On Tractable XAI Queries based on Compiled Representations

Gilles Audemard[1] , **Frédéric Koriche**[1] , **Pierre Marquis**[1,2]
[1]CRIL, Université d'Artois & CNRS, France
[2]Institut 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 term 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. Assessing the robustness of the predictions (i.e., would a small change in the input instance question the predicted label?) is also out of reach. 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 $\boldsymbol{C}$ of various types (including black boxes) can be associated with Boolean circuits Σ (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, Σ as an abstraction of $\boldsymbol{C}$ is typically far less complex than $\boldsymbol{C}$; on the other hand, the Boolean nature of Σ 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 $\boldsymbol{C}$ and for each of them, to identify some sufficient conditions on the language $\mathcal{L}$ used to represent Σ that render the query tractable. For the sake of generality, we do not make any strong assumption on the nature of the classifier: $\boldsymbol{C}$ can be a feedforward neural network, a Bayesian network classifier, a random forest, etc. We just assume that $\boldsymbol{C}$ is a discrete multi-label classifier from which a Boolean circuit Σ having the same input-output behaviour can been generated. More precisely, each input instance is a vector $\boldsymbol{x}$ of n values assigned to Boolean features from a set $X = \{x_1, \cdots, x_n\}$ and each corresponding output is a vector $\boldsymbol{y} \in \boldsymbol{Y}$ of m values taken from a set $Y = \{y_1, \cdots, y_m\}$ of Boolean labels.[1] Given the prediction $\boldsymbol{y} = \boldsymbol{C}(\boldsymbol{x})$, the instance $\boldsymbol{x} \in \boldsymbol{X}$ is classified by $\boldsymbol{C}$ as an element of the jth class if and only if $y_j = 1$.

The variables of Σ are split into three categories:

(1) *Input variables* used to describe the input features of $\boldsymbol{C}$. 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;

[1]When an input feature x_i is not Boolean, but takes values into a finite domain $\{v_1, \cdots, v_p\}$, one can represent it using p Boolean features $x_i^1, \cdots, x_i^p$ 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 $\bigwedge_{j=1}^{p-1} \bigwedge_{k=j+1}^{p} (\neg x_i^j \vee \neg x_i^k)$ and $\bigvee_{j=1}^{p} x_i^j$ ensuring that x_i takes exactly one value in its domain.

(3) *Intermediate variables* used to capture all other features of the classifier $\boldsymbol{C}$, and corresponding to the wires of Σ that are not output ones. Each intermediate variable is also assumed to take a Boolean value, and we use Z to denote the set of those variables.

As Σ and $\boldsymbol{C}$ are supposed to have the same input-output behaviour, for any pair $(\boldsymbol{x}, \boldsymbol{y})$ for which $\boldsymbol{y} = \boldsymbol{C}(\boldsymbol{x})$, we have $y_j = 1$ precisely when the output variable y_j of the circuit Σ on the input $\boldsymbol{x}$ is set to 1. Importantly, we make no assumption about the number of predicted labels: for an input $\boldsymbol{x}$, several coordinates – resp. no coordinate – of the associated output $\boldsymbol{y}$ can be set to 1, which means that $\boldsymbol{C}$ (and hence Σ) recognizes $\boldsymbol{x}$ as a member of several (resp. no) class(es). In practice, $\boldsymbol{C}$ is often associated with additional information about $\boldsymbol{y}$, taking the form of weights representing confidences or probabilities about predicted labels. Such weights are usually exploited by the encoding scheme for generating Σ from $\boldsymbol{C}$, but they are *not* encoded into Σ. Thus, Σ is viewed as a *compact representation* of the classes Y, as they are recognized by $\boldsymbol{C}$.

In the following, we present a number of XAI queries of practical interest for explaining classifications achieved by $\boldsymbol{C}$, or for assessing the robustness of $\boldsymbol{C}$. These queries are addressed by considering the circuit Σ associated with $\boldsymbol{C}$, instead of $\boldsymbol{C}$ itself. For the sake of clarity, we split XAI queries into *explanation* queries and *verification* queries. Explanation queries are concerned by the local interpretability issue, i.e., they are about a given input $\boldsymbol{x}$, while verification queries are concerned by 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 Σ 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 Σ with a given random forest $\boldsymbol{C}$ 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 Σ 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.

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 Σ, we denote by $Var(\Sigma)$ the subset of variables of PS occurring in Σ. As usual, atomic representations include propositional variables in PS, and Boolean constants in $\{\top, \bot\}$. 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 ω from X to $\mathbb{B} = \{0, 1\}$. Propositional representations are interpreted in a classical way. For a representation Σ and an interpretation over $X = Var(\Sigma)$, we use $\omega \models \Sigma$ to denote the fact that ω if a model of Σ according to the semantics of propositional logic. That is, assigning the variables of Σ to truth values as specified by ω makes Σ true. By $[\Sigma]$ we denote the *set* of models of Σ over $Var(\Sigma)$, and by $\|\Sigma\|$ we denote the *number* of models of Σ over $Var(\Sigma)$. In particular, Σ is *inconsistent* if $\|\Sigma\| = 0$, and *consistent* otherwise. A representation Σ_2 is a *logical consequence* of a representation Σ_1 (denoted $\Sigma_1 \models \Sigma_2$) if $\Sigma_1 \wedge \neg\Sigma_2$ is inconsistent. Σ_1 and Σ_2 are *logically equivalent* (denoted $\Sigma_1 \equiv \Sigma_2$) if they are logical consequences of each other.

Given a representation Σ and a consistent term γ, the *conditioning* of Σ by γ is the representation obtained by replacing in Σ every occurrence of a variable $x \in Var(\gamma)$ by $\top$ if x is a positive literal of γ and by $\bot$ if $\neg x$ is a negative literal of γ. Finally, when X is a subset of propositional variables from PS, Σ is said to be *independent* of X if there is a representation Φ logically equivalent to Σ such that $Var(\Phi) \cap X = \varnothing$. The *forgetting* of X in Σ, denoted $\exists X.\Sigma$, is the *most general consequence* of Σ that is independent of X (see e.g., (Lang, Liberatore, and Marquis 2003)). The *projection* of Σ onto X is the forgetting of $\overline{X}$ in Σ, where $\overline{X}$ denotes the set $PS \setminus X$. We mention in passing that $\exists X.\Sigma$ can be computed as a propositional representation, thanks to the following inductive characterization:

- $\exists \varnothing.\Sigma \equiv \Sigma$,
- $\exists \{x\}.\Sigma \equiv (\Sigma \mid \neg x) \vee (\Sigma \mid x)$,
- $\exists (X' \cup \{x\}).\Sigma \equiv \exists X'.(\exists \{x\}.\Sigma)$.

Knowledge Compilation Map. Introduced by Darwiche and Marquis (2002), and extended in a number of papers (e.g., (Darwiche and Marquis 2004; Niveau et al. 2010; Darwiche 2011; Fargier, Marquis, and Niveau 2013; Fargier and Marquis 2014; Koriche et al. 2016)), the knowledge compilation (KC) map is a multicriteria evaluation of propositional languages. Various languages have been considered as target languages for knowledge compilation. They include, among others, `DNF` (disjunctions of terms), `DNNF` (decomposable normal form circuits) (Darwiche 2001), `FBDD` (free binary decision diagrams) (Gergov and Meinel 1994), and `OBDD` (ordered binary decision diagrams) (Bryant 1986). In the KC map, a propositional language $\mathcal{L}$ is evaluated according to properties it offers, or not, depending on the existence of a polynomial-time algorithm for achieving some treatment of interest. These properties are usually decomposed into queries and transformations. In this paper, we focus on the

following queries and transformations:

- *Queries*
  - **consistency:** $\mathcal{L}$ satisfies **CO** if and only if there is a polynomial-time algorithm that maps every representation Σ from $\mathcal{L}$ to 1 if Σ is consistent, and to 0 otherwise.
  - **implicant:** $\mathcal{L}$ satisfies **IM** if and only if there is a polynomial-time algorithm that maps every representation Σ from $\mathcal{L}$ and every term γ to 1 if $\gamma \models \Sigma$ holds, and to 0 otherwise.
  - **sentential entailment:** $\mathcal{L}$ satisfies **SE** if and only if there is a polynomial-time algorithm that maps any two representations Σ_1 and Σ_2 from $\mathcal{L}$ to 1 if $\Sigma_1 \models \Sigma_2$ holds, and to 0 otherwise.
  - **equivalence:** $\mathcal{L}$ satisfies **EQ** if and only if there is a polynomial-time algorithm that maps any two representations Σ_1 and Σ_2 from $\mathcal{L}$ to 1 if $\Sigma_1 \equiv \Sigma_2$ holds, and to 0 otherwise.
  - **model counting:** $\mathcal{L}$ satisfies **CT** if and only if there is a polynomial-time algorithm that maps every representation Σ from $\mathcal{L}$ to a nonnegative integer (in binary notation) corresponding to the number of models $\|\Sigma\|$ of Σ over $Var(\Sigma)$.
  - **model enumeration:** $\mathcal{L}$ satisfies **ME** if and only if there is an enumeration algorithm with polynomial delay for the set $[\Sigma]$ of models of Σ over $Var(\Sigma)$. Here, the algorithm must generate all models in sequence, without any model occurring more than once. Additionally, the algorithm must guarantee that each delay between the generation of two successive models and between the generation of the last model and the notification of termination, is polynomial in the size of Σ.[2]
- *Transformations*
  - **conditioning:** $\mathcal{L}$ satisfies **CD** if and only if there is a polynomial-time algorithm that maps every representation Σ from $\mathcal{L}$ and every consistent term γ to a representation from $\mathcal{L}$ that is logically equivalent to $\Sigma \mid \gamma$.
  - **bounded conjunction:** $\mathcal{L}$ satisfies $\wedge$**BC** if and only if there is a polynomial-time algorithm that maps every pair of representations Σ_1 and Σ_2 from $\mathcal{L}$ to a representation from $\mathcal{L}$ that is logically equivalent to $\Sigma_1 \wedge \Sigma_2$.
  - **forgetting:** $\mathcal{L}$ satisfies **FO** if and only if there is a polynomial-time algorithm that maps every representation Σ from $\mathcal{L}$ and every subset X of variables from PS to a representation from $\mathcal{L}$ equivalent to $\exists X.\Sigma$.
  - **optimization:** $\mathcal{L}$ satisfies **OPT** if and only if there exists a polynomial-time algorithm that maps every representation Σ from $\mathcal{L}$ and any linear function f over $Var(\Sigma)$ into a representation from $\mathcal{L}$ for which the models are those of Σ 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:** $\mathcal{L}$ satisfies $\wedge$**DC** if and only if there is a polynomial-time algorithm that maps k representations $\Sigma_1, \cdots, \Sigma_k$ from $\mathcal{L}$ such that $Var(\Sigma_i) \cap Var(\Sigma_j) = \varnothing$ (for $i, j \in \{1, \cdots, k\}, i \neq j$) to a representation from $\mathcal{L}$ 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 DNNF is strictly more succinct than OBDD and 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 Σ with a given random forest $\boldsymbol{C}$ 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 $\boldsymbol{x}$ as an element of class $y_j \in Y$ if the unique path of T that is compatible with $\boldsymbol{x}$ (i.e., that contains a pair $x = v$ only if x takes the value v in $\boldsymbol{x}$) leads to a leaf node whose label contains y_j.

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

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 $X = \{\text{SW}(\textit{eet taste}), \text{GR}(\textit{een}), \text{RO}(\textit{und shape})\}$, and the set of output labels $Y = \{\text{GRA}(\textit{pe fruit}), \text{BAN}(\textit{ana}), \text{APP}(\textit{le})\}$. An instance $\boldsymbol{x}$ is classified as a label $y \in Y$ 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 $\boldsymbol{x}$ 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 $\boldsymbol{x}_1$ 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 GRA *(resp.* GRA *or* BAN*,* 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 Σ plus the size of the corresponding set of models. Note that we have that $\mathcal{L}$ satisfies **ME** (in the strong sense) whenever it satisfies **CD** and **CO** (see (Fargier and Marquis 2014) for details).

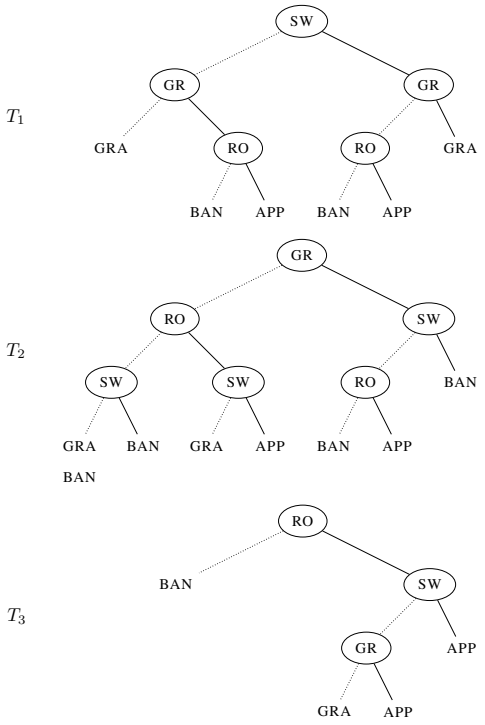

Figure 1: A random forest with three decision trees T_1, T_2, T_3.

instance $\boldsymbol{x}_2$ 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* GRA *(resp.* BAN*,* BAN*). Thus, R classifies $\boldsymbol{x}_1$ as an element of* GRA *or* BAN*, and R classifies $\boldsymbol{x}_2$ as an element of* 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* GRA *(resp.* BAN*,* 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 Σ over the set of propositional variables $PS = X \cup Y \cup Z$, where Z captures the intermediate variables. Σ is the conjunction of two kinds of constraints:

- for every $j \in \{1, \ldots, m\}$, Σ includes the constraint

$$y_j \Leftrightarrow \left(\sum_{k=1}^{p} z_j^k \geq \tau \right)$$

  where variable z_j^k is true if and only if the input instance $\boldsymbol{x}$ 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 $\boldsymbol{x}$ as an element of y_j is at least equal to the threshold τ;
- for every $j \in \{1, \ldots, m\}$ and every $k \in \{1, \ldots, p\}$, Σ includes a constraint

$$z_j^k \Leftrightarrow \left(\bigvee_{\gamma \in \Gamma(T_k, y_j)} \gamma \right)$$

  indicating that z_j^k is set to true (i.e., $\boldsymbol{x}$ is classified as an element of y_j by T_k) if and only if $\boldsymbol{x}$ 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 γ encoding the paths of T_k which lead to leaf nodes labelled by y_j.

SW	GR	RO	GRA	BAN	APP
0	0	0	1	1	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	0	1	0
1	1	1	0	0	0

Table 1: The projection of Σ over $X \cup Y$.

For our running example, R is associated with the representation Σ, which is given by the conjunction of the following constraints:

$$\begin{aligned}
\text{GRA} \Leftrightarrow \quad & (\text{GRA}^1 + \text{GRA}^2 + \text{GRA}^3 \geq 2) \\
\text{BAN} \Leftrightarrow \quad & (\text{BAN}^1 + \text{BAN}^2 + \text{BAN}^3 \geq 2) \\
\text{APP} \Leftrightarrow \quad & (\text{APP}^1 + \text{APP}^2 + \text{APP}^3 \geq 2) \\
\text{GRA}^1 \Leftrightarrow \quad & ((\overline{\text{SW}} \wedge \overline{\text{GR}}) \vee (\text{SW} \wedge \text{GR})) \\
\text{BAN}^1 \Leftrightarrow \quad & ((\overline{\text{SW}} \wedge \text{GR} \wedge \overline{\text{RO}}) \vee (\text{SW} \wedge \overline{\text{GR}} \wedge \overline{\text{RO}})) \\
\text{APP}^1 \Leftrightarrow \quad & ((\overline{\text{SW}} \wedge \text{GR} \wedge \text{RO}) \vee (\text{SW} \wedge \overline{\text{GR}} \wedge \text{RO})) \\
\text{GRA}^2 \Leftrightarrow \quad & ((\overline{\text{SW}} \wedge \overline{\text{GR}} \wedge \overline{\text{RO}}) \vee (\overline{\text{SW}} \wedge \overline{\text{GR}} \wedge \text{RO})) \\
\text{BAN}^2 \Leftrightarrow \quad & ((\text{SW} \wedge \overline{\text{GR}} \wedge \overline{\text{RO}}) \vee (\overline{\text{SW}} \wedge \text{GR} \wedge \overline{\text{RO}}) \\
& \vee (\text{SW} \wedge \text{GR})) \\
\text{APP}^2 \Leftrightarrow \quad & ((\text{SW} \wedge \overline{\text{GR}} \wedge \text{RO}) \vee (\overline{\text{SW}} \wedge \text{GR} \wedge \text{RO})) \\
\text{GRA}^3 \Leftrightarrow \quad & (\overline{\text{SW}} \wedge \overline{\text{GR}} \wedge \text{RO}) \\
\text{BAN}^3 \Leftrightarrow \quad & \overline{\text{RO}} \\
\text{APP}^3 \Leftrightarrow \quad & ((\overline{\text{SW}} \wedge \text{GR} \wedge \text{RO}) \vee (\text{SW} \wedge \text{RO}))
\end{aligned}$$

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

$$\begin{aligned}
\text{GRA} \Leftrightarrow \quad & (\overline{\text{SW}} \wedge \overline{\text{GR}}) \\
\text{BAN} \Leftrightarrow \quad & \overline{\text{RO}} \\
\text{APP} \Leftrightarrow \quad & ((\overline{\text{SW}} \wedge \text{GR}) \vee (\text{SW} \wedge \overline{\text{GR}})) \wedge \text{RO}
\end{aligned}$$

Equivalently, the projection of Σ 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 $\boldsymbol{x}$ to a class of Y as predicted by $\boldsymbol{C}$ can be translated in logical terms when Σ (represented using a propositional language $\mathcal{L}$) is considered instead of $\boldsymbol{C}$.

To this point, observe that any input instance $\boldsymbol{x}$ 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 $\boldsymbol{x}$ to avoid heavy notations. Specifically, $\boldsymbol{x}$ is given by $\bigwedge_{j=1}^{n} \ell_j$ where $\ell_j = x_j$ if $v_j = 1$ and $\ell_j = \overline{x}_j$ otherwise.

Now, since Σ has the same input-output behaviour as $\boldsymbol{C}$, we know that every variable from $Y \cup Z$ is *defined* in Σ in terms of X. In other words, for any input $\boldsymbol{x}$, we have $\boldsymbol{C}(\boldsymbol{x}) = (y_1 = c_1, \ldots, y_m = c_m)$ if and only if the unique model of Σ over $X \cup Y \cup Z$ which is compatible with $\boldsymbol{x}$ 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 φ_u built upon variables of X, only, that is such that $\Sigma \models u \Leftrightarrow \varphi_u$ (Beth 1953; Lang and Marquis 2008).

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

- *abductive characterization*: $\boldsymbol{x}$ is an abductive explanation of y_j with respect to Σ provided that the set of assumptions is the set of all literals over the variables from X. Indeed, if y_j is true in $\boldsymbol{C}(\boldsymbol{x})$ then we have $\Sigma \wedge \boldsymbol{x} \models y_j$. Note here that $\Sigma \wedge \boldsymbol{x}$ is consistent for every input instance $\boldsymbol{x}$ over X, since Σ is a Boolean circuit with inputs in X.
- *model-based characterization*: $\boldsymbol{x}$ is a model over X of $\exists \overline{X}.(\Sigma \wedge y_j)$, or equivalently a model over X of $\exists \overline{X}.(\Sigma \mid y_j)$. Thus, the projection of $\Sigma \mid y_j$ onto X can be viewed as a compact representation of all the input instances $\boldsymbol{x}$ viewed as elements of y_j by $\boldsymbol{C}$.

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 $\boldsymbol{C}$ (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 $\overline{\text{SW}} \wedge \overline{\text{GR}} \wedge \overline{\text{RO}}$ is classified as an element of GRA, and hence, she would like to gather more information about instances labeled by this class.

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

Proof. Counting the inputs $\boldsymbol{x}$ associated by $\boldsymbol{C}$ with a given output y_j amounts to computing $\|\Sigma \mid y_j\|$. Enumerating those inputs boils down to enumerating the models of $\exists \overline{X}.(\Sigma \mid y_j)$ with polynomial delay, which is feasible as soon as $\mathcal{L}$ satisfies **CD**, **FO**, and **ME**. Since every variable of $Var(\Sigma)$ not belonging to X is defined from X in Σ, 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 $\mathcal{L}$ 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 $\boldsymbol{C}$ is. Indeed, as much $\boldsymbol{C}$ 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 $\overline{\text{SW}} \wedge \overline{\text{GR}} \wedge \overline{\text{RO}}$, is ambiguously classified by $\boldsymbol{C}$. Similarly, only one instance, given by $\text{SW} \wedge \text{GR} \wedge \text{RO}$, is not recognized by $\boldsymbol{C}$ as a member of any class from Y.

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

Proof. Counting the inputs for which $\boldsymbol{C}$ remains mute amounts to computing $A = \left\| \Sigma \mid \bigwedge_{j=1}^{m} \overline{y_j} \right\|$. Furthermore, the number of inputs mapped by $\boldsymbol{C}$ into a single class y_j is given by $B_j = \left\| \Sigma \mid (y_j \wedge \bigwedge_{k \in \{1,\ldots,m\} \mid k \neq j} \overline{y}_k) \right\|$. Therefore, counting the number of inputs for which $\boldsymbol{C}$ provides ambiguous outputs amounts to evaluating $|\boldsymbol{X}| - A - \sum_{j=1}^{m} B_j$. The instances classified as elements of the intersection of a given set $S \subseteq Y$ 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 $\boldsymbol{C}$ remains mute correspond to the models over X of $\Sigma \mid \bigwedge_{j=1}^{m} \overline{y_j}$. Thus, they can be enumerated with polynomial delay from Σ when $\mathcal{L}$ 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 SW (resp. GR, RO) in class 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 (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 $\mathcal{L}$ satisfies* **CD** *and* **CT**.
- *Deciding whether a feature (or a combination of features) is mandatory or forbidden in a given class (or combination of classes) is in* P *when $\mathcal{L}$ satisfies* **CD** *and* **CO**.

Proof.

- The frequency of x_k in y_j is given by $\frac{\|\Sigma|(y_j \wedge x_k)\|}{\|\Sigma|y_j\|}$. When a more complex combination of features / classes is considered, it is sufficient to replace the terms $y_j \wedge x_k$ (resp.

y_j) in this equation by the terms denoting the combination of features and classes (resp. classes) to get the right value. This value can be computed in polynomial time when $\mathcal{L}$ satisfies **CD** and **CT**.

- Determining whether x_k is mandatory (resp. forbidden) for y_j amounts to deciding whether or not $\Sigma \mid (y_j \wedge \overline{x}_k)$ is inconsistent (resp. $\Sigma \mid (y_j \wedge x_k)$ is inconsistent), which can be achieved in polynomial time when $\mathcal{L}$ satisfies **CD** and **CO**. When a more complex combination of features / 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 otherwise 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 instance $\boldsymbol{x}$ does not depend on the value of its feature x_i.

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

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

When considering Boolean features, this proposition coheres with a result reported in (Shih, Choi, and Darwiche 2018a), showing that identifying irrelevant features is tractable when an OBDD representation of $\exists \overline{X}.(\Sigma \mid y_j)$ 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 $\boldsymbol{C}$ is *monotone* (resp. *anti-monotone*) for a given class y_j, with respect an input feature x_i, if for any input instance $\boldsymbol{x}$ such that $\boldsymbol{C}(\boldsymbol{x}) = \boldsymbol{y}$ with $y_j = 1$ implies that $\boldsymbol{C}(\boldsymbol{x}[x_i \leftarrow 1]) = \boldsymbol{y}'$ (resp. $\boldsymbol{C}(\boldsymbol{x}[x_i \leftarrow 0]) = \boldsymbol{y}'$) with $y'_j = 1$.[3]

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* P *whenever $\mathcal{L}$ satisfies* **CD**, **FO**, *and* **SE**.

Proof. At the semantics level, conditioning a propositional representation φ by a literal ℓ (resp. $\neg\ell$) consists first in selecting the models ω of φ over $Var(\varphi)$ where ℓ is assigned to 1 (resp. 0), then in forgetting ℓ in the result (which leads to add to this set of models every interpretation that coincides with one of the ω but is such that ℓ is assigned to 0 (resp. 1) in it). Using symbols, $\varphi \mid x \equiv \exists\{x\}.(\varphi \wedge x)$ and $\varphi \mid \overline{x} \equiv \exists\{x\}.(\varphi \wedge \overline{x})$. Accordingly, if $\boldsymbol{C}$ is not monotonic (resp. anti-monotonic) for y_j with respect to x_i, then there is a model of $\exists \overline{X}.(\Sigma \mid y_j)$ which is not a model of $\exists \overline{X}.(\Sigma \mid y_j)$ conditioned by x_i (resp. $\overline{x}_i$). Thus, checking whether $\boldsymbol{C}$ is monotonic (resp. anti-monotonic) for y_j w.r.t. x_i amounts to testing whether $\exists \overline{X}.(\Sigma \mid y_j) \models (\exists \overline{X}.(\Sigma \mid y_j)) \mid x_i$ (resp. $\exists \overline{X}.(\Sigma \mid y_j) \models (\exists \overline{X}.(\Sigma \mid y_j)) \mid \overline{x}_i$). 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 monotonicity has already been considered in (Shih, Choi, and Darwiche 2018a) for binary classifiers (i.e., when $m = 1$).

Determining how much 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 ordinal and takes the form of a set of pre-orders $\leq_{y_j}$ over Y, one for each class $j \in \{1, \cdots, m\}$. By writing $y_k \leq_{y_j} y_l$, we mean that y_k is at least as close to y_j as to y_l. Thus, the user may believe that APP $\leq_{\text{GRA}}$ BAN. Of course, it is expected that $y_j \leq_{y_j} y_l$ whatever $y_l \in Y$. Clearly enough, every cardinal similarity σ between classes of Y (i.e., a symmetric mapping associating a non-negative number with a couple of classes), induces an ordinal similarity given by $y_k \leq_{y_j} y_l$ if and only if $\sigma(y_k, y_j) \geq \sigma(y_l, y_j)$.

Several cardinal similarities can be defined and evaluated from Σ. A first one is given by the *Jaccard index* $J(y_k, y_l)$: it is the number of elements belonging to both classes, divided 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). $J(y_k, y_l)$ indicates the proportion of inputs that are (possibly wrongly)

[3] If $\boldsymbol{x} = (x_1, \cdots, x_n)$, then $\boldsymbol{x}[x_k \leftarrow v]$ is the same vector as $\boldsymbol{x}$, except that the jth coordinate x_k of $\boldsymbol{x}[x_k \leftarrow v]$ 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 Σ as done in our setting, we get:

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

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 $J(\text{BAN}, \text{GRA}) = \frac{1}{5}$ while $J(\text{APP}, \text{GRA}) = 0$. 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.[4] In doing so, let $d_H(\boldsymbol{x}, \boldsymbol{x}')$ denote the number of bits that differ between the inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$ from $\boldsymbol{X}$. This distance can be easily lifted to propositional representations Σ_1 and Σ_2 over X encoding sets of such inputs:[5]

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

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

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

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 $H(\text{BAN}, \text{GRA}) = 1$ (since both classes share an element) while $H(\text{APP}, \text{GRA}) = \frac{2}{3}$. From the point of view of H similarity, bananas are also viewed as closer to grape fruits than apples.

Proposition 6.

- MCJ*: computing $J(y_k, y_l)$ is in* P *when $\mathcal{L}$ satisfies* $\mathbf{CD}$ *and* $\mathbf{CT}$.
- MCH*: computing $H(y_k, y_l)$ is in* P *when $\mathcal{L}$ satisfies* $\mathbf{CD}$, $\wedge\mathbf{BC}$, $\wedge\mathbf{DC}$, $\mathbf{OPT}$, *and* $\mathbf{ME}$.

Proof.

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

$$\frac{\|\Sigma \mid (y_k \wedge y_l)\|}{\|\Sigma \mid (y_k \wedge y_l)\| + \|\Sigma \mid (y_k \wedge \overline{y_l})\| + \|\Sigma \mid (\overline{y}_k \wedge y_l)\|}.$$

- The computation of $H(y_k, y_l)$ is a bit more subtle. Consider a representation Σ in $\mathcal{L}$ and a "clone" Σ' of it obtained by renaming (in a uniform way) every variable v occurring in Σ into v'. Then condition Σ by y_k and Σ' by y'_l, and compute a representation α in $\mathcal{L}$ of the conjunction of the resulting representations. This can be done in time polynomial in the size of Σ when $\mathcal{L}$ satisfies $\mathbf{CD}$ and $\wedge\mathbf{BC}$. The next step consists in computing a representation in $\mathcal{L}$ of each formula $z_i \oplus (x_i \Leftrightarrow x'_i)$ where $i \in \{1, \cdots, n\}$ 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 $\mathcal{L}$ satisfies $\wedge\mathbf{DC}$, a representation β in $\mathcal{L}$ of $\bigwedge_{i=1}^{n}(z_i \oplus (x_i \Leftrightarrow x'_i))$ can be computed in polynomial time. Finally, a representation γ in $\mathcal{L}$ of $\alpha \wedge \beta$ can be computed in polynomial time when $\mathcal{L}$ satisfies $\wedge\mathbf{BC}$. Consider now a linear function f over $Var(\alpha \wedge \beta)$ associating with every variable in it the weight 0, except the variables $z_1, \cdots, z_n$ that have weight 1. Since $\mathcal{L}$ satisfies $\mathbf{OPT}$, a representation δ in $\mathcal{L}$ having for models those of γ minimizing the value of f can be computed in polynomial time from γ and f. By construction, every such model contains a minimum number of variables z_i set to 1, and this number is equal to $d_H(\exists \overline{X}.(\Sigma \mid y_k), \exists \overline{X}.(\Sigma \mid y_l))$ (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 $\mathcal{L}$ satisfies $\mathbf{ME}$, a model ω of δ 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 $d_H(\exists \overline{X}.(\Sigma \mid y_k), \exists \overline{X}.(\Sigma \mid y_l))$, 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 element 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 $\text{SW} \wedge \overline{\text{GR}} \wedge \text{RO}$. Based on this background information, it is interesting to determine how much each class (as it is recognized by $\boldsymbol{C}$) 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 $\text{SW} \wedge \overline{\text{GR}} \wedge \text{RO}$ classified as an apple by R coincides with the prototype, while the other instance $\overline{\text{SW}} \wedge \text{GR} \wedge \text{RO}$ classified as an apple by R is at a distance of 2 from the prototype $\text{SW} \wedge \overline{\text{GR}} \wedge \text{RO}$. 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 $\boldsymbol{C}$.

Proposition 7. *Computing how far a given class y_j is from a given prototype $\boldsymbol{x}$ is in* P *when $\mathcal{L}$ satisfies* $\mathbf{CD}$, $\mathbf{OPT}$, *and* $\mathbf{ME}$.

Proof. The approach for computing this value is as follows. One first generates a representation α in $\mathcal{L}$ of $\Sigma \mid y_j$, which

[4] 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^j obtained using the direct encoding (just consider a weight of $\frac{1}{2}$ for each x_i^j).

[5] We define $d_H(\Sigma_1, \Sigma_2) = +\infty$ whenever Σ_1 is inconsistent or Σ_2 is inconsistent.

is feasible in polynomial time when $\mathcal{L}$ satisfies $\mathbf{CD}$. Assuming that $\boldsymbol{x} = (x_1 = v_1, \cdots, x_n = v_n)$, one then considers a linear function $f_{\boldsymbol{x}}$ that associates a null weight with every variable from $Y \cup Z$, and with every x_i ($i \in \{1, \cdots, n\}$), a weight equal to -1 if $v_i = 1$ and a weight equal to 1 when $v_i = 0$. Accordingly, for every truth assignment ω over $X \cup Y \cup Z$, we have that $f_{\boldsymbol{x}}(\omega)$ + $\Sigma_{i=1}^{n} v_i$ is equal to the Hamming distance between the projection of ω onto X and $\boldsymbol{x}$. Since $\Sigma_{i=1}^{n} v_i$ does not depend on ω, the projections onto X of the interpretations ω maximizing the value of $f_{\boldsymbol{x}}$ are those maximizing the Hamming distance to $\boldsymbol{x}$. Obviously enough, they are also (precisely) the projections onto X of the interpretations ω minimizing the value of $-f_{\boldsymbol{x}}$, which is a linear function as well. When $\mathcal{L}$ satisfies $\mathbf{OPT}$, a representation β from $\mathcal{L}$ of the models of α minimizing the value of $-f_{\boldsymbol{x}}$ can be computed in polynomial time. Finally, when $\mathcal{L}$ satisfies $\mathbf{ME}$, a model ω' of β can be generated in polynomial time; the Hamming distance between the projection of ω' onto X and $\boldsymbol{x}$ is the maximal distance between an element of the class y_j and $\boldsymbol{x}$, i.e., the value we look for. □

4.2 Explanation Queries

We now turn to explanation queries, which are related to a specific input $\boldsymbol{x}$ and the way it has been classified by $\boldsymbol{C}$. Often, the reason why $\boldsymbol{x}$ has been classified as an element of a class y_j does not depend on all features over X but only on some of them. This leads to the two following notions of *minimum-cardinality explanations* and of *prime implicant explanations* introduced in (Shih, Choi, and Darwiche 2018b). When the prediction achieved by $\boldsymbol{C}$ is somewhat unexpected, the user may also ask for *counterfactual explanations*. Whatever the type of explanation that is looked for (minimum-cardinality, prime implicant, counterfactual), an input instance can have exponentially many explanations.

Enumerating minimum-cardinality explanations (EMC). Given an input $\boldsymbol{x}$ such that $\boldsymbol{C}(\boldsymbol{x}) = \boldsymbol{y}$ with $y_j = 1$, a minimum-cardinality explanation of $\boldsymbol{x}$ classified as an y_j is an input instance $\boldsymbol{x}'$ such that $\boldsymbol{C}(\boldsymbol{x}') = \boldsymbol{y}'$ with $y'_j = 1$ (i.e., $\boldsymbol{x}'$ is classified as an y_j as well), $\boldsymbol{x}'$ coheres with $\boldsymbol{x}$ on the ones in the sense that for any $k \in \{1, \ldots, n\}$, if $x'_k = 1$ then $x_k = 1$, and $\boldsymbol{x}'$ has a minimal number of coordinates set to 1. Roughly speaking, the features that are set to 1 in $\boldsymbol{x}'$ are enough to explain why $\boldsymbol{x}$ has been classified by $\boldsymbol{C}$ as an element of y_j. In our running example, a minimum-cardinality explanation of input $\text{SW} \wedge \text{GR} \wedge \overline{\text{RO}}$ (classified as an element of BAN by R) is $\overline{\text{RO}}$.

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

Proof. First, one computes in polynomial time a representation α in $\mathcal{L}$ 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 $\mathcal{L}$ satisfies $\mathbf{OPT}$, a representation β in $\mathcal{L}$ having for models those of α minimizing the value of f can be computed in polynomial time from β and f. By construction, the projection of every such model onto X is a minimum-cardinality explanation of $\boldsymbol{x}$. Furthermore, since every variable from $Y \cup Z$ is defined from X in Σ, two distinct models of β over $Var(\beta)$ necessarily have two distinct projections onto X. Thus, enumerating the models of β and projecting each of them onto X as soon as it is generated leads to enumerating the minimum-cardinality explanations of $\boldsymbol{x}$. The fact that $\mathcal{L}$ satisfies $\mathbf{ME}$ 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 $\mathcal{L}$ = OBDD. Indeed, OBDD satisfies $\mathbf{CD}$, $\mathbf{OPT}$, and $\mathbf{ME}$, but many other languages in the DNNF family (including DNNF itself) are strictly more succinct than 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 $\boldsymbol{x}$ necessarily contains every feature that is mandatory for being recognized as an element of y_j by $\boldsymbol{C}$. Similarly, no irrelevant feature for y_j participates in any minimum-cardinality explanation.

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

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

Proof. The computation of a prime implicant explanation of $\boldsymbol{x}$ is a follows. First, compute a representation α in $\mathcal{L}$ of $\exists \overline{X}.(\Sigma \mid y_j)$. This is feasible in polynomial time since $\mathcal{L}$ is supposed to satisfy $\mathbf{CD}$ and $\mathbf{FO}$. Then, the problem amounts to computing a prime implicant of α. This can be done in polynomial time using a greedy algorithm when $\mathcal{L}$ satisfies $\mathbf{IM}$: start with $\boldsymbol{x}$, which is a model (hence an implicant) of α, and for every literal of $\boldsymbol{x}$, test in polynomial time whether $\boldsymbol{x}$ deprived from this literal is still an implicant of the formula (this is feasible in polynomial time when $\mathcal{L}$ satisfies $\mathbf{IM}$); if the resulting term still is an implicant of α, 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 $\boldsymbol{x}$ contains every feature that is mandatory for being recognized as an element of y_j by $\boldsymbol{C}$. Furthermore, no prime implicant explanation of $\boldsymbol{x}$ contains a feature that is forbidden or irrelevant for being recognized as an element of

y_j by $\boldsymbol{C}$ (if $\exists \overline{X}.(\Sigma \mid y_j)$ is independent from x_k, x_k does not occur in any prime implicant of $\exists \overline{X}.(\Sigma \mid y_j)$). 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., $\mathcal{L}$ = OBDD) as far as we know. Especially, though the enumeration of the models of Σ 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 Σ. 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 $\boldsymbol{C}$ has classified a given instance $\boldsymbol{x}_1$. 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 $\boldsymbol{x}_2$ which is classified as an apple (and not as a grape fruit), and is as close as possible to $\boldsymbol{x}_1$ in terms of the number of common values of the features in the two inputs (alias the Hamming distance $d_H(\boldsymbol{x}_1, \boldsymbol{x}_2)$). Indeed, the set of features that differ in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ can be viewed as an explanation of why $\boldsymbol{x}_1$ 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 $\boldsymbol{x}_1 = \overline{\text{SW}} \wedge \overline{\text{GR}} \wedge \overline{\text{RO}}$ w.r.t. d_H is $\overline{\text{SW}} \wedge \text{GR} \wedge \text{RO}$ (it is at Hamming distance 2 of $\overline{\text{SW}} \wedge \overline{\text{GR}} \wedge \overline{\text{RO}}$). Another counterfactual explanation for $\boldsymbol{x}_1$ is $\text{SW} \wedge \overline{\text{GR}} \wedge \text{RO}$.

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

Proof. One first generates a representation α in $\mathcal{L}$ of $\Sigma \mid (y_k \wedge \overline{y_j})$, which is feasible in polynomial time when $\mathcal{L}$ satisfies **CD**. Then one considers the same function $f_{\boldsymbol{x}}$ as in the proof of Proposition 7. For every truth assignment ω over $X \cup Y \cup Z$, the value $f_{\boldsymbol{x}}(\omega) + \Sigma_{i=1}^{n} v_i$ is equal to the Hamming distance between the projection of ω onto X and $\boldsymbol{x}$. Since $\Sigma_{i=1}^{n} v_i$ does not depend on ω, the projections onto X of the interpretations ω minimizing the value of $f_{\boldsymbol{x}}$ are those minimizing the Hamming distance to $\boldsymbol{x}$. When $\mathcal{L}$ satisfies **OPT**, a representation β from $\mathcal{L}$ of the models of α minimizing the value of $f_{\boldsymbol{x}}$ can be computed in polynomial time. In addition, when $\mathcal{L}$ satisfies **ME**, starting from β, those models can be enumerated with a polynomial delay. Since every variable from $Y \cup Z$ is defined from X in Σ, two distinct models of β over $Var(\beta)$ necessarily have two distinct projections onto X. Hence the counterfactual explanations of $\boldsymbol{x}$ can be enumerated with a polynomial delay. □

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 $\boldsymbol{x}$ as an y_j defined as $min_{k=1,\ldots,m|k\neq j} d_H(\boldsymbol{x}, \Sigma \mid (y_k \wedge \overline{y_j}))$, i.e., the Hamming distance between $\boldsymbol{x}$ and a closest instance that has been recognized by $\boldsymbol{C}$ as belonging to a class distinct from y_j. In essence, this amounts to computing in sequence a counterfactual explanation of $\boldsymbol{x}$ for every y_k distinct from y_j, and for each of them to compute its Hamming distance to $\boldsymbol{x}$ 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 $\mathcal{L}$ = 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 Σ (i.e., when one just assumes that Σ 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 Σ 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 Σ.

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 Σ must be implemented. Hopefully, this is also the case.

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

A Decision-DNNF representation Σ is a Boolean circuit with a single root, leaves are labeled by literals over PS or Boolean constants, and internal nodes are decomposable $\wedge$-nodes or decision nodes of the form $ite(x, n_1, n_2)$, where ite stands for "*if ... then ... else ...*". In a decision node $ite(x, n_1, n_2)$, x is any propositional variable of PS (the decision variable), and the Decision-DNNF representations Σ_1 and Σ_2 rooted (respectively) at n_1 and n_2 do not contain any occurrence of x. ite can be considered as a ternary connective so that every node of the form $ite(x, n_1, n_2)$ can be viewed as a short for $(\bar{x} \wedge \Sigma_1) \vee (x \wedge \Sigma_2)$. Since in the latter statement, the $\wedge$-nodes are decomposable nodes, and the $\vee$-node is a deterministic one, every Decision-DNNF circuit can be considered as a d−DNNF circuit. Formally, there exists a linear-time equivalence-preserving translation from Decision-DNNF to d−DNNF. Structural restrictions on DNNF circuits based on a concept of vtree can also be imposed, leading to languages of structured DNNF representations that offer additional queries and transformations.

Thanks to the decomposability/determinism/structure conditions, languages of the DNNF family offer many queries and transformations of interest.[6] Thus, DNNF satisfies **CD**, **CO**, **FO**, **OPT**, $\wedge$**DC**, and d−DNNF offers all the queries and transformations listed in the paper, except **SE**, **EQ**, **FO**, $\wedge$**BC** (Darwiche and Marquis 2002; Darwiche and Marquis 2004). Structured d−DNNF circuits (given a fixed vtree) offers in addition **SE**, **EQ**, and $\wedge$**BC**. Accordingly, under the restriction when Σ is given as a DNNF circuit (resp. a d−DNNF circuit), the XAI queries EMC, ECO, EIN, IMA, and MCP (resp. CIN, MFR, and MCJ) are tractable. MCH is tractable when Σ is given as a structured d−DNNF circuit. Finally, though Decision-DNNF does not offer **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 $\overline{X}$ by a $\vee$-node (while keeping the same two children) turns the Decision-DNNF circuit at hand into a d−DNNF circuit (see (Lagniez, Lonca, and Marquis 2020) for details). As a consequence, DPI is tractable when Σ is given as a Decision-DNNF circuit, and IIR and IMO are tractable when Σ is given as a structured Decision-DNNF circuit.

[6] In addition, 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 Σ is given at start as a CNF formula (Darwiche 2001; Darwiche 2004; Pipatsrisawat and Darwiche 2010; Muise et al. 2012; Lagniez and Marquis 2017).

XAI query	Conditions on $\mathcal{L}$ making the query tractable
EMC	**CD**, **OPT**, **ME**
DPI	**CD**, **FO**, **IM**
ECO	**CD**, **OPT**, **ME**
CIN	**CD**, **CT**
EIN	**CD**, **ME**
CAM	**CD**, **CT**
EAM	**CD**, **ME**
MFR	**CD**, **CT**
IMA	**CD**, **CO**
IIR	**CD**, **FO**, **EQ**
IMO	**CD**, **FO**, **SE**
MCJ	**CD**, **CT**
MCH	**CD**, $\wedge$**BC**, $\wedge$**DC**, **OPT**, **ME**
MCP	**CD**, **OPT**, **ME**

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

6 Conclusion

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

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 Σ 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 DNNF appear as valuable, ensuring that many XAI queries from the table are tractable whenever Σ is represented in $\mathcal{L}$.

This work calls for many perspectives. Designing encoding schemas suited to other types of classifiers (e.g., other ensemble methods) is one of them. From the theory side, deriving bounds on the size of the compiled representations in various target languages of the corresponding encodings is an interesting issue we would like to address. From a more practical point of view, we plan to make experiments to determine the scalability of the compilation-based approach to XAI for several encoding schemas (associated with classifiers of various types) and for several target languages for knowledge compilation.

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

Beth, E. 1953. On Padoa's method in the theory of definition. *Indagationes mathematicae* 15:330–339.

Bova, S.; Capelli, F.; Mengel, S.; and Slivovsky, F. 2016. Knowledge compilation meets communication complexity. In *Proc. of IJCAI'16*, 1008–1014.

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

Breiman, L. 1996. Bagging predictors. *Machine Learning* 24(2):123–140.

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

Bryant, R. E. 1986. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers* C-35(8):677–692.

Bunel, R.; Turkaslan, I.; Torr, P. H. S.; Kohli, P.; and Mudigonda, P. K. 2018. A unified view of piecewise linear neural network verification. In *Proc. of NeurIPS'18*, 4795–4804.

Chan, H., and Darwiche, A. 2003. Reasoning about bayesian network classifiers. In *Proc. of UAI'03*, 107–115.

Choi, A.; Shih, A.; Goyanka, A.; and Darwiche, A. 2020. On symbolically encoding the behavior of random forests. In *Proc. of FoMLAS'20, 3rd Workshop on Formal Methods for ML-Enabled Autonomous Systems, Workshop at CAV'20*. To appear.

Darwiche, A., and Hirth, A. 2020. On the reasons behind decisions. In *Proc. of ECAI'20*. To appear.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17:229–264.

Darwiche, A., and Marquis, P. 2004. Compiling propositional weighted bases. *Artificial Intelligence* 157(1-2):81–113.

Darwiche, A. 2001. Decomposable negation normal form. *Journal of the Association for Computing Machinery* 48(4):608–647.

Darwiche, A. 2004. New advances in compiling CNF into decomposable negation normal form. In *Proc. of ECAI'04*, 328–332.

Darwiche, A. 2011. SDD: A new canonical representation of propositional knowledge bases. In *Proc. of IJCAI'11*, 819–826.

de Kleer, J. 1989. A comparison of ATMS and CSP techniques. In *Proc. of IJCAI'89*, 290–296.

Fargier, H., and Marquis, P. 2014. Disjunctive closures for knowledge compilation. *Artificial Intelligence* 216:129–162.

Fargier, H.; Marquis, P.; and Niveau, A. 2013. Towards a knowledge compilation map for heterogeneous representation languages. In *Proc. of IJCAI'13*, 877–883.

Garey, M., and Johnson, D. 1979. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman.

Gergov, J., and Meinel, C. 1994. Efficient analysis and manipulation of OBDDs can be extended to FBDDs. *IEEE Transactions on Computers* 43(10):1197–1209.

Goldsmith, J.; Hagen, M.; and Mundhenk, M. 2005. Complexity of DNF and isomorphism of monotone formulas. In *Proc. of MFCS'05*, 410–421.

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

Ho, T. K. 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(8):832–844.

Huang, J., and Darwiche, A. 2007. The language of search. *Journal of Artificial Intelligence Research (JAIR)* 29:191–219.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019. Abduction-based explanations for machine learning models. In *Proc. of AAAI'19*, 1511–1519.

Katz, G.; Huang, D. A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljic, A.; Dill, D. L.; Kochenderfer, M. J.; and Barrett, C. W. 2019. The marabou framework for verification and analysis of deep neural networks. In *Proc. of CAV'19*, 443–452.

Koriche, F.; Berre, D. L.; Lonca, E.; and Marquis, P. 2016. Fixed-parameter tractable optimization under DNNF constraints. In *Proc. of ECAI'16*, 1194–1202.

Lagniez, J.-M., and Marquis, P. 2017. An Improved Decision-DNNF Compiler. In *Proc. of IJCAI'17*, 667–673.

Lagniez, J.; Lonca, E.; and Marquis, P. 2020. Definability for model counting. *Artificial Intelligence* 281:103229.

Lang, J., and Marquis, P. 2008. On propositional definability. *Artificial Intelligence* 172(8-9):991–1017.

Lang, J.; Liberatore, P.; and Marquis, P. 2003. Propositional independence: Formula-variable independence and forgetting. *Journal of Artificial Intelligence Research* 18:391–443.

Leofante, F.; Narodytska, N.; Pulina, L.; and Tacchella, A. 2018. Automated verification of neural networks: Advances, challenges and perspectives. *CoRR* abs/1805.09938.

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

Molnar, C.; Casalicchio, G.; and Bischl, B. 2018. iml: An R package for interpretable machine learning. *Journal of Open Source Software* 3(26):786.

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

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

Dsharp: Fast d-DNNF compilation with sharpSAT. In *Proc. of AI'12*, 356–361.

Narodytska, N.; Kasiviswanathan, S. P.; Ryzhyk, L.; Sagiv, M.; and Walsh, T. 2018. Verifying properties of binarized deep neural networks. In *Proc. of AAAI'18*, 6615–6624.

Narodytska, N.; Shrotri, A. A.; Meel, K. S.; Ignatiev, A.; and Marques-Silva, J. 2019. Assessing heuristic machine learning explanations with model counting. In *Proc. of SAT'19*, 267–278.

Niveau, A.; Fargier, H.; Pralet, C.; and G.Verfaillie. 2010. Knowledge compilation using interval automata and applications to planning. In *Proc. of ECAI'10*, 459–464.

Oztok, U., and Darwiche, A. 2014. On compiling CNF into Decision-DNNF. In *Proc. of CP'14*, 42–57.

Pipatsrisawat, K., and Darwiche, A. 2008. New compilation languages based on structured decomposability. In *Proc. of AAAI'08*, 517–522.

Pipatsrisawat, K., and Darwiche, A. 2010. Top-down algorithms for constructing structured DNNF: Theoretical and practical implications. In *Proc. of ECAI'10*, 3–8.

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

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-precision model-agnostic explanations. In *Proc. of AAAI'18*, 1527–1535.

Shi, W.; Shih, A.; Darwiche, A.; and Choi, A. 2020. On tractable representations of binary neural networks. In *Proc. of KR'20*. To appear.

Shih, A.; Choi, A.; and Darwiche, A. 2018a. Formal verification of Bayesian network classifiers. In *Proc. of PGM'18*, 427–438.

Shih, A.; Choi, A.; and Darwiche, A. 2018b. A symbolic approach to explaining Bayesian network classifiers. In *Proc. of IJCAI'18*, 5103–5111.

Shih, A.; Choi, A.; and Darwiche, A. 2019. Compiling Bayesian networks into decision graphs. In *Proc. of AAAI'19*, 7966–7974.

Shih, A.; Darwiche, A.; and Choi, A. 2019. Verifying binarized neural networks by Angluin-style learning. In *Proc. of SAT'19*, 354–370.

Walsh, T. 2000. SAT v CSP. In *Proc. of CP'00*, 441–456.