View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by INRIA a CCSD electronic archive server

The Tractability of SHAP-Score-Based Explanations over Deterministic and Decomposable Boolean Circuits

Marcelo Arenas, Pablo Barceló, Leopoldo Bertossi, Mikaël Monet

To cite this version:

Marcelo Arenas, Pablo Barceló, Leopoldo Bertossi, Mikaël Monet. The Tractability of SHAP-Score-Based Explanations over Deterministic and Decomposable Boolean Circuits. AAAI 2021 - 35th Conference on Artificial Intelligence, Feb 2021, Virtual, France. hal-03147623

HAL Id: hal-03147623

https://hal.inria.fr/hal-03147623

Submitted on 20 Feb 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

arXiv:2007.14045v2 [cs.AI] 10 Dec 2020

The Tractability of SHAP-Score-Based Explanations over Deterministic and Decomposable Boolean Circuits

Marcelo Arenas[1,2,3], **Pablo Barceló**[2,3], **Leopoldo Bertossi**[4,3], **Mikaël Monet**[5]

[1] Department of Computer Science, Universidad Católica de Chile
[2] Institute for Mathematical and Computational Engineering, Universidad Católica de Chile
[3] IMFD Chile
[4] Universidad Adolfo Ibáñez, FIC, Chile
[5] Inria Lille – Nord Europe & UMR 9189 CRIStAL, France

Abstract

Scores based on Shapley values are widely used for providing explanations to classification results over machine learning models. A prime example of this is the influential SHAP-score, a version of the Shapley value that can help explain the result of a learned model on a specific entity by assigning a score to every feature. While in general computing Shapley values is a computationally intractable problem, it has recently been claimed that the SHAP-score can be computed in polynomial time over the class of decision trees. In this paper, we provide a proof of a stronger result over Boolean models: the SHAP-score can be computed in polynomial time over *deterministic and decomposable Boolean circuits*. Such circuits, also known as *tractable Boolean circuits*, generalize a wide range of Boolean circuits and binary decision diagrams classes, including binary decision trees, Ordered Binary Decision Diagrams (OBDDs) and Free Binary Decision Diagrams (FBDDs). We also establish the computational limits of the notion of SHAP-score by observing that, under a mild condition, computing it over a class of Boolean models is always polynomially as hard as the model counting problem for that class. This implies that both determinism and decomposability are essential properties for the circuits that we consider, as removing one or the other renders the problem of computing the SHAP-score intractable (namely, #P-hard).

1 Introduction

Explainable artificial intelligence has become an active area of research. Central to it is the observation that artificial intelligence (AI) and machine learning (ML) models cannot always be blindly applied without being able to interpret and explain their results. For example, when someone applies for a loan and sees their application rejected by an algorithmic decision-making system, the system should be able to provide an explanation for that decision. Explanations can be *global* – focusing on the general input/output relation of the model –, or *local* – focusing on how features affect the decision of the model for a specific input. Recent literature has strengthened the importance of the latter by showing their ability to provide explanations that are often overlooked by global explanations (Molnar 2020).

One natural way of providing local explanations for classification models consists in assigning *numerical scores* to the feature values of an entity that has gone through the classification process. Intuitively, the higher the score of a feature value, the more relevant it should be considered. It is in this context that the SHAP-*score* has been introduced (Lundberg and Lee 2017; Lundberg et al. 2020). This recent notion has rapidly gained attention and is becoming influential. There are two properties of the SHAP-score that support its rapid adoption. First, its definition is quite general and can be applied to any kind of classification model. Second, the definition of the SHAP-score is grounded on the well-known *Shapley value* (Shapley 1953; Roth 1988), that has already been used successfully in several domains of computer science; see, e.g., (Hunter and Konieczny 2010; Livshits et al. 2020; Michalak et al. 2013; Cesari et al. 2018). Thus, SHAP-scores have a clear, intuitive, combinatorial meaning, and inherit all the desirable properties of the Shapley value.

For a given classifier M, entity $\mathbf{e}$ and feature x, the SHAP-score $\mathsf{SHAP}(M, \mathbf{e}, x)$ intuitively represents the importance of the feature value $\mathbf{e}(x)$ to the classification result $M(\mathbf{e})$. In its general formulation, $\mathsf{SHAP}(M, \mathbf{e}, x)$ is a weighted average of differences of expected values of the outcomes (c.f. Section 2 for its formal definition). Unfortunately, computing quantities that are based on the notion of Shapley value is in general intractable. Indeed, in many scenarios the computation turns out to be #P-hard (Faigle and Kern 1992; Deng and Papadimitriou 1994; Livshits et al. 2020; Bertossi et al. 2020), which makes the notion difficult to use – if not impossible – for practical purposes (Arora and Barak 2009). Therefore, a natural question is: For what kinds of classification models the computation of the SHAP-score can be done efficiently? This is the subject of this paper.

In this work, we focus on classifiers working with *binary feature values* (i.e., propositional features that can take the values 0 or 1), and that return 1 (accept) or 0 (reject) for each entity. We will call these *Boolean classifiers*. The second assumption that we make is that the underlying probability distribution on the population of entities is what we call a *product distribution*, where each binary feature x has a probability $\mathrm{p}(x)$ of being equal to 1, independently of the other features. We note here that the restriction to binary inputs can be relevant in many practical scenarios where the features are of a propositional nature.

More specifically, we investigate Boolean classifiers defined as *deterministic and decomposable Boolean circuits*, a widely studied model in *knowledge compilation* (Darwiche 2001; Darwiche and Marquis 2002). Such circuits encompass a wide range of Boolean models and binary decision diagrams classes that are considered in knowledge compilation, and in AI more generally. For instance, they generalize *binary decision trees*, *ordered binary decision diagrams* (OBDDs), *free binary decision diagrams* (FBDDs), and *deterministic and decomposable negation normal norms* (d-DNNFs) (Darwiche 2001; Amarilli et al. 2020; Darwiche and Hirth 2020). These circuits are also known under the name of *tractable Boolean circuits*, that is used in recent literature (Shih, Darwiche, and Choi 2019; Shi et al. 2020; Shih, Choi, and Darwiche 2018b,a; Shih et al. 2019; Peharz et al. 2020). We provide an example of a deterministic and decomposable Boolean circuit next (and give the formal definition in Section 2).

Example 1.1. We want to classify papers submitted to a conference as rejected (Boolean value 0) or accepted (Boolean value 1). Papers are described by features **fg**, **dtr**, **nf** and **na**, which stand for "follows guidelines", "deep theoretical result", "new framework" and "nice applications", respectively. The Boolean classifier for the papers is given by the Boolean circuit in Figure 1. The input of this circuit are the features **fg**, **dtr**, **nf** and **na**, each of which can take value either 0 or 1, depending on whether the feature is present (1) or absent (0). The nodes with labels $\neg$, $\vee$ or $\wedge$ are logic gates, and the associated Boolean value of each one of them depends on the logical connective represented by its label and the Boolean values of its inputs. The output value of the circuit is given by the top node in the figure.

The Boolean circuit in Figure 1 is said to be *decomposable*, because for each $\wedge$-gate, the sets of features of its inputs are pairwise disjoint. For instance, in the case of the top node in Figure 1, the left-hand side input has $\{\mathbf{fg}\}$ as its set of features, while its right-hand side input has $\{\mathbf{dtr}, \mathbf{nf}, \mathbf{na}\}$ as its set of features, which are disjoint. Also, this circuit is said to be *deterministic*, which means that for every $\vee$-gate, two (or more) of its inputs cannot be given value 1 by the same Boolean assignment for the features. For instance, in the case of the only $\vee$-gate in Figure 1, if a Boolean assignment for the features gives value 1 to its left-hand side input, then feature **dtr** has to be given value 1 and, thus, such an assignment gives value 0 to the right-hand side input of the $\vee$-gate. In the same way, it can be shown that if a Boolean assignment for the features gives value 1 to the right-hand side input of this $\vee$-gate, then it gives value 0 to its left-hand side input. □

Readers who are not familiar with knowledge compilation can simply think about deterministic and decomposable circuits as a tool for establishing in a uniform manner the tractability of computing SHAP-scores on several Boolean classifier classes. Our main contributions are the following:

1. We provide a polynomial time algorithm that computes the SHAP-score for deterministic and decomposable Boolean circuits, in the special case of *uniform probability distributions* (that is, when each $\mathrm{p}(x)$ is $\frac{1}{2}$). In particular, this provides a precise proof of the claim made in (Lundberg et al. 2020) that the SHAP-score for Boolean classifiers given as decision trees can be computed in polynomial time. Moreover, we also obtain as a corollary that the SHAP-score for Boolean classifiers given as OBDDs, FBDDs and d-DNNFs can be computed in polynomial time.
2. We observe that computing the SHAP-score on Boolean circuits in a class is always polynomially as hard as the *model counting* problem for that class (under a mild condition). By using this observation, we obtain that each one of the *determinism* assumption and the *decomposability* assumption is necessary for tractability.
3. Last, we show that the results above (and most interestingly, the polynomial-time algorithm) can be extended to the SHAP-score defined on product distributions for the entity population.

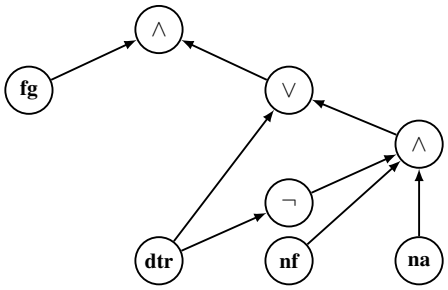


Figure 1: A deterministic and decomposable Boolean Circuit as a classifier.

Our contributions should be compared to the results obtained in the contemporaneous paper (Van den Broeck et al. 2020). There, the authors establish the following theorem: for every class $\mathcal{C}$ of classifiers and under product distributions, the problem of computing the SHAP-score for $\mathcal{C}$ is polynomial-time equivalent to the problem of computing the expected value for the models in $\mathcal{C}$. Since computing expectations is in polynomial time for tractable Boolean circuits, this in particular implies that computing the SHAP-score is in polynomial time for the circuits that we consider; in other words, their results capture ours. However, there is a fundamental difference in the approach taken to show tractability: their reduction uses multiple oracle calls to the problem of computing expectations, whereas we provide a more direct algorithm to compute the SHAP-score on these circuits.

Our algorithm for computing the SHAP-score could be used in practical scenarios. Indeed, recently, some classes of classifiers have been compiled into tractable Boolean circuits. This is the case, for instance, of Bayesian Classifiers (Shih, Choi, and Darwiche 2018a), Binary Neural Networks (Shi et al. 2020), and Random Forests (Choi et al. 2020). The idea is to start with a Boolean classifier M given in a formalism that is hard to interpret – for instance a Binary neural network – and to compute a tractable Boolean circuit M' that is equivalent to M (this computation can be

expensive). One can then use M' and the nice properties of tractable Boolean circuits to interpret the decisions of the model. Hence, this makes it possible to apply the results in this paper on the SHAP-score to those classes of classifiers.

Paper structure. We give preliminaries in Section 2. In Section 3, we prove that the SHAP-score can be computed in polynomial time for deterministic and decomposable Boolean circuits for uniform probability distributions. In Section 4 we establish the limits of the tractable computation of the SHAP-score. Next we show in Section 5 that our results extend to the setting where we consider product distributions. We conclude and discuss future work in Section 6.

2 Preliminaries

2.1 Entities, distributions and classifiers

Let X be a finite set of *features*, also called *variables*. An *entity* over X is a function $\mathbf{e} : X \to \{0,1\}$. We denote by $\mathsf{ent}(X)$ the set of all entities over X. On this set, we consider the *uniform probability distribution*, i.e., for an event $E \subseteq \mathsf{ent}(X)$, we have that $P(E) := \frac{|E|}{2^{|X|}}$. We will come back to this assumption in Section 5, where we will consider the more general product distributions (we start with the uniform distribution to ease the presentation).

A *Boolean classifier* M over X is a function $M : \mathsf{ent}(X) \to \{0,1\}$ that maps every entity over X to 0 or 1. We say that M *accepts* an entity $\mathbf{e}$ when $M(\mathbf{e}) = 1$, and that it *rejects* it if $M(\mathbf{e}) = 0$. Since we consider $\mathsf{ent}(X)$ to be a probability space, M can be regarded as a random variable.

2.2 The SHAP-score over Boolean classifiers

Let $M : \mathsf{ent}(X) \to \{0,1\}$ be a Boolean classifier over the set X of features. Given an entity $\mathbf{e}$ over X and a subset $S \subseteq X$ of features, the set $\mathsf{cw}(\mathbf{e}, S) := \{\mathbf{e}' \in \mathsf{ent}(X) \mid \mathbf{e}'(x) = \mathbf{e}(x) \text{ for each } x \in S\}$ contains those entities that coincide with $\mathbf{e}$ over each feature in S. In other words, $\mathsf{cw}(\mathbf{e}, S)$ is the set of entities that are *consistent with* $\mathbf{e}$ on S. Then, given an entity $\mathbf{e} \in \mathsf{ent}(X)$ and $S \subseteq X$, we define the *expected value of M over $X \setminus S$ with respect to* $\mathbf{e}$ as

$$\phi(M, \mathbf{e}, S) := \mathbb{E}\big[M(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S)\big].$$

Since we consider the uniform distribution over $\mathsf{ent}(X)$, we have that

$$\phi(M, \mathbf{e}, S) = \sum_{\mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S)} \frac{1}{2^{|X \setminus S|}} M(\mathbf{e}').$$

Intuitively, $\phi(M, \mathbf{e}, S)$ is the probability that $M(\mathbf{e}') = 1$, conditioned on the inputs $\mathbf{e}' \in \mathsf{ent}(X)$ to coincide with $\mathbf{e}$ over each feature in S. This function is then used in the general formula of the Shapley value (Shapley 1953; Roth 1988) to obtain the SHAP-score for feature values in $\mathbf{e}$.

Definition 2.1. *Given a Boolean classifier M over a set of features X, an entity $\mathbf{e}$ over X, and a feature $x \in X$, the* SHAP score of feature x on $\mathbf{e}$ with respect to M *is defined as*

$$\mathsf{SHAP}(M, \mathbf{e}, x) := \sum_{S \subseteq X \setminus \{x\}} \frac{|S|!\,(|X| - |S| - 1)!}{|X|!} \Big(\phi(M, \mathbf{e}, S \cup \{x\}) - \phi(M, \mathbf{e}, S) \Big). \quad (1)$$

Thus, $\mathsf{SHAP}(M, \mathbf{e}, x)$ is a weighted average of the contribution of feature x on $\mathbf{e}$ to the classification result, i.e., of the differences between having it and not, under all possible permutations of the other feature values. Observe that, from this definition, a high positive value of $\mathsf{SHAP}(M, \mathbf{e}, x)$ intuitively means that setting x to $\mathbf{e}(x)$ strongly leans the classifier towards acceptance, while a high negative value of $\mathsf{SHAP}(M, \mathbf{e}, x)$ means that setting x to $\mathbf{e}(x)$ strongly leans the classifier towards rejection.

2.3 Deterministic and decomposable Boolean circuits

A Boolean circuit over a set of variables X is a directed acyclic graph C such that

(i) Every node without incoming edges is either a *variable gate* or a *constant gate*. A variable gate is labeled with a variable from X, and a constant gate is labeled with either 0 or 1;

(ii) Every node with incoming edges is a *logic gate*, and is labeled with a symbol $\wedge$, $\vee$ or $\neg$. If it is labeled with the symbol $\neg$, then it has exactly one incoming edge;[1]

(iii) Exactly one node does not have any outgoing edges, and this node is called the *output gate of C*.

Such a Boolean circuit C represents a Boolean classifier in the expected way – we assume the reader to be familiar with Boolean logic –, and we write $C(\mathbf{e})$ for the value in $\{0,1\}$ of the output gate of C when we evaluate C over the entity $\mathbf{e}$.

Several restrictions of Boolean circuits with good computational properties have been studied. Let C be a Boolean circuit over a set of variables X and g a gate of C. The Boolean circuit C_g over X is defined by considering the subgraph of C induced by the set of gates g' in C for which there exists a path from g' to g in C. Notice that g is the output gate of C_g. The set $\mathsf{var}(g)$ is defined as the set of variables $x \in X$ such that there exists a variable gate with label x in C_g. Then, an $\vee$-gate g of C is said to be *deterministic* if for every pair g_1, g_2 of distinct input gates of g, the Boolean circuits C_{g_1} and C_{g_2} are disjoint in the sense that there is no entity $\mathbf{e}$ that is accepted by both C_{g_1} and C_{g_2} (that is, there is no entity $\mathbf{e} \in \mathsf{ent}(X)$ such that $C_{g_1}(\mathbf{e}) = C_{g_2}(\mathbf{e}) = 1$). The circuit C is called *deterministic* if every $\vee$-gate of C is deterministic. An $\wedge$-gate g of C is said to be *decomposable* if for every pair g_1, g_2 of distinct input gates of g, we have that $\mathsf{var}(g_1) \cap \mathsf{var}(g_2) = \emptyset$. Then, C is called *decomposable* if every $\wedge$-gate of C is decomposable.

[1] Recall that the fan-in of a gate is the number of its input gates. In our definition of Boolean circuits, we allow unbounded fan-in $\wedge$- and $\vee$-gates.

Example 2.2. In Example 1.1, we explained at an intuitive level why the Boolean circuit in Figure 1 is deterministic and decomposable. By using the terminology defined in the previous paragraph, it can be formally checked that this Boolean circuit indeed satisfies these conditions. □

As mentioned before, deterministic and decomposable Boolean circuits generalize many decision diagrams and Boolean circuits classes. We refer to (Darwiche 2001; Amarilli et al. 2020) for detailed studies of knowledge compilation classes and of their precise relationships. For the reader's convenience, we explain in the supplementary material how FBDDs and binary decision trees can be encoded in linear time as deterministic and decomposable Boolean circuits.

3 Tractable Computation of the SHAP-Score

In this section, we prove our first tractability result, namely, that computing the SHAP-score for Boolean classifiers given as deterministic and decomposable Boolean circuits can be done in polynomial time, for uniform probability distributions. Formally:

Theorem 3.1. *The following problem can be solved in polynomial time. Given as input a deterministic and decomposable Boolean circuit C over a set of features X, an entity $\mathbf{e} : X \to \{0,1\}$, and a feature $x \in X$, compute the value* $\mathsf{SHAP}(C, \mathbf{e}, x)$.

In particular, since binary decision trees, OBDDs, FBDDs and d-DNNFs are all restricted kinds of deterministic and decomposable circuits, we obtain as a consequence of Theorem 3.1 that this problem is also in polynomial time for these classes. For instance, for binary decision trees we obtain:

Corollary 3.2. *The following problem can be solved in polynomial time. Given as input a binary decision tree T over a set of features X, an entity $\mathbf{e} : X \to \{0,1\}$, and a feature $x \in X$, compute the value* $\mathsf{SHAP}(T, \mathbf{e}, x)$.

The authors of (Lundberg et al. 2020) give a proof of this result, but, unfortunately, with few details to fully understand it. Moreover, it is important to notice that Theorem 3.1 is a nontrivial extension of the result for decision trees, as it is known that deterministic and decomposable circuits can be exponentially more succinct than binary decision trees (in fact, than FBDDs) at representing Boolean classifiers (Darwiche 2001; Amarilli et al. 2020).

In order to prove Theorem 3.1, we need to introduce some notation. Let M be a Boolean classifier over a set of features X. We write $\mathsf{SAT}(M) \subseteq \mathsf{ent}(X)$ for the set of entities that are accepted by M, and $\#\mathsf{SAT}(M)$ for the cardinality of this set. Let $\mathbf{e}, \mathbf{e}' \in \mathsf{ent}(X)$ be a pair of entities over X. We define $\mathsf{sim}(\mathbf{e}, \mathbf{e}') := \{x \in X \mid \mathbf{e}(x) = \mathbf{e}'(x)\}$ to be the set of features on which $\mathbf{e}$ and $\mathbf{e}'$ coincide. Given a Boolean classifier M over X, an entity $\mathbf{e} \in \mathsf{ent}(X)$ and a natural number $k \leq |X|$, we define the set $\mathsf{SAT}(M, \mathbf{e}, k) := \mathsf{SAT}(M) \cap \{\mathbf{e}' \in \mathsf{ent}(X) \mid |\mathsf{sim}(\mathbf{e}, \mathbf{e}')| = k\}$, in other words, the set of entities $\mathbf{e}'$ that are accepted by M and which coincide with $\mathbf{e}$ in exactly k features. Naturally, we write $\#\mathsf{SAT}(M, \mathbf{e}, k)$ for the size of $\mathsf{SAT}(M, \mathbf{e}, k)$.

Example 3.3. Let M be the Boolean classifier represented by the circuit in Example 1.1. Then $\mathsf{SAT}(M)$ is the set containing all papers that are accepted according to M, so that $\#\mathsf{SAT}(M) = 5$. Now, consider the entity $\mathbf{e}$ such that $\mathbf{e}(\mathbf{fg}) = 1$, $\mathbf{e}(\mathbf{dtr}) = 1$, $\mathbf{e}(\mathbf{nf}) = 0$ and $\mathbf{e}(\mathbf{na}) = 1$. Then one can check that $\#\mathsf{SAT}(M, \mathbf{e}, 0) = 0$, $\#\mathsf{SAT}(M, \mathbf{e}, 1) = 0$, $\#\mathsf{SAT}(M, \mathbf{e}, 2) = 2$, $\#\mathsf{SAT}(M, \mathbf{e}, 3) = 2$ and $\#\mathsf{SAT}(M, \mathbf{e}, 4) = 1$. □

Our proof of Theorem 3.1 is technical and is divided into two modular parts. The first part, which is developed in Section 3.1, consists in showing that the problem of computing $\mathsf{SHAP}(\cdot,\cdot,\cdot)$ can be reduced in polynomial time to that of computing $\#\mathsf{SAT}(\cdot,\cdot,\cdot)$. This part of the proof is a sequence of formula manipulations, and it only uses the fact that deterministic and decomposable circuits can be efficiently *conditioned* on a variable value (to be defined in Section 3.1). In the second part of the proof, which is developed in Section 3.2, we show that computing $\#\mathsf{SAT}(\cdot,\cdot,\cdot)$ can be done in polynomial time for deterministic and decomposable Boolean circuits. It is in this part that the properties of deterministic and decomposable circuits are really used.

3.1 Reducing $\mathsf{SHAP}(\cdot,\cdot,\cdot)$ to $\#\mathsf{SAT}(\cdot,\cdot,\cdot)$

In this section, we show that for deterministic and decomposable Boolean circuits, the computation of the SHAP-score can be reduced in polynomial time to the computation of $\#\mathsf{SAT}(\cdot,\cdot,\cdot)$. To achieve this, we will need two more definitions. Let M be a Boolean classifier over a set of features X and $x \in X$, and let Boolean classifiers $M_{+x} : \mathsf{ent}(X \setminus \{x\}) \to \{0,1\}$ and $M_{-x} : \mathsf{ent}(X \setminus \{x\}) \to \{0,1\}$ be defined as follows. For $\mathbf{e} \in \mathsf{ent}(X \setminus \{x\})$, we write $\mathbf{e}_{+x}$ and $\mathbf{e}_{-x}$ the entities over X such that $\mathbf{e}_{+x}(x) = 1$, $\mathbf{e}_{-x}(x) = 0$ and $\mathbf{e}_{+x}(y) = \mathbf{e}_{-x}(y) = \mathbf{e}(y)$ for every $y \in X \setminus \{x\}$. Then define $M_{+x}(\mathbf{e}) := M(\mathbf{e}_{+x})$ and $M_{-x}(\mathbf{e}) := M(\mathbf{e}_{-x})$. In the literature, M_{+x} (resp., M_{-x}) is called the *conditioning by x (resp., by $\neg x$) of M*. Conditioning can be done in linear time for a Boolean circuit C by replacing every gate with label x by a constant gate with label 1 (resp., 0). We write C_{+x} (resp., C_{-x}) for the Boolean circuit obtained via this transformation. One can easily check that, if C is deterministic and decomposable, then C_{+x} and C_{-x} are deterministic and decomposable as well.

We now introduce the second definition needed for the proof. For a Boolean classifier M over a set of variables X, an entity $\mathbf{e} \in \mathsf{ent}(X)$ and an integer $k \leq |X|$, we define

$$\mathsf{H}(M, \mathbf{e}, k) := \sum_{\substack{S \subseteq X \\ |S| = k}} \sum_{\mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S)} M(\mathbf{e}'). \tag{2}$$

We first explain how computing $\mathsf{SHAP}(\cdot,\cdot,\cdot)$ can be reduced in polynomial time to the problem of computing $\mathsf{H}(\cdot,\cdot,\cdot)$, and then how computing $\mathsf{H}(\cdot,\cdot,\cdot)$ can be reduced in polynomial time to computing $\#\mathsf{SAT}(\cdot,\cdot,\cdot)$.

Reducing from $\mathsf{SHAP}(\cdot,\cdot,\cdot)$ to $\mathsf{H}(\cdot,\cdot,\cdot)$. We need to compute $\mathsf{SHAP}(C, \mathbf{e}, x)$, for a given deterministic and decomposable circuit C over a set of variables X, entity $\mathbf{e} \in$

$\mathsf{ent}(X)$, and feature $x \in X$. Let $n = |X|$, and define

$$\mathsf{Diff}_k(C, \mathbf{e}, x) \coloneqq \sum_{\substack{S \subseteq X \setminus \{x\} \\ |S|=k}} (\phi(C, \mathbf{e}, S \cup \{x\}) - \phi(C, \mathbf{e}, S)).$$

Then by the definition of the SHAP-score in (1), we have:

$$\mathsf{SHAP}(C, \mathbf{e}, x) = \sum_{k=0}^{n-1} \frac{k!(n-k-1)!}{n!} \mathsf{Diff}_k(C, \mathbf{e}, x).$$

Observe that all arithmetical terms (such as $k!$ or $n!$) can be computed in polynomial time: this is simply because n is given in unary, as it is bounded by the size of the circuit. Therefore, it is enough to show how to compute in polynomial time the quantities $\mathsf{Diff}_k(C, \mathbf{e}, x)$ for each $k \in \{0, \ldots, n-1\}$, as $n = |X|$ is bounded by the size of the input $(C, \mathbf{e}, x)$. By definition of $\phi(\cdot, \cdot, \cdot)$, we have that $\mathsf{Diff}_k(C, \mathbf{e}, x) = \alpha - \beta$, where:

$$\alpha = \sum_{\substack{S \subseteq X \setminus \{x\} \\ |S|=k}} \frac{1}{2^{n-(k+1)}} \sum_{\mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S \cup \{x\})} C(\mathbf{e}')$$

$$\beta = \sum_{\substack{S \subseteq X \setminus \{x\} \\ |S|=k}} \frac{1}{2^{n-k}} \sum_{\mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S)} C(\mathbf{e}').$$

Next we show how the computation of α and β can be reduced in polynomial-time to the computation of $\mathsf{H}(\cdot, \cdot, \cdot)$. For an entity $\mathbf{e} \in \mathsf{ent}(X)$ and $S \subseteq X$, let $\mathbf{e}_{|S}$ be the entity over S that is obtained by restricting $\mathbf{e}$ to the domain S (that is, formally $\mathbf{e}_{|S} \in \mathsf{ent}(S)$ and $\mathbf{e}_{|S}(y) \coloneqq \mathbf{e}(y)$ for every $y \in S$). Then, starting with β, we have that:

$$\begin{aligned}
\beta &= \sum_{\substack{S \subseteq X \setminus \{x\} \\ |S|=k}} \frac{1}{2^{n-k}} \sum_{\mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S)} C(\mathbf{e}') \\
&= \Bigg[\sum_{\substack{S \subseteq X \setminus \{x\} \\ |S|=k}} \frac{1}{2^{n-k}} \sum_{\substack{\mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S) \\ \mathbf{e}'(x)=1}} C(\mathbf{e}') \Bigg] \\
&\qquad + \Bigg[\sum_{\substack{S \subseteq X \setminus \{x\} \\ |S|=k}} \frac{1}{2^{n-k}} \sum_{\substack{\mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S) \\ \mathbf{e}'(x)=0}} C(\mathbf{e}') \Bigg] \\
&= \Bigg[\frac{1}{2^{n-k}} \sum_{\substack{S \subseteq X \setminus \{x\} \\ |S|=k}} \sum_{\mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|X \setminus \{x\}}, S)} C_{+x}(\mathbf{e}'') \Bigg] \\
&\qquad + \Bigg[\frac{1}{2^{n-k}} \sum_{\substack{S \subseteq X \setminus \{x\} \\ |S|=k}} \sum_{\mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|X \setminus \{x\}}, S)} C_{-x}(\mathbf{e}'') \Bigg] \\
&= \frac{1}{2^{n-k}} \Big(\mathsf{H}(C_{+x}, \mathbf{e}_{|X \setminus \{x\}}, k) + \mathsf{H}(C_{-x}, \mathbf{e}_{|X \setminus \{x\}}, k) \Big).
\end{aligned}$$

The last equality is obtained by using the definition of $\mathsf{H}(\cdot, \cdot, \cdot)$. A similar analysis allows us to conclude that:

$$\alpha = \begin{cases} \frac{1}{2^{n-(k+1)}} \mathsf{H}(C_{+x}, \mathbf{e}_{|X \setminus \{x\}}, k), & \text{if } C(\mathbf{e}) = 1 \\ \frac{1}{2^{n-(k+1)}} \mathsf{H}(C_{-x}, \mathbf{e}_{|X \setminus \{x\}}, k), & \text{if } C(\mathbf{e}) = 0 \end{cases}.$$

Hence, if we can compute in polynomial time $\mathsf{H}(\cdot, \cdot, \cdot)$ for deterministic and decomposable Boolean circuits, then we can compute α and β in polynomial time (because C_{+x} and C_{-x} can be computed in linear time from C, and they are deterministic and decomposable as well). Thus, we can compute $\mathsf{Diff}_k(C, \mathbf{e}, x)$ in polynomial time for each $k \in \{0, \ldots, n-1\}$ and, hence, $\mathsf{SHAP}(C, \mathbf{e}, x)$ as well. In conclusion, $\mathsf{SHAP}(C, \mathbf{e}, x)$ can be computed in polynomial time if there is a polynomial-time algorithm to compute $\mathsf{H}(\cdot, \cdot, \cdot)$ for deterministic and decomposable Boolean circuits.

Reducing from $\mathsf{H}(\cdot, \cdot, \cdot)$ to $\#\mathsf{SAT}(\cdot, \cdot, \cdot)$. We now show that computing $\mathsf{H}(\cdot, \cdot, \cdot)$ can be reduced in polynomial time to computing $\#\mathsf{SAT}(\cdot, \cdot, \cdot)$. Given as input a deterministic and decomposable circuit C over a set of variables X, an entity $\mathbf{e} \in \mathsf{ent}(X)$, and an integer $k \leq |X|$, recall the definition of $\mathsf{H}(C, \mathbf{e}, x)$ in (2). Then consider an entity $\mathbf{e}'' \in \mathsf{ent}(X)$ and reason about how many times $\mathbf{e}''$ will occur as a summand in the expression (2). First of all, it is clear that if $|\mathsf{sim}(\mathbf{e}, \mathbf{e}'')| < k$, then $\mathbf{e}''$ will not appear in the sum; this is because if $\mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S)$ for some $S \subseteq X$ such that $|S| = k$, then $S \subseteq \mathsf{sim}(\mathbf{e}, \mathbf{e}')$ and, thus, $k \leq |\mathsf{sim}(\mathbf{e}, \mathbf{e}')|$. Now, how many times does an entity $\mathbf{e}'' \in \mathsf{ent}(X)$ such that $|\mathsf{sim}(\mathbf{e}, \mathbf{e}'')| \geq k$ occur as a summand in the expression? The answer is simple: once per $S \subseteq \mathsf{sim}(\mathbf{e}, \mathbf{e}'')$ of size k. Since there are $\binom{|\mathsf{sim}(\mathbf{e}, \mathbf{e}'')|}{k}$ such sets S, we obtain that $\mathsf{H}(C, \mathbf{e}, k)$ is equal to

$$\begin{aligned}
&\sum_{\substack{\mathbf{e}'' \in \mathsf{ent}(X) \\ |\mathsf{sim}(\mathbf{e}, \mathbf{e}'')| \geq k}} \binom{|\mathsf{sim}(\mathbf{e}, \mathbf{e}'')|}{k} \cdot C(\mathbf{e}'') \\
&= \sum_{\substack{\mathbf{e}'' \in \mathsf{SAT}(C) \\ |\mathsf{sim}(\mathbf{e}, \mathbf{e}'')| \geq k}} \binom{|\mathsf{sim}(\mathbf{e}, \mathbf{e}'')|}{k} \\
&= \sum_{\ell=k}^{n} \sum_{\substack{\mathbf{e}'' \in \mathsf{SAT}(C) \\ |\mathsf{sim}(\mathbf{e}, \mathbf{e}'')| = \ell}} \binom{|\mathsf{sim}(\mathbf{e}, \mathbf{e}'')|}{k} \\
&= \sum_{\ell=k}^{n} \binom{\ell}{k} \sum_{\substack{\mathbf{e}'' \in \mathsf{SAT}(C) \\ |\mathsf{sim}(\mathbf{e}, \mathbf{e}'')| = \ell}} 1 = \sum_{\ell=k}^{n} \binom{\ell}{k} \cdot \#\mathsf{SAT}(C, \mathbf{e}, \ell),
\end{aligned}$$

with the last equality being obtained by using the definition of $\#\mathsf{SAT}(\cdot, \cdot, \cdot)$. This concludes the reduction of this section and, hence, the first part of the proof.

3.2 Computing $\#\mathsf{SAT}(\cdot, \cdot, \cdot)$ in polynomial time

We now take care of the second part of the proof of Theorem 3.1, i.e., proving that computing $\#\mathsf{SAT}(\cdot, \cdot, \cdot)$ for deterministic and decomposable Boolean circuits can be done in polynomial time. To do this, given a deterministic and decomposable Boolean circuit C, we first perform two preprocessing steps on C, which will simplify the proof.

- **Rewriting to fan-in at most 2.** First, we modify the circuit C so that the fan-in of every $\vee$- and $\wedge$-gate is at most 2. This can simply be done in linear time by rewriting every $\wedge$-gate (resp., and $\vee$-gate) of fan-in $m > 2$ with

a chain of $m-1$ $\wedge$-gates (resp., $\vee$-gates) of fan-in 2. It is clear that the resulting Boolean circuit is deterministic and decomposable. Hence, from now on we assume that the fan-in of every $\vee$- and $\wedge$-gate of C is at most 2.

- **Smoothing the circuit.** A deterministic and decomposable circuit C is *smooth* (Darwiche 2001; Shih et al. 2019) if for every $\vee$-gate g and input gates g_1, g_2 of g, we have that $\mathsf{var}(g_1) = \mathsf{var}(g_2)$, and we call such an $\vee$-gate smooth. A standard construction allows to transform in polynomial time a deterministic and decomposable Boolean circuit C into an equivalent smooth deterministic and decomposable Boolean circuit, and where each gate has fan-in at most 2. Thus, from now on we also assume that C is smooth. We illustrate how the construction works in Example 3.4 . Full details can be found in the supplementary material (namely, in Section E.2, paragraph *Smoothing the circuit*).

We have all the ingredients to prove that $\#\mathsf{SAT}(\cdot,\cdot,\cdot)$ can be computed in polynomial time. Let C be a deterministic and decomposable Boolean circuit over a set of variables X, $\mathbf{e} \in \mathsf{ent}(X)$, ℓ a natural number such that $\ell \leq |X|$ and $n = |X|$. For a gate g of C, let R_g be the Boolean circuit over $\mathsf{var}(g)$ that is defined by considering the subgraph of C induced by the set of gates g' in C for which there exists a path from g' to g in C. Notice that R_g is a deterministic and decomposable Boolean circuit with output gate g. Moreover, for a gate g and natural number $k \leq |\mathsf{var}(g)|$, define $\alpha_g^k := \#\mathsf{SAT}(R_g, \mathbf{e}_{|\mathsf{var}(g)}, k)$, which we recall is the number of entities $\mathbf{e}' \in \mathsf{ent}(\mathsf{var}(g))$ such that $\mathbf{e}'$ satisfies R_g and $|\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g)}, \mathbf{e}')| = k$. We will show how to compute all the values α_g^k for every gate g of C and $k \in \{0,\ldots,|\mathsf{var}(g)|\}$ in polynomial time. This will conclude the proof since, for the output gate g_{out} of C, we have that $\alpha_{g_{\mathsf{out}}}^{\ell} = \#\mathsf{SAT}(C, \mathbf{e}, \ell)$. Next we explain how to compute these values in a bottom-up manner.

Variable gate. g is a variable gate with label $y \in X$, so that $\mathsf{var}(g) = \{y\}$. Then $\alpha_g^0 = 1 - \mathbf{e}(y)$ and $\alpha_g^1 = \mathbf{e}(y)$.

Constant gate. g is a constant gate with label $a \in \{0,1\}$. Then $\mathsf{var}(g) = \emptyset$ and $\alpha_g^0 = a$.[2]

$\neg$-gate. g is a $\neg$-gate with input gate g'. Then $\mathsf{var}(g) = \mathsf{var}(g')$, and the values $\alpha_{g'}^k$ for $k \in \{0,\ldots,|\mathsf{var}(g)|\}$ have already been computed. Fix $k \in \{0,\ldots,|\mathsf{var}(g)|\}$. Since $\binom{|\mathsf{var}(g)|}{k}$ is equal to the number of entities $\mathbf{e}' \in \mathsf{ent}(\mathsf{var}(g))$ such that $|\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g)}, \mathbf{e}')| = k$, we have that

$$\alpha_g^k = \binom{|\mathsf{var}(g)|}{k} - \alpha_{g'}^k.$$

Therefore, given that $\binom{|\mathsf{var}(g)|}{k}$ can be computed in polynomial time since $k \leq |\mathsf{var}(g)| \leq n = |X|$, we have an efficient way to compute α_g^k.

$\vee$-gate. g is an $\vee$-gate. By assumption, g is deterministic, smooth and has fan-in at most 2. If g has only one input g', then clearly $\mathsf{var}(g) = \mathsf{var}(g')$ and $\alpha_g^k = \alpha_{g'}^k$ for every $k \in \{0,\ldots,|\mathsf{var}(g)|\}$. Thus, assume that g has exactly two input gates g_1 and g_2, and recall that $\mathsf{var}(g_1) = \mathsf{var}(g_2) = \mathsf{var}(g)$, because g is smooth. Also, recall that $\alpha_{g_1}^k$ and $\alpha_{g_2}^k$, for each $k \in \{0,\ldots,|\mathsf{var}(g)|\}$, have already been computed. Fix $k \in \{0,\ldots,|\mathsf{var}(g)|\}$. Given that g is deterministic and smooth, we have that $\mathsf{SAT}(R_g) = \mathsf{SAT}(R_{g_1}) \cup \mathsf{SAT}(R_{g_2})$, where $\mathsf{SAT}(R_{g_1}) \cap \mathsf{SAT}(R_{g_2}) = \emptyset$. By intersecting these three sets with the set $\{\mathbf{e}' \in \mathsf{var}(g) \mid |\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g)}, \mathbf{e}')| = k\}$, we obtain that $\mathsf{SAT}(R_g, \mathbf{e}_{|\mathsf{var}(g)}, k) = \mathsf{SAT}(R_{g_1}, \mathbf{e}_{|\mathsf{var}(g)}, k) \cup \mathsf{SAT}(R_{g_2}, \mathbf{e}_{|\mathsf{var}(g)}, k)$, where $\mathsf{SAT}(R_{g_1}, \mathbf{e}_{|\mathsf{var}(g)}, k) \cap \mathsf{SAT}(R_{g_2}, \mathbf{e}_{|\mathsf{var}(g)}, k) = \emptyset$. Hence:

$$\#\mathsf{SAT}(R_g, \mathbf{e}_{|\mathsf{var}(g)}, k) = \\ \#\mathsf{SAT}(R_{g_1}, \mathbf{e}_{|\mathsf{var}(g)}, k) + \#\mathsf{SAT}(R_{g_2}, \mathbf{e}_{|\mathsf{var}(g)}, k),$$

or, in other words, we have that $\alpha_g^k = \alpha_{g_1}^k + \alpha_{g_2}^k$. Hence, we have an efficient way to compute α_g^k.

$\wedge$-gate. g is an $\wedge$-gate. By assumption, recall that g is decomposable and has fan-in at most 2. If g has only one input g', then clearly $\mathsf{var}(g) = \mathsf{var}(g')$ and $\alpha_g^k = \alpha_{g'}^k$ for every $k \in \{0,\ldots,|\mathsf{var}(g)|\}$. Thus, assume that g has exactly two input gates g_1 and g_2. Recall then that the values $\alpha_{g_1}^i$ and $\alpha_{g_2}^j$, for each $i \in \{0,\ldots,|\mathsf{var}(g_1)|\}$ and $j \in \{0,\ldots,|\mathsf{var}(g_2)|\}$, have already been computed. Fix $k \in \{0,\ldots,|\mathsf{var}(g)|\}$. Given that g is a decomposable $\wedge$-gate, in this case it is possible to prove that:

$$\alpha_g^k = \sum_{\substack{i \in \{0,\ldots,|\mathsf{var}(g_1)|\} \\ j \in \{0,\ldots,|\mathsf{var}(g_2)|\} \\ i+j=k}} \alpha_{g_1}^i \cdot \alpha_{g_2}^j. \tag{3}$$

The complete proof of this property can be found in Appendix B. Therefore, as in the previous cases, we conclude that there is an efficient way to compute α_g^k.

This concludes the proof that $\#\mathsf{SAT}(\cdot,\cdot,\cdot)$ can be computed in polynomial time for deterministic and decomposable Boolean circuits and, hence, the proof of Theorem 3.1.

Example 3.4. We illustrate how the algorithm for computing the SHAP-score operates on the Boolean circuit C given in Example 1.1. Recall that C is defined over $X = \{\mathbf{fg}, \mathbf{dtr}, \mathbf{nf}, \mathbf{na}\}$, and assume we want to compute $\mathsf{SHAP}(C, \mathbf{e}, \mathbf{nf})$ for the entity $\mathbf{e}$ with $\mathbf{e}(x) = 1$ for each $x \in X$. By the polynomial time reductions shown in Section 3.1, to compute $\mathsf{SHAP}(C, \mathbf{e}, \mathbf{nf})$ it suffices to compute $\mathsf{H}(C_{-\mathbf{nf}}, \mathbf{e}_{|X\setminus\{\mathbf{nf}\}}, \ell)$ and $\mathsf{H}(C_{+\mathbf{nf}}, \mathbf{e}_{|X\setminus\{\mathbf{nf}\}}, \ell)$ for each $\ell \in \{0,\ldots,3\}$, which in turn reduces to the computation of $\#\mathsf{SAT}(C_{-\mathbf{nf}}, \mathbf{e}_{|X\setminus\{\mathbf{nf}\}}, \ell)$ and $\#\mathsf{SAT}(C_{+\mathbf{nf}}, \mathbf{e}_{|X\setminus\{\mathbf{nf}\}}, \ell)$ for each $\ell \in \{0,\ldots,3\}$. In what follows, we show how to compute the values $\#\mathsf{SAT}(C_{+\mathbf{nf}}, \mathbf{e}_{|X\setminus\{\mathbf{nf}\}}, \ell)$.

For the sake of presentation, let $D := C_{+\mathbf{nf}}$ and $\mathbf{e}^\star = \mathbf{e}_{|X\setminus\{\mathbf{nf}\}}$, so that we need to compute $\#\mathsf{SAT}(D, \mathbf{e}^\star, \ell)$ for

[2] We recall the mathematical convention that there is a unique function with the empty domain and, hence, a unique entity over $\emptyset$.

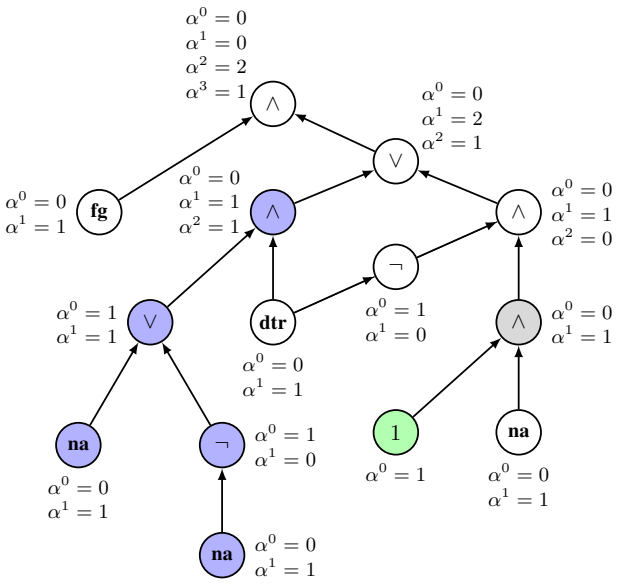


Figure 2: Execution of our algorithm to compute $\#\mathsf{SAT}(\cdot, \cdot, \cdot)$ over the Boolean circuit $C_{+\mathbf{nf}}$ from Example 3.4.

each $\ell \in \{0, \ldots, 3\}$. Notice that the values to be computed are $\#\mathsf{SAT}(D, \mathbf{e}^\star, 0) = 0$, $\#\mathsf{SAT}(D, \mathbf{e}^\star, 1) = 0$, $\#\mathsf{SAT}(D, \mathbf{e}^\star, 2) = 2$ and $\#\mathsf{SAT}(D, \mathbf{e}^\star, 3) = 1$. To compute $\#\mathsf{SAT}(D, \mathbf{e}^\star, \ell)$, we first need to replace feature **nf** by constant 1 in C to generate $D = C_{+\mathbf{nf}}$, and then we need to transform D into a Boolean circuit that is smooth and where each gate has fan-in at most 2. The result of this process is shown in Figure 2, where the green node is added when replacing feature **nf** by constant 1, the gray node is added to satisfy the restriction that each gate has fan-in at most 2, and the blue nodes are added to have a smooth Boolean circuit.

The algorithm to compute $\#\mathsf{SAT}(D, \mathbf{e}^\star, \ell)$ runs in a bottom-up fashion on the Boolean circuit, computing for each gate g the values α_g^k for $k \in \{0, \ldots, |\mathsf{var}(g)|\}$. We show these values next to each node in Figure 2, but omitting gate subscripts. For instance, for a variable gate g with label **na**, we have that $\mathsf{var}(g) = \{\mathbf{na}\}$, $\alpha_g^0 = 0$ and $\alpha_g^1 = 1$, given that $\mathbf{e}^\star_{|\mathsf{var}(g)}(\mathbf{na}) = \mathbf{e}^\star(\mathbf{na}) = 1$. Notice that for the output gate g_{out} of the Boolean circuit, which is its top gate, we have that $\#\mathsf{SAT}(D, \mathbf{e}^\star, \ell) = \alpha_{g_{\mathsf{out}}}^\ell$ for each $\ell \in \{0, \ldots, 3\}$, which were the values to be computed. □

4 Limits on the Tractable Computation of the SHAP-Score

We have shown that the SHAP-score can be computed in polynomial time for deterministic and decomposable circuits. A natural question, then, is whether both determinism and decomposability are necessary for this positive result to hold. In this section we show this to be case, at least under standard complexity assumptions. Recall that #P consists of the class of functions that can be defined by counting the number of accepting paths of a non-deterministic Turing machine that works in polynomial time. The notion of hardness for the class #P is defined in terms of polynomial time *Turing reductions*. Under widely-held complexity assumptions, #P-hard problems cannot be solved in polynomial time (Arora and Barak 2009). We can then prove the following:

Theorem 4.1. *The following problems are #P-hard.*

1. *Given as input a decomposable Boolean circuit C over a set of features X, an entity $\mathbf{e} : X \to \{0, 1\}$, and a feature $x \in X$, compute the value $\mathsf{SHAP}(C, \mathbf{e}, x)$.*
2. *Given as input a deterministic Boolean circuit C over a set of features X, an entity $\mathbf{e} : X \to \{0, 1\}$, and a feature $x \in X$, compute the value $\mathsf{SHAP}(C, \mathbf{e}, x)$.*

To prove Theorem 4.1, we start by showing that there is a polynomial-time reduction from the problem of computing the number of entities that satisfy M, for M an arbitrary Boolean classifier, to the problem of computing the SHAP-score over M. This holds under the mild condition that $M(\mathbf{e})$ can be computed in polynomial time for an input entity $\mathbf{e}$, which is satisfied for all the Boolean circuits and binary decision diagrams classes considered in this paper. The proof of this result follows from well-known properties of Shapley values. (A closely related result can be found as Theorem 5.1 in (Bertossi et al. 2020)).

Lemma 4.2. *Let M be a Boolean classifier over a set of features X. Then for every $\mathbf{e} \in \mathsf{ent}(X)$ we have:*

$$\#\mathsf{SAT}(M) \;=\; 2^{|X|}\Big(M(\mathbf{e}) - \sum_{x \in X} \mathsf{SHAP}(M, \mathbf{e}, x)\Big).$$

We prove Lemma 4.2 in the supplementary material. Item (1) in Theorem 4.1 follows then by the following two facts: (a) Counting the number of entities that satisfy a DNF formula is a #P-hard problem (Provan and Ball 1983), and (b) DNF formulae are particular kinds of decomposable Boolean circuits. Analogously, item (2) in Theorem 4.1 can be obtained from the following two facts: (a) Counting the number of entities that satisfy a 3-CNF formula is a #P-hard problem, and (b) from every 3-CNF formula ψ, we can build in polynomial time an equivalent deterministic Boolean circuit C_ψ. Details can be found in the supplementary material.

5 Tractability for the Product Distribution

In Section 2, we introduce the uniform distribution, and used it so far as a basis for the SHAP-score. Another probability space that is often considered on $\mathsf{ent}(X)$ is the *product distribution*, defined as follows. Let $\mathrm{p} : X \to [0, 1]$ be a function that associates to every feature $x \in X$ a value $\mathrm{p}(x) \in [0, 1]$; intuitively, the probability that x takes value 1. Then, the *product distribution generated by* p is the probability distribution Π_{p} over $\mathsf{ent}(X)$ such that, for every $\mathbf{e} \in \mathsf{ent}(X)$,

$$\Pi_{\mathrm{p}}(\mathbf{e}) \;:=\; \Big(\prod_{\substack{x \in X \\ \mathbf{e}(x)=1}} \mathrm{p}(x)\Big) \cdot \Big(\prod_{\substack{x \in X \\ \mathbf{e}(x)=0}} (1 - \mathrm{p}(x))\Big).$$

That is, the product distribution that is determined by prespecified marginal distributions, and that makes the features take values independently from each other. Observe

the effect of the probability distribution on the SHAP-score: intuitively, the higher the probability of an entity, the more impact this entity will have on the computation. This can be used, for instance, to avoid bias in the explanations (Lundberg and Lee 2017; Bertossi et al. 2020).

Notice that the uniform space is a special case of product space, with Π_{p} invoking $\mathrm{p}(x) := 1/2$ for every $x \in X$. Thus, our hardness results from Theorem 4.1 also hold in the case where the probabilities $\mathrm{p}(x)$ are given as input. What is more interesting is the fact that our tractability result from Theorem 3.1 extends to product distributions. Formally:

Theorem 5.1. *The following problem can be solved in polynomial time. Given as input a deterministic and decomposable circuit C over a set of features X, rational probability values $\mathrm{p}(x)$ for every feature $x \in X$, an entity $\mathbf{e} : X \to \{0,1\}$, and a feature $x \in X$, compute the value $\mathsf{SHAP}(C, \mathbf{e}, x)$ under the probability distribution Π_{p}.*

The proof of Theorem 5.1 is more involved than that of Theorem 3.1, and is provided in the supplementary material. In particular, the main difficulty is that $\phi(M, \mathbf{e}, S)$ is no longer equal to $\sum_{\mathbf{e}' \in \mathsf{cw}(\mathbf{e},S)} \frac{1}{2^{|X \setminus S|}} M(\mathbf{e}')$ (as it was the case for the uniform space), because the entities do not all have the same probability. This prevents us from being able to reduce to the computation of $\#\mathsf{SAT}(\cdot,\cdot,\cdot)$. Instead, we use a different definition of $\mathsf{H}(\cdot,\cdot,\cdot)$, and prove that it can directly be computed in a bottom-up fashion on the circuits. We show in Algorithm 1 our algorithm to compute the SHAP score for deterministic and decomposable Boolean circuits under product distributions, which can be extracted from the proof in the supplementary material. Notice that by using the techniques presented in Section 3, the first step of the algorithm transforms the input circuit C into an equivalent smooth circuit D where each $\vee$-gate and $\wedge$-gate has fan-in 2.

6 Extensions and Future Work

We leave open many interesting directions for future work. For instance, we intend to extend our algorithm for efficiently computing the SHAP-score to work with non-Boolean classifiers, and to consider more general probability distributions that could better capture possible correlations and dependencies between features. We also aim to provide an experimental comparison of our algorithm, but specialized for decision trees, with the one provided in (Lundberg et al. 2020, Alg. 2). Last, we intend to test our algorithm on real-world scenarios.

Acknowledgments

Pablo Barceló was funded by Fondecyt grant 1200967.

References

Amarilli, A.; Capelli, F.; Monet, M.; and Senellart, P. 2020. Connecting knowledge compilation classes and width parameters. *Theory Comput. Syst.* 64(5): 861–914.

Arora, S.; and Barak, B. 2009. *Computational Complexity - A Modern Approach*. Cambridge University Press.

Bertossi, L.; Li, J.; Schleich, M.; Suciu, D.; and Vagena, Z. 2020. Causality-based explanation of classification

Algorithm 1: SHAP-scores for deterministic and decomposable Boolean circuits

Input : Deterministic and decomposable Boolean circuit C over features X with ouput gate g_{out}, rational probability values $\mathrm{p}(x)$ for all $x \in X$, entity $\mathbf{e} \in \mathsf{ent}(X)$, and feature $x \in X$.

Output: The value $\mathsf{SHAP}(C, \mathbf{e}, x)$ under the probability distribution Π_{p}.

Transform C into an equivalent smooth circuit D where each $\vee$-gate and $\wedge$-gate has fan-in 2;

Compute the set $\mathsf{var}(g)$ for every gate g in D;

Compute values γ_g^ℓ and δ_g^ℓ for every gate g in D and $\ell \in \{0, \ldots, |\mathsf{var}(g) \setminus \{x\}|\}$ by bottom-up induction on D as follows:

- **if** *g is a constant gate with label $a \in \{0,1\}$* **then**
  - $\gamma_g^0,\ \delta_g^0 \leftarrow a$;
- **else if** *g is a variable gate with $\mathsf{var}(g) = \{x\}$* **then**
  - $\gamma_g^0 \leftarrow 1$;
  - $\delta_g^0 \leftarrow 0$;
- **else if** *g is a variable gate with $\mathsf{var}(g) = \{y\}$ and $y \neq x$* **then**
  - $\gamma_g^0,\ \delta_g^0 \leftarrow \mathrm{p}(y)$;
  - $\gamma_g^1,\ \delta_g^1 \leftarrow \mathbf{e}(y)$;
- **else if** *g is a $\neg$-gate with input gate g'* **then**
  - **for** $\ell \in \{0, \ldots, |\mathsf{var}(g) \setminus \{x\}|\}$ **do**
    - $\gamma_g^\ell \leftarrow \binom{|\mathsf{var}(g)\setminus\{x\}|}{\ell} - \gamma_{g'}^\ell$;
    - $\delta_g^\ell \leftarrow \binom{|\mathsf{var}(g)\setminus\{x\}|}{\ell} - \delta_{g'}^\ell$;
  - **end**
- **else if** *g is an $\vee$-gate with input gates g_1, g_2* **then**
  - **for** $\ell \in \{0, \ldots, |\mathsf{var}(g) \setminus \{x\}|\}$ **do**
    - $\gamma_g^\ell \leftarrow \gamma_{g_1}^\ell + \gamma_{g_2}^\ell$;
    - $\delta_g^\ell \leftarrow \delta_{g_1}^\ell + \delta_{g_2}^\ell$;
  - **end**
- **else if** *g is an $\wedge$-gate with input gates g_1, g_2* **then**
  - **for** $\ell \in \{0, \ldots, |\mathsf{var}(g) \setminus \{x\}|\}$ **do**
    - $\gamma_g^\ell \leftarrow \sum_{\substack{\ell_1 \in \{0,\ldots,|\mathsf{var}(g_1)\setminus\{x\}|\} \\ \ell_2 \in \{0,\ldots,|\mathsf{var}(g_2)\setminus\{x\}|\} \\ \ell_1+\ell_2=\ell}} \gamma_{g_1}^{\ell_1} \cdot \gamma_{g_2}^{\ell_2}$;
    - $\delta_g^\ell \leftarrow \sum_{\substack{\ell_1 \in \{0,\ldots,|\mathsf{var}(g_1)\setminus\{x\}|\} \\ \ell_2 \in \{0,\ldots,|\mathsf{var}(g_2)\setminus\{x\}|\} \\ \ell_1+\ell_2=\ell}} \delta_{g_1}^{\ell_1} \cdot \delta_{g_2}^{\ell_2}$;
  - **end**

end

return

$$\sum_{k=0}^{|X|-1} \frac{k!\,(|X|-k-1)!}{|X|!} \cdot \left[(\mathbf{e}(x) - \mathrm{p}(x))(\gamma_{g_{\mathsf{out}}}^k - \delta_{g_{\mathsf{out}}}^k)\right];$$

outcomes. In *Proceedings of the Fourth Workshop on Data Management for End-To-End Machine Learning, DEEM@SIGMOD 2020*, 6:1–6:10.

Cesari, G.; Algaba, E.; Moretti, S.; and Nepomuceno, J. A. 2018. An application of the Shapley value to the analysis of co-expression networks. *Applied network science* 3(1): 35.

Choi, A.; Shih, A.; Goyanka, A.; and Darwiche, A. 2020. On Symbolically Encoding the Behavior of Random Forests. *CoRR* abs/2007.01493.

Darwiche, A. 2001. On the tractable counting of theory models and its application to truth maintenance and belief revision. *J. Applied Non-Classical Logics* 11(1-2).

Darwiche, A.; and Hirth, A. 2020. On the reasons behind decisions. *arXiv preprint arXiv:2002.09284* .

Darwiche, A.; and Marquis, P. 2002. A Knowledge Compilation Map. *J. Artif. Intell. Res.* 17: 229–264.

Deng, X.; and Papadimitriou, C. H. 1994. On the complexity of cooperative solution concepts. *Mathematics of operations research* 19(2): 257–266.

Faigle, U.; and Kern, W. 1992. The Shapley value for cooperative games under precedence constraints. *International Journal of Game Theory* 21(3): 249–266.

Hunter, A.; and Konieczny, S. 2010. On the measure of conflicts: Shapley inconsistency values. *Artificial Intelligence* 174(14): 1007–1026.

Livshits, E.; Bertossi, L. E.; Kimelfeld, B.; and Sebag, M. 2020. The Shapley value of tuples in query answering. In *23rd International Conference on Database Theory, ICDT 2020, March 30-April 2, 2020, Copenhagen, Denmark*, volume 155, 20:1–20:19.

Lundberg, S. M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J. M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; and Lee, S.-I. 2020. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence* 2(1): 2522–5839.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, 4765–4774.

Michalak, T. P.; Aadithya, K. V.; Szczepanski, P. L.; Ravindran, B.; and Jennings, N. R. 2013. Efficient computation of the Shapley value for game-theoretic network centrality. *Journal of Artificial Intelligence Research* 46: 607–650.

Molnar, C. 2020. *Interpretable machine learning: A guide for making black box models explainable*. https://christophm.github.io/interpretable-ml-book.

Peharz, R.; Lang, S.; Vergari, A.; Stelzner, K.; Molina, A.; Trapp, M.; Van den Broeck, G.; Kersting, K.; and Ghahramani, Z. 2020. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. *arXiv preprint arXiv:2004.06231* .

Provan, J. S.; and Ball, M. O. 1983. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing* 12(4): 777–788.

Roth, A. E. 1988. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press.

Shapley, L. S. 1953. A value for n-person games. *Contributions to the Theory of Games* 2(28): 307–317.

Shi, W.; Shih, A.; Darwiche, A.; and Choi, A. 2020. On tractable representations of binary neural networks. *arXiv preprint arXiv:2004.02082* .

Shih, A.; Choi, A.; and Darwiche, A. 2018a. A symbolic approach to explaining Bayesian network classifiers. In *Proceedings IJCAI*, 5103–5111.

Shih, A.; Choi, A.; and Darwiche, A. 2018b. Formal verification of Bayesian network classifiers. In *International Conference on Probabilistic Graphical Models*, 427–438.

Shih, A.; Darwiche, A.; and Choi, A. 2019. Verifying binarized neural networks by Angluin-style learning. In *International Conference on Theory and Applications of Satisfiability Testing*, 354–370. Springer.

Shih, A.; Van den Broeck, G.; Beame, P.; and Amarilli, A. 2019. Smoothing structured decomposable circuits. In *Advances in Neural Information Processing Systems*, 11416–11426.

Van den Broeck, G.; Lykov, A.; Schleich, M.; and Suciu, D. 2020. On the Tractability of SHAP Explanations. *arXiv preprint arXiv:2009.08634* .

Supplementary Material: Technical Appendix

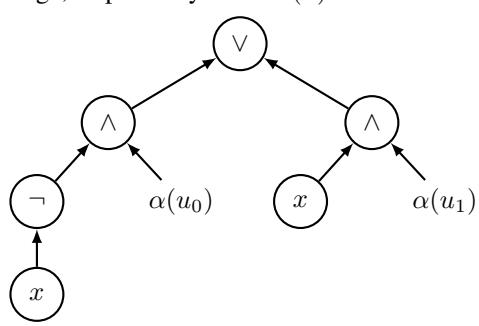
A Encoding Binary Decision Trees and FBDDs into Deterministic and Decomposable Boolean Circuits

In this appendix, we explain why binary decision trees and free binary decision diagrams (FBDDs) are special kinds of deterministic and decomposable Boolean circuits. First we need to define these formalisms.

Binary Decision Diagrams. A *binary decision diagram* (BDD) over a set of variables X is a rooted directed acyclic graph D such that: (i) each internal node is labeled with a variable from X, and has exactly two outgoing edges: one labeled 0, the other one labeled 1; and (ii) each leaf is labeled either 0 or 1. Such a BDD represents a Boolean classifier in the following way. Let $\mathbf{e}$ be an entity over X, and let $\pi_{\mathbf{e}} = u_1, \ldots, u_m$ be the unique path in D satisfying the following conditions: (a) u_1 is the root of D; (b) u_m is a leaf of D; and (c) for every $i \in \{1, \ldots, m-1\}$, if the label of u_i is $x \in X$, then the label of the edge (u_i, u_{i+1}) is equal to $\mathbf{e}(x)$. Then the *value of* $\mathbf{e}$ *in* D, denoted by $D(\mathbf{e})$, is defined as the label of the leaf u_m. Moreover, a binary decision diagram D is *free* (FBDD) if for every path from the root to a leaf, no two nodes on that path have the same label, and *a binary decision tree* is an FBDD whose underlying graph is a tree.

As we show next, FBDDs can be encoded in linear time as deterministic and decomposable Boolean circuits.

Encoding FBDDs into deterministic and decomposable Boolean circuits (Folklore). Given an FBDD D over a set of variables X, we explain how D can be encoded as a deterministic and decomposable Boolean circuit C over X. Notice that the technique used in this example also apply to binary decision trees, as they are a particular case of FBDDs. The construction of C is done by traversing the structure of D in a bottom-up manner. In particular, for every node u of D, we construct a deterministic and decomposable circuit $\alpha(u)$ that is equivalent to the FBDD represented by the subgraph of D rooted at u. More precisely, for a leaf u of D that is labeled with $\ell \in \{0, 1\}$, we define $\alpha(u)$ to be the Boolean circuit consisting of only one constant gate with label ℓ. For an internal node u of D labeled with variable $x \in X$, let u_0 and u_1 be the nodes that we reach from u by following the 0- and 1-labeled edge, respectively. Then $\alpha(u)$ is the Boolean circuit depicted in the following figure:



It is clear that the circuit that we obtain is equivalent to the input FBDD. We now argue that this circuit is deterministic and decomposable. For the $\vee$-gate shown in the figure, if an entity $\mathbf{e}$ is accepted by the Boolean circuit in its left-hand size, then $\mathbf{e}(x) = 0$, while if an entity $\mathbf{e}$ is accepted by the Boolean circuit in its right-hand size, then $\mathbf{e}(x) = 1$. Hence, we have that this $\vee$-gate is deterministic, from which we conclude that $\alpha(u)$ is deterministic, as $\alpha(u_0)$ and $\alpha(u_1)$ are also deterministic by construction. Moreover, the $\wedge$-gates shown in the figure are decomposable as variable x is mentioned neither in $\alpha(u_0)$ nor in $\alpha(u_1)$: this is because D is a *free* BDD. Thus, we conclude that $\alpha(u)$ is decomposable, as $\alpha(u_0)$ and $\alpha(u_1)$ are decomposable by construction. Finally, if u_{root} is the root of D, then by construction we have that $\alpha(u_{\mathsf{root}})$ is a deterministic and decomposable Boolean circuit equivalent to D. Note that this encoding can trivially be done in linear time. Thus, we often say, by abuse of terminology, that "FBDDs (or binary decision trees) are restricted kinds of deterministic and decomposable circuits".

B Proof of Theorem 3.1

To complete the proof of Theorem 3.1, we need to prove equation (3). Recall that in this case, we have that g is an $\wedge$-gate, which is decomposable and has fan-in at most 2. Moreover, we assume that g has exactly two input gates g_1 and g_2, and we fix $k \in \{0, \ldots, |\mathsf{var}(g)|\}$.

To prove equation (3), we need the following notation. For two disjoint sets of variables X_1, X_2 and entities $\mathbf{e}_1 \in \mathsf{ent}(X_1)$, $\mathbf{e}_2 \in \mathsf{ent}(X_2)$, we denote by $\mathbf{e}_1 \cup \mathbf{e}_2$ the entity over $X_1 \cup X_2$ that coincides with $\mathbf{e}_1$ over X_1 and with $\mathbf{e}_2$ over X_2 (that is, $\mathbf{e}_1 \cup \mathbf{e}_2 \in \mathsf{ent}(X_1 \cup X_2)$, $(\mathbf{e}_1 \cup \mathbf{e}_2)(x_1) = \mathbf{e}_1(x_1)$ for every $x_1 \in X_1$, and $(\mathbf{e}_1 \cup \mathbf{e}_2)(x_2) = \mathbf{e}_2(x_2)$ for every $x_2 \in X_2$). Moreover, for two sets $S_1 \subseteq \mathsf{ent}(X_1)$, $S_2 \subseteq \mathsf{ent}(X_2)$, we denote by $S_1 \otimes S_2$ the set of entities over $X_1 \cup X_2$ defined as

$$S_1 \otimes S_2 \;:=\; \{\mathbf{e}_1 \cup \mathbf{e}_2 \mid \mathbf{e}_1 \in S_1 \text{ and } \mathbf{e}_2 \in S_2\}.$$

Given that g is a decomposable $\wedge$-gate, we have that:

$$\mathsf{SAT}(R_g) \;=\; \mathsf{SAT}(R_{g_1}) \otimes \mathsf{SAT}(R_{g_2}).$$

Moreover, we have that $\mathsf{SAT}(R_g, \mathbf{e}_{|\mathsf{var}(g)}, k) = \mathsf{SAT}(R_g) \cap \{\mathbf{e}' \in \mathsf{var}(g) \mid |\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g)}, \mathbf{e}')| = k\}$ and

$$\begin{aligned}
&\big(\mathsf{SAT}(R_{g_1}) \otimes \mathsf{SAT}(R_{g_2})\big) \cap \{\mathbf{e}' \in \mathsf{var}(g) \mid |\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g)}, \mathbf{e}')| = k\} \\
&\quad = \{\mathbf{e}_1 \cup \mathbf{e}_2 \mid \mathbf{e}_1 \in \mathsf{SAT}(R_{g_1}) \text{ and } \mathbf{e}_2 \in \mathsf{SAT}(R_{g_2})\} \cap \{\mathbf{e}' \in \mathsf{var}(g) \mid |\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g)}, \mathbf{e}')| = k\} \\
&\quad = \{\mathbf{e}_1 \cup \mathbf{e}_2 \mid \mathbf{e}_1 \in \mathsf{SAT}(R_{g_1}), \mathbf{e}_2 \in \mathsf{SAT}(R_{g_2}), \text{ and } |\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g)}, \mathbf{e}_1 \cup \mathbf{e}_2)| = k\} \\
&\quad = \{\mathbf{e}_1 \cup \mathbf{e}_2 \mid \mathbf{e}_1 \in \mathsf{SAT}(R_{g_1}), \mathbf{e}_2 \in \mathsf{SAT}(R_{g_2}), \text{ and there exist } i \in \{0, \ldots, |\mathsf{var}(g_1)|\}, \\
&\qquad\quad j \in \{0, \ldots, |\mathsf{var}(g_2)|\} \text{ such that } |\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g_1)}, \mathbf{e}_1)| = i, |\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g_2)}, \mathbf{e}_2)| = j, \text{ and } i + j = k\} \\
&\quad = \bigcup_{\substack{i \in \{0,\ldots,|\mathsf{var}(g_1)|\} \\ j \in \{0,\ldots,|\mathsf{var}(g_2)|\} \\ i+j=k}} \{\mathbf{e}_1 \mid \mathbf{e}_1 \in \mathsf{SAT}(R_{g_1}) \text{ and } |\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g_1)}, \mathbf{e}_1)| = i\} \otimes \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \{\mathbf{e}_2 \mid \mathbf{e}_2 \in \mathsf{SAT}(R_{g_2}) \text{ and } |\mathsf{sim}(\mathbf{e}_{|\mathsf{var}(g_2)}, \mathbf{e}_2)| = j\} \\
&\quad = \bigcup_{\substack{i \in \{0,\ldots,|\mathsf{var}(g_1)|\} \\ j \in \{0,\ldots,|\mathsf{var}(g_2)|\} \\ i+j=k}} \mathsf{SAT}(R_{g_1}, \mathbf{e}_{|\mathsf{var}(g_1)}, i) \otimes \mathsf{SAT}(R_{g_2}, \mathbf{e}_{|\mathsf{var}(g_2)}, j).
\end{aligned}$$

Combining the previous results, we obtain that

$$\mathsf{SAT}(R_g, \mathbf{e}_{|\mathsf{var}(g)}, k) \;=\; \bigcup_{\substack{i \in \{0,\ldots,|\mathsf{var}(g_1)|\} \\ j \in \{0,\ldots,|\mathsf{var}(g_2)|\} \\ i+j=k}} \mathsf{SAT}(R_{g_1}, \mathbf{e}_{|\mathsf{var}(g_1)}, i) \otimes \mathsf{SAT}(R_{g_2}, \mathbf{e}_{|\mathsf{var}(g_2)}, j).$$

Thus, given that for every pair $i_1, i_2 \in \{0, \ldots, |\mathsf{var}(g_1)|\}$ such that $i_1 \neq i_2$, it holds that

$$\mathsf{SAT}(R_{g_1}, \mathbf{e}_{|\mathsf{var}(g_1)}, i_1) \cap \mathsf{SAT}(R_{g_1}, \mathbf{e}_{|\mathsf{var}(g_1)}, i_2) \;=\; \emptyset$$

(and similarly for R_{g_2}), we conclude by the definitions of α_g^k, $\alpha_{g_1}^i$, $\alpha_{g_2}^j$ that

$$\alpha_g^k \;=\; \sum_{\substack{i \in \{0,\ldots,|\mathsf{var}(g_1)|\} \\ j \in \{0,\ldots,|\mathsf{var}(g_2)|\} \\ i+j=k}} \alpha_{g_1}^i \cdot \alpha_{g_2}^j,$$

which was to be shown.

C Proof of Lemma 4.2

The validity of the equation from Lemma 4.2 will be consequence of the following property of the SHAP-score: for every Boolean classifier M over X, entity $\mathbf{e} \in \mathsf{ent}(X)$ and feature $x \in X$, it holds that

$$\sum_{x \in X} \mathsf{SHAP}(M, \mathbf{e}, x) \;=\; \phi(M, \mathbf{e}, X) - \phi(M, \mathbf{e}, \emptyset). \tag{4}$$

This property is often called the *efficiency* property of the Shapley value. Although this is folklore, we prove Equation (4) here for the reader's convenience. For a permutation $\pi : X \to \{1, \ldots, n\}$ and $x \in X$, let S_π^x denote the set of features that appear before x in π. Formally, $S_\pi^x := \{y \in X \mid \pi(y) < \pi(x)\}$. Then, letting $\Pi(X)$ be the set of all permutations $\pi : X \to \{1, \ldots, n\}$, observe that the definition of SHAP-score from Definition 2.1 can be rewritten as

$$\mathsf{SHAP}(M, \mathbf{e}, x) \;=\; \frac{1}{n!} \sum_{\pi \in \Pi(X)} \big(\phi(M, \mathbf{e}, S_\pi^x \cup \{x\}) - \phi(M, \mathbf{e}, S_\pi^x)\big).$$

Hence, we have that

$$\begin{aligned}
\sum_{x \in X} \mathsf{SHAP}(M, \mathbf{e}, x) \;&=\; \frac{1}{n!} \sum_{x \in X} \sum_{\pi \in \Pi(X)} \big(\phi(M, \mathbf{e}, S_\pi^x \cup \{x\}) - \phi(M, \mathbf{e}, S_\pi^x)\big) \\
&=\; \frac{1}{n!} \sum_{\pi \in \Pi(X)} \sum_{x \in X} \big(\phi(M, \mathbf{e}, S_\pi^x \cup \{x\}) - \phi(M, \mathbf{e}, S_\pi^x)\big)
\end{aligned}$$

$$= \frac{1}{n!} \sum_{\pi \in \Pi(X)} \big(\phi(M, \mathbf{e}, X) - \phi(M, \mathbf{e}, \emptyset)\big),$$

where the last equality is obtained by noticing that the inner sum is a telescoping sum. This establishes Equation (4). Now, we simply use the definition of $\phi(\cdot, \cdot, \cdot)$ in this equation to obtain

$$\begin{aligned} \sum_{x \in X} \mathsf{SHAP}(M, \mathbf{e}, x) &= M(\mathbf{e}) - \frac{1}{2^n} \sum_{\mathbf{e}' \in \mathsf{ent}(X)} M(\mathbf{e}') \\ &= M(\mathbf{e}) - \frac{\#\mathsf{SAT}(M)}{2^n}, \end{aligned}$$

thus proving Lemma 4.2.

D Proof of Theorem 4.1

We have already explained in the body of this article why Item (1) of Theorem 4.1 holds. We now justify that (2) holds, by proving that from every 3-CNF formula ψ, we can build in polynomial time an equivalent deterministic Boolean circuit C_ψ.

Given a clause $\gamma = (\ell_1 \vee \ell_2 \vee \ell_3)$ consisting of three literals, define $d(\gamma)$ as the propositional formula

$$(\ell_1 \wedge \ell_2 \wedge \ell_3) \vee (\overline{\ell_1} \wedge \ell_2 \wedge \ell_3) \vee (\ell_1 \wedge \overline{\ell_2} \wedge \ell_3) \vee (\ell_1 \wedge \ell_2 \wedge \overline{\ell_3}) \vee (\overline{\ell_1} \wedge \overline{\ell_2} \wedge \ell_3) \vee (\overline{\ell_1} \wedge \ell_2 \wedge \overline{\ell_3}) \vee (\ell_1 \wedge \overline{\ell_2} \wedge \overline{\ell_3}),$$

where $\overline{x} = \neg x$ and $\overline{\neg x} = x$, for each propositional variable x. Clearly, γ and $d(\gamma)$ are equivalent formulae. Moreover, given a propositional formula $\psi = \gamma_1 \wedge \cdots \wedge \gamma_k$ in 3-CNF, where each γ_i is a clause with three literals, define $d(\psi)$ as the propositional formula $d(\gamma_1) \wedge \cdots \wedge d(\gamma_k)$. Clearly, ψ and $d(\psi)$ are equivalent formulae, from which we have that $\#\mathsf{SAT}(\psi) = \#\mathsf{SAT}(d(\psi))$. Moreover, $d(\psi)$ can be directly transformed into a deterministic Boolean Circuit $C_{d(\psi)}$. Hence, from the fact that $C_{d(\psi)}$ can be constructed in polynomial time from an input propositional formula ψ in 3-CNF, and the fact that $\#\mathsf{SAT}(\cdot)$ is #P-hard for 3-CNFs, we have that Theorem 4.1 (2) holds from Lemma 4.2.

E Proof of Theorem 5.1

In this section, we prove that computing the SHAP-score for Boolean classifiers given as deterministic and decomposable Boolean circuits can be done in polynomial time for product distributions; see Theorem 5.1 for the formal statement. As mentioned in Section 5, the proof will be slightly more involved than that of Theorem 3.1; this is because not all entities have the same probability, and this prevents us from reducing to $\#\mathsf{SAT}(\cdot, \cdot, \cdot)$. Instead, we will use a different definition of $\mathsf{H}(\cdot, \cdot, \cdot)$ and show that it can directly be computed bottom-up on the circuits.

But before that, we introduce new notation that will be more convenient for this proof. For a Boolean classifier M over features X, probability distribution[3] $\mathcal{D} : \mathsf{ent}(X) \to [0, 1]$, entity $\mathbf{e} \in \mathsf{ent}(X)$ and set $S \subseteq X$, we define

$$\phi_{\mathcal{D}}(M, \mathbf{e}, S) \coloneqq \mathbb{E}_{\mathbf{e}' \sim \mathcal{D}}\big[M(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S)\big].$$

Notice that we now use the notation $\mathbb{E}_{\mathbf{e}' \sim \mathcal{D}}[f(\mathbf{e}')]$ for expected value of a random variable f, instead of the simpler $\mathbb{E}[f(\mathbf{e}')]$ that we used in the body of the paper. This is because we will sometimes need to make explicit what is the probability distribution to consider. Then given a Boolean classifier M over a set of features X, a probability distribution $\mathcal{D}$ over $\mathsf{ent}(X)$, an entity $\mathbf{e}$ over X, and a feature $x \in X$, the *Shapley value of feature x in $\mathbf{e}$ with respect to M under $\mathcal{D}$* is defined as

$$\mathsf{SHAP}_{\mathcal{D}}(M, \mathbf{e}, x) \coloneqq \sum_{S \subseteq X \setminus \{x\}} \frac{|S|!\,(|X| - |S| - 1)!}{|X|!} \Big(\phi_{\mathcal{D}}(M, \mathbf{e}, S \cup \{x\}) - \phi_{\mathcal{D}}(M, \mathbf{e}, S)\Big). \tag{5}$$

Note that by taking $\mathcal{D}$ to be the uniform probability distribution on $\mathsf{ent}(X)$, we obtain the definition that we considered in Section 2. In this section we will consider the product distributions Π_{p} as defined in Section 5. With these notation in place, we can now start the proof.

For a Boolean classifier M over a set of variables X, a probability distribution $\mathcal{D}$ over $\mathsf{ent}(X)$, an entity $\mathbf{e} \in \mathsf{ent}(X)$ and a natural number $k \leq |X|$, define

$$\mathsf{H}_{\mathcal{D}}(M, \mathbf{e}, k) \coloneqq \sum_{\substack{S \subseteq X \\ |S| = k}} \mathbb{E}_{\mathbf{e}' \sim \mathcal{D}}[M(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S)].$$

[3]Note that $\mathcal{D} : \mathsf{ent}(X) \to [0, 1]$ is actually a *probability mass function*, but we will abuse notation to simplify the presentation.

Our proof of Theorem 5.1 is divided into two modular parts. The first part, which is developed in Section E.1, consists in showing that the problem of computing $\mathsf{SHAP}_{\Pi.}(\cdot,\cdot,\cdot)$ can be reduced in polynomial time to that of computing $\mathsf{H}_{\Pi.}(\cdot,\cdot,\cdot)$. This part of the proof is again a sequence of formula manipulations, and it only uses the fact that deterministic and decomposable circuits can be efficiently conditioned on a variable value. In the second part of the proof, which is developed in Section E.2, we show that computing $\mathsf{H}_{\Pi.}(\cdot,\cdot,\cdot)$ can be done in polynomial time for deterministic and decomposable Boolean circuits. It is in this part that the magic of deterministic and decomposable circuits really operates.

E.1 Reducing in polynomial-time from $\mathsf{SHAP}_{\Pi.}(\cdot,\cdot,\cdot)$ to $\mathsf{H}_{\Pi.}(\cdot,\cdot,\cdot)$

In this section, we show that for deterministic and decomposable Boolean circuits and under product distributions, the computation of the SHAP-score can be reduced in polynomial time to the computation of $\mathsf{H}_{\Pi.}(\cdot,\cdot,\cdot)$. We wish to compute $\mathsf{SHAP}_{\Pi_p}(C,\mathbf{e},x)$, for a given deterministic and decomposable circuit C over a set of variables X, probability mapping $p : X \to [0,1]$, entity $\mathbf{e} \in \mathsf{ent}(X)$ and feature $x \in X$. Define

$$\mathsf{Diff}_k(C,\mathbf{e},x) \;:=\; \sum_{\substack{S\subseteq X\setminus\{x\}\\ |S|=k}} (\phi_{\Pi_p}(C,\mathbf{e},S\cup\{x\}) - \phi_{\Pi_p}(C,\mathbf{e},S)),$$

and let $n = |X|$. We then have

$$\begin{aligned}\mathsf{SHAP}_{\Pi_p}(C,\mathbf{e},x) \;&=\; \sum_{S\subseteq X\setminus\{x\}} \frac{|S|!(n-|S|-1)!}{n!}(\phi_{\Pi_p}(C,\mathbf{e},S\cup\{x\}) - \phi_{\Pi_p}(C,\mathbf{e},S))\\ &=\; \sum_{k=0}^{n-1} \frac{k!(n-k-1)!}{n!}\mathsf{Diff}_k(C,\mathbf{e},x).\end{aligned}$$

Therefore, it is enough to show how to compute in polynomial time the quantities $\mathsf{Diff}_k(C,\mathbf{e},x)$ for each $k \in \{0,\ldots,n-1\}$. By definition of $\phi_{\Pi.}(\cdot,\cdot,\cdot)$ we have that

$$\begin{aligned}\mathsf{Diff}_k(C,\mathbf{e},x) \;=\; &\Bigg[\sum_{\substack{S\subseteq X\setminus\{x\}\\ |S|=k}} \mathbb{E}_{\mathbf{e}'\sim\Pi_p}[C(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e},S\cup\{x\})]\Bigg]\\ &- \Bigg[\sum_{\substack{S\subseteq X\setminus\{x\}\\ |S|=k}} \mathbb{E}_{\mathbf{e}'\sim\Pi_p}[C(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e},S)]\Bigg].\end{aligned} \tag{†}$$

In this expression, let α and β be the left- and right-hand side terms in the subtraction. For a set of features X, mapping $p : X \to [0,1]$ and $S \subseteq X$, we write $p_{|S} : S \to [0,1]$ for the mapping that is the restriction of p to S, and $\Pi_{p_{|S}} : \mathsf{ent}(S) \to [0,1]$ for the corresponding product distribution on $\mathsf{ent}(S)$. Looking closer at β, we have that

$$\begin{aligned}\beta \;&=\; \sum_{\substack{S\subseteq X\setminus\{x\}\\ |S|=k}} \mathbb{E}_{\mathbf{e}'\sim\Pi_p}[C(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e},S)]\\ &=\; p(x)\cdot \sum_{\substack{S\subseteq X\setminus\{x\}\\ |S|=k}} \mathbb{E}_{\mathbf{e}'\sim\Pi_p}[C(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e},S) \text{ and } \mathbf{e}'(x)=1]\\ &\qquad\qquad + (1-p(x))\cdot \sum_{\substack{S\subseteq X\setminus\{x\}\\ |S|=k}} \mathbb{E}_{\mathbf{e}'\sim\Pi_p}[C(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e},S) \text{ and } \mathbf{e}'(x)=0]\\ &=\; p(x)\cdot \sum_{\substack{S\subseteq X\setminus\{x\}\\ |S|=k}} \mathbb{E}_{\mathbf{e}''\sim\Pi_{p_{|X\setminus\{x\}}}}[C_{+x}(\mathbf{e}'') \mid \mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|X\setminus\{x\}},S)]\\ &\qquad\qquad + (1-p(x))\cdot \sum_{\substack{S\subseteq X\setminus\{x\}\\ |S|=k}} \mathbb{E}_{\mathbf{e}''\sim\Pi_{p_{|X\setminus\{x\}}}}[C_{-x}(\mathbf{e}'') \mid \mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|X\setminus\{x\}},S)]\\ &=\; p(x)\cdot \mathsf{H}_{\Pi_{p_{|X\setminus\{x\}}}}(C_{+x},\mathbf{e}_{|X\setminus\{x\}},k) \;+\; (1-p(x))\cdot \mathsf{H}_{\Pi_{p_{|X\setminus\{x\}}}}(C_{-x},\mathbf{e}_{|X\setminus\{x\}},k),\end{aligned}$$

where the last equality is obtained simply by using the definition of $\mathsf{H}_{\cdot}(\cdot,\cdot,\cdot)$. Hence, if we could compute in polynomial time $H_{\Pi_{\cdot}}(\cdot,\cdot,\cdot)$ for deterministic and decomposable Boolean circuits, then we could compute β in polynomial time as C_{+x} and C_{-x} can be computed in linear time from C, and they are deterministic and decomposable Boolean circuits as well. We now inspect the term α, which we recall is

$$\alpha \;=\; \sum_{\substack{S \subseteq X \setminus \{x\} \\ |S|=k}} \mathbb{E}_{\mathbf{e}' \sim \Pi_p}[C(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S \cup \{x\})].$$

But then observe that, for $S \subseteq X \setminus \{x\}$ and $\mathbf{e}' \in \mathsf{cw}(\mathbf{e}, S \cup \{x\})$, it holds that

$$C(\mathbf{e}') \;=\; \begin{cases} C_{+x}(\mathbf{e}'_{|X \setminus \{x\}}) & \text{if } \mathbf{e}(x) = 1 \\ C_{-x}(\mathbf{e}'_{|X \setminus \{x\}}) & \text{if } \mathbf{e}(x) = 0 \end{cases}.$$

Therefore, if $\mathbf{e}(x) = 1$, we have that

$$\begin{aligned} \alpha \;&=\; \sum_{\substack{S \subseteq X \setminus \{x\} \\ |S|=k}} \mathbb{E}_{\mathbf{e}'' \sim \Pi_{p_{|X \setminus \{x\}}}}[C_{+x}(\mathbf{e}'') \mid \mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|X \setminus \{x\}}, S)] \\ &= \mathsf{H}_{\Pi_{p_{|X \setminus \{x\}}}}(C_{+x}, \mathbf{e}_{|X \setminus \{x\}}, k) \end{aligned}$$

whereas if $\mathbf{e}(x) = 0$, we have that

$$\alpha \;=\; \mathsf{H}_{\Pi_{p_{|X \setminus \{x\}}}}(C_{-x}, \mathbf{e}_{|X \setminus \{x\}}, k).$$

Hence, again, if we were able to compute in polynomial time $\mathsf{H}_{\Pi_{\cdot}}(\cdot,\cdot,\cdot)$ for deterministic and decomposable Boolean circuits, then we could compute α in polynomial time (as deterministic and decomposable Boolean circuits C_{+x} and C_{-x} can be computed in linear time from C). But then we deduce from (†) that $\mathsf{Diff}_k(C, \mathbf{e}, x)$ could be computed in polynomial time for each $k \in \{0, \ldots, n-1\}$, from which we have that $\mathsf{SHAP}_{\Pi_p}(C, \mathbf{e}, x)$ could be computed in polynomial time, therefore concluding the existence of the reduction claimed in this section.

E.2 Computing $\mathsf{H}_{\Pi_{\cdot}}(\cdot,\cdot,\cdot)$ in polynomial time

We now take care of the second part of the proof of Theorem 5.1, i.e., proving that computing $\mathsf{H}_{\Pi_{\cdot}}(\cdot,\cdot,\cdot)$ for deterministic and decomposable Boolean circuits can be done in polynomial time. Formally:

Lemma E.1. *The following problem can be solved in polynomial time. Given as input a deterministic and decomposable Boolean circuit C over a set of variables X, rational probability values $p(x)$ for each $x \in X$, an entity $\mathbf{e} \in \mathsf{ent}(X)$ and a natural number $k \leq |X|$, compute the quantity $\mathsf{H}_{\Pi_p}(C, \mathbf{e}, k)$.*

We first perform two preprocessing steps on C, which will simplify the proof. These are the same preprocessing steps that we did in Section 3, but we added more details for the reader's convenience.

Rewriting to fan-in at most 2. First, we modify the circuit C so that the fan-in of every $\vee$- and $\wedge$-gate is at most 2. This can simply be done in linear time by rewriting every $\wedge$-gate (resp., and $\vee$-gate) of fan-in $m > 2$ with a chain of $m-1$ $\wedge$-gates (resp., $\vee$-gates) of fan-in 2. It is clear that the resulting Boolean circuit is deterministic and decomposable. Hence, from now on we assume that the fan-in of every $\vee$- and $\wedge$-gate of C is at most 2.

Smoothing the circuit. Recall that a deterministic and decomposable circuit C is *smooth* if for every $\vee$-gate g and input gates g_1, g_2 of g, we have that $\mathsf{var}(g_1) = \mathsf{var}(g_2)$, and we call such an $\vee$-gate smooth. We modify as follows the circuit C so that it becomes smooth. Recall that by the previous paragraph, we assume that the fan-in of every $\vee$-gate is at most 2. For an $\vee$-gate g of C having two input gates g_1, g_2 violating the smoothness condition, define $S_1 := \mathsf{var}(g_1) \setminus \mathsf{var}(g_2)$ and $S_2 := \mathsf{var}(g_2) \setminus \mathsf{var}(g_1)$, and let d_{S_1}, d_{S_2} be Boolean circuits defined as follows. If $S_1 = \emptyset$, then d_{S_1} consist of the single constant gate 1. Otherwise, d_{S_1} encodes the propositional formula $\wedge_{x \in S_1}(x \vee \neg x)$, but it is constructed in such a way that every $\wedge$- and $\vee$-gate has fan-in at most 2. Boolean circuit d_{S_2} is constructed exactly as d_{S_1} but considering the set of variables S_2 instead of S_1. Observe that $\mathsf{var}(d_{S_1}) = S_1$, $\mathsf{var}(d_{S_2}) = S_2$ and d_{S_1}, d_{S_2} always evaluate to 1. Then, we transform g into a smooth $\vee$-gate by replacing gate g_1 by a decomposable $\wedge$-gate $(g_1 \wedge d_{S_2})$, and gate g_2 by a decomposable $\wedge$-gate $(g_2 \wedge d_{S_1})$. This does not change the Boolean classifier computed. Moreover, since $\mathsf{var}(g_1 \wedge d_{S_2}) = \mathsf{var}(g_2 \wedge d_{S_1}) = \mathsf{var}(g_1) \cup \mathsf{var}(g_2)$, we have that g is now smooth. Finally, the resulting Boolean circuit is deterministic and decomposable. Hence, by repeating the previous procedure for each non-smooth $\vee$-gate, we conclude that C can be transformed into an equivalent smooth Boolean circuit in polynomial time, which is deterministic and decomposable, and where each gate has fan-in at most 2. Thus, from now on we also assume that C is smooth.

Proof of Lemma E.1. Let C be a deterministic and decomposable Boolean circuit C over a set of variables X, $p : X \to [0,1]$ be a rational probability mapping, $\mathbf{e} \in \mathsf{ent}(X)$ and k a natural number such that $k \leq |X|$, and let $n = |X|$. For a gate g of C, let R_g be the Boolean circuit over $\mathsf{var}(g)$ that is defined by considering the subgraph of C induced by the set of gates g' in C for which there exists a path from g' to g in C.[4] Notice that R_g is a deterministic and decomposable Boolean circuit with output gate g. Moreover, for a gate g and natural number $l \leq |\mathsf{var}(g)|$, define $\alpha_g^l \coloneqq \mathsf{H}_{\Pi_{p_{|\mathsf{var}(g)}}}(R_g, \mathbf{e}_{|\mathsf{var}(g)}, l)$, which we recall is equal, by definition, to

$$\mathsf{H}_{\Pi_{p_{|\mathsf{var}(g)}}}(R_g, \mathbf{e}_{|\mathsf{var}(g)}, l) = \sum_{\substack{S \subseteq \mathsf{var}(g) \\ |S|=l}} \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\mathsf{var}(g)}}}[R_g(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g)}, S)].$$

We will show how to compute all the values α_g^l for every gate g of C and $l \in \{0, \ldots, |\mathsf{var}(g)|\}$ in polynomial time. This will conclude the proof since, for the output gate g_{out} of C, we have that $\alpha_{g_{\mathsf{out}}}^k = \mathsf{H}_{\Pi_p}(C, \mathbf{e}, k)$. Next we explain how to compute these values by bottom-up induction on C.

Variable gate. g is a variable gate with label $y \in X$, so that $\mathsf{var}(g) = \{y\}$. Then for $\mathbf{e}' \in \mathsf{ent}(\{y\})$ we have $R_g(\mathbf{e}') = \mathbf{e}'(y)$, therefore

$$\begin{aligned}
\alpha_g^0 &= \sum_{\substack{S \subseteq \{y\} \\ |S|=0}} \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\{y\}}}}[\mathbf{e}'(y) \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\{y\}}, S)] \\
&= \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\{y\}}}}[\mathbf{e}'(y) \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\{y\}}, \emptyset)] \\
&= \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\{y\}}}}[\mathbf{e}'(y)] \\
&= 1 \cdot p(y) + 0 \cdot (1 - p(y)) \\
&= p(y)
\end{aligned}$$

and

$$\begin{aligned}
\alpha_g^1 &= \sum_{\substack{S \subseteq \{y\} \\ |S|=1}} \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\{y\}}}}[\mathbf{e}'(y) \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\{y\}}, S)] \\
&= \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\{y\}}}}[\mathbf{e}'(y) \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\{y\}}, \{y\})] \\
&= \mathbf{e}(y).
\end{aligned}$$

Constant gate. g is a constant gate with label $a \in \{0,1\}$, and $\mathsf{var}(g) = \emptyset$. We recall the mathematical convention that there is a unique function with the empty domain and, hence, a unique entity over $\emptyset$. But then

$$\begin{aligned}
\alpha_g^0 &= \sum_{\substack{S \subseteq \emptyset \\ |S|=0}} \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\emptyset}}}[a \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\emptyset}, S)] \\
&= \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\emptyset}}}[a \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\emptyset}, \emptyset)] \\
&= a.
\end{aligned}$$

$\neg$-gate. g is a $\neg$-gate with input gate g'. Notice that $\mathsf{var}(g) = \mathsf{var}(g')$. Then, since for $\mathbf{e}' \in \mathsf{ent}(\mathsf{var}(g))$ we have that $R_g(\mathbf{e}') = 1 - R_{g'}(\mathbf{e}')$, we have

$$\alpha_g^l = \sum_{\substack{S \subseteq \mathsf{var}(g) \\ |S|=l}} \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\mathsf{var}(g)}}}[1 - R_{g'}(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g)}, S)].$$

By linearity of expectations we deduce that

$$\begin{aligned}
\alpha_g^l = &\sum_{\substack{S \subseteq \mathsf{var}(g) \\ |S|=l}} \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\mathsf{var}(g)}}}[1 \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g)}, S)] \\
&- \sum_{\substack{S \subseteq \mathsf{var}(g) \\ |S|=l}} \mathbb{E}_{\mathbf{e}' \sim \Pi_{p_{|\mathsf{var}(g)}}}[R_{g'}(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g)}, S)]
\end{aligned}$$

[4] The only difference between R_g and C_g (defined in Section 2) is that we formally regard R_g as a Boolean classifier over $\mathsf{var}(g)$, while we formally regarded C_g as a Boolean classifier over X.

$$= \Big[\sum_{\substack{S\subseteq \mathsf{var}(g)\\ |S|=l}} 1\Big] - \alpha^l_{g'}$$
$$= \binom{|\mathsf{var}(g)|}{l} - \alpha^l_{g'}$$

for every $l \in \{0,\ldots,|\mathsf{var}(g)|\}$. By induction, the values $\alpha^l_{g'}$ for $l \in \{0,\ldots,|\mathsf{var}(g)|\}$ have already been computed. Thus, we can compute all the values α^l_g for $l \in \{0,\ldots,|\mathsf{var}(g)|\}$ in polynomial time.

$\vee$-gate. g is an $\vee$-gate. By assumption, recall that g is deterministic, smooth and has fan-in at most 2. If g has only one input g', then clearly $\mathsf{var}(g) = \mathsf{var}(g')$ and $\alpha^l_g = \alpha^l_{g'}$ for every $l \in \{0,\ldots,|\mathsf{var}(g)|\}$. Thus, assume that g has exactly two input gates g_1 and g_2, and recall that $\mathsf{var}(g_1) = \mathsf{var}(g_2) = \mathsf{var}(g)$, because g is smooth. Given that g is deterministic, observe that for every $\mathbf{e}' \in \mathsf{ent}(\mathsf{var}(g))$ we have $R_g(\mathbf{e}') = R_{g_1}(\mathbf{e}') + R_{g_1}(\mathbf{e}')$. But then for $l \in \{0,\ldots,|\mathsf{var}(g)|\}$ we have

$$\begin{aligned}
\alpha^l_g &= \sum_{\substack{S\subseteq \mathsf{var}(g)\\ |S|=l}} \mathbb{E}_{\mathbf{e}'\sim\Pi_{p_{|\mathsf{var}(g)}}}[R_{g_1}(\mathbf{e}') + R_{g_2}(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g)}, S)]\\
&= \sum_{\substack{S\subseteq \mathsf{var}(g)\\ |S|=l}} \mathbb{E}_{\mathbf{e}'\sim\Pi_{p_{|\mathsf{var}(g)}}}[R_{g_1}(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g)}, S)]\\
&\qquad + \sum_{\substack{S\subseteq \mathsf{var}(g)\\ |S|=l}} \mathbb{E}_{\mathbf{e}'\sim\Pi_{p_{|\mathsf{var}(g)}}}[R_{g_2}(\mathbf{e}') \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g)}, S)]\\
&= \alpha^l_{g_1} + \alpha^l_{g_2},
\end{aligned}$$

where the second equality is by linearity of the expectation, and the last equality is valid because g is smooth. By induction, the values $\alpha^l_{g_1}$ and $\alpha^l_{g_2}$, for each $l \in \{0,\ldots,|\mathsf{var}(g)|\}$, have already been computed. Therefore, we can compute all the values α^l_g for $l \in \{0,\ldots,|\mathsf{var}(g)|\}$ in polynomial time.

$\wedge$-gate. g is an $\wedge$-gate. By assumption, recall that g is decomposable and has fan-in at most 2. If g has only one input g', then clearly $\mathsf{var}(g) = \mathsf{var}(g')$ and $\alpha^l_g = \alpha^l_{g'}$ for every $l \in \{0,\ldots,|\mathsf{var}(g)|\}$. Thus, assume that g has exactly two input gates g_1 and g_2. For $\mathbf{e}' \in \mathsf{ent}(\mathsf{var}(g))$ we have that $R_g(\mathbf{e}') = R_{g_1}(\mathbf{e}'_{|\mathsf{var}(g_1)}) \cdot R_{g_2}(\mathbf{e}'_{|\mathsf{var}(g_2)})$. Moreover, since $\mathsf{var}(g) = \mathsf{var}(g_1) \cup \mathsf{var}(g_2)$ and $\mathsf{var}(g_1) \cap \mathsf{var}(g_2) = \emptyset$ (because g is decomposable), observe that every $S \subseteq \mathsf{var}(g)$ can be uniquely decomposed into $S_1 \subseteq \mathsf{var}(g_1)$, $S_2 \subseteq \mathsf{var}(g_2)$ such that $S = S_1 \cup S_2$. Henceforth, for $l \in \{0,\ldots,|\mathsf{var}(g)|\}$ we have

$$\alpha^l_g = \sum_{\substack{S_1\subseteq \mathsf{var}(g_1)\\ |S_1|\leq l}} \sum_{\substack{S_2\subseteq \mathsf{var}(g_2)\\ |S_2|=|\mathsf{var}(g)|-|S_1|}} \mathbb{E}_{\mathbf{e}'\sim\Pi_{p_{|\mathsf{var}(g)}}}[R_{g_1}(\mathbf{e}'_{|\mathsf{var}(g_1)}) \cdot R_{g_2}(\mathbf{e}'_{|\mathsf{var}(g_2)}) \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g)}, S_1 \cup S_2)].$$

But, by definition of the product distribution $\Pi_{p_{|\mathsf{var}(g)}}$, we have that $R_{g_1}(\mathbf{e}'_{|\mathsf{var}(g_1)})$ and $R_{g_2}(\mathbf{e}'_{|\mathsf{var}(g_2)})$ are independent random variables, hence we deduce

$$\begin{aligned}
\alpha^l_g &= \sum_{\substack{S_1\subseteq \mathsf{var}(g_1)\\ |S_1|\leq l}} \sum_{\substack{S_2\subseteq \mathsf{var}(g_2)\\ |S_2|=|\mathsf{var}(g)|-|S_1|}} \Big[\mathbb{E}_{\mathbf{e}'\sim\Pi_{p_{|\mathsf{var}(g)}}}[R_{g_1}(\mathbf{e}'_{|\mathsf{var}(g_1)}) \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g)}, S_1 \cup S_2)]\\
&\qquad \times \mathbb{E}_{\mathbf{e}'\sim\Pi_{p_{|\mathsf{var}(g)}}}[R_{g_2}(\mathbf{e}'_{|\mathsf{var}(g_2)}) \mid \mathbf{e}' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g)}, S_1 \cup S_2)]\Big].\\
&= \sum_{\substack{S_1\subseteq \mathsf{var}(g_1)\\ |S_1|\leq l}} \sum_{\substack{S_2\subseteq \mathsf{var}(g_2)\\ |S_2|=|\mathsf{var}(g)|-|S_1|}} \Big[\mathbb{E}_{\mathbf{e}''\sim\Pi_{p_{|\mathsf{var}(g_1)}}}[R_{g_1}(\mathbf{e}'') \mid \mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g_1)}, S_1)]\\
&\qquad \times \mathbb{E}_{\mathbf{e}''\sim\Pi_{p_{|\mathsf{var}(g_2)}}}[R_{g_2}(\mathbf{e}'') \mid \mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g_2)}, S_2)]\Big],
\end{aligned}$$

where the last equality is simply by definition of the product distributions, and because $R_{g_1}(\mathbf{e}'_{|\mathsf{var}(g_1)})$ is independent of the value $\mathbf{e}'_{|\mathsf{var}(g_2)}$, and similarly for $R_{g_2}(\mathbf{e}'_{|\mathsf{var}(g_2)})$. But then

$$\alpha^l_g = \sum_{\substack{S_1\subseteq \mathsf{var}(g_1)\\ |S_1|\leq l}} \mathbb{E}_{\mathbf{e}''\sim\Pi_{p_{|\mathsf{var}(g_1)}}}[R_{g_1}(\mathbf{e}'') \mid \mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g_1)}, S_1)] \times \sum_{\substack{S_2\subseteq \mathsf{var}(g_2)\\ |S_2|=|\mathsf{var}(g)|-|S_1|}}$$

$$
\begin{aligned}
&\qquad\qquad\qquad\qquad\qquad\qquad \mathbb{E}_{\mathbf{e}'' \sim \Pi_{p_{|\mathsf{var}(g_2)}}}[R_{g_2}(\mathbf{e}'') \mid \mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g_2)}, S_2)] \\
&= \sum_{\substack{S_1 \subseteq \mathsf{var}(g_1) \\ |S_1| \leq l}} \mathbb{E}_{\mathbf{e}'' \sim \Pi_{p_{|\mathsf{var}(g_1)}}}[R_{g_1}(\mathbf{e}'') \mid \mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g_1)}, S_1)] \times \alpha_{g_2}^{|\mathsf{var}(g)| - |\mathsf{var}(S_1)|} \\
&= \sum_{l_1=0}^{l} \alpha_{g_2}^{|\mathsf{var}(g)| - l_1} \times \sum_{\substack{S_1 \subseteq \mathsf{var}(g_1) \\ |S_1| = l_1}} \mathbb{E}_{\mathbf{e}'' \sim \Pi_{p_{|\mathsf{var}(g_1)}}}[R_{g_1}(\mathbf{e}'') \mid \mathbf{e}'' \in \mathsf{cw}(\mathbf{e}_{|\mathsf{var}(g_1)}, S_1)] \\
&= \sum_{l_1=0}^{l} \alpha_{g_2}^{|\mathsf{var}(g)| - l_1} \times \alpha_{g_1}^{l_1} \\
&= \sum_{\substack{l_1 \in \{0, \ldots, |\mathsf{var}(g_1)|\} \\ l_2 \in \{0, \ldots, |\mathsf{var}(g_2)|\} \\ l_1 + l_2 = l}} \alpha_{g_1}^{l_1} \cdot \alpha_{g_2}^{l_2}.
\end{aligned}
$$

By induction, the values $\alpha_{g_1}^{l_1}$ and $\alpha_{g_2}^{l_2}$, for each $l_1 \in \{0, \ldots, |\mathsf{var}(g_1)|\}$ and $l_2 \in \{0, \ldots, |\mathsf{var}(g_2)|\}$, have already been computed. Therefore, we can compute all the values α_g^l for $l \in \{0, \ldots, |\mathsf{var}(g)|\}$ in polynomial time.

This concludes the proof of Lemma E.1 and, hence, the proof of Theorem 5.1. □