On Tractable XAI Queries based on Compiled Representations

Gilles Audemard¹, Frédéric Koriche¹, Pierre Marquis¹,²
¹CRIL, Université d’Artois & CNRS, France
²Institut Universitaire de France, France
{audemard, koriche, marquis}@cril.fr

Abstract

One of the key purposes of eXplainable AI (XAI) is to develop techniques for understanding predictions made by Machine Learning (ML) models and for assessing how much reliable they are. Several encoding schemas have recently been pointed out, showing how ML classifiers of various types can be mapped to Boolean circuits exhibiting the same input-output behaviours. Thanks to such mappings, XAI queries about classifiers can be delegated to the corresponding circuits. In this paper, we present some explanation queries and verification queries about classifiers. We show how they can be addressed by combining queries and transformations about the associated Boolean circuits. Taking advantage of previous results from the knowledge compilation map, this allows us to identify a number of cases for which XAI queries are tractable provided that the circuit has been first turned into a compiled representation.

1 Introduction

In the past decade, ML techniques have revolutionized vision, speech, language understanding, and many other fields. However, the most powerful ML models (e.g., deep neural nets) in terms of quality of predictions are poorly explainable. For such black boxes, it is humanly impossible to understand the sequence of operations which are performed for predicting the label of an input instance. Thus, explaining the predictions made by such ML models (i.e., why a given instance is classified as such?) is not feasible in general. In essence, the impossibility to explain predictions and to determine the extent to which they are reliable forms a major obstacle in a number of applications for which safety is of paramount importance (like autonomous cars or medical diagnosis).

This calls for explanation technologies, enabling for reasoning about the decisions made by the classifier, measuring how robust they are, testing whether the classifier is biased and how much it complies with some prior knowledge about classes, etc. Accordingly, there has been a growing body of work on explainable and robust AI (XAI) for the past couple of years (Katz et al. 2019; Ignatiev, Narodytska, and Marques-Silva 2019; Bunel et al. 2018; Leofante et al. 2018; Molnar, Casalicchio, and Bischl 2018; Shih, Darwiche, and Choi 2019; Guidotti et al. 2019; Miller 2019; Molnar 2019). In this research topic, recent works have shown how ML classifiers $C$ of various types (including black boxes) can be associated with Boolean circuits $\Sigma$ (alias transparent or “white” boxes), exhibiting the same input-output behaviours (Narodytska et al. 2018; Shih, Choi, and Darwiche 2018a; Shih, Choi, and Darwiche 2019). Thanks to such mappings, XAI queries about classifiers can be delegated to the corresponding circuits (Darwiche and Hirth 2020). The rationale for this approach is twofold: on the one hand, $\Sigma$ as an abstraction of $C$ is typically far less complex than $C$; on the other hand, the Boolean nature of $\Sigma$ makes it amenable to various reasoning tasks required for generating explanations and addressing reliability issues.

The present study is relevant to this research trend within XAI. Our objective is to state in a formal way a number of XAI queries about $C$ and for each of them, to identify some sufficient conditions on the language $L$ used to represent $\Sigma$ that render the query tractable. For the sake of generality, we do not make any strong assumption on the nature of the classifier: $C$ can be a feedforward neural network, a Bayesian network classifier, a random forest, etc. We just assume that $C$ is a discrete multi-label classifier from which a Boolean circuit $\Sigma$ having the same input-output behaviour can be generated. More precisely, each input instance is a vector $x$ of $n$ values assigned to Boolean features from a set $X = \{x_1, \ldots, x_n\}$ and each corresponding output is a vector $y \in Y$ of $m$ values taken from a set $Y = \{y_1, \ldots, y_m\}$ of Boolean labels. Given the prediction $y = C(x)$, the instance $x \in X$ is classified by $C$ as an element of the $j$th class if and only if $y_j = 1$.

The variables of $\Sigma$ are split into three categories:

1. **Input variables** used to describe the input features of $C$.
   - Each input variable is a Boolean variable from $X$;

2. **Output variables** used to describe classes or labels. Each output variable is a Boolean variable from $Y$;

---

¹When an input feature $x_i$ is not Boolean, but takes values into a finite domain $\{v_i, 1, \ldots, v_i\}$, one can represent it using $p$ Boolean features $x_i^1, \ldots, x_i^p$ so that $x_i^j$ is true if and only if $x_i$ takes value $v_j$. The direct – or sparse – encoding (de Kleer 1989; Walsh 2000) of $x_i$ is given by the constraints $\wedge_{j=1}^{p-1} \wedge_{k=j+1}^{p} (\neg x_i^j \lor \neg x_i^k)$ and $\vee_{j=1}^{p} x_i^j$ ensuring that $x_i$ takes exactly one value in its domain.
### 2 Formal Preliminaries

**Propositional Languages.** The languages considered in this paper are defined over a finite set \( PS \) of propositional variables and a finite set of connectives. The elements of a propositional language \( \mathcal{L} \) are called representations, and for any such representation \( \Sigma \), we denote by \( Var(\Sigma) \) the subset of variables of \( PS \) occurring in \( \Sigma \). As usual, atomic representations include propositional variables in \( PS \), and Boolean constants in \{0, 1\}. A literal is a propositional variable, possibly negated, or a Boolean constant. Any propositional variable \( x \) is called a positive literal, and the negation of \( x \), denoted \( \neg x \) or \( \overline{x} \), is called a negative literal. A term is a conjunction of literals, and a clause is a disjunction of literals.

Given a set of variables \( X \subseteq PS \), an interpretation over \( X \) is a mapping \( \omega \) from \( X \) to \( \mathbb{B} = \{0, 1\} \). Propositional representations are interpreted in a classical way. For a representation \( \Sigma \) and an interpretation over \( X = Var(\Sigma) \), we use \( \omega \models \Sigma \) to denote the fact that \( \omega \) satisfies a model of \( \Sigma \) according to the semantics of propositional logic. That is, assigning the variables of \( \Sigma \) to truth values as specified by \( \omega \) makes \( \Sigma \) true. By \( [\Sigma] \) we denote the set of models of \( \Sigma \) over \( Var(\Sigma) \), and by \( \|\Sigma\| \) we denote the number of models of \( \Sigma \) over \( Var(\Sigma) \).

In the following, we present a number of XAI queries of practical interest for explaining classifications achieved by \( C \), or for assessing the robustness of \( C \). These queries are addressed by considering the circuit \( \Sigma \) associated with \( C \), instead of \( C \) itself. For the sake of clarity, we split XAI queries into explanation queries and verification queries. Explanation queries are concerned with the local interpretability issue, i.e., they are about a given input \( x \), while verification queries are concerned with the global interpretability issue, i.e., they are independent of any input. Some of the XAI queries considered in the following are brand new, others have been introduced in the recent past. In both cases, we show how these explanation and verification queries can be modeled using basic operations over Boolean circuits. Taking advantage of previous results from the knowledge compilation map for propositional languages \( \mathcal{L} \) (Darwiche and Marquis 2002), this allows us to identify a number of XAI queries which are tractable, provided that the circuit \( \Sigma \) has been first turned into a compiled representation from \( \mathcal{L} \).

The rest of the paper is organized as follows. After some formal preliminaries about propositional logic and the knowledge compilation map, we present a simple encoding schema that can be used to associate a Boolean circuit \( \Sigma \) with a given random forest \( C \) over Boolean features. A toy example is provided, that will serve as a running example for illustrating the XAI queries presented afterwards. Then, we successively present some explanation queries and some verification queries, and for each of them, we identify some sufficient conditions on the propositional language used to represent \( \Sigma \) which ensure that the query is tractable (i.e., it can be answered using a polynomial-time algorithm). Finally, we discuss the results and conclude the paper.
following queries and transformations:

- **Queries**

  - **consistency:** \( L \) satisfies CO if and only if there is a polynomial-time algorithm that maps every representation \( \Sigma \) from \( L \) to 1 if \( \Sigma \) is consistent, and to 0 otherwise.

  - **implicant:** \( L \) satisfies IM if and only if there is a polynomial-time algorithm that maps every representation \( \Sigma \) from \( L \) and every term \( \gamma \) to 1 if \( \gamma |\Sigma \) holds, and to 0 otherwise.

  - **sentential entailment:** \( L \) satisfies SE if and only if there is a polynomial-time algorithm that maps any two representations \( \Sigma_1 \) and \( \Sigma_2 \) from \( L \) to 1 if \( \Sigma_1 \models \Sigma_2 \) holds, and to 0 otherwise.

  - **equivalence:** \( L \) satisfies EQ if and only if there is a polynomial-time algorithm that maps any two representations \( \Sigma_1 \) and \( \Sigma_2 \) from \( L \) to 1 if \( \Sigma_1 \equiv \Sigma_2 \) holds, and to 0 otherwise.

  - **model counting:** \( L \) satisfies CT if and only if there is a polynomial-time algorithm that maps every representation \( \Sigma \) from \( L \) to a nonnegative integer (in binary notation) corresponding to the number of models \( ||\Sigma|| \) of \( \Sigma \) over \( Var(\Sigma) \).

  - **model enumeration:** \( L \) satisfies ME if and only if there is an enumeration algorithm with polynomial description from \( \Sigma \) corresponding to the number of models \( ||\Sigma|| \) of \( \Sigma \) over \( Var(\Sigma) \).

- **Transformations**

  - **conditioning:** \( L \) satisfies CD if and only if there is a polynomial-time algorithm that maps every representation \( \Sigma \) from \( L \) and every consistent term \( \gamma \) to a representation from \( L \) that is logically equivalent to \( \Sigma \models \gamma \).

  - **bounded conjunction:** \( L \) satisfies \( \land BC \) if and only if there is a polynomial-time algorithm that maps every pair of representations \( \Sigma_1 \) and \( \Sigma_2 \) from \( L \) to a representation from \( L \) that is logically equivalent to \( \Sigma_1 \land \Sigma_2 \).

  - **forgetting:** \( L \) satisfies FO if and only if there is a polynomial-time algorithm that maps every representation \( \Sigma \) from \( L \) and every subset \( X \) of variables from \( PS \) to a representation from \( L \) equivalent to \( \exists X. \Sigma \).

  - **optimization:** \( L \) satisfies OPT if and only if there exists a polynomial-time algorithm that maps every representation \( \Sigma \) from \( L \) and any linear function \( f \) over \( Var(\Sigma) \) into a representation from \( L \) for which the models are those of \( \Sigma \) minimizing the value of \( f \).

We also add to the list of properties the following transformation that will be useful in the rest of the paper:

- **decomposable conjunction:** \( L \) satisfies \( \land DC \) if and only if there is a polynomial-time algorithm that maps \( k \) representations \( \Sigma_1, \cdots, \Sigma_k \) from \( L \) such that \( Var(\Sigma_i) \cap Var(\Sigma_j) = \emptyset \) (for \( i, j \in \{1, \cdots, k\}, i \neq j \)) to a representation from \( L \) that is logically equivalent to \( \bigwedge_{i=1}^{k} \Sigma_i \).

In the KC map, the space efficiency or succinctness of propositional languages is also assessed. Informally, the succinctness captures the relative ability of propositional languages to encode pieces of information using little space. For example, it is known that \( \text{DNF} \) is strictly more succinct than \( \text{OBDD} \) and \( \text{DNF} \). A language suited for a given application is then viewed as one of the most succinct languages offering the queries and transformations required by the application (Darwiche and Marquis 2002).

### 3 Encoding Random Forests

In order to illustrate the approach into which our work takes place and the forthcoming XAI queries, let us first explain how to associate a Boolean circuit \( \Sigma \) with a given random forest \( C \) over Boolean features (the proposed encoding is quite easy).

A decision tree (here, a classification tree) (Breiman et al. 1984; Quinlan 1986) is a finite tree \( T \) where each internal node is a decision node labelled by a feature \( x \) from \( X \) and having as many children as the cardinality of the domain of \( x \), and each leaf node is labelled by a non-empty subset of \( Y \). The arcs from a decision node to its children are labelled by pairs noted \( x = v \) where \( v \) is an element of the domain of the feature \( x \) labelling the node (one value \( v \) per arc). A path of \( T \) is a (finite) sequence of labels \( x = v \) of the arcs encountered from the root of \( T \) to one of its leaves.

A decision tree classifies an input \( x \) as an element of class \( y \) if the unique path of \( T \) that is compatible with \( x \) (i.e., that contains a pair \( x = v \) only if \( x \) takes the value \( v \) in \( x \)) leads to a leaf node whose label contains \( y \).

A random forest \( R \) (Ho 1998; Breiman 1996; Breiman 2001) is a finite set of \( p \) decision trees over the set \( S \) of features. Given a threshold \( \tau \in \{1, \ldots, p\} \), a random forest \( R = \{T_1, \ldots, T_p\} \) classifies an input \( x \) as \( y \in Y \) if at least \( \tau \) decision trees of \( R \) classify \( x \) as \( y \).

**Example 1.** As a matter of example, consider a fruit classification task and the random forest \( R \) given by the trees \( T_1, T_2 \) and \( T_3 \), illustrated in Figure 1. Each decision tree is defined over the set of input features \( X = \{\text{sw(eet taste)}, \text{gr(een)}, \text{ro(und shape)}\} \), and the set of output labels \( Y = \{\text{GRA(pe fruit)}, \text{BAN(ana)}, \text{APP(le)}\} \). An instance \( x \) is classified as a label \( y \) if and only if it is classified as such by a strict majority of decision trees in \( R \) (i.e., at least \( \tau = 2 \) decision trees of \( R \) classify \( x \) as \( y \)).

In Figure 1, the labels \( x = v \) of the arcs are not explicitly represented. A dashed line from a decision node labelled by \( x \) corresponds to the label \( x = 0 \), while a plain line corresponds to the label \( x = 1 \).

Based on this example, the input instance \( x_1 \) given by \( (\text{sw} = 0, \text{gr} = 0, \text{ro} = 0) \) is classified by \( T_1 \) (resp. \( T_2, T_3 \)) as an element of \( \text{GRA} \) (resp. \( \text{GRA} \) or \( \text{BAN} \), \( \text{BAN} \)). The

\(^2\)Actually, this requirement is stronger than the one considered in (Darwiche and Marquis 2002) which only required that the set of models can be enumerated in time polynomial in the size of \( \Sigma \) plus the size of the corresponding set of models. Note that we have that \( L \) satisfies ME (in the strong sense) whenever it satisfies CD and CO (see (Fargier and Marquis 2014) for details).
For every \( \Sigma \) is the conjunction of two kinds of constraints:

1. For every \( x_2 \) defined by \((\text{SW} = 1, \text{GR} = 1, \text{RO} = 0)\) is classified by \( T_1 \) (resp. \( T_2, T_3 \)) as an element of \( \text{GRA} \) (resp. \( \text{BAN}, \text{BAN} \)). Thus, \( R \) classifies \( x_1 \) as an element of \( \text{GRA} \) or \( \text{BAN} \), and \( R \) classifies \( x_2 \) as an element of \( \text{BAN} \). Note that \( R \) may remain mute about a given instance, meaning that there is no evidence enough to classify it as any element of \( Y \). For instance, \( R \) does not predict anything about the input instance \((\text{SW} = 1, \text{GR} = 1, \text{RO} = 1)\), since this input is classified by \( T_1 \) (resp. \( T_2, T_3 \)) as an element of \( \text{GRA} \) (resp. \( \text{BAN}, \text{APP} \)). Thus, the strict majority threshold of 2 is not met for this instance.

Based on these considerations, let us present an encoding schema which maps any random forest \( R \), defined on the input variables \( X \) and the output variables \( Y \), to a representation \( \Sigma \) over the set of propositional variables \( PS = X \cup Y \cup Z \), where \( Z \) captures the intermediate variables. \( \Sigma \) is the conjunction of two kinds of constraints:

- for every \( j \in \{1, \ldots, m\}, \Sigma \) includes the constraint

\[
y_j \iff \left( \sum_{k=1}^{p} z_{j}^{k} \geq \tau \right)
\]

where variable \( z_{j}^{k} \) is true if and only if the input instance \( x \) is classified as an element of class \( y_j \) by \( T_k \). Accordingly, the input is classified as \( y_j \) by \( R \) if and only if the number of decision trees of \( R \) classifying \( x \) as an element of \( y_j \) is at least equal to the threshold \( \tau \);

- for every \( j \in \{1, \ldots, m\} \) and every \( k \in \{1, \ldots, p\}, \Sigma \) includes a constraint

\[
z_{j}^{k} \iff \left( \gamma \in \Gamma(T_k, y_j) \right)
\]

indicating that \( z_{j}^{k} \) is set to true (i.e., \( x \) is classified as an element of \( y_j \) by \( T_k \)) if and only if \( x \) is compatible with a path of \( T_k \) leading to a leaf node labelled by \( y_j \). Here, \( \Gamma(T_k, y_j) \) denotes the set of all terms \( \gamma \) encoding the paths of \( T_k \) which lead to leaf nodes labelled by \( y_j \).

For our running example, \( R \) is associated with the representation \( \Sigma \), which is given by the conjunction of the following constraints:

\[
\begin{align*}
\text{GRA} &\iff (\text{GRA}^1 + \text{GRA}^2 + \text{GRA}^3 \geq 2) \\
\text{BAN} &\iff (\text{BAN}^1 + \text{BAN}^2 + \text{BAN}^3 \geq 2) \\
\text{APP} &\iff (\text{APP}^1 + \text{APP}^2 + \text{APP}^3 \geq 2) \\
\text{GRA}^1 &\iff (\text{SW} \land \text{GR} \lor (\text{SW} \land \text{GR} \land \text{RO})) \\
\text{BAN}^1 &\iff (\text{SW} \land \text{GR} \land \text{RO} \lor (\text{SW} \land \text{GR} \land \text{RO})) \\
\text{APP}^1 &\iff (\text{SW} \land \text{GR} \lor (\text{SW} \land \text{GR} \land \text{RO})) \\
\text{GRA}^2 &\iff (\text{SW} \land \text{GR} \lor (\text{SW} \land \text{GR} \land \text{RO})) \\
\text{BAN}^2 &\iff (\text{SW} \land \text{GR} \land \text{RO} \lor (\text{SW} \land \text{GR} \land \text{RO})) \\
\text{GR}^3 &\iff (\text{SW} \land \text{GR} \land \text{RO}) \\
\text{BAN}^3 &\iff (\text{SW} \land \text{GR} \land \text{RO}) \\
\text{APP}^3 &\iff (\text{SW} \land \text{GR} \land \text{RO} \lor (\text{SW} \land \text{GR} \land \text{RO}))
\end{align*}
\]

From \( \Sigma \), the following constraints expressing how each class is inferred from the input features can be derived as logical consequences:

\[
\begin{align*}
\text{GRA} &\iff (\text{SW} \land \text{GR}) \\
\text{BAN} &\iff (\text{SW} \land \text{GR} \lor (\text{SW} \land \text{GR} \land \text{RO})) \\
\text{APP} &\iff (\text{SW} \land \text{GR} \lor (\text{SW} \land \text{GR} \land \text{RO})) \land \text{RO}
\end{align*}
\]

Equivalently, the projection of \( \Sigma \) onto \( X \cup Y \) is given by the truth assignments in Table 1.

### 4 XAI Queries

After illustrating how multi-labels classifiers can be encoded into propositional representations, we are now in position to examine XAI queries. We first present a couple of results, showing how the notion of membership of an instance \( x \) to a class of \( Y \) as predicted by \( C \) can be translated in logical terms when \( \Sigma \) (represented using a propositional language \( L \)) is considered instead of \( C \).

To this point, observe that any input instance \( x \) of the form \((x_1 = v_1, \ldots, x_n = v_n)\) can be encoded as a term over the literals on \( X \), also denoted \( x \) to avoid heavy notations. Specifically, \( x \) is given by \( \bigwedge_{j=1}^{n} \ell_j \) where \( \ell_j = x_j \) if \( v_j = 1 \) and \( \ell_j = \pi_j \) otherwise.

<table>
<thead>
<tr>
<th>SW</th>
<th>GR</th>
<th>RO</th>
<th>GRA</th>
<th>BAN</th>
<th>APP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: The projection of \( \Sigma \) over \( X \cup Y \).
Now, since $\Sigma$ has the same input-output behaviour as $C$, we know that every variable from $Y \cup Z$ is defined in $\Sigma$ in terms of $X$. In other words, for any input $x$, we have $C(x) = (y_1 = c_1, \ldots, y_m = c_m)$ if and only if the unique model of $\Sigma$ over $X \cup Y \cup Z$ which is compatible with $x$ assigns to each output variable $y_j$ ($j \in \{1, \ldots, m\}$) the truth value $c_j$. Being “defined” means here that for any variable $u \in Y \cup Z$ there exists a formula $\varphi_u$ built upon variables of $X$, only, that is such that $\Sigma \models u \iff \varphi_u$ (Beth 1953; Lang and Marquis 2008).

Based on these notions, the input instances $x$ which are classified by $C$ as elements of class $y_j$ in the sense that $y_j = 1$ when $C(x) = (y_1 = c_1, \ldots, y_m = c_m)$ can be characterized in several ways using $\Sigma$:

- **abductive characterization**: $x$ is an abductive explanation of $y_j$ with respect to $\Sigma$ provided that the set of assumptions is the set of all literals over the variables from $X$. Indeed, if $y_j$ is true in $C(x)$ then we have $\Sigma \land x \models y_j$. Note here that $\Sigma \land x$ is consistent for every input instance $x$ over $X$, since $\Sigma$ is a Boolean circuit with inputs in $X$.

- **model-based characterization**: $x$ is a model over $X$ of $\exists x.(\Sigma \land y_j)$, or equivalently a model over $X$ of $\exists x.(\Sigma \mid y_j)$. Thus, the projection of $\Sigma \mid y_j$ onto $X$ can be viewed as a compact representation of all the input instances $x$ viewed as elements of $y_j$ by $C$.

### 4.1 Verification Queries

As indicated above, the XAI queries under consideration in this study are separated into verification and explanation queries. We start with the verification queries.

**Counting the inputs (CIN) / enumerating the inputs (EIN) associated with a given class.** These queries are useful for a user in order to figure out the classes of $Y$ as they are recognized by $C$ (which may differ from what the user has in mind). For instance, in our running example, the user might be surprised by the fact that the instance given by $\text{SW} \land \text{GR} \land \text{RO}$ is classified as an element of $\text{GRA}$, and hence, she would like to gather more information about instances labeled by this class.

**Proposition 1.** Counting the inputs associated by $C$ with a given class is in $P$ when $L$ satisfies CD and CT. Enumerating those inputs with polynomial delay is feasible when $L$ satisfies CD and ME.

**Proof.** Counting the inputs $x$ associated by $C$ with a given output $y_j$ amounts to computing $|\Sigma \mid y_j|$. Enumerating those inputs boils down to enumerating the models of $\exists x.(\Sigma \mid y_j)$ with polynomial delay, which is feasible as soon as $L$ satisfies CD, FO, and ME. Since every variable of $\text{Var}(\Sigma)$ not belonging to $X$ is defined from $X$ in $\Sigma$, we can just enumerate the models of $\Sigma \mid y_j$, filtering out from each of them all the literals over the intermediate variables from $Z$. Thus, the enumeration query is tractable as soon as $L$ satisfies CD and ME.

**Counting the inputs (CAM) / enumerating the inputs (EAM) for which the classifier provides ambiguous outputs / remains mute.** The CAM query is important for estimating how resolute the classifier $C$ is. Indeed, as much $C$ provides ambiguous outputs or remains mute for input instances, as much it will be hard to take advantage of it for making decisions. The EAM query is also useful, as it points out instances for which ambiguities or omissions arise. In our running example, only one instance, given by the term $\text{SW} \land \text{GR} \land \text{RO}$, is ambiguously classified by $C$. Similarly, only one instance, given by $\text{SW} \land \text{GR} \land \text{RO}$, is not recognized by $C$ as a member of any class from $Y$.

**Proposition 2.** Counting the inputs for which $C$ provides ambiguous outputs / remains mute is in $P$ when $L$ satisfies CD and CT. Enumerating with polynomial delay the inputs that are ambiguously recognized by $C$ as elements of the intersection of a given set of classes / the inputs for which $C$ remains mute, is feasible when $L$ satisfies CD and ME.

**Proof.** Counting the inputs for which $C$ remains mute amounts to computing $A = |\Sigma \mid \bigwedge_{j=1}^m y_j|$. Furthermore, the number of inputs mapped by $C$ into a single class $y_j$ is given by $B_j = |\Sigma \mid y_j \land \bigwedge_{k \in \{1, \ldots, m\} \setminus \{j\}} y_k|$. Therefore, counting the number of inputs for which $C$ provides ambiguous outputs amounts to evaluating $|X| - A - \sum_{j=1}^m B_j$. The instances classified as elements of the intersection of a given set of $S \subseteq Y$ of classes are given by the models over $X$ of $\Sigma \mid \gamma_S$, where $\gamma_S = \bigwedge_{y_j \in S} y_j$, and those for which $C$ remains mute correspond to the models over $X$ of $\Sigma \mid \bigwedge_{j=1}^m y_j$. Thus, they can be enumerated with polynomial delay from $S$ when $L$ satisfies CD and ME.

**Measuring the frequency of some feature in a given class (MFR) / identifying mandatory or forbidden features (IMA).** MFR aims to evaluate how significant is the presence of each feature $x_k$ in each class of interest $y_j$. More generally, one can consider a combination of features (each of them being either present or absent) instead of a single feature being present. We can also consider a combination of classes (again, each of them being either met or not). When the frequency is equal to 1, $x_k$ is mandatory for being recognized as an element of $y_j$, while when it is equal to 0, it is forbidden for being recognized as an element of $y_j$. In our running example, the frequency of $\text{SW}$ (resp. $\text{GR}$, $\text{RO}$) in class $\text{APP}$ is $\frac{1}{2}$ (resp. $\frac{1}{2}$, 1). Thus, none of the three features is forbidden for the class of apples, but having a round shape ($\text{RO}$) is mandatory.

**Proposition 3.** Computing the frequency of a feature (or a combination of features) in a given class (or combination of classes) is in $P$ when $L$ satisfies CD and CT.

**Proof.**
- The frequency of $x_k$ in $y_j$ is given by $\frac{|\Sigma \mid (y_j \land x_k)|}{|\Sigma \mid y_j|}$. When a more complex combination of features $l$ classes is considered, it is sufficient to replace the terms $y_j \land x_k$ (resp.
Obdd is tractable when an
wiche 2018a), showing that identifying irrelevant features
heres with a result reported in (Shih, Choi, and Dar-
• Determining whether $x_k$ is mandatory (resp. forbidden)
for $y_j$, amounts to deciding whether or not $\Sigma \mid (y_j \land x_k)$
is inconsistent (resp. $\Sigma \mid (y_j \land x_k)$ is inconsistent), which
can be achieved in polynomial time when $\mathcal{L}$ satisfies CD and CO.
When a more complex combination of features
i classes is considered, one consistency test per feature
must be done, replacing $y_j$ by the combination of classes
under consideration. Since the combination of features is
part of the input, the problem can still be solved in
polynomial time when $\mathcal{L}$ satisfies CD and CO.

Identifying irrelevant features for a given class (IIR).
In many scenarios, the user has some beliefs about the features
that must be relevant (or not), for some classes. For instance,
she might expect that the shape of a fruit is relevant for
determining whether it is a banana - this is indeed the case in
our running example. Dually, the user could believe that a
given class should not depend on some features, which oth-
erwise would reveal a bias in the classifier. For instance, she
might expect that the class of bananas should not depend on
Gr - again, this is the case in our running example.

Recall that an input feature $x_i$ is irrelevant for a class $y_j$
if the value of $x_i$ can be switched without changing the fact
that the instance is predicted by the classifier as an element
of $y_j$. Stated otherwise, the membership in $y_j$ of any in-
stance $x$ does not depend on the value of its feature $x_i$.

Proposition 4. Determining the features that are considered
as useless by $C$ for a given class is in $\mathbb{P}$ whenever $\mathcal{L}$ satisfies
CD, FO, and EQ.

Proof. $x_i$ is irrelevant for $y_j$ if and only if $\exists \mathcal{X} \cdot (\Sigma \mid y_j)$
is independent from $x_i$ (Lang, Liberatore, and Marquis 2003).
This amounts to testing whether ($\exists \mathcal{X} \cdot (\Sigma \mid y_j)$) $\wedge (\exists \mathcal{X} \cdot (\Sigma \mid y_j))$ is equivalent, or not. This can be done
in polynomial time when $\mathcal{L}$ satisfies CD, FO, and EQ.

When considering Boolean features, this proposition co-
heres with a result reported in (Shih, Choi, and Dar-
wo 2018a), showing that identifying irrelevant features is tractable when an OBDD representation of $\exists \mathcal{X} \cdot (\Sigma \mid y_j)$ is
provided.

Identifying monotone/anti-monotone features (IMO).
In many applications, it is believed that increasing the value
of some feature does not change the membership to some
class. Dually, one might also expect that decreasing the
value of another feature does not change the membership
to that class. For example, if a given fruit with a non-green
color is recognized as an apple, then switching its color to
green should not change the fact that it is still an apple. By
contrast, if a fruit with sweet taste is recognized as a grape,
then it is reasonable to expect that it would remain a grape
fruit if its taste was sour (i.e., not sweet).

In more formal terms, a classifier $C$ is monotone (resp.
anti-monotone) for a given class $y_j$, with respect an input
feature $x_i$, if for any input instance $x$ such that $C(x) = y$
with $y_j = 1$ implies that $C(x[x_i ← 1]) = y'$ (resp. $C(x[x_i ← 0]) = y'$) with $y'_j = 1$.

Proposition 5. Checking whether the classifier is monotone
(or anti-monotone) for a class $y_j$ with respect to an input
feature $x_i$ is in $\mathbb{P}$ whenever $\mathcal{L}$ satisfies CD, FO, and SE.

Proof. At the semantics level, conditioning a propositional
representation $\varphi$ by a literal $\ell$ (resp. $\neg \ell$) consists first in
selecting the models $\omega$ of $\varphi$ over $\mathcal{V} \mathcal{A} \varphi (\varphi)$ where $\ell$ is assigned to
1 (resp. 0), then in forgetting $\ell$ in the result (which leads to
add this to set of models every interpretation that coincides
with one of the $\omega$ but is such that $\ell$ is assigned to 0 (resp.
1 in it). Using symbols, $\varphi \mid x \equiv \exists \{x\}, (\varphi \land x)$ and $\varphi \mid \neg \ell \equiv \exists \{x\}, (\varphi \land \neg \ell)$). Accordingly, if $C$ is not monotonic
(resp. anti-monotonic) for $y_j$ with respect to $x_i$, then there is
a model of $\exists \mathcal{X} \cdot (\Sigma \mid y_j)$ which is not a model of $\exists \mathcal{X} \cdot (\Sigma \mid y_j)$
conditioned by $x_i$ (resp. $\neg x_i$). Thus, checking whether $C$
is monotonic (resp. anti-monotonic) for $y_j$ w.r.t. $x_i$ amounts
to testing whether $\exists \mathcal{X} \cdot (\Sigma \mid y_j) \models (\exists \mathcal{X} \cdot (\Sigma \mid y_j)) \wedge (\exists \mathcal{X} \cdot (\Sigma \mid y_j)) \equiv \exists \mathcal{X} \cdot (\Sigma \mid y_j) \mid x_i$ (resp. $\exists \mathcal{X} \cdot (\Sigma \mid y_j) \models (\exists \mathcal{X} \cdot (\Sigma \mid y_j)) \mid \neg x_i$). This can be done in
polynomial time when $\mathcal{L}$ satisfies CD, FO, and SE.

In our running example, one can check that the random
forest $R$ is monotonic for the class GRA with respect to the
feature SW. We mention in passing that the notion of mono-
tonicity has already been considered in (Shih, Choi, and Dar-
wo 2018a) for binary classifiers (i.e., when $m = 1$).

Determining how many classes are close to each other
(MCJ, MCH).
In multi-label classification, the user may
have some beliefs about the extent to which the classes of $Y$
are close to each other. For example, it can be expected that
the class of apples is at least as close to the class of grape
fruits as to the class of bananas. Thus, it is interesting to
determine whether or not the closeness of classes (as they are
identified by the predictor) is compatible with those beliefs.
Such a consideration requires a formal characterization of
the notion of “similarity”.

A simple, yet poorly informative notion of similarity is
ordained and takes the form of a set of pre-orders $\leq y_j$ over $Y$,
one for each class $y_j \in \{1, \cdots, m\}$. By writing $y_k \leq y_j$ $y_j$,$y_i$, we mean that $y_k$ is at least as close to $y_j$ as to $y_i$. Thus,
the user may believe that $\text{app} \leq \text{GRA}$ BAN. Of course, it is
expected that $y_j \leq y_j$ $y_i$ whatever $y_i \in Y$. Clearly enough,
every cardinal similarity $\sigma$ between classes of $Y$ (i.e., a sym-
metric mapping associating a non-negative number with a
couple of classes), induces an ordinal similarity given by
$y_k \leq y_j$ $y_j$ if and only if $\sigma(y_k, y_j) \geq \sigma(y_j, y_j)$.

Several cardinal similarities can be defined and evaluated
from $\Sigma$. A first one is given by the Jaccard index $J(y_k, y_j)$:
it is the number of elements belonging to both classes, di-
vided by the number of elements belonging to at least one
of them (i.e., the cardinality of the intersection of the two
classes divided by the cardinality of their union). $J(y_k, y_j)$
indicates the proportion of inputs that are (possibly wrongly)

- If $x = (x_1, \cdots, x_n)$, then $x[x_k ← v]$ is the same vector as $x$,
except that the $j$th coordinate $x_k$ of $x[x_k ← v]$ has value $v$.

\footnote{If $x = (x_1, \cdots, x_n)$, then $x[x_k ← v]$ is the same vector as $x$,
except that the $j$th coordinate $x_k$ of $x[x_k ← v]$ has value $v$.}
recognized as both $y_k$ and $y_l$ among those which are recognized as belonging to at least one of the two classes. $J(y_k, y_l)$ varies between 0 and 1, the higher the more similar $y_k$ and $y_l$. When classes are not given explicitly as the sets of their elements but represented implicitly using a circuit $\Sigma$ as done in our setting, we get:

$$J(y_k, y_l) = \frac{||\Sigma \wedge (y_k \land y_l)||}{||\Sigma \wedge (y_k \lor y_l)||}$$

Clearly, when the classes $y_k$ and $y_l$ are pairwise disjoint (as it should be for apples and bananas, ideally) one must have $J(y_k, y_l) = 0$. In the running example, we can check from Table 1 that $J(BAN, GRA) = \frac{1}{3}$ while $J(APP, GRA) = 0$. Thus, from the point of view of Jaccard index, bananas are viewed as closer to grape fruits than apples, which can be considered as counter-intuitive by the user and reflect a problem in the predictor.

Sometimes, it makes sense to evaluate the distance between two classes $y_k$ and $y_l$ as an aggregation of a more elementary distance between the elements of the two classes. For example, we can take the Hamming distance between an element of $y_k$ and an element of $y_l$. In doing so, let $d_H(x, x')$ denote the number of bits that differ between the inputs $x$ and $x'$ from $X$. This distance can be easily lifted to propositional representations $\Sigma_1$ and $\Sigma_2$ over $X$ encoding sets of such inputs:

$$d_H(\Sigma_1, \Sigma_2) = \min_{\omega_1 \in [\Sigma_1]} \min_{\omega_2 \in [\Sigma_2]} d_H(\omega_1, \omega_2)$$

and then the cardinal similarity $H$ between classes can be defined as follows:

$$H(y_k, y_l) = \frac{n - d_H(\exists X. (\Sigma \mid y_k), \exists X. (\Sigma \mid y_l))}{n}$$

Thus, $H(y_k, y_l)$ gives the proportion of (Boolean) features that do not need to be switched to go from an element of $y_k$ to an element of $y_l$. In the running example, we can check from Table 1 that $H(BAN, GRA) = 1$ (since both classes share an element) while $H(APP, GRA) = \frac{2}{3}$. From the point of view of $H$ similarity, bananas are also viewed as closer to grape fruits than apples.

**Proposition 6.**
- MCI: computing $J(y_k, y_l)$ is in $P$ when $\Lambda$ satisfies CD and CT.
- MCH: computing $H(y_k, y_l)$ is in $P$ when $\Lambda$ satisfies CD, $\land BC$, $\land DC$, OPT, and ME.

**Proof.**
- As to $J(y_k, y_l)$, it is enough to observe that $J(y_k, y_l)$ is also equal to

$$\frac{||\Sigma \mid (y_k \land y_l)||}{||\Sigma \mid (y_k \land y_l)|| + ||\Sigma \mid (y_k \lor y_l)|| + ||\Sigma \mid (\overline{y_k} \land y_l)||}.$$  

$^3$Note that the Hamming distance between two assignments of a feature $x_i$ over a finite, yet non-Boolean domain, can be computed as the weighted Hamming distance between the corresponding assignments of the Boolean variables $x'_i$ obtained using the direct encoding (just consider a weight of $\frac{1}{2}$ for each $x'_i$).

$^3$We define $d_H(\Sigma_1, \Sigma_2) = +\infty$ whenever $\Sigma_1$ is inconsistent or $\Sigma_2$ is inconsistent.

- The computation of $H(y_k, y_l)$ is a bit more subtle. Consider a representation $\Sigma$ in $\Lambda$ and a “clone” $\Sigma'$ of it obtained by renaming (in a uniform way) every variable $v$ occurring in $\Sigma$ into $v'$. Then condition $\Sigma$ by $y_k$ and $\Sigma'$ by $y_l$, and compute a representation $\alpha$ in $\Lambda$ of the conjunction of the resulting representations. This can be done in time polynomial in the size of $\Sigma$ when $\Lambda$ satisfies CD and $\land ABC$. The next step consists in computing a representation in $\Lambda$ of each formula $z_i \oplus (x_i \Leftrightarrow x'_i)$ where $i \in \{1, \ldots, n\}$ and $z_i$ is a fresh variable of $PS$. This can be done in constant time since every such formula has a fixed size and involves only 3 variables. Then since $\Lambda$ satisfies $\land DC$, a representation $\beta$ in $\Lambda$ of $\bigwedge_{i=1}^n (z_i \oplus (x_i \Leftrightarrow x'_i))$ can be computed in polynomial time. Finally, a representation $\gamma$ in $\Lambda$ of $\alpha \land \beta$ can be computed in polynomial time when $\Lambda$ satisfies $\land BC$. Consider now a linear function $f$ over $Var(\alpha \land \beta)$ associating with every variable in it the weight 0, except the variables $z_1, \ldots, z_n$ that have weight 1. Since $\Lambda$ satisfies OPT, a representation $\delta$ in $\Lambda$ having for models those of $\gamma$ minimizing the value of $f$ can be computed in polynomial time from $\gamma$ and $f$. By construction, every such model contains a minimum number of variables $z_i$ set to 1, and this number is equal to $d_H(\exists X. (\Sigma \mid y_k), \exists X. (\Sigma \mid y_l))$ (note the the projection onto the variables of $X$ is done implicitly by the optimization transformation when the variables out of $X$ are given a null weight in the linear function). Because $\Lambda$ satisfies ME, a model $\omega$ of $\delta$ can be generated in polynomial time. Once computed, it is enough to count the number of variables $z_i$ set to 1 in it to get $d_H(\exists X. (\Sigma \mid y_k), \exists X. (\Sigma \mid y_l))$, from which $H(y_k, y_l)$ can be easily derived.

□

**Computing how far a class is from a given prototype (MCP).** In many cases, the user has in mind a prototypical example of the class. For instance, she may believe that an apple has a round shape, is not green, and has a sweet taste, which yields to the prototype $SW \land GR \land RO$. Based on this background information, it is interesting to determine how much each class (as it is recognized by $C$) complies with this prototype, which can be evaluated by computing the Hamming distance between every element of the class and the prototype, and considering the maximal distance. In our running example, the instance $SW \land GR \land RO$ classified as an apple by $R$ coincides with the prototype, while the other instance $SW \land GR \land RO$ classified as an apple by $R'$ is at a distance of 2 from the prototype $SW \land GR \land RO$. Thus the distance of the class of apples (as they are recognized by $R$) to the corresponding prototype is 2. A large value of this distance can be considered as unacceptable, as it may reflect a bias in the data used to train $C$.

**Proposition 7.** Computing how far a given class $y_l$ is from a given prototype $x$ is in $P$ when $\Lambda$ satisfies CD, OPT, and ME.

**Proof.** The approach for computing this value is as follows. One first generates a representation $\alpha$ in $\Lambda$ of $\Sigma \mid y_l$, which
is feasible in polynomial time when $L$ satisfies CD. Assuming that $x = (x_1 = v_1, \ldots , x_n = v_n)$, one then considers a linear function $f_a$ that associates a null weight with every variable from $Y \cup Z$, and with every $x_i$ ($i \in \{1, \ldots , n\}$), a weight equal to $-1$ if $v_i = 1$ and a weight equal to $1$ when $v_i = 0$. Accordingly, for every truth assignment $\omega$ over $X \cup Y \cup Z$, we have that $f_a(\omega) + \sum_{i=1}^{n} v_i$ is equal to the Hamming distance between the projection of $\omega$ onto $X$ and $x$. Since $\sum_{i=1}^{n} v_i$ does not depend on $\omega$, the projections onto $X$ of the interpretations $\omega$ minimizing the value of $f_a$ are those maximizing the Hamming distance to $x$. Obviously enough, they are also (precisely) the projections onto $X$ of the interpretations $\omega$ with $x$ coheres with $x$ on the ones in the sense that for any $k \in \{1, \ldots , n\}$, if $x'_k = 0$ then $x_k = 1$, and $x'$ has a minimal number of coordinates set to $1$. Roughly speaking, the features that are set to $1$ in $x'$ are enough to explain why $x$ has been classified by $C$ as an element of $y_j$. In our running example, a minimum-cardinality explanation of input $SW \land GR \land RO$ (classified as an element of $ban$ by $R$) is $RO$.

**Proposition 8.** The minimum-cardinality explanations of $x$ can be enumerated with polynomial delay from $\Sigma$ given $y_j$ as soon as $L$ satisfies CD, OPT, and ME.

**Proof.** First, one computes in polynomial time a representation $\alpha$ in $L$ of $\Sigma \mid y_j$. Then one considers the linear function $f$ associating with every variable in $Y \cup Z$ the weight $0$, except the variables of $X$ that have weight $1$. Since $L$ satisfies OPT, a representation $\beta$ in $L$ having for models those of $\alpha$ minimizing the value of $f$ can be computed in polynomial time from $\beta$ and $f$. By construction, the projection of every such model onto $X$ is a minimum-cardinality explanation of $x$. Furthermore, since every variable from $Y \cup Z$ is defined from $X$ in $\Sigma$, two distinct models of $\beta$ over $\text{Var}(\beta)$ necessarily have two distinct projections onto $X$. Thus, enumerating the models of $\beta$ and projecting each of them onto $X$ as soon as it is generated leads to enumerating the minimum-cardinality explanations of $x$. The fact that $L$ satisfies ME is therefore enough to conclude the proof.

The above proposition extends a result from (Shih, Choi, and Darwiche 2018b), where it was shown that the enumeration of minimum-cardinality explanations is feasible with a polynomial delay when $L = \text{OBDD}$. Indeed, $\text{OBDD}$ satisfies CD, OPT, and ME, but many other languages in the DNF family (including DNNF itself) are strictly more succinct than $\text{OBDD}$ and also satisfy those properties.

Note that mandatory features are relevant to minimum-cardinality explanations in the sense that every minimum-cardinality explanation of $x$ necessarily contains every feature that is mandatory for being recognized as an element of $y_j$ by $C$. Similarly, no irrelevant feature for $y_j$ participates in any minimum-cardinality explanation.

**Deriving a prime implicant explanation (DPI).** Given an input $x$ such that $C(x) = y$ with $y_j = 1$, a prime implicant explanation is a subset-minimal partial assignment $x'$ which is coherent with $x$ (i.e., $x$ and $x'$ give the same values to the variables that are assigned in $x'$) and which satisfies the property that for every extension $x''$ of $x'$ over $X$, we have $C(x'') = y''$ with $y''_j = 1$. The features assigned in $x'$ (and the way they are assigned) can be viewed as explaining why $x$ has been classified as an element of $y_j$. Thus, in the running example, a prime implicant explanation of input $SW \land GR \land RO$ (classified as an element of GRA by $R$) is $SW \land GR$. The fact that $RO$ holds or not does not matter.

**Proposition 9.** Deriving a prime implicant explanation of $x$ from $\Sigma$ given $y_j$ is in $\mathbb{P}$ when $L$ satisfies CD, FO, and IM.

**Proof.** The computation of a prime implicant explanation of $x$ is a follows. First, compute a representation $\alpha$ in $L$ of $\exists X. (\Sigma \mid y_j)$. This is feasible in polynomial time since $L$ is supposed to satisfy CD and FO. Then, the problem amounts to computing a prime implicant of $\alpha$. This can be done in polynomial time using a greedy algorithm when $L$ satisfies IM: start with $x$, which is a model (hence an implicant) of $\alpha$, and for every literal of $x$, test in polynomial time whether $x$ deprived from this literal is still an implicant of the formula (this is feasible in polynomial time when $L$ satisfies IM); if the resulting term still is an implicant of $\alpha$, then remove the literal, otherwise keep it and resume.

Prime implicant explanations are also referred to as sufficient reasons for a decision in (Darwiche and Hirth 2020). Interestingly, the explanations of classifications as produced by Anchor (Ribeiro, Singh, and Guestrin 2018) can be viewed as approximations of prime implicant explanations (Narodytska et al. 2019). Note that every prime implicant explanation of $x$ contains every feature that is mandatory for being recognized as an element of $y_j$ by $C$. Furthermore, no prime implicant explanation of $x$ contains a feature that is forbidden or irrelevant for being recognized as an element of $x$. Furthermore, since every variable from $Y \cup Z$ is defined from $X$ in $\Sigma$, two distinct models of $\beta$ over $\text{Var}(\beta)$ necessarily have two distinct projections onto $X$. Thus, enumerating the models of $\beta$ and projecting each of them onto $X$ as soon as it is generated leads to enumerating the minimum-cardinality explanations of $x$. The fact that $L$ satisfies ME is therefore enough to conclude the proof.

The above proposition extends a result from (Shih, Choi, and Darwiche 2018b), where it was shown that the enumeration of minimum-cardinality explanations is feasible with a polynomial delay when $L = \text{OBDD}$. Indeed, $\text{OBDD}$ satisfies CD, OPT, and ME, but many other languages in the DNF family (including DNNF itself) are strictly more succinct than $\text{OBDD}$ and also satisfy those properties.

Note that mandatory features are relevant to minimum-cardinality explanations in the sense that every minimum-cardinality explanation of $x$ necessarily contains every feature that is mandatory for being recognized as an element of $y_j$ by $C$. Similarly, no irrelevant feature for $y_j$ participates in any minimum-cardinality explanation.

**Deriving a prime implicant explanation (DPI).** Given an input $x$ such that $C(x) = y$ with $y_j = 1$, a prime implicant explanation is a subset-minimal partial assignment $x'$ which is coherent with $x$ (i.e., $x$ and $x'$ give the same values to the variables that are assigned in $x'$) and which satisfies the property that for every extension $x''$ of $x'$ over $X$, we have $C(x'') = y''$ with $y''_j = 1$. The features assigned in $x'$ (and the way they are assigned) can be viewed as explaining why $x$ has been classified as an element of $y_j$. Thus, in the running example, a prime implicant explanation of input $SW \land GR \land RO$ (classified as an element of GRA by $R$) is $SW \land GR$. The fact that $RO$ holds or not does not matter.

**Proposition 9.** Deriving a prime implicant explanation of $x$ from $\Sigma$ given $y_j$ is in $\mathbb{P}$ when $L$ satisfies CD, FO, and IM.

**Proof.** The computation of a prime implicant explanation of $x$ is a follows. First, compute a representation $\alpha$ in $L$ of $\exists X. (\Sigma \mid y_j)$. This is feasible in polynomial time since $L$ is supposed to satisfy CD and FO. Then, the problem amounts to computing a prime implicant of $\alpha$. This can be done in polynomial time using a greedy algorithm when $L$ satisfies IM: start with $x$, which is a model (hence an implicant) of $\alpha$, and for every literal of $x$, test in polynomial time whether $x$ deprived from this literal is still an implicant of the formula (this is feasible in polynomial time when $L$ satisfies IM); if the resulting term still is an implicant of $\alpha$, then remove the literal, otherwise keep it and resume.

Prime implicant explanations are also referred to as sufficient reasons for a decision in (Darwiche and Hirth 2020). Interestingly, the explanations of classifications as produced by Anchor (Ribeiro, Singh, and Guestrin 2018) can be viewed as approximations of prime implicant explanations (Narodytska et al. 2019). Note that every prime implicant explanation of $x$ contains every feature that is mandatory for being recognized as an element of $y_j$ by $C$. Furthermore, no prime implicant explanation of $x$ contains a feature that is forbidden or irrelevant for being recognized as an element of $x$. Furthermore, since every variable from $Y \cup Z$ is defined from $X$ in $\Sigma$, two distinct models of $\beta$ over $\text{Var}(\beta)$ necessarily have two distinct projections onto $X$. Thus, enumerating the models of $\beta$ and projecting each of them onto $X$ as soon as it is generated leads to enumerating the minimum-cardinality explanations of $x$. The fact that $L$ satisfies ME is therefore enough to conclude the proof.
y_j by C (if \( \exists \Sigma (y \mid y_j) \) is independent from \( x_k \), \( x_k \) does not occur in any prime implicant of \( \exists \Sigma (y \mid y_j) \)). Finally, it can be noted that discrete, yet non-Boolean features raise some representation issues when dealing with prime implicant explanations (it turns out that the prime implicants over the Boolean features obtained using the direct encoding do not capture the intended explanations, see (Choi et al. 2020)).

From a computational point of view, the enumeration of prime implicant explanations looks as a much more demanding issue than the enumeration of minimum-cardinality explanations. Though algorithms for computing the set of prime implicant explanations exist (see (Shih, Choi, and Darwiche 2018b; Darwiche and Hirth 2020)), the question of the existence of an algorithm with polynomial delay for enumerating the elements of this set from a representation \( \Sigma \in \mathcal{L} \) is open for many \( \mathcal{L} \) (e.g., \( \mathcal{L} = \text{OBDD} \)) as far as we know. Especially, though the enumeration of the models of \( \Sigma \) with a polynomial delay is possible as soon as \( \mathcal{L} \) satisfies CD and CO (Darwiche and Marquis 2002), those conditions on \( \mathcal{L} \) do not appear as sufficient for ensuring the existence of an enumeration algorithm with polynomial delay for the prime implicants of \( \Sigma \). Indeed, it is known that unless \( P = NP \) there is no input-output polynomial time algorithm for generating the prime implicants of a monotone formula (Goldsmith, Hagen, and Mundhenk 2005), and the language of monotone formulae satisfies CD and CO.

Enumerating counterfactual explanations (ECO). Occasionally, the user may be surprised by the way \( C \) has classified a given instance \( x_1 \). She was expecting the input instance to be recognized as an apple (\( y_k \)), but it has been classified as a grape fruit (\( y_j \)). In such a case, the user is likely to ask for a counterfactual explanation: one is interested in identifying an instance \( x_2 \) which is classified as an apple (and not as a grape fruit), and is as close as possible to \( x_1 \) in terms of the number of common values of the features in the two inputs (alias the Hamming distance \( d_H(x_1, x_2) \)). Indeed, the set of features that differ in \( x_1 \) and \( x_2 \) can be viewed as an explanation of why \( x_1 \) has not been classified as \( y_k \). In the running example, an input classified as an element of APP and that is as close as possible to \( x_1 = \text{SW} \wedge \text{GR} \wedge \text{RO} \) w.r.t. \( d_H \) is \( \text{SW} \wedge \text{GR} \wedge \text{RO} \) (it is at Hamming distance 2 of \( \text{SW} \wedge \text{GR} \wedge \text{RO} \)). Another counterfactual explanation for \( x_1 \) is \( \text{SW} \wedge \text{GR} \wedge \text{RO} \).

**Proposition 10.** The counterfactual explanations of \( x \) can be enumerated with polynomial delay from \( \Sigma \) given \( y_j \) and \( y_k \) as soon as \( \mathcal{L} \) satisfies CD, OPT, and ME.

**Proof:** One first generates a representation \( \alpha \in \mathcal{L} \) of \( \Sigma \mid (y_k \wedge \neg y_j) \), which is feasible in polynomial time when \( \mathcal{L} \) satisfies CD. Then one considers the same function \( f_x \) as in the proof of Proposition 7. For every truth assignment \( \omega \) over \( X \cup Y \cup Z \), the value \( f_x(\omega) + \sum_{i=1}^{n} v_i \) is equal to the Hamming distance between the projection of \( \omega \) onto \( X \) and \( x \). Since \( \sum_{i=1}^{n} v_i \) does not depend on \( \omega \), the projections onto \( X \) of the interpretations minimizing the value of \( f_x \) are those minimizing the Hamming distance to \( x \). When \( \mathcal{L} \) satisfies OPT, a representation \( \beta \) from \( \mathcal{L} \) of the models of \( \alpha \) minimizing the value of \( f_x \) can be computed in polynomial time.

In addition, when \( \mathcal{L} \) satisfies ME, starting from \( \beta \), those models can be enumerated with a polynomial delay. Since every variable from \( Y \cup Z \) is defined from \( X \) in \( \Sigma \), two distinct models of \( \beta \) over \( \text{Var}(\beta) \) necessarily have two distinct projections onto \( X \). Hence the counterfactual explanations of \( x \) can be enumerated with a polynomial delay. \( \square \)

As a direct consequence of Proposition 10, when \( \mathcal{L} \) satisfies CD, OPT, and ME, one can easily compute the robustness (Shih, Choi, and Darwiche 2018a) of the classification of \( x \) as an \( y_j \) defined as \( m_{m=1,...,m} k \neq j d_H(x, \Sigma \mid (y_k \wedge \neg y_j)) \), i.e., the Hamming distance between \( x \) and a closest instance that has been recognized by \( C \) as belonging to a class distinct from \( y_j \). In essence, this amounts to computing in sequence a counterfactual explanation of \( x \) for every \( y_k \) distinct from \( y_j \), and for each of them to compute its Hamming distance to \( x \) while memorizing the minimum distance. The computation of this robustness can clearly be done in polynomial time, extending to the Boolean case a result from (Shih, Choi, and Darwiche 2018a) in which it is shown that this is indeed the case for \( \mathcal{L} = \text{OBDD} \) (which satisfies CD, OPT, and ME).

## 5 Discussion

The XAI queries considered in the paper correspond to computation problems of various types: decision problems (IMA, IIR, IMO), counting problems (CIN, CAM), function problems (DPI, MFR, MCJ, MCH, MCP), enumeration problems (EMC, ECO, EIN, EA). Whatever its type, each of those queries is NP-hard in the broad sense (Garey and Johnson 1979) when no restrictions are imposed on the circuit \( \Sigma \) (i.e., when one just assumes that \( \Sigma \) is a circuit with inputs in \( X \), outputs in \( Y \), and intermediate variables in \( Z \)). More precisely, we can prove that for any of the queries associated with decision, counting, or function problems, if a polynomial-time algorithm for solving the query existed, then we would have \( P = NP \). In addition, we can also prove that for any of the queries associated with an enumeration problem, if an algorithm for enumerating solutions with polynomial delay existed, then we would have \( P = NP \). Thus, making additional assumptions on \( \Sigma \) looks as mandatory to make tractable any of the 14 XAI queries listed in Table 2. This is the path followed in the paper. Indeed, the results reported in the previous propositions identify conditions under which the XAI queries are tractable, those conditions taking the form of queries and transformations offered by the language \( \mathcal{L} \) used to represent \( \Sigma \).

To make such results significant from the practical side, two conditions must be fulfilled. On the one hand, some encoding schemas associating a circuit with a given classifier must be available. This is already the case for several families of classifiers, including Bayes classifiers (Chan and Darwiche 2003; Shih, Choi, and Darwiche 2019), binary neural networks (Narodytska et al. 2018; Shih, Darwiche, and Choi 2019; Shi et al. 2020), and random forests (as shown in the paper). On the other hand, languages \( \mathcal{L} \) satisfying the queries and transformations supporting the XAI queries under consideration must exist and translators (alias
compilers) for generating representations in $\mathcal{L}$ of $\Sigma$ must be implemented. Hopefully, this is also the case.

In this respect, the family of $\text{DNNF}$ languages (Darwiche 2001; Huang and Darwiche 2007; Pipatsrisawat and Darwiche 2008; Oztok and Darwiche 2014) appears as particularly interesting. A $\text{DNNF}$ representation $\Sigma$ is a Boolean circuit with a single root, internal nodes labelled by connectives in $\{\land, \lor\}$, leaves labeled by literals over $PS$ or Boolean constants, and satisfying the following decomposability condition: the $\text{DNNF}$ representations rooted at any $\land$-node in $\Sigma$ do not share any variable. A $d$-$\text{DNNF}$ representation is a $\text{DNNF}$ representation $\Sigma$ that satisfies the following determinism condition: the $d$-$\text{DNNF}$ representations rooted at any $\lor$-node in $\Sigma$ are pairwise inconsistent.

A Decision-$\text{DNNF}$ representation $\Sigma$ is a Boolean circuit with a single root, leaves are labeled by literals over $PS$ or Boolean constants, and internal nodes are decomposable $\land$-nodes or decision nodes of the form $\text{ite}(x, n_1, n_2)$, where $\text{ite}$ stands for "if ... then ... else ...". In a decision node $\text{ite}(x, n_1, n_2)$, $x$ is any propositional variable of $PS$ (the decision variable), and the Decision-$\text{DNNF}$ representations $\Sigma_1$ and $\Sigma_2$ rooted (respectively) at $n_1$ and $n_2$ do not contain any occurrence of $x$. $\text{ite}$ can be considered as a ternary connective so that every node of the form $\text{ite}(x, n_1, n_2)$ can be viewed as a short for $(x \land \Sigma_1) \lor (x \land \Sigma_2)$. Since in the latter statement, the $\land$-nodes are decomposable nodes, and the $\lor$-node is a deterministic one, every Decision-$\text{DNNF}$ circuit can be considered as a $d$-$\text{DNNF}$ circuit. Formally, there exists a linear-time equivalence-preserving translation from Decision-$\text{DNNF}$ to $d$-$\text{DNNF}$. Structural restrictions on $\text{DNNF}$ circuits based on a concept of vtree can also be imposed, leading to languages of structured $\text{DNNF}$ representations that offer additional queries and transformations.

Thanks to the decomposability/determinism/structure conditions, languages of the $\text{DNNF}$ family offer many queries and transformations of interest. Thus, $\text{DNNF}$ satisfies $\text{CD}$, $\text{CO}$, $\text{FO}$, $\text{OPT}$, $\land\text{DC}$, and $d$-$\text{DNNF}$ offers all the queries and transformations listed in the paper, except $\text{SE}$, $\text{EQ}$, $\text{FO}$, $\land\text{BC}$ (Darwiche and Marquis 2002; Darwiche and Marquis 2004). Structured $d$-$\text{DNNF}$ circuits (given a fixed vtree) offers in addition $\text{SE}$, $\text{EQ}$, and $\land\text{BC}$. Accordingly, under the restriction when $\Sigma$ is given as a $\text{DNNF}$ circuit (resp. a $d$-$\text{DNNF}$ circuit), the XAI queries $\text{EMC}$, $\text{ECO}$, $\text{EIN}$, $\text{IMA}$, and $\text{MCP}$ (resp. $\text{CIN}$, $\text{MFR}$, and $\text{MCJ}$) are tractable. $\text{MCH}$ is tractable when $\Sigma$ is given as a structured $d$-$\text{DNNF}$ circuit. Finally, though Decision-$\text{DNNF}$ does not offer $\text{FO}$ in the general case, wherever the forgetting transformation is used for solving the XAI queries considered above, it concerns variables that are defined from unforgotten ones (those of $X$). In such a case, applying the forgetting algorithm that consists in replacing every decision node labelled by a variable from $X$ by a $\lor$-node (while keeping the same two children) turns the Decision-$\text{DNNF}$ circuit at hand into a $d$-$\text{DNNF}$ circuit (see (Lagniez, Lonca, and Marquis 2020) for details). As a consequence, DPI is tractable when $\Sigma$ is given as a Decision-$\text{DNNF}$ circuit, and IIR and IMO are tractable when $\Sigma$ is given as a structured Decision-$\text{DNNF}$ circuit.

### 6 Conclusion

In this paper, we have presented a number of XAI queries, which are useful for explaining classifications achieved by a predictor $C$ (the queries are EMC, DPI, ECO) or for assessing the robustness of $C$ (the queries are CIN, EIN, CAM, EAM, MFR, IMA, IIR, IMO, MCJ, MCH, MCP). All those queries are delegated to a Boolean circuit $\Sigma$ exhibiting the same input-output behaviour as $C$.

We have shown how these XAI queries can be addressed by combining operations on Boolean circuits. Especially, taking advantage of previous results reported in the knowledge compilation literature, we have identified some conditions on the language $\mathcal{L}$ used for representing $\Sigma$ that prove enough for ensuring the tractability of the XAI queries. Table 2 summarizes the XAI queries considered in the paper and associates with each of them some conditions on $\mathcal{L}$ that are sufficient to make it tractable. Several subsets $\mathcal{L}$ of $\text{DNNF}$ appear as valuable, ensuring that many XAI queries from the table are tractable whenever $\Sigma$ is represented in $\mathcal{L}$.

<table>
<thead>
<tr>
<th>XAI query</th>
<th>Conditions on $\mathcal{L}$ making the query tractable</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMC</td>
<td>CD, OPT, ME</td>
</tr>
<tr>
<td>DPI</td>
<td>CD, FO, IM</td>
</tr>
<tr>
<td>ECO</td>
<td>CD, OPT, ME</td>
</tr>
<tr>
<td>CIN</td>
<td>CD, CT</td>
</tr>
<tr>
<td>EIN</td>
<td>CD, ME</td>
</tr>
<tr>
<td>CAM</td>
<td>CD, CT</td>
</tr>
<tr>
<td>EAM</td>
<td>CD, ME</td>
</tr>
<tr>
<td>MFR</td>
<td>CD, CT</td>
</tr>
<tr>
<td>IMA</td>
<td>CD, CO</td>
</tr>
<tr>
<td>IIR</td>
<td>CD, FO, EQ</td>
</tr>
<tr>
<td>IMO</td>
<td>CD, FO, SE</td>
</tr>
<tr>
<td>MCJ</td>
<td>CD, CT</td>
</tr>
<tr>
<td>MCH</td>
<td>CD, $\land\text{BC}$, $\land\text{DC}$, OPT, ME</td>
</tr>
<tr>
<td>MCP</td>
<td>CD, OPT, ME</td>
</tr>
</tbody>
</table>

Table 2: XAI queries and conditions on $\mathcal{L}$ that are sufficient to make them tractable.

In addition, $\text{DNNF}$ languages are quite succinct – actually, more than other candidates like OBDD or FBDD (Darwiche and Marquis 2002; Bova et al. 2016) – , and there exist compilers targeting those languages when $\Sigma$ is given at start as a $\text{CNF}$ formula (Darwiche 2001; Darwiche 2004; Pipatsrisawat and Darwiche 2010; Muise et al. 2012; Lagniez and Marquis 2017).
Acknowledgements

Many thanks to the anonymous reviewers for their comments and suggestions. This work has benefited from the support of the AI Chair EXPEKCTATION of the French National Research Agency (ANR).

References


Muise, C.; McIlraith, S.; Beck, J.; and Hsu, E. 2012.


