^f, it is useless to consider any explanation candidate that generalises t.

Deciding whether an explanation candidate t is an abductive explanation (i.e., determining whether t is an implicant of f^* modulo Σ^f) amounts to testing the inconsistency of $\Sigma^f \wedge t \wedge \overline{f^*}$. This can be done using various kinds of solvers (for instance, an SMT solver [Ignatiev *et al.*, 2019], a MIP solver [Cui *et al.*, 2015; Parmentier and Vidal, 2021], a SAT solver [Audemard *et al.*, 2022] or a MaxSAT solver [Ignatiev *et al.*, 2022] can be used).

Exploring in a greedy fashion the set of most specific generalisations of the current explanation candidate t (starting with $t = t_{\boldsymbol{x}}$) until the implicant test fails ensures that no relevant part of the search space is left unexplored for generating a CPI-Xp for $\boldsymbol{x}$ given f and Σ^f. Algorithm 1 is explained in the following example.

Example 3 (Example 1 cont'ed)**.** *Suppose that $\boldsymbol{x} = (1, 1, 1)$. Since $f(\boldsymbol{x}) = 1$, we have $f^* = f$. At start (line 2), t is equal to $\{B_1 = (A_1 > 20), B_2 = (A_1 > 30), B_3 = (A_2 = 1)\}$. Suppose that A_2 is processed before A_1 in the main for-loop (line 4). One first looks for the most specific generalisations of $t' = t_{A_2} = \{B_3 = (A_2 = 1)\}$ given Σ^f. Since $\mathbf{msg}(t', \Sigma^f) = \{\emptyset\}$, there is a single generalisation of t' to be considered at line 12 (the inner for-loop), namely $\emptyset$. Since $\{B_1 = (A_1 > 20), B_2 = (A_1 > 30)\}$ is an implicant of f^* modulo Σ^f, t' is set to $\emptyset$ at line 15. Then one looks for the most specific generalisations of $t' = \emptyset$. Since $t' = \emptyset$, the test at line 7 leads to leave the while-loop. At that stage, cx is still equal to $\emptyset$.*
Then A_1 is processed in the main for-loop. One first looks for the most specific generalisations of $t' = t_{A_1} = \{B_1 = (A_1 > 20), B_2 = (A_1 > 30)\}$ given Σ^f. Since $\mathbf{msg}(t', \Sigma^f) = \{\{B_1 = (A_1 > 20)\}\}$, there is a single generalisation of t' to be considered at line 12 (the inner for-loop), namely $\{B_1 = (A_1 > 20)\}$. Since $\{B_1 = (A_1 > 20)\}$ is an implicant of f^* modulo Σ^f, t' is set to $\{B_1 = (A_1 > 20)\}$ at line 15. Then one looks for the most specific generalisations of $t' = \{B_1 = (A_1 > 20)\}$. Since $\mathbf{msg}(t', \Sigma^f) = \{\emptyset\}$, there is a single generalisation of t' to be considered at line 12 (the inner for-loop), namely $\emptyset$. Since $\emptyset$ is not an implicant of f^* modulo Σ^f, found remains false, so that cx is updated to $\{B_1 = (A_1 > 20)\}$ (line 19). Finally, since $t' = \emptyset$, the test at line 7 leads to leave the while-loop. At that stage, cx is equal to $\{B_1 = (A_1 > 20)\}$. Since every attribute $A_i \in \mathcal{A}$ has been processed in the main for loop, $cx = \{B_1 = (A_1 > 20)\}$ is returned at line 21.

Proposition 4. **CPI-Xp-TBM** *is correct w.r.t. its specification: it always terminates and returns a CPI-Xp for $\boldsymbol{x}$ given f and Σ^f.*

For each attribute $A_i \in \mathcal{A}$, the number of calls to the implicant test **imp** at line 14 is upper bounded by $c(i)$, thus linear in the number of Boolean conditions in $\mathcal{B}$ that are about A_i. Therefore, the total number of calls to the implicant test is linear in the number of Boolean conditions in $\mathcal{B}$. Given that the implicant test can be achieved in time polynomial in $|f| + |\Sigma^f|$ when f is a decision tree [Audemard *et al.*, 2024], **CPI-Xp-TBM** is a polynomial-time algorithm when f is a decision tree. In the case of random forests or boosted trees, the implicant test is coNP-complete, making unlikely the existence of a (deterministic) polynomial-time algorithm for it. Thus, **CPI-Xp-TBM** does not run in polynomial time when such models are considered as input. However, if one is ready to lose a bit of generality in the explanations in order to ensure a polynomial-time behaviour of **CPI-Xp-TBM** (thus, to improve its scalability), a way consists in replacing the test $\mathbf{imp}(cx \cup g \cup \bigcup_{j>i} t_{A_j}, f^*, \Sigma^f)$ by a lower approximation of it that can be evaluated efficiently. Such polynomial-time approximations exist for random forests (resp. boosted trees) leading to the characterisation of abductive explanations called majoritary (resp. tree-specific) explanations (see [Audemard *et al.*, 2022; Audemard *et al.*, 2023] for details). The resulting explanations would not be CPI-Xps in general but the fact that they are abductive (wAXpc) would be guaranteed.

Dataset	$\|D\|$	$\|\mathcal{A}\|$	$\|\mathcal{B}_{DT}\|$	$\|\mathcal{B}_{RF}\|$	$\|\mathcal{B}_{BT}\|$
arrowhead_0_vs_1	146	249	11.1	880.6	90.2
arrowhead_0_vs_2	146	249	6.9	595.3	68.2
arrowhead_1_vs_2	130	249	9.6	789.6	83.2
australian	690	38	71.8	1563.1	427.2
balance_0_vs_1	625	4	16.6	28.0	16.0
balance_0_vs_2	337	4	16.5	28.0	16.0
balance_1_vs_2	576	4	18.5	28.0	16.0
bank	4521	48	221.4	5327.3	833.9
biodegradation	1055	41	106.7	5716.1	854.2
breast-tumor	286	37	55.8	115.5	71.5
bupa	345	5	43.7	420.7	150.3
cleveland_nominal	303	23	40.4	576.8	187.1
cnae	1080	856	15.7	554.3	24.2
compas	6172	11	45.3	69.2	43.8
contraceptive	1473	21	80.2	109.5	75.4
dexter	600	20000	36.3	7907.2	388.8
divorce	170	54	2.8	119.3	15.2
german	1000	58	42.4	530.7	166.2
melb_data_0_vs_1	13580	61	712.0	28394.0	2172.6
melb_data_0_vs_2	13580	61	644.0	23285.0	2028.4
melb_data_1_vs_2	13580	61	581.6	24101.6	2044.6

Table 2: Dataset statistics. $|D|$: number of instances, $|\mathcal{A}|$: number of primitive features, $|\mathcal{B}_{DT}|/|\mathcal{B}_{RF}|/|\mathcal{B}_{BT}|$: average numbers of binary features in the decision trees, random forests, and boosted trees that have been generated.

	Cooper-Amgoud			CPI-Xp-TBM		
Dataset	Time	Std	TO%	Time	Std	TO%
arrowhead_0_vs_1	**0.004**	0.004	**0**	0.007	0.006	**0**
arrowhead_0_vs_2	**0.003**	0.005	**0**	0.005	0.006	**0**
arrowhead_1_vs_2	**0.005**	0.006	**0**	0.010	0.008	**0**
australian	7.541	11.774	38	**0.071**	0.026	**0**
balance_0_vs_1	0.496	1.674	**0**	**0.011**	0.007	**0**
balance_0_vs_2	0.100	0.082	**0**	**0.015**	0.009	**0**
balance_1_vs_2	0.109	0.092	**0**	**0.011**	0.006	**0**
bank	-	-	100	**0.909**	0.326	**0**
biodegradation	13.565	13.393	59	**0.129**	0.040	**0**
breast-tumor	5.899	7.560	87	**0.055**	0.020	**0**
bupa	9.961	14.395	30	**0.020**	0.009	**0**
cleveland_nominal	1.094	3.656	2	**0.019**	0.008	**0**
cnae	**0.008**	0.006	**0**	0.019	0.009	**0**
compas	1.073	0.462	96	**0.097**	0.071	**0**
contraceptive	-	-	100	**0.248**	0.111	**0**
dexter	**0.067**	0.027	**0**	0.540	0.161	**0**
divorce	**4.8e-04**	0.002	**0**	0.001	0.002	**0**
german	0.853	1.779	1	**0.025**	0.010	**0**
melb_data_0_vs_1	-	-	100	**12.773**	5.071	**0**
melb_data_0_vs_2	-	-	100	**8.835**	3.438	**0**
melb_data_1_vs_2	-	-	100	**7.635**	2.746	**0**

Table 3: Performance comparison on decision trees: **Cooper-Amgoud** vs. **CPI-Xp-TBM**. Time (resp. Std) is the average computation time (in seconds) to derive one CPI-Xp (resp. the corresponding standard deviation). TO% is the number of timeouts (out of 100) that have been reached.

Our algorithm **CPI-Xp-TBM** (Algorithm 1) differs from the general-purpose procedure introduced in [Cooper and Amgoud, 2024] in several key aspects. While the latter is designed for any classifier and relies on a counter-example search strategy to iteratively replace weak explanations with strictly more general ones, our approach specifically exploits the structural properties of tree-based models and their associated domain theories. This specialization reduces the computational complexity from Π_2^p-completeness in the general case to a problem solvable with only polynomially many calls to an NP oracle when dealing with tree ensembles.

Finally, as this is the case for the CPI-Xps derived using the algorithm presented in [Cooper and Amgoud, 2024], it is easy to show that the CPI-Xps that are generated by Algorithm 1 are not necessarily minimal ones. However, we know that for every CPI-Xp t for $\boldsymbol{x}$ given f and Σ^f, there exists a mCPI-Xp $t_{min} \subseteq t$ for $\boldsymbol{x}$ given f and Σ^f such that $t_{min} \equiv_{\Sigma^f} t$. The point is that t_{min} can be easily extracted from t using a greedy "deletion-based" algorithm (see [Cooper and Amgoud, 2024] for details).

4 Experiments

Empirical protocol. In our experiments, we considered 21 datasets for binary classification, coming from standard repositories: UCI (https://archive.ics.uci.edu/ml/index.php), openML (https://www.openml.org/), and UCR (https://www.timeseriesclassification.com/). Instances of these datasets are based on primitive attributes of various types (numerical, categorical, Boolean) forming a set $\mathcal{A}$.

Categorical attributes have been one-hot encoded and numerical attributes have been binarized on-the-fly by the learning algorithm used to generate the classifier f, given by a tree-based model f.

Decision trees and random forests f have been learned using the version 1.6.1 of the Scikit-Learn library [Pedregosa *et al.*, 2011]. All hyperparameters are set to their default values. As for decision trees, the Gini impurity is used as the splitting criterion, and no maximum depth is enforced (nodes are expanded until all leaves are pure or contain fewer than 2 samples). Regarding random forests, 100 trees per forest are generated, and the same default node expansion strategy is applied (nodes are expanded until all leaves are pure or until all leaves contain less than *min_samples_split* samples, set to 2). Finally, regarding boosted trees, we used the XGBoost implementation (version 1.7.3) [Chen and Guestrin, 2016] with its default configuration: the model consists of 100 boosting rounds, a learning rate of 0.3, and a maximum tree depth of 6. For every dataset, a 10-fold cross validation process has been performed: for each model under consideration, 10 classifiers have been learned from a training set representing a specific subset (picked up uniformly at random) of $\frac{9}{10}$ of the total number of instances in the dataset. For each classifier, 10 instances satisfying the corresponding domain theory Σ^f have been generated, leading to 100 instances per dataset.

The implicant test $\mathbf{imp}(cx \cup g \cup \bigcup_{j>i} t_{A_j}, f^*, \Sigma^f)$ is one of the core components of Algorithm 1. In our implementation of **imp**, we use a SAT encoding when f is a decision tree or a random forest (extra variables that are existentially quantified are used in the latter case) to exploit highly optimized Boolean logic; a MIP encoding (close to those presented in [Cui *et al.*, 2015; Parmentier and Vidal, 2021]) is used when f is a boosted tree to natively and efficiently handle its continuous linear sum constraints ($\sum_i T_i(\boldsymbol{x}) > 0$). Based on those encodings, the implicant tests achieved in the experiments are performed, respectively, using the Glucose 3.0 SAT solver [Audemard *et al.*, 2013] and the SCIP solver [Achterberg, 2009] (via the OR-Tools wrapper).

For each classifier f and instance $\boldsymbol{x}$, the time spent by **CPI-Xp-TBM** to generate one CPI-Xp for $\boldsymbol{x}$ given f and Σ^f has been measured. Similar measurements have also been

	Cooper-Amgoud			CPI-Xp-TBM		
Dataset	**Time**	**Std**	**TO%**	**Time**	**Std**	**TO%**
arrowhead_0_vs_1	-	-	100	**0.358**	0.066	**0**
arrowhead_0_vs_2	-	-	100	**0.229**	0.030	**0**
arrowhead_1_vs_2	-	-	100	**0.344**	0.075	**0**
australian	-	-	100	**9.322**	10.520	**4**
balance_0_vs_1	0.801	1.124	3	**0.126**	0.044	**0**
balance_0_vs_2	0.881	1.829	**0**	**0.067**	0.029	**0**
balance_1_vs_2	0.474	0.560	**0**	**0.069**	0.026	**0**
bank	-	-	100	**22.095**	15.734	**19**
biodegradation	-	-	100	**12.971**	12.383	**9**
breast-tumor	-	-	100	**0.208**	0.078	**0**
bupa	-	-	100	**0.417**	0.369	**0**
cleveland_nominal	-	-	100	**0.539**	0.343	**0**
cnae	47.111	6.233	56	**1.553**	1.747	**0**
compas	**0.562**	0.282	98	0.660	0.220	**0**
contraceptive	0.669	0	99	**0.576**	0.144	**0**
dexter	-	-	100	**53.148**	5.610	**88**
divorce	12.044	16.095	28	**0.053**	0.020	**0**
german	-	-	100	**24.449**	14.761	**21**
melb_data_0_vs_1	-	-	100	**31.444**	10.406	**84**
melb_data_0_vs_2	-	-	100	**40.995**	12.747	**98**
melb_data_1_vs_2	-	-	100	**47.999**	8.358	**77**

Table 4: Performance comparison on random forests: **Cooper-Amgoud** vs. **CPI-Xp-TBM**. Time (resp. Std) is the average computation time (in seconds) to derive one CPI-Xp (resp. the corresponding standard deviation). TO% is the number of timeouts (out of 100) that have been reached.

	Cooper-Amgoud			CPI-Xp-TBM		
Dataset	**Time**	**Std**	**TO%**	**Time**	**Std**	**TO%**
arrowhead_0_vs_1	24.448	17.465	54	**2.173**	1.090	**0**
arrowhead_0_vs_2	18.062	14.061	33	**3.626**	3.280	**0**
arrowhead_1_vs_2	35.251	15.191	68	**4.380**	3.337	**0**
australian	-	-	100	**31.036**	15.979	**57**
balance_0_vs_1	30.909	11.031	24	**2.743**	1.513	**0**
balance_0_vs_2	19.741	10.944	1	**1.288**	0.677	**0**
balance_1_vs_2	17.363	10.908	9	**1.175**	0.758	**0**
bank	-	-	100	**30.431**	14.107	**79**
biodegradation	-	-	100	**37.484**	11.428	**54**
breast-tumor	-	-	100	**9.652**	9.702	**2**
bupa	35.478	0	99	**9.848**	8.914	**0**
cleveland_nominal	30.909	0	99	**12.332**	7.765	**0**
cnae	10.895	7.026	**0**	**3.124**	1.700	**0**
compas	30.484	14.459	96	**7.487**	8.057	**0**
contraceptive	-	-	100	**18.288**	13.203	**4**
dexter	-	-	100	**26.592**	11.159	**4**
divorce	3.419	0.853	**0**	**1.411**	0.152	**0**
german	36.192	0	99	**28.240**	14.940	**21**
melb_data_0_vs_1	-	-	100	**46.492**	8.703	**96**
melb_data_0_vs_2	-	-	100	**36.364**	0	**99**
melb_data_1_vs_2	-	-	100	**51.927**	9.313	**98**

Table 5: Performance comparison on boosted trees: **Cooper-Amgoud** vs. **CPI-Xp-TBM**. Time (resp. Std) is the average computation time (in seconds) to derive one CPI-Xp (resp. the corresponding standard deviation). TO% is the number of timeouts (out of 100) that have been reached.

achieved using this time our implementation of the generic algorithm **Cooper-Amgoud** for deriving CPI-Xps, as presented in [Cooper and Amgoud, 2024]. To make a fair comparison between the two algorithms, the implicant test required by **Cooper-Amgoud** has been implemented using the same function **imp** as used in **CPI-Xp-TBM**. A timeout of 60 seconds has been considered for each computation.

Empirical results. Table 2 summarizes the characteristics of the datasets used in our experiments. Column $|D|$ reports the total number of instances, while $|A|$ represents the count of primitive features. $|\mathcal{B}_{DT}|$, $|\mathcal{B}_{RF}|$ and $|\mathcal{B}_{BT}|$ denote respectively the average number of binary features occurring in the decision trees, random forests and boosted trees that have been generated. In our approach, combinatorial hardness is driven by the number of primitive features and resulting Boolean conditions rather than the number of instances.

Performance comparisons are then detailed in the subsequent tables. Table 3 focuses on DTs, comparing our approach (**CPI-Xp-TBM**) against the generic algorithm proposed by Cooper and Amgoud. It reports for each dataset the average computation time (Time, in seconds) and the corresponding standard deviation (Std). Only the instances (out of the 100 instances generated for the dataset) for which the approach terminated in due time have been considered in the computation of Time and Std. Table 3 also reports the percentage of instances reaching the 60-second timeout (TO%). Finally, Tables 4 and 5 present similar measurements for RFs and for BTs, respectively.

The results show that for each dataset used in the experiments and each of the three tree-based models under consideration, **Cooper-Amgoud** never succeeded in solving more instances than **CPI-Xp-TBM** within 60 seconds. When the computations succeeded, **Cooper-Amgoud** turned out to be slightly more efficient on average than **CPI-Xp-TBM** only for a couple of datasets when f is a DT, while **CPI-Xp-TBM** was more efficient than **Cooper-Amgoud** in all the remaining cases when f is an RF or a BT. This comparison highlights the robustness of our approach, particularly in the cases of RFs and BTs where the increased number of implicant tests and the computational cost of those tests cause the generic method to timeout for the majority of datasets and instances tested. Thus, in our experiments, the computation of a CPI-Xp using **Cooper-Amgoud** succeeded in due time (60 seconds) for 61.29% of the instances tested in the DT case, 19.81% in the RF case, and 29.43% in the BT case. In contrast, **CPI-Xp-TBM** maintains quite a high efficiency across "mildly more complex" tree-based models (i.e., based on hundreds of Boolean conditions). Consequently, the computation of a CPI-Xp using **CPI-Xp-TBM** succeeded in due time for 100% of the instances tested in the DT case, 80.95% in the RF case, and 75.52% in the BT case. Timeouts in **CPI-Xp-TBM** are entirely due to the calls to the **imp** oracle: while our approach reduces the number of oracle calls, each implicant test remains inherently intractable for RFs and BTs.

5 Conclusion

We presented a greedy algorithm that exploits the specific domain theory of tree-based models to efficiently derive CPI-Xps. Unlike what happens when the domain theory under consideration is not constrained, our algorithm requires only polynomially many calls to the implicant test. An empirical evaluation across tree ensembles confirms that our approach scales better than the existing general-purpose algorithm for deriving CPI-Xps.

Several perspectives for further research can be envisioned. One of them is to look for alternative conditions on the domain theory that would be sufficient to preserve the correctness of the algorithm. Another perspective is to exploit local CPI-Xps as building blocks for deriving more global explanations via set-cover algorithms.

Acknowledgements

Many thanks to the anonymous reviewers for their numerous comments and insights. This work has benefited from the support of the AI Chair EXPEKCTATION (ANR-19-CHIA-0005-01) and of the France 2030 MAIA Project (ANR-22-EXES-0009) of the French National Research Agency (ANR).

References

[Achterberg, 2009] Tobias Achterberg. Scip: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[Arrieta *et al.*, 2020] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[Audemard *et al.*, 2013] Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon. Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *Proceedings of SAT'13*, pages 309–317, 2013.

[Audemard *et al.*, 2022] Gilles Audemard, Steve Bellart, Louenas Bounia, Frédéric Koriche, Jean-Marie Lagniez, and Pierre Marquis. Trading complexity for sparsity in random forest explanations. In *Proceedings of AAAI'22*, pages 5461–5469, 2022.

[Audemard *et al.*, 2023] Gilles Audemard, Jean-Marie Lagniez, Pierre Marquis, and Nicolas Szczepanski. Computing abductive explanations for boosted trees. In *Proceedings of AISTATS'23*, pages 4699–4711, 2023.

[Audemard *et al.*, 2024] Gilles Audemard, Jean-Marie Lagniez, Pierre Marquis, and Nicolas Szczepanski. Deriving provably correct explanations for decision trees: The impact of domain theories. In *Proceedings of IJCAI'24*, pages 3688–3696, 2024.

[Audemard *et al.*, 2026] Gilles Audemard, Sylvie Coste-Marquis, Pierre Marquis, Mehdi Sabiri, and Nicolas Szczepanski. Computing coverage-based prime implicant explanations for tree-based models. https://archive.softwareheritage.org/swh:1:dir:46ffd51e87f2b334be056aaeed27eef32a0e281a, 2026.

[Bodria *et al.*, 2023] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *Data Mining and Knowledge Discovery*, 37(5):1719–1778, 2023.

[Breiman *et al.*, 1984] Leo Breiman, Jerome Friedman, R.A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[Breiman, 2001] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of KDD'16*, page 785–794, 2016.

[Cooper and Amgoud, 2023] Martin C. Cooper and Leila Amgoud. Abductive explanations of classifiers under constraints: Complexity and properties. In *Proceedings of ECAI'23*, pages 469–476, 2023.

[Cooper and Amgoud, 2024] Martin C. Cooper and Leila Amgoud. Abductive explanations of classifiers under constraints: Complexity and properties. *CoRR*, abs/2409.12154, 2024.

[Cooper and Marques-Silva, 2023] Martin C. Cooper and Joao Marques-Silva. Tractability of explaining classifier decisions. *Artificial Intelligence*, 316:103841, 2023.

[Cui *et al.*, 2015] Zhicheng Cui, Wenlin Chen, Yujie He, and Yixin Chen. Optimal action extraction for random forests and boosted trees. In *Proceedings of KDD'15*, pages 179–188, 2015.

[Freund and Schapire, 1997] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[Friedman, 2001] Jerome H. Friedman. Greedy function approximation: A gradient boosted machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[Guidotti *et al.*, 2019] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):93:1–93:42, 2019.

[Gunning, 2019] David Gunning. DARPA's explainable artificial intelligence (XAI) program. In *Proceedings of IUI'19*, 2019.

[Hsieh *et al.*, 2024] Weiche Hsieh, Ziqian Bi andChuanqi Jiang, Junyu Liu, Benji Peng, Sen Zhang, Xuanhe Pan, Jiawei Xu, Jinlang Wang, Keyu Chen, Pohsun Feng, Yizhu Wen, Xinyuan Song, Tianyang Wang andMing Liu, Junjie Yang, Ming Li, Bowen Jing, Jintao Ren, Junhao Song, Hong-Ming Tseng, Yichao Zhang, Lawrence K.Q. Yan, Qian Niu, Silin Chen, Yunze Wang, and Chia Xin Liang. A comprehensive guide to explainable AI: From classical models to LLMs. *CoRR*, abs/2412.00800, 2024.

[Ignatiev *et al.*, 2019] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. On validating, repairing and refining heuristic ML explanations. *CoRR*, abs/1907.02509, 2019.

[Ignatiev *et al.*, 2020] Alexey Ignatiev, Nina Narodytska, Nicholas Asher, and Joao Marques-Silva. On relating 'why?' and 'why not?' explanations. *CoRR*, abs/2012.11067, 2020.

[Ignatiev *et al.*, 2022] Alexey Ignatiev, Yacine Izza, Peter J. Stuckey, and Joao Marques-Silva. Using MaxSAT for efficient explanations of tree ensembles. In *Proceedings of AAAI'22*, pages 3776–3785, 2022.

[Miller, 2019] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[Molnar, 2019] Christoph Molnar. *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*. Leanpub, 2019.

[Nauta *et al.*, 2023] Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable AI. *ACM Computing Surveys*, 55(13s), 2023.

[Parmentier and Vidal, 2021] Axel Parmentier and Thibaut Vidal. Optimal counterfactual explanations in tree ensembles. In *Proceedings of ICML'21*, volume 139, 2021.

[Pedregosa *et al.*, 2011] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[Quinlan, 1986] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Schapire and Freund, 2014] Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2014.