On the Computational Tractability of the (Many) Shapley Values

Reda Marzouk*[1]   **Shahaf Bassan*[2]**   **Guy Katz†[2]**   **Colin de la Higuera†[1]**

[1] LS2N, Université de Nantes, France   [2] The Hebrew University of Jerusalem, Israel

Abstract

Recent studies have examined the computational complexity of computing Shapley additive explanations (also known as SHAP) across various models and distributions, revealing their tractability or intractability in different settings. However, these studies primarily focused on a specific variant called Conditional SHAP, though many other variants exist and address different limitations. In this work, we analyze the complexity of computing a much broader range of such variants, including Conditional, Interventional, and Baseline SHAP, while exploring both local and global computations. We show that both local and global Interventional and Baseline SHAP can be computed in polynomial time for various ML models under Hidden Markov Model distributions, extending popular algorithms such as TreeSHAP beyond empirical distributions. On the downside, we prove intractability results for these variants over a wide range of neural networks and tree ensembles. We believe that our results emphasize the intricate diversity of computing Shapley values, demonstrating how their complexity is substantially shaped by both the specific SHAP variant, the model type, and the distribution.

A prominent method for providing post-hoc explanations for ML models is via Shapley additive explanations (SHAP) (Lundberg and Lee, 2017). However, a major limitation of SHAP is the significantly high computational complexity of computing these explanations (Bertossi et al., 2020). Practical methods — like those in the popular SHAP library (Lundberg and Lee, 2017) — typically address this computational bur-

den in one of two ways. The first is through approximation techniques, such as KernelSHAP (Lundberg and Lee, 2017), which offer greater scalability but lack formal guarantees for the resulting explanations. The second approach involves designing algorithms tailored to specific, simpler model types (e.g., tree-based models or linear models), which are more computationally feasible. However, these methods also typically rely on underlying assumptions. For example, the popular TreeSHAP algorithm (Lundberg et al., 2020) assumes explanations are based on empirical distributions, while LinearSHAP (Lundberg and Lee, 2017) assumes feature independence.

These model-specific algorithms have sparked interest in developing a deeper theoretical understanding of the computational complexity involved in calculating Shapley values for various types of models and under different distributions. One of the early works by Van den Broeck et al. (2022) presented tractable results for a range of models while also establishing NP-hardness for relatively simple settings, such as computing SHAP for decision trees with naive Bayes modeled distributions. Additionally, Arenas et al. (2023) demonstrated that computing SHAP is tractable for Decomposable Deterministic Boolean Circuits under independent distributions, while Marzouk and De La Higuera (2024) reported similar positive complexity results for Weighted Automata under Markovian distributions.

However, a key limitation of these previous computational complexity works is that they have primarily focused on a specific SHAP variant known as Conditional SHAP (Sundararajan and Najmi, 2020). The explainable AI community has explored a variety of SHAP variants, including Conditional, Interventional (Janzing et al., 2020), and Baseline (Sundararajan and Najmi, 2020) SHAP, among many others (Frye et al., 2020b; Albini et al., 2022). These variants were introduced mainly to address the axiomatic limitations of the original Conditional SHAP formulation (Sundararajan and Najmi, 2020; Huang and Marques-Silva, 2023). Analyzing these variants is

---

*Equal contribution (first authors), †Equal contribution (last authors).

Table 1: Summary of complexity results for Baseline, Interventional, and Conditional SHAP (Base, Interv, Cond) applied to decision trees (`DT`), tree ensembles for regression and classification (`ENS-DT`$_R$, `ENS-DT`$_C$), linear regression (`LIN`$_R$), weighted automata (`WA`), and neural networks (`NN-SIGMOID`, `RNN-ReLU`). Results are analyzed under independent (`IND`), empirical (`EMP`), or hidden Markov model (`HMM`) distributions, with novel findings of this work highlighted in blue. L and G denote local and global SHAP, respectively (L* or G* if only one is covered). While most settings are either tractable (PTIME) or intractable (NP-H, coNP-H, NTIME-H, #P-H) for all SHAP variants, we indicate cases where a *strict complexity gap* between SHAP variants exists ($\downarrow$ notation).

		IND	EMP	HMM
`DT, ENS-DT`$_R$`, LIN`$_R$	**Base**	PTIME (L,G)	PTIME (L,G)	PTIME (L,G)
	Interv	PTIME (L,G)	PTIME (L,G) $\downarrow$	PTIME (L,G) $\downarrow$
	Cond	PTIME (L,G*)	NP-H (L) $\downarrow$	#P-H (L) $\downarrow$
`WA`	**Base**	PTIME (L,G)	PTIME (L,G)	PTIME (L,G)
	Interv	PTIME (L,G)	PTIME (L,G) $\downarrow$	PTIME (L,G) $\downarrow$
	Cond	PTIME (L,G*)	NP-H (L) $\downarrow$	#P-H (L) $\downarrow$
`ENS-DT`$_C$	**Base**	—	NP-H (L,G)	NP-H (L,G)
	Interv	#P-H	NP-H (L)	#P-H (L)
	Cond	#P-H	NP-H (L)	#P-H (L)
`NN-SIGMOID`	**Base**	—	NTIME-H (L,G)	NTIME-H (L,G)
	Interv	NP- H (L)	NTIME-H (L)	NP-H (L)
	Cond	NP-H (L)	—	NP-H (L)
`RNN-ReLU`	**Base**	—	coNP-H (L,G)	coNP-H (L,G)
	Interv	NP-H (L)	coNP-H (L)	NP-H (L)
	Cond	NP-H (L)	—	NP-H (L)

vital as many popular SHAP algorithms incorporate them. For example, KernelSHAP (Lundberg and Lee, 2017) and TreeSHAP (Lundberg et al., 2020) typically compute *interventional* SHAP values, and not conditional ones.

The aim of this paper is to provide a comprehensive, multi-dimensional perspective on the problem of SHAP computation, analyzed through the lens of formal computational theory (Arora and Barak, 2009). This analysis spans four key dimensions: (i) the variants of the Shapley value, including Baseline, Interventional, and Conditional SHAP; (ii) the class of models to be interpreted; (iii) the distributional assumptions regarding the input data generation process; and (iv) the scope of the explanatory analysis, which can be either *local* or *global*, with the global scope measured as an aggregate of *local* SHAP values relative to the input data-generating distribution (Frye et al., 2020b).

Our contributions. The complexity results of this work, summarized in Table 1, are:

- *On the positive side,* in Section 2 we prove that both local and global Interventional and Baseline SHAP can be computed in polynomial time for the family of weighted automata, decision trees, tree ensembles in regression tasks, and linear regression models when input instances are assumed to be generated from a distribution modeled by a

Hidden Markov model.

- *On the negative side,* in Section 3 we establish intractability results, including NP-hardness, coNP-hardness, etc., for computing not only Conditional SHAP but also Interventional and Baseline SHAP across a range of neural networks (e.g., ReLU, Sigmoid, RNN-ReLU) and tree ensemble classifiers (e.g., Random Forests, XGBoost). These results hold even under strict conditions, such as uniform distributions or feature independence.

- Finally, in Section 4, we present generalized computational complexity relationships between SHAP variants, demonstrating that some are more or less tractable than others in certain distributional scenarios. Using these findings, we establish new complexity results for computing various SHAP variants across different models.

Key Takeaways

The obtained complexity results provide both theoretical and practical insights regarding the problem of SHAP computation:

1. **There are substantial complexity differences between SHAP variants.** Particularly, we show *stark complexity gaps* between obtaining Conditional SHAP, which is NP-Hard, and both Interventional SHAP and Baseline SHAP, which

can be solved in polynomial time — demonstrating substantial deviations among the SHAP variants. Our findings additionaly pinpoint the specific settings where these gaps occur and where they do not.

2. **The distributional assumptions made by TreeSHAP and LinearSHAP can be extended to cover substantially more expressive classes of distributions.** Specifically, we demonstrate that local and global Interventional and Baseline SHAP can be solved efficiently in polynomial time for certain model families, including XGBoost trees, while surpassing the distributional scope of the widely used TreeSHAP algorithm, limited to empirical distributions. In addition, our results also relax the feature independence requirement in LinearSHAP. These results could significantly improve SHAP value distribution modeling, which is essential for computing faithful explanations (Aas et al., 2021).

3. **Obtaining SHAP is strictly easier in soft-voting tree ensembles compared to hard-voting tree ensembles.** Many of our findings highlight significant differences in the complexity of obtaining SHAP for tree ensembles used in *regression* versus *classification*. These findings extend some of the conclusions made in (Huang and Marques-Silva, 2024b) to a more general setting, and underline the feasibility of obtaining SHAP explanations for soft-voting ensembles as compared to hard-voting ensembles, where these prove to be intractable.

4. **Obtaining SHAP for neural networks is hard, even in highly simplified settings.** We prove various intractbility results (e.g., NP, #P-Hardness, etc.) for different neural networks, demonstrating that this hardness persists in different settings, and even for *baseline* SHAP, the simplest SHAP variant.

5. **Obtaining Global SHAP is often tractable, despite the additional expectation factor.** Interestingly, we prove that in many scenarios, the tractability of local SHAP extends to the global SHAP variant explored in (Frye et al., 2020b), despite the added complexity of computing an expectation over the entire data distribution.

Due to space limitations, we provide only a brief summary of the proofs for our claims in the paper, with the full detailed proofs included in the appendix.

1 Background

1.1 Model Types

We examine the following types of models: (i) linear regression models ($\mathtt{LIN_R}$); (ii) decision trees ($\mathtt{DT}$); (iii) tree ensembles (including both majority voting ensembles such as Random Forests and weighted voting ensembles like XGBoost) used for regression or classification ($\mathtt{ENS\text{-}DT_R}$, $\mathtt{ENS\text{-}DT_C}$); (iv) (feed-forward/recurrent) neural networks with ReLU/Sigmoid activations ($\mathtt{NN\text{-}SIGMOID}$, $\mathtt{RNN\text{-}ReLU}$); and (v) Weighted Automata ($\mathtt{WA}$). A complete formalization of all models is provided in Appendix 3.

Although WAs may be considered a niche model family within the broader ML community, a significant portion of our paper focuses on establishing tractability results for them. From the perspective of Explainable AI, our interest in WAs stems from two factors: Firstly, WAs have been proposed as abstractions of neural networks (Okudono et al., 2020; Eyraud and Ayache, 2024; Weiss et al., 2019; Lacroce et al., 2021), offering enhanced transparency. Secondly, and importantly, WAs can be reduced to various other popular ML models like decision trees, linear regression, and tree ensembles, making tractability results for WAs applicable to a wide range of models. We begin by defining N-Alphabet WAs, a generalization of WAs:

Definition 1 (N-Alphabet Weighted Automata). *For $n, N \in \mathbb{N}$, and $\{\Sigma_i\}_{i \in [N]}$, a collection of finite alphabets, an N-Alphabet Weighted Automaton A over the product $\Sigma_1 \times \ldots \times \Sigma_N$ is defined by the tuple $\langle \alpha, A_{\sigma_1, \ldots, \sigma_N}, \beta \rangle$, where $(\alpha, \beta) \in \mathbb{R}^n \times \mathbb{R}^n$ are the initial and final state vectors, and $A_{\sigma_1, \ldots, \sigma_N} \in \mathbb{R}^{n \times n}$ are transition matrices. The N-Alphabet WA A computes a function over $\Sigma_1 \times \ldots \times \Sigma_N$ as:*

$$f_A(w^{(1)}, \ldots, w^{(N)}) \overset{\text{def}}{=} \alpha^T \cdot \prod_{i=1}^{L} A_{w_i^{(1)} \ldots w_i^{(N)}} \cdot \beta$$

where $(w^{(1)}, \ldots, w^{(N)}) \in \Sigma_1 \times \ldots \times \Sigma_N$ and $|w^{(1)}| = \ldots = |w^{(N)}| = L$.

The integer n denotes the size of the WA A, denoted as $\mathtt{size}(A)$. For $N = 1$, 1-Alphabet WAs match the classical definition of WA (Droste et al., 2009), so we use "WA" and "1-letter WA" interchangeably.

1.2 Distributions

Our analysis briefly touches on several types of distributions, primarily for comparison purposes. These include (i) *independent distributions* ($\mathtt{IND}$), where all features are assumed to be independent of one another; (ii) *empirical distributions* ($\mathtt{EMP}$), i.e., the family of distributions induced from finite datasets; and

(iii) *Markovian distributions* (MARKOV), i.e., distributions where the future state depends only on the current state, independent of past states. A full formalization of these distribution families is in Appendix 3.

Previous studies explored the complexity of these three distributions for the conditional SHAP variant (Arenas et al., 2023; Van den Broeck et al., 2022; Marzouk and De La Higuera, 2024; Huang and Marques-Silva, 2024b). However, the tractability results here apply to a broader class of distributions, specifically those modeled by *Hidden Markov Models* (HMMs). HMMs are more expressive than standard Markovian distributions, as they incorporate hidden states to model sequences influenced by latent variables. In Appendix 5, we prove that HMMs include independent, Markovian, and empirical distributions.

HMM distributions. HMMs (Rabiner and Juang, 1986) are a popular class of sequential latent probabilistic models used in various applications (Knill and Young, 1997; De Fonzo et al., 2007). For an alphabet Σ (also referred to as the observation space), an HMM defines a probabilistic function over Σ^∞. Formally, an HMM of size n over Σ is a tuple $\langle \alpha, T, O \rangle$, where: (i) $\alpha \in \mathbb{R}^n$, the initial state vector, represents a probability distribution over $[n]$; and (ii) $T \in \mathbb{R}^{n \times n}$, $O \in \mathbb{R}^{n \times |\Sigma|}$ are stochastic matrices, with each row encoding a probability distribution.

HMMs and WAs. The WA formalism in Definition 1 suffices to cover HMMs, up to reparametrization. Indeed, it has been proven that the probability that an HMM $M = \langle \alpha, T, O \rangle$ generates a prefix $w \in \Sigma^*$ is: $\mathbf{1}^T \cdot \prod_{i=1}^{|w|} A_{w_i} \cdot \alpha$ (Hsu et al., 2012), where $\mathbf{1}$ is a row vector with all 1's, and for any $\sigma \in \Sigma$, $A_\sigma \stackrel{\text{def}}{=} T \cdot \text{Diag}(O[:, \sigma])$. The matrix $\text{Diag}(O[:, \sigma])$ is the diagonal matrix formed from the column vector in O indexed by σ. We follow this parameterization of HMMs and assume they are parameterized by the 1-Alphabet WA formalism in Definition 1. For *non-sequential models*, we assume the family of HMMs, denoted $\overrightarrow{\text{HMM}}$, represents latent variable models describing probability distributions over random vectors in a finite domain.

Definition 2. $(\overrightarrow{\text{HMM}})$ *Let* $(n, N) \in \mathbb{N}^2$ *be two integers, and* $\mathbb{D}$ *a finite set. An* $\overrightarrow{\text{HMM}}$ *over* $\mathbb{D}^n$ *is parameterized by the tuple* $\langle \pi, \alpha, \{T_i\}_{i \in [n]}, \{O_i\}_{i \in [n]} \rangle$, *where* π *is a permutation on* $[n]$, *and for each* $i \in [n]$, T_i *and* O_i *are stochastic matrices in* $\mathbb{R}^N$ *and* $\mathbb{R}^{N \times |\mathbb{D}|}$, *respectively. A model* M *in* $\overrightarrow{\text{HMM}}$ *computes the following probability distribution over* $\mathbb{D}^n$:

$$P_M(x_1, \ldots, x_n) := \mathbf{1}^T \cdot \prod_{i=1}^{n} A_{i, x_{\pi(i)}} \cdot \alpha$$

where: $A_{i,x} \stackrel{\text{def}}{=} T_i \cdot Diag(O_i[:, x])$.

In essence, models in the family $\overrightarrow{\text{HMM}}$ are non-stationary HMMs where observations are ordered by a permutation π. They include a stopping probability mechanism, terminating after the n-th symbol with probability 1. Like HMMs, $\overrightarrow{\text{HMM}}$ includes independent, empirical, and Markovian distributions (see proof in Appendix 5).

1.3 The (Many) Shapley Values

Local Shapley values. Let there be a discrete input space $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ and a model f, which can be either a regression model $f : \mathcal{X} \to \mathbb{R}$ or a classification model $f : \mathcal{X} \to [c]$ for a certain set of classes $[c]$ ($c \in \mathbb{N}$), along with a specific *local* instance $\mathbf{x} \in \mathcal{X}$. Then, the (local) Shapley value attribution for a feature $i \in [n]$ with respect to $\langle f, \mathbf{x} \rangle$ is defined as:

$$\phi(f, \mathbf{x}, i) \stackrel{\text{def}}{=} \sum_{S \subseteq [n] \setminus \{i\}} \frac{|S|! \cdot (n - |S| - 1)!}{n!} \cdot$$
$$[v(f, \mathbf{x}, S \cup \{i\}) - v(f, \mathbf{x}, S)]$$

where v is referred to as the *value function* of ϕ. The primary versatility of Shapley values lies in the various ways v can be defined. Typically, Shapley values are computed with respect to a distribution $\mathcal{D}_p$ over $\mathcal{X}$, meaning v is determined by this distribution, i.e., $v(f, \mathbf{x}, S, \mathcal{D}_p)$. A common definition of v is through conditional expectation, referred to here as *Conditional SHAP*, also known as Conditional Expectation SHAP (CES) (Sundararajan and Najmi, 2020):

$$v_c(f, \mathbf{x}, S, \mathcal{D}_p) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{z} \sim \mathcal{D}_p}[f(\mathbf{z}) | \mathbf{z}_S = \mathbf{x}_S] \quad (1)$$

where $\mathbf{z}_S = \mathbf{x}_S$ indicates that the values of the features S in $\mathbf{z}$ are set to those in $\mathbf{x}$. Another approach for computing the value function is *Interventional SHAP* (Janzing et al., 2020), also known as Random-Baseline SHAP (Sundararajan and Najmi, 2020), used in practical algorithms like KernelSHAP, LinearSHAP, and TreeSHAP (Lundberg and Lee, 2017; Lundberg et al., 2020). In interventional SHAP, when a feature $j \in \overline{S}$ is missing, it is replaced with a reference value independently drawn from a predefined distribution, breaking dependencies with other features. Formally:

$$v_i(f, \mathbf{x}, S, \mathcal{D}_p) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{z} \sim \mathcal{D}_p}[f(\mathbf{x}_S; \mathbf{z}_{\overline{S}})] \quad (2)$$

where $(\mathbf{x}_S; \mathbf{z}_{\overline{S}})$ represents a vector in which the features in S are fixed to the values in $\mathbf{x}$, and the features in $\overline{S}$ are fixed to the values in $\mathbf{z}$. When the distribution $\mathcal{D}_p$ assumes feature independence, interventional

and conditional SHAP coincide, i.e., $v_i(f, \mathbf{x}, S, \mathcal{D}_p) = v_c(f, \mathbf{x}, S, \mathcal{D}_p)$ (Sundararajan and Najmi, 2020). However, this alignment does not typically hold in many real-world distributions.

Finally, instead of defining Shapley values with respect to a distribution $\mathcal{D}_p$, they can be defined using an auxiliary baseline $\mathbf{z}^{\text{ref}}$ that captures the "missingness" of features in $\overline{S}$. *Baseline SHAP* (Sundararajan and Najmi, 2020) is defined as follows:

$$v_b(f, \mathbf{x}, S, \mathbf{z}^{\text{ref}}) \stackrel{\text{def}}{=} f(\mathbf{x}_S; \mathbf{z}^{\text{ref}}_{\overline{S}}) \qquad (3)$$

By substituting these value function definitions into the generic Shapley value formula (Equation 1), we obtain $\phi_c(f, i, \mathcal{D}_p)$, $\phi_i(f, i, \mathcal{D}_p)$, and $\phi_b(f, i, \mathbf{z}^{\text{ref}})$, corresponding to the (local) conditional, interventional, and baseline Shapley values, respectively.

Global Shapley values. Shapley values $\phi(f, \mathbf{x}, i)$ offer *local* explainability for the model's prediction on a specific data point $\mathbf{x}$. One approach to deriving a global importance indicator for input features from their local Shapley values consists at aggregating these values over the input space, weighted by the target data generating distribution (Frye et al., 2020b):[1]

$$\Phi(f, i, \mathcal{D}_p) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_p}[\phi(f, \mathbf{x}, i)] \qquad (4)$$

Note that the global Shapley value is always computed with respect to a distribution $\mathcal{D}_p$, as the inputs $\mathbf{x}$ are aggregated over it. This gives rise to $\Phi_c(f, i, \mathcal{D}_p)$, $\Phi_i(f, i, \mathcal{D}_p)$, and $\Phi_b(f, i, \mathcal{D}_p, \mathbf{z}^{\text{ref}})$, representing the (global) conditional, interventional, and baseline Shapley values, respectively. For *sequential models*, f accepts input vectors of arbitrary length n, i.e., $|\mathbf{x}| = n$. While local Shapley values are computed for a specific input $\mathbf{x}$, global Shapley values pose a challenge as they are input-independent. One approach is to fix the feature space length n and compute the global Shapley value for an input of that size. In this case, the global Shapley value Φ incorporates n as part of its input: $\Phi(f, i, \mathcal{D}_p, n) \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_p^{(n)}}[\phi(f, \mathbf{x}, i)]$, where $i \in [n]$, $\mathcal{D}_p$ is a probability distribution over an infinite alphabet Σ^∞, and $\mathcal{D}_p^{(n)}$ is the probability of generating an infinite sequence prefixed by $\mathbf{x} \in \Sigma^n$.

Shapley values for sequential models. For complexity results on sequential models (WAs and RNNs), we build on prior work (Marzouk and De La Higuera, 2024), which uses the *pattern* formalism to analyze the complexity of obtaining Shapley values for these

models. Formally, let Σ be a finite alphabet, with its elements called symbols. The set of all finite (and infinite) sequences from Σ is denoted by Σ (and Σ^∞). For any integer $n > 0$, Σ^n represents sequences of length n. For a sequence $w \in \Sigma$, $|w|$ is its length, $w_{i:j}$ is the subsequence from the i-th to the j-th symbol, and w_i is its i-th symbol. A pattern p over Σ is a regular expression in the form $\#^{i_1} w_1 \ldots \#^{i_n} w_n \#^{i_{n+1}}$, where $\#$ is a placeholder symbol (i.e., $\# = \Sigma$), $\{i_k\}_{k \in [n+1]}$ are integers, and $\{w_k\}_{k \in [n+1]}$ are sequences over Σ^*. The extended alphabet $\Sigma \cup \#$ is $\Sigma\#$. The language of a pattern p is Lp, with $|p|$ as its length and $|p|_\#$ indicating the number of $\#$ symbols.

We define two operators on patterns: (i) The *swap* operator, which takes a tuple $(p, \sigma, i) \in \Sigma_\#^* \times \Sigma \times \mathbb{N}$ with $i \leq |p|$, and returns a pattern where the i-th symbol of p is replaced by σ. For example, with $\Sigma = \{0, 1\}$: $\mathtt{swap}(0\#0\#, 1, 2) = 0\mathbf{1}0\#$; (ii) The *do* operator, which takes a tuple $(p, w', w) \in \Sigma_\#^* \times \Sigma^* \times \Sigma^*$ with $|w| = |w'| = |p|$, and returns a sequence u where $u_i = w'_i$ if $p_i = \#$, and $u_i = w_i$ otherwise. For example, with $\Sigma = \{0, 1\}$: $\mathtt{do}(0\#0\#, 1100, 1111) = 1110$. We represent a coalition S in the SHAP formulation using patterns. For instance, with the alphabet $\Sigma = \{0, 1\}$ and the sequence $w = 0011$, the coalition of the first and third symbols is represented by the pattern $0\#1\#$.

SHAP as a computational problem. As outlined in the introduction, this work aims to provide a comprehensive computational analysis of the SHAP problem across several dimensions: (i) the class of models being interpreted; (ii) the underlying data-generating distributions; (iii) the specific SHAP variant; and (iv) its scope (global or local). Each combination of these dimensions gives rise to a distinct formal computational problem. To navigate this multi-dimensional landscape of computational problems, we adopt the following notation: A (variant) of the SHAP computational problem shall be denoted as $\mathtt{(LOC|GLOB)}$-$\mathtt{(I|B|C)}$-$\mathtt{SHAP}(\mathcal{M}, \mathcal{P})$, where $\mathtt{LOC}$ and $\mathtt{GLOB}$ refer to local and global SHAP, respectively, while $\mathtt{I}$, $\mathtt{B}$, and $\mathtt{C}$ correspond to the interventional, baseline, and conditional SHAP variants. The symbols $\mathcal{M}$ and $\mathcal{P}$ represent the class of models and the class of feature distributions, respectively. Under this notation, $\mathtt{LOC}$-$\mathtt{I}$-$\mathtt{SHAP}(\mathtt{WA}, \mathtt{HMM})$ refers to the problem of computing local interventional SHAP for the family of weighted automata under Hidden Markov Model distributions.

A variant of the SHAP computational problem takes as input instance a model $M \in \mathcal{M}$, a data-generating distribution $\mathrm{P} \in \mathcal{P}$ [2], an index specifying the input

---

[1] Several other methods exist for computing global feature importance using SHAP (Covert et al., 2020; Lundberg and Lee, 2017; Lundberg et al., 2020), but they fall outside the scope of this work.

[2] Note that in the formulation of Local Baseline SHAP variants of the SHAP problem, the data-generating distribution P is replaced by a reference input instance.

feature of interest, and, in the case of local SHAP variants, the model's input undergoing explanatory analysis. The computational complexity of the problem is measured with respect to the size of M, the size of P, and the dimensionality of the input space[3]. A variant is considered tractable if it can be solved in polynomial time with respect to these parameters.

For completeness, a summary of all complexity classes discussed in this article (PTIME, NP, coNP, NTIME, and #P) is provided in Appendix 3.

2 Positive Complexity Results

This section highlights configurations that allow polynomial-time computation of various Shapley value variants. We first show that both local and global interventional and baseline SHAP values for WAs under HMM-modeled distributions can be computed in polynomial time (Theorem 1). By reductions, this result extends to various other ML models (Theorem 2).

2.1 Tractability for WAs

This main result of this subsection is given in the following theorem:

Theorem 1. *The following computational problems, which include* `LOC-I-SHAP(WA,HMM)`, `GLO-I-SHAP(WA,HMM)`, `LOC-B-SHAP(WA)`, *as well as* `GLO-B-SHAP(WA,HMM)` *are poly-time computable with respect to the size of the WA, the size of the HMM, the sequence length and the size of the alphabet.*

The remainder of this section provides a proof sketch for Theorem 1, with the complete proof available in Appendix 4. The proof is constructive and draws heavily on techniques from the theory of rational languages (Berstel and Reutenauer, 1988). We begin by linking the previously defined swap and do operators for patterns to the computation of (interventional) SHAP values. The corresponding relations for baseline SHAP are provided in Appendix 4 due to space constraints.

Lemma 1. *For a sequential model f, a string w (representing an input $\boldsymbol{x} \in \mathcal{X}$), a pattern p (representing a coalition $S \subseteq [n]$), and a distribution $\mathcal{D}_p$ over $\mathcal{X}$, then the following relations hold:*

$$v_i(f, w, p, \mathcal{D}_p) = \mathbb{E}_{z \sim \mathcal{D}_p^{|w|}} \left[f(\boldsymbol{do}(p, \boldsymbol{z}, w)) \right];$$

$$\phi_i(f, w, i, \mathcal{D}_p) = \mathbb{E}_{p \sim \mathcal{P}_i^x} [v_i(f, w, \boldsymbol{swap}(p, w_i, i), \mathcal{D}_p)$$
$$- v_i(f, w, p, \mathcal{D}_p)]$$

---

[3]For sequential models, where inputs are sequences, we assume the input space's dimensionality equals the sequence length under analysis.

where:

$$\mathcal{P}_i^w(p) \stackrel{\text{def}}{=} \begin{cases} \frac{(|p|_{\#}-1)! \cdot (|w|-|p|_{\#})!}{|w|!} & \text{if } w \in L_p \\ 0 & \text{otherwise} \end{cases}$$

To develop an algorithm for computing v_i and ϕ_i in polynomial time for the class of WA under HMM distributions, we utilize two operations on N-Alphabet WA, parameterized by the input instance:

Definition 3. *Let $N > 0$ be an integer and $\{\Sigma_i\}_{i \in [N]}$ be a collection of N alphabets.*

1. ***The Kronecker product operation*** *of two N-Alphabet WAs A and B over $\Sigma_1 \times \ldots \times \Sigma_N$ at index $i \in [N]$ returns an N-Alphabet WA over $\Sigma_1 \times \ldots \times \Sigma_N$, denoted $A \otimes B$ implementing the function:*

$$f_{A \otimes B}(w^{(1)}, \ldots, w^{(n)}) := f_A(w^{(1)}, \ldots, w^{(n)})$$
$$\cdot f_B(w^{(1)}, \ldots, w^{(n)})$$

2. ***The projection operation*** *of a 1-Alphabet WA A over an N-Alphabet WA T over $\Sigma_1 \times \ldots \times \Sigma_N$ at index $i \in [N]$, returns an (N-1)-Alphabet WA over $\Sigma_1 \times \ldots \times \Sigma_{i-1} \times \Sigma_{i+1} \times \ldots \times \Sigma_N$, denoted $\Pi_i(A, T)$, implementing the following function:*

$$g(w^{(1)}, \ldots, w^{(i-1)}, w^{(i+1)}, w^{(N)}) :=$$
$$\sum_{w \in \Sigma_i^L} f_A(w) \cdot f_T(w^{(1)}, \ldots w^{(i)}, w, w^{(i+1)}, \ldots, w^{(N)})$$

where $(w^{(1)}, \ldots, w^{(i-1)}, w^{(i+1)}, \ldots w^{(N)}) \in \Sigma_1^* \times \ldots \times \Sigma_{i-1}^* \times \Sigma_{i+1}^* \ldots \times \Sigma_N^*$ such that $|w^{(1)}| = \ldots = |w^{(i-1)}| = |w^{(i+1)}| = \ldots = |w^{(N)}| = L$.

Note that if T is a 1-Alphabet WA, then the returned model $\Pi_1(A, T)$ is a 0-Alphabet WA. For simplicity, we denote: $\Pi_1(A, T) \stackrel{\text{def}}{=} \sum_{w \in \Sigma_1} f_A(w) \cdot f_T(w)$, yielding a scalar. Additionally, we define $\Pi_0(A) \stackrel{\text{def}}{=} \Pi_1(A, \mathbf{1})$, where $\mathbf{1}$ is a WA that assigns 1 to all sequences in Σ. The next proposition provides a useful intermediary result, with its proof in Appendix 4:

Proposition 1. *If $N = O(1)$, the projection and Kronecker product operations are poly-time computable.*

Interventional and Baseline SHAP of WAs in terms of N-Alphabet WA operators. As mentioned earlier, the algorithmic construction for `LOC-I-SHAP(WA,HMM)`, and `GLO-I-SHAP(WA,HMM)` will take the form of efficiently computable operations over N-Alphabet WAs parameterized by the input instance of the corresponding problem. The main lemma formalizing this fact is given as follows:

Lemma 2. *Fix a finite alphabet Σ. Let f be a WA over Σ, and consider a sequence $(w, w^{reff}) \in \Sigma^* \times \Sigma$ (representing an input and a basline $\boldsymbol{x}, \boldsymbol{x}^{reff} \in \mathcal{X}$) such that $|w| = |w^{reff}|$. Let $i \in [|w|]$ be an integer, and $\mathcal{D}_P$ be a distribution modeled by an HMM over Σ. Then:*

$$\phi_i(f, w, i, \mathcal{D}_P) = \\ \Pi_1(A_{w,i}, \Pi_2(\mathcal{D}_P, \Pi_3(f, T_{w,i}) - \Pi_3(f, T_w)));$$
$$\Phi_i(f, i, n, \mathcal{D}_P) = \\ \Pi_0(\Pi_2(\mathcal{D}_P, A_{i,n} \otimes \Pi_2(\mathcal{D}_P, \Pi_3(f, T_i) - \Pi_3(f, T))));$$
$$\phi_b(f, w, i, w^{reff}) = \\ \Pi_1(A_{w,i}, \Pi_2(f_{w^{reff}}, \Pi_3(f, T_{w,i}) - \Pi_3(f, T_w)));$$
$$\Phi_b(f, i, n, w^{reff}, \mathcal{D}_P) = \\ \Pi_0(\Pi_2(\mathcal{D}_P, A_{i,n} \otimes \Pi_2(f_{w^{reff}}, \Pi_3(f, T_i) - \Pi_3(f, T))))$$

where:

- *$A_{w,i}$ is a 1-Alphabet WA over $\Sigma_\#$ implementing the uniform distribution over coalitions excluding the feature i (i.e., $f_{A_{w,i}} = \mathcal{P}_i^w$);*

- *T_w is a 3-Alphabet WA over $\Sigma_\# \times \Sigma \times \Sigma$ implementing the function: $g_w(p, w', u) := I(\boldsymbol{do}(p, w', w) = u)$.*

- *$T_{w,i}$ is a 3-Alphabet WA over $\Sigma_\# \times \Sigma \times \Sigma$ implementing the function: $g_{\boldsymbol{x},i}(p, w', u) := I(\boldsymbol{do}(\boldsymbol{swap}(p, w_i, i), w', w) = u)$.*

- *T is a 4-Alphabet WA over $\Sigma_\# \times \Sigma \times \Sigma \times \Sigma$ given as: $g(p, w', u, w) := g_w(p, w', u)$.*

- *T_i is a 4-Alphabet WA over $\Sigma_\# \times \Sigma \times \Sigma \times \Sigma$ given as: $g_i(p, w', u, w) := g_{w,i}(p, w', u)$.*

- *$A_{i,n}$ is a 2-Alphabet WA over $\Sigma_\# \times \Sigma$ implementing the function: $g_{i,n}(p, w) := I(p \in \mathcal{L}_i^w) \cdot \mathcal{P}_i^w(p)$, where $|w| = |p| = n$.*

- *$f_{w^{reff}}$ is an HMM such that the probability of generating w^{reff} as a prefix is equal to 1.*

The proof of Lemma 2 is in Appendix 4. In essense, it reformulates the computation of both local and global interventional and baseline SHAP using operations on N-Alphabet WAs, depending on the input instance, particularly involving $A_{w,i}$, T_w, $T_{w,i}$, T, T_i, $A_{i,n}$, and $f_{w^{ref}}$. The final step to complete the proof of Theorem 1 is to show that these WAs can be constructed in polynomial time relative to the input size.

Proposition 2. *The N-Alphabet WAs $A_{w,i}$, T_w, $T_{w,i}$, T, T_i, $A_{i,n}$ (defined in Lemma 2) and the HMM $f_{w^{reff}}$ can be constructed in polynomial time with respect to $|w|$ and $|\Sigma|$.*

The full proof of Proposition 2 can be found in Appendix 4. Theorem 1 is a direct corollary of Lemma 2, Proposition 1, and Proposition 2.

2.2 Tractability for other ML models.

Beyond the proper interest of the result in Theorem 1 regarding WAs, it also yields interesting results about the computational complexity of obtaining interventional and baseline SHAP variants for other popular ML models which include decision trees, tree ensembles for regression tasks (e.g. Random forests or XG-Boost), and linear regression models:

Theorem 2. *Let $\mathbb{S} := \{\texttt{LOC}, \texttt{GLO}\}$, $\mathbb{V} := \{\texttt{B}, \texttt{I}\}$, $\mathbb{P} := \{\texttt{EMP}, \overrightarrow{\texttt{HMM}}\}$, and $\mathbb{F} := \{\texttt{DT}, \texttt{ENS-DT}_\texttt{R}, \texttt{Lin}_\texttt{R}\}$. Then, for any $\texttt{S} \in \mathbb{S}$, $\texttt{V} \in \mathbb{V}$, $\texttt{P} \in \mathbb{P}$, and $\texttt{F} \in \mathbb{F}$ the problem $\texttt{S-V-SHAP(F,P)}$ can be solved in polynomial time.*

The proof of Theorem 2 is provided in Appendix 5, where a poly-time construction of either a decision tree, an ensemble of decision trees for regression, or a linear regression model into a WA is detailed. This result brings forward two interesting outcomes. First, it expands the distributional assumptions of some popular SHAP algorithms such as LinearSHAP and TreeSHAP. Let us denote TREE as the family of all decision trees DT and regression tree ensembles $\texttt{ENS-DT}_\texttt{R}$. Then:

Corollary 1. *While the TreeSHAP (Lundberg et al., 2020) algorithm solves $\texttt{LOC-I-SHAP(TREE,EMP)}$ and $\texttt{GLO-I-SHAP(TREE,EMP)}$ in poly-time, Theorem 2 establishes that $\texttt{LOC-I-SHAP(TREE,}\overrightarrow{\texttt{HMM}})$ and $\texttt{GLO-I-SHAP(TREE,}\overrightarrow{\texttt{HMM}})$ can be solved in poly-time.*

Since empirical distributions are strictly contained within HMMs, i.e., $\texttt{EMP} \subsetneq \texttt{HMM}$ and $\texttt{EMP} \subsetneq \overrightarrow{\texttt{HMM}}$ (see proof in Appendix 5), Corollary 1 significantly broadens the distributional assumption of the TreeSHAP algorithm beyond just empirical distributions. Similar conclusions can be drawn for LinearSHAP:

Corollary 2. *While the LinearSHAP (Lundberg and Lee, 2017) algorithm solves $\texttt{LOC-I-SHAP(LIN}_\texttt{R}\texttt{,IND)}$ in polynomial time, Theorem 2 establishes that $\texttt{LOC-I-SHAP(LIN}_\texttt{R}\texttt{,}\overrightarrow{\texttt{HMM}})$ and $\texttt{GLO-I-SHAP(LIN}_\texttt{R}\texttt{,}\overrightarrow{\texttt{HMM}})$ can be solved in polynomial time.*

Which, again, demonstrates the expansion of the distributional assumption of LinearSHAP, as $\texttt{IND} \subsetneq \overrightarrow{\texttt{HMM}}$ (see Appendix 5). Lastly, Theorem 2 establishes a *strict computational complexity gap* between computing *conditional* SHAP, which remains intractable even for simple models like decision trees under HMM distributions (and even under empirical distributions (Van den Broeck et al., 2022)), whereas computing both local interventional and baseline SHAP values are shown to be tractable. This suggests that interventional and baseline SHAP are strictly more efficient to compute in these settings.

Corollary 3. *If $f \in \{\texttt{WA}, \texttt{DT}, \texttt{ENS-DT}_\texttt{R}, \texttt{Lin}_\texttt{R}\}$, $\mathcal{D}_p := \overrightarrow{\texttt{HMM}}$ (or $\mathcal{D}_p := \texttt{HMM}$ for $\texttt{WA}$), and assuming $P \neq NP$, then*

computing local interventional SHAP or local baseline SHAP for f is strictly more computationally tractable than computing local conditional SHAP for f.

3 Negative Complexity Results

3.1 When Interventional SHAP is Hard

In this subsection, we present intractability results for interventional SHAP.

Theorem 3. *The decision version of the problem* LOC-I-SHAP(RNN-ReLu, IND) *is* NP-Hard.

Recall that an RNN-ReLu model R over Σ is parametrized as: $\langle h_{init}, W, \{v_\sigma\}_{\sigma \in \Sigma}, O \rangle$ and processes a sequence sequentially from left-to-right such that $h_\epsilon = h_{init}$, $h_{w'\sigma} = ReLu(W \cdot h_{w'} + v_\sigma)$, and $f_R(w) = I(O^T \cdot h_w \geq 0)$. The proof of Theorem 3 leverages the efficiency axiom of Shapley values. Specifically, for a sequential binary classifier f over alphabet Σ, an integer n, and $i \in [n]$, the efficiency property can be expressed as (Arenas et al., 2023):

$$\sum_{i=1}^{n} \phi_i(f, \mathbf{x}, i, P_{unif}) = f(\mathbf{x}) - P_{unif}^{(n)}(f(\mathbf{x}) = 1) \quad (5)$$

where P_{unif} denotes the uniform distribution over Σ^∞. This property will be used to reduce the EMPTY problem to the computation of interventional SHAP in our context. The EMPTY problem is defined as follows: Given a set of models $\mathcal{F}$, EMPTY takes as input some $f \in \mathcal{F}$ and an integer $n > 0$ and asks if f is empty on the support Σ^n (i.e., is the set $\{\mathbf{x} \in \Sigma^n : f(\mathbf{x}) = 1\}$ empty?). The connection between local interventional SHAP and the emptiness problem is outlined in the following proposition:

Proposition 3. *Let $\mathcal{F}$ be a class of sequential binary classifiers. Then,* EMPTY($\mathcal{F}$) *can be reduced in polynomial time to the problem* LOC-I-SHAP($\mathcal{F}$, IND).

Proposition 3 suggests that proving the NP-Hardness of the problem EMPTY(RNN-ReLu) is a sufficient condition to yield the result of Theorem 3. This condition is asserted in the following lemma:

Lemma 3. EMPTY(RNN-ReLu) *is NP-Hard.*

The proof of Lemma 3 is done via a reduction from the *closest string problem* (CSP), which is NP-Hard (Li et al., 2002). CSP takes as input a set of strings $S := \{w_i\}_{i \in [m]}$ of length n and an integer $k > 0$. The goal is to determine if there exists a string $w' \in \Sigma^n$ such that for all $w_i \in S$, $d_H(w_i, w') \leq k$, where $d_H(.,.)$ is the Hamming distance: $d_H(w, w') := \sum_{i=1}^{[|w|]} 1_{w_j}(w'_j)$. We conclude our reduction by proving the following proposition, with the proof in Appendix 6:

Proposition 4. CSP *can be reduced in polynomial time to* EMPTY(RNN-ReLu).

3.2 When Baseline SHAP is Hard

In this subsection, we turn our focus to Baseline SHAP, which can be seen as a special case of Interventional SHAP when confined to empirical distributions induced by a reference input $\mathbf{x}^{\text{reff}}$. Hence, it is expected to be computationally simpler to compute than Interventional SHAP. However, we identify specific scenarios where calculating Baseline SHAP remains computationally challenging:

Theorem 4. *(i) Unless P=co-NP, the problem* LOC-B-SHAP(NN-SIGMOID) *can not be solved in polynomial time; (ii) The decision versions of the problems* LOC-B-SHAP(RNN-ReLu) *and* LOC-B-SHAP(ENS-DT$_C$) *are co-NP-Hard and NP-Hard respectively.*

We begin with results for LOC-B-SHAP(NN-SIGMOID) and LOC-B-SHAP(RNN-ReLu). Our proofs reduce from the Dummy Player problem of Weighted Majority Games (Freixas et al., 2011):

Definition 4. *A Weighted Majority Game (WMG) G is a coalitional game defined by the tuple $\langle N, \{n_i\}_{i \in [N]}, q \rangle$, where: (i) N is the number of players; (ii) n_i is the voting power of player i, for $i \in [N]$; (iii) q is the winning quota. The value function $v_G(S)$ is 1 if $\sum_{i \in S} n_i \geq q$, and 0 otherwise.*

A *dummy player* in a WMG G is one whose voting power adds no value to any coalition. Formally, i is a dummy in G if $\forall S \subseteq [N] \setminus \{i\}$, it holds that $v_G(S \cup \{i\}) = v_G(S)$. Determining if a player i is a dummy in G is co-NP-Complete (Freixas et al., 2011).

Reducing the dummy problem in WMG to both of the problems LOC-B-SHAP(NN-SIGMOID) **and** LOC-B-SHAP(RNN-ReLu). Informally, given a WMG $G := \langle N, \{n_j\}_{j \in [N]}, q \rangle$, the reduction constructs a model f_G over the input set $\{0, 1\}^N$ from the target model family to simulate G's value function using chosen input instances $\mathbf{x}, \mathbf{x}^{\text{reff}} \in \{0, 1\}^n$, i.e., $f_G(\mathbf{x}_S; \mathbf{x}^{\text{reff}}_{\bar{S}}) \approx v(S)$. The properties of this reduction for both model families are formally stated as follows:

Proposition 5. *There are poly-time algorithms that: (i) Given a WMG G and player i, return a sigmoidal neural network f_G over $\{0, 1\}^N$, $\mathbf{x}, \mathbf{x}^{reff} \in \{0, 1\}^N$ and $\epsilon \in \mathbb{R}$ such that i is not dummy iff $\phi_b(f_G, i, \mathbf{x}, \mathbf{x}^{reff}) > \epsilon$; (ii) Given a WMG G and player i, return an RNN-ReLU f_G over $\{0, 1\}^N$, $\mathbf{x}, \mathbf{x}^{reff} \in \{0, 1\}^N$ such that i is not dummy iff $\phi_b(f_G, i, \mathbf{x}, \mathbf{x}^{reff}) > 0$.*

The complexity of computing LOC-B-SHAP(SIGMOID) and LOC-B-SHAP(RNN-ReLu) from Theorem 4 are corollaries of Proposition 5.

The problem LOC-B-SHAP(ENS-DT$_C$) is NP-Hard.
This segment is dedicated to proving the remaining point of Theorem 4 stating the NP-Hardness of LOC-B-SHAP(ENS-DT$_C$). We prove this by reducing from the classical NP-Complete 3-SAT problem. The reduction strategy is illustrated in Algorithm 4.

Algorithm 1 3-SAT to LOC-B-SHAP(ENS-DT$_C$)

Input: A CNF Formula Ψ of m clauses
Output: An instance of LOC-B-SHAP(ENS-DT$_C$)
1: $\mathbf{x} \leftarrow [1, \dots, 1]$
2: $\mathbf{x}^{\text{reff}} \leftarrow [0, \dots, 0]$
3: $i \leftarrow n + 1$
4: $\mathcal{T} \leftarrow \emptyset$
5: **for** $j \in [1, m]$ **do**
6: Construct a DT T_j that assigns 1 to variable assignments satisfying the formula: $C_j \wedge \mathbf{x}_{n+1}$.
7: $\mathcal{T} \leftarrow \mathcal{T} \cup \{T\}_j$
8: **end for**
9: Construct a null decision tree T_{null} that assigns a label 0 to all variable assignments
10: Add $m - 1$ copies of T_{null} to $\mathcal{T}$
11: **return** $\langle \mathcal{T}, i, \mathbf{x}, \mathbf{x}^{\text{reff}} \rangle$

The next proposition establishes a property of ENS-DT$_C$ used in Lemma 4 to derive our complexity results:

Proposition 6. *Let Ψ be an arbitrary CNF formula over n boolean variables, and $\mathcal{T}$ be the ensemble of decision trees outputted by Algorithm 4 for the input Ψ. We have that $f_{\mathcal{T}}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \boldsymbol{x}_{n+1}) = 1$ if $\boldsymbol{x}_{n+1} = 1 \wedge \boldsymbol{x} \models \Psi$, and $f_{\mathcal{T}}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \boldsymbol{x}_{n+1}) = 0$, otherwise.*

Using the result of Proposition 6, the following lemma directly establishes the NP-hardness of the decision problem for LOC-B-SHAP(ENS-DT$_C$):

Lemma 4. *Let Ψ be an arbitrary CNF formula of n variables, and $\langle \mathcal{T}, n + 1, \boldsymbol{x}, \boldsymbol{x}^{reff} \rangle$ be the output of Algorithm 4 for the input Ψ. We have that $\phi_b(f_{\mathcal{T}}, n + 1, \boldsymbol{x}, \boldsymbol{x}^{reff}) > 0$ iff $\exists \boldsymbol{x} \in \{0,1\}^n : \boldsymbol{x} \models \Psi$.*

4 Generalized Complexity Relations of SHAP Variants

While the previous sections presented specific results on the complexity of generating various SHAP variants for different models and distributions, this section aims to establish *general* relationships concerning the complexity of different SHAP variants. We will then leverage these insights to derive corollaries for other SHAP contexts not explicitly covered in the paper.

Proposition 7. *Let $\mathcal{M}$ be a class of models and $\mathcal{P}$ a class of probability distributions such that EMP $\preceq_P \mathcal{P}$. Then, LOC-B-SHAP($\mathcal{M}$) $\preceq_P$ GLO-B-SHAP($\mathcal{M},\mathcal{P}$) and LOC-B-SHAP($\mathcal{M}$) $\preceq_P$ LOC-I-SHAP($\mathcal{M},\mathcal{P}$).*

In other words, assume a class of probability distributions $\mathcal{P}$ is "harder" (under polynomial reductions) than the class of empirical distributions EMP, i.e., any $P \in \mathcal{P}$ can be reduced in poly-time to some $P' \in$ EMP. Then, global baseline SHAP is at least as hard to compute as local baseline SHAP and local interventional SHAP is at least as hard to compute as local baseline SHAP (both under polynomial reductions). Since EMP $\preceq_P$ HMM (proof in Appendix 5), these corollaries follow from Theorem 4 and proposition 7:

Corollary 4. *Let there be some P $\in \{$EMP, HMM$\}$, and $\overrightarrow{\text{P}} \in \{$EMP, $\overrightarrow{\text{HMM}}\}$. Then it holds that: (i) Unless P=co-NP, the problems GLO-B-SHAP(SIGMOID, $\overrightarrow{\text{P}}$) and LOC-I-SHAP(NN-SIGMOID, $\overrightarrow{\text{P}}$) can not be computed exactly in polynomial time; (ii) The decision version of the problems GLO-B-SHAP(RNN-ReLu, P) and LOC-I-SHAP(RNN-ReLu, P) are co-NP-Hard; (iii) The decision version of the problems GLO-B-SHAP(M, P) and LOC-I-SHAP(ENS-DT$_C$, P) are NP-Hard.*

Conclusion

This paper aims to enhance our understanding of the computational complexity of computing various Shapley value variants. We found that for various ML models — including decision trees, regression tree ensembles, weighted automata, and linear regression — both local and global interventional and baseline SHAP can be computed in polynomial time under HMM modeled distributions. This extends popular algorithms, such as TreeSHAP, beyond their empirical distributional scope. We also establish strict complexity gaps between the various SHAP variants (baseline, interventional, and conditional) and prove the intractability of computing SHAP for tree ensembles and neural networks in simplified scenarios. Overall, we present SHAP as a versatile framework whose complexity depends on four key factors: (i) model type, (ii) SHAP variant, (iii) distribution modeling approach, (iv) and local vs. global explanations. We believe this perspective provides deeper insight into the computational complexity of SHAP, paving the way for future work.

Our work opens promising directions for future research. First, expanding our computational analysis to other SHAP-related metrics, such as asymmetric SHAP (Frye et al., 2020b) and SAGE (Covert et al., 2020), would be valuable. Additionally, we aim to explore more expressive distribution classes and relaxed assumptions beyond those in Section 2 while maintaining tractable SHAP computation. Finally, when exact computation is intractable (Section 3), investigating the approximability of SHAP metrics through approximation and parameterized complexity theory (Downey and Fellows, 2012) is an important direction.

Acknowledgments

References

K. Aas, M. Jullum, and A. Løland. Explaining Individual Predictions when Features are Dependent: More Accurate Approximations to Shapley Values. *Artificial Intelligence*, 2021.

O. Abramovich, D. Deutch, N. Frost, A. Kara, and D. Olteanu. Banzhaf Values for Facts in Query Answering. In *2nd Proc. ACM on Management of Data (PACMMOD)*, pages 1–26, 2024.

F. Adolfi, M. Vilas, and T. Wareham. The Computational Complexity of Circuit Discovery for Inner Interpretability. 2024. Technical Report. `https://arxiv.org/abs/2410.08025`.

E. Albini, J. Long, D. Dervovic, and D. Magazzeni. Counterfactual Shapley Additive Explanations. In *Proc. Conf. on Fairness, Accountability, and Transparency (FAccT)*, pages 1054–1070, 2022.

G. Amir, S. Bassan, and G. Katz. Hard to Explain: On the Computational Hardness of In-Distribution Model Interpretation. In *Proc. 27th European Conf. on Artificial Intelligence (ECAI)*, pages 818–825, 2024.

M. Arenas, P. Barcelo, L. Bertossi, and M. Monet. On the Complexity of SHAP-Score-Based Explanations: Tractability via Knowledge Compilation and Non-Approximability Results. *Journal of Machine Learning Research (JMLR)*, pages 1–58, 2023.

S. Arora and B. Barak. *Computational Complexity: a Modern Approach*. Cambridge University Press, 2009.

G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Lagniez, and P. Marquis. On Preferred Abductive Explanations for Decision Trees and Random Forests. In *Proc. 31st Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 643–650, 2022a.

G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Lagniez, and P. Marquis. Trading Complexity for Sparsity in Random Forest Explanations. In *Proc. 36th AAAI Conf. on Artificial Intelligence*, pages 5461–5469, 2022b.

P. Barceló, M. Monet, J. Pérez, and B. Subercaseaux. Model Interpretability Through the Lens of Computational Complexity. In *Proc. 34th Conf. on Advances in Neural Information Processing Systems (Neurips)*, pages 15487–15498, 2020.

P. Barceló, R. Cominetti, and M. Morgado. When is the Computation of a Feature Attribution Method Tractable? 2025a. Technical Report. `https://arxiv.org/abs/2501.02356`.

P. Barceló, A. Kozachinskiy, M. R. Orth, B. Subercaseaux, and J. Verschae. Explaining k-Nearest Neighbors: Abductive and Counterfactual Explanations. 2025b. Technical Report. `https://arXiv:2501.06078`.

C. Barrett and C. Tinelli. Satisfiability Modulo Theories. *Handbook of model checking*, 2018.

S. Bassan and G. Katz. Towards Formal XAI: Formally Approximate Minimal Explanations of Neural Networks. In *Proc. 29th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 187–207, 2023.

S. Bassan, G. Amir, D. Corsi, I. Refaeli, and G. Katz. Formally Explaining Neural Networks within Reactive Systems. In *Proc. 23rd Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 1–13, 2023.

S. Bassan, G. Amir, and G. Katz. Local vs. Global Interpretability: A Computational Complexity Perspective. In *Proc. 41st Int. Conf. on Machine Learning (ICML)*, 2024.

S. Bassan, R. Eliav, and S. Gur. Explain Yourself, Briefly! Self-Explaining Neural Networks with Concise Sufficient Reasons. 2025. Technical Report. `https://arxiv.org/abs/2502.03391`.

J. Berstel and C. Reutenauer. *Rational Series and their Languages*. Springer-Verlag, 1988.

L. Bertossi, J. Li, M. Schleich, and D. S. Z. Vagena. Causality-Based Explanation of Classification Outcomes. In *Proc. 4th Int. Workshop on Data Management for End-to-End Machine Learning (DEEM'20)*, pages 1–10, 2020.

L. Bertossi, B. Kimelfeld, E. Livshits, and M. Monet. The Shapley Value in Database Management. *ACM Sigmod Record*, pages 6–17, 2023.

G. Blanc, J. Lange, and L.-Y. Tan. Provably Efficient, Succinct, and Precise Explanations. In *Proc. 35th Conf. on Advances in Neural Information Processing Systems (Neurips)*, pages 6129–6141, 2021.

M. A. Burgess and A. C. Chapman. Approximating the Shapley Value Using Stratified Empirical Bernstein Sampling. In *Proc. 30th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 73–81, 2021.

M. Calautti, E. Malizia, and C. Molinaro. On the Complexity of Global Necessary Reasons to Explain Classification. 2025. Technical Report. `https://arXivpreprintarXiv:2501.06766`.

M. C. Cooper and J. Marques-Silva. Tractability of Explaining Classifier Decisions. *Artificial Intelligence*, 2023.

I. Covert, S. M. Lundberg, and S.-I. Lee. Understanding Global Feature Contributions with Additive Importance Measures. In *Proc. 34th Int. Conf. on Advances in Neural Information Processing Systems (NeurIPS)*, pages 17212–17223, 2020.

A. Darwiche and A. Hirth. On the Reasons Behind Decisions. In *Proc. 24th European Conf. on Artifical Intelligence (ECAI)*, pages 712–720, 2020.

A. Darwiche and C. Ji. On the Computation of Necessary and Sufficient Explanations. In *Proc. 36th AAAI Conf. on Artificial Intelligence*, pages 5582–5591, 2022.

V. De Fonzo, F. Aluffi-Pentini, and V. Parisi. Hidden Markov Models in Bioinformatics. *Current Bioinformatics*, pages 49–61, 2007.

D. Deutch, N. Frost, B. Kimelfeld, and M. Monet. Computing the Shapley Value of Facts in Query Answering. In *Proc. Int. Conf. on Management of Data (MOD)*, pages 1570–1583, 2022.

R. Downey and M. Fellows. *Parameterized Complexity*. Springer Science & Business Media, 2012.

M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata*. Springer Publishing Company, Incorporated, 1st edition, 2009.

R. Eyraud and S. Ayache. Distillation of Weighted Automata from Recurrent Neural Networks using a Spectral Approach. *Machine Learning*, pages 3233–3266, 2024.

J. Freixas, X. Molinero, M. Olsen, and M. Serna. On the Complexity of Problems on Simple Games. *RAIRO-Operations Research*, pages 295–314, 2011.

C. Frye, D. de Mijolla, T. Begley, L. Cowton, M. Stanley, and I. Feige. Shapley Explainability on the Data Manifold. In *Proc. 11th Int. Conf. on Learning Representations (ICLR)*, 2020a.

C. Frye, C. Rowat, and I. Feige. Asymmetric Shapley Values: Incorporating Causal Knowledge into Model-Agnostic Explainability. In *Proc. 34th Int. Conf. on Advances in Neural Information Processing Systems (NeurIPS)*, pages 1229–1239, 2020b.

D. Fryer, I. Strümke, and H. Nguyen. Shapley Values for Feature Selection: The Good, the Bad, and the Axioms. *IEEE Access*, pages 144352–144360, 2021.

F. Fumagalli, M. Muschalik, P. Kolpaczki, E. Hüllermeier, and B. Hammer. SHAP-IQ: Unified Approximation of Any-Order Shapley Interactions. In *Proc. 38th Int. Conf. on Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

T. Heskes, E. Sijben, I. G. Bucur, and T. Claassen. Causal Shapley Values: Exploiting Causal Knowledge to Explain Individual Predictions of Complex Models. In *Proc. 34th Int. Conf. on Advances in Neural Information Processing Systems (NeurIPS)*, pages 4778–4789, 2020.

D. Hsu, S. M. Kakade, and T. Zhang. A Spectral Algorithm for Learning Hidden Markov Models. 2012. Technical Report. `https://arxiv.org/abs/0811.4413`.

X. Huang and J. Marques-Silva. The Inadequacy of Shapley Values for Explainability. 2023. Technical Report. `https://arxiv.org/abs/2302.08160`.

X. Huang and J. Marques-Silva. On the Failings of Shapley Values for Explainability. *Int. Journal of Approximate Reasoning (IJAR)*, 2024a.

X. Huang and J. Marques-Silva. Updates on the Complexity of SHAP Scores. In *Proc. 33rd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 403–412, 2024b.

A. Ignatiev. Towards Trustable Explainable AI. In *Proc. 29th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 5154–5158, 2020.

A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based Explanations for Machine Learning Models. In *Proc. 33rd AAAI Conf. on Artificial Intelligence*, pages 1511–1519, 2019.

Y. Izza, X. Huang, A. Morgado, J. Planes, A. Ignatiev, and J. Marques-Silva. Distance-Restricted Explanations: Theoretical Underpinnings & Efficient Implementation. 2024. Technical Report. `https://arXiv:2405.08297`.

D. Janzing, L. Minorics, and P. Bloebaum. Feature Relevance Quantification in Explainable AI: A Causal

Problem. In *Proc. 23rd Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 2907–2916, 2020.

A. Kara, D. Olteanu, and D. Suciu. From Shapley Value to Model Counting and Back. In *2nd Proc. ACM on Management of Data (PACMMOD)*, pages 1–23, 2024.

P. Karmakar, M. Monet, P. Senellart, and S. Bressan. Expected Shapley-like Scores of Boolean Functions: Complexity and Applications to Probabilistic Databases. In *2nd Proc. ACM on Management of Data (PACMMOD)*, pages 1–26, 2024.

G. Katz, C. Barrett, D. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Proc. 29th Int. Conf. on Computer Aided Verification (CAV)*, pages 97–117, 2017.

K. Knill and S. Young. Hidden Markov Models in Speech and Language Processing. *Corpus-Based Methods in Language and Speech Processing*, pages 27–68, 1997.

I. Kumar, S. Venkatasubramanian, C. Scheidegger, and S. Friedler. Problems with Shapley-value-based Explanations as Feature Importance Measures. In *Proc. 37th Int. Conf. on Machine Learning (ICML)*, pages 5491–5500, 2020.

Y. Kwon, M. Rivas, and J. Zou. Efficient Computation and Analysis of Distributional Shapley Values. In *Proc. 24th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 793–801, 2021.

C. Lacroce, P. Panangaden, and G. Rabusseau. Extracting Weighted Automata for Approximate Minimization in Language Modelling. In *Proc. 15th Int. Conf. on Grammatical Inference (ICGI)*, pages 92–112, 2021.

M. Li, B. Ma, and L. Wang. On the Closest String and Substring Problems. *Journal of the ACM (JACM)*, pages 157–171, 2002.

E. Livshits, L. Bertossi, B. Kimelfeld, and M. Sebag. The Shapley Value of Tuples in Query Answering. *Logical Methods in Computer Science*, 2021.

S. Lundberg and S.-I. Lee. A Unified Approach to Interpreting Model Predictions. In *Proc. 31st Int. Conf. on Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

S. Lundberg, E. Gabriel, H. Chen, A. DeGrave, J. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From Local Explanations to Global Understanding with Explainable AI for Trees. *Nature Machine Intelligence*, pages 56–67, 2020.

J. Marques-Silva. Logic-based Explainability in Machine Learning. In *Proc. 18th Int. Summer School of Reasoning Web. Causality, Explanations and Declarative Knowledge*, pages 24–104, 2023.

J. Marques-Silva and X. Huang. Explainability is NOT a Game. *Communications of the ACM*, pages 66–75, 2024.

R. Marzouk and C. De La Higuera. On the Tractability of SHAP Explanations under Markovian Distributions. In *Proc. 41st Int. Conf. on Machine Learning (ICML)*, 2024.

T. Okudono, M. Waga, T. Sekiyama, and I. Hasuo. Weighted Automata Extraction from Recurrent Neural Networks via Regression on State Spaces. In *Proc. 34th AAAI Conf. on Artificial Intelligence*, pages 5306–5314, 2020.

S. Ordyniak, G. Paesani, and S. Szeider. The Parameterized Complexity of Finding Concise Local Explanations. In *Proc. 32nd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 3312–3320, 2023.

L. Rabiner and B. Juang. An Introduction to Hidden Markov Models. *IEEP ASSP Magazine*, pages 4–16, 1986.

M. Sundararajan and A. Najmi. The Many Shapley Values for Model Explanation. In *Proc. 37th Int. Conf. on Machine Learning (ICML)*, pages 9269–9278, 2020.

M. Sundararajan, K. Dhamdhere, and A. Agarwal. The Shapley Taylor Interaction Index. In *Proc. 37th Int. Conf. on Machine Learning (ICML)*, pages 9259–9268, 2020.

M. Taufiq, P. Blöbaum, and L. Minorics. Manifold Restricted Interventional Shapley Values. In *Proc. 26th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 5079–5106, 2023.

G. Van den Broeck, A. Lykov, M. Schleich, and D. Suciu. On the Tractability of SHAP Explanations. *Journal of Artificial Intelligence Research (JAIR)*, pages 851–886, 2022.

S. Wäldchen, J. Macdonald, S. Hauch, and G. Kutyniok. The Computational Complexity of Understanding Binary Classifier Decisions. *Journal of Artificial Intelligence Research (JAIR)*, 70:351–387, 2021.

S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, and J. Z. Kolter. Beta-Crown: Efficient Bound Propagation with Per-Neuron Split Constraints for Neural Network Robustness Verification. In *Proc. 35th Conf. on Advances in Neural Information Processing Systems (NeurIPS)*, pages 29909–29921, 2021.

G. Weiss, Y. Goldberg, and E. Yahav. Learning Deterministic Weighted Automata with Queries and Counterexamples. In *Proc. 33rd Int. Conf. on Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

H. Wu, O. Isac, A. Zeljić, T. Tagomori, M. Daggitt, W. Kokke, I. Refaeli, G. Amir, K. Julian, S. Bassan, et al. Marabou 2.0: A Versatile Formal Analyzer of Neural Networks. In *Proc. 36th Int. Conf. on Computer Aided Verification (CAV)*, pages 249–264, 2024.

Checklist

1. For all models and algorithms presented, check if you include:

   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] Mainly in Section 1 and the appendix.

   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] See Section 2, Section 3, Section 4, and the appendix.

   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:

   (a) Statements of the full set of assumptions of all theoretical results. [Yes] Mainly in Section 1 and the appendix.

   (b) Complete proofs of all theoretical results. [Yes] Because of space constraints, we provide only a brief summary of the proofs for our claims in the paper, with the complete detailed proofs available in the appendix.

   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:

   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Not Applicable]

   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable]

   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]

   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

   (a) Citations of the creator If your work uses existing assets. [Not Applicable]

   (b) The license information of the assets, if applicable. [Not Applicable]

   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

   (d) Information about consent from data providers/curators. [Not Applicable]

   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Not Applicable]

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix

The appendix provides formalizations, supplementary background, and gathers the proofs of several mathematical statements referenced either implicitly or explicitly throughout the main article. It is structured as follows:

- Appendix 1 discusses the limitations of this work and outlines potential directions for future research.
- Appendix 3 provides the technical background and preliminary results that will be referenced throughout the appendix.
- Appendix 4 includes the proofs of intermediate mathematical statements that demonstrate the tractability of computing both local and global interventional SHAP and baseline SHAP variants (Theorem 1).
- Appendix 5 elaborates on the reduction strategy from WAs to decision trees, linear regression models, and tree ensembles employed for regression (Theorem 2). Moreover, it presents a polynomial-time reduction between the distribution families discussed throughout the main paper, along with corollaries that can be derived from these relationships.
- Appendix 6 provides proofs of the intermediary results that validate the reduction strategy from the closest string problem to `LOC-I-SHAP(RNN-ReLu, IND)` (Theorem 3).
- Appendix 7 is focused on providing the complete proofs of intermediate results related to the complexity of computing local Baseline SHAP for Sigmoidal Neural Networks, RNN-ReLUs, and tree ensemble classifiers (Theorem 4).
- Appendix 8, presents the proof for the main result discussed in the section on generalized relations of SHAP variants (Section 4) of the main article, specifically Proposition 7.

1   Limitations and Future Work

First, while our work presents novel complexity results for the computation of various Shapley value variants, there are many other variants that we did not address, such as asymmetric Shapley values (Frye et al., 2020b), counterfactual Shapley values (Albini et al., 2022), and others. Additionally, in Section 4, we provided new complexity relationships between SHAP variants, but many more connections remain unexplored. Extensions of our work could also consider different model types and distributional assumptions beyond those discussed here.

A second limitation is that, while we thoroughly examine theoretical strict complexity gaps between polynomial-time and non-polynomial-time (i.e., NP-Hard) computations of SHAP variants, we did not focus on techincal optimizations of the specific poly-time algorithms proposed in this paper. Some of the polynomial factors of these algorithms, though constant, may be improved (see Section 4 for more details), and we believe improving them, as well as exploring potential parallelization techniques for solving them presents an exciting direction for future research.

Finally, we adopt the common convention used in all previous works on the computational complexity of computing Shapley values (Arenas et al., 2023; Van den Broeck et al., 2022; Marzouk and De La Higuera, 2024; Huang and Marques-Silva, 2024b), assuming a discrete input space to simplify the technical aspects of the proofs. However, we emphasize that many of our findings also extend to continuous domains. Specifically, for both decision trees and tree ensembles, the complexity results remain consistent across both discrete and continuous domains, as the tractability of the underlying models is unaffected. On the other hand, while this assumption does not generally apply to linear models and neural networks, we stress that the computational *hardness* results we present for these models continue to hold in continuous settings. The same, however, cannot be said for membership proofs. Extending the membership proofs for linear models and neural networks, as presented in this work, to continuous domains and exploring other computational complexity frameworks for these models offers a promising direction for future research.

2 Extended Related Work

In this section, we include a more elaborate discussion of related work and key complexity results examined in prior research.

SHAP values. Building on the initial SHAP framework introduced by (Lundberg and Lee, 2017) for deriving explanations of ML models, various subsequent works extensively investigated the application of SHAP across various contexts within the XAI literature. Many efforts have concentrated on developing numerous other SHAP variants beyond Conditional SHAP (Sundararajan and Najmi, 2020; Janzing et al., 2020; Heskes et al., 2020), aligning SHAP with the distribution manifold (Frye et al., 2020a; Taufiq et al., 2023), and enhancing the approximation of its calculation (Fumagalli et al., 2024; Sundararajan et al., 2020; Burgess and Chapman, 2021; Kwon et al., 2021). Additionally, the literature has explored various limitations of SHAP in different contexts (Fryer et al., 2021; Huang and Marques-Silva, 2024a; Kumar et al., 2020; Marques-Silva and Huang, 2024).

The Complexity of SHAP. Notably, (Van den Broeck et al., 2022) investigates Conditional SHAP, presenting a range of tractability and intractability results for different ML models, with a key insight being that computing Conditional SHAP under independent distributions is as complex as computing the conditional expectation. Later, Arenas et al. (2023) generalizes these findings, showing that the tractability results for Conditional SHAP align with the class of Decomposable Deterministic Boolean Circuits and establishing that both the Decomposability and Determinism properties are necessary for tractability. More recently, Marzouk and De La Higuera (2024) moves beyond the independent distribution assumption, extending the analysis to *Markovian* distributions, which are significantly less expressive than the HMM-modeled distributions considered in our work. Additionally, Huang and Marques-Silva (2024b) introduces distinctions between regression and classification in tree ensembles for Conditional SHAP and extends previous results to diverse input and output settings. Other relevant, but less direct extensions of these complexity results are extensions to the domain of databse tuples (Deutch et al., 2022; Livshits et al., 2021; Bertossi et al., 2023; Kara et al., 2024; Karmakar et al., 2024), as well as obtaining the complexity of other interaraction values other than Shapley values (Abramovich et al., 2024; Barceló et al., 2025a). Our work provides a novel analysis of all three major SHAP variants (baseline, interventional, and conditional), broadening distributional assumptions and offering new insights into local and global SHAP values. We frame SHAP computation as a multifaceted process shaped by (i) model type; (ii) SHAP variant; (iii) distributional assumptions; and (iv) the local-global distinction.

Formal XAI. More broadly, our work falls within the subdomain of interest known as *formal XAI* (Marques-Silva, 2023), which aims to generate explanations for ML models with formal guarantees (Ignatiev, 2020; Bassan and Katz, 2023; Darwiche and Hirth, 2020; Darwiche and Ji, 2022; Ignatiev et al., 2019; Audemard et al., 2022a). These explanations are often derived using formal reasoning tools, such as SMT solvers (Barrett and Tinelli, 2018) (e.g., for explaining tree ensembles (Audemard et al., 2022b)) or neural network verifiers (Katz et al., 2017; Wu et al., 2024; Wang et al., 2021) (e.g., for explaining neural networks (Izza et al., 2024; Bassan et al., 2023)). A key focus within formal XAI is analyzing the computational complexity of obtaining such explanations (Barceló et al., 2020; Wäldchen et al., 2021; Cooper and Marques-Silva, 2023; Bassan et al., 2024; Blanc et al., 2021; Amir et al., 2024; Bassan et al., 2025; Adolfi et al., 2024; Barceló et al., 2025b; Calautti et al., 2025; Ordyniak et al., 2023).

3 Complexity Classes, Models, and Distributions

In this section, we introduce the general preliminary notations used throughout our paper, including those related to model types, distributional assumptions, and complexity classes.

3.1 Computational Complexity Classes

In this work, we assume that readers are familiar with standard complexity classes, including polynomial time (PTIME) and non-deterministic polynomial time (NP and coNP). We also discuss the complexity class NTIME, which refers to the set of decision problems solvable by a non-deterministic Turing machine within a specified time bound, such as polynomial time. Additionally, we cover the class #P, which counts the number of accepting paths of a non-deterministic Turing machine and can be seen as the "counting" counterpart of NP. It is known that PTIME is contained within NP, coNP, NTIME, and #P, but it is widely believed that these containments are strict, i.e., PTIME $\subsetneq$ NP, coNP, NTIME, #P (Arora and Barak, 2009). We use the standard notation

$L_1 \preceq_P L_2$ to indicate that a polynomial-time reduction exists from the computational problem L_1 to L_2.

3.2 Models

In this subsection, we describe the families of model types used throughout the paper, including decision trees (`DT`), tree ensembles for classification (`ENS-DT`$_{\mathtt{C}}$) and regression (`ENS-DT`$_{\mathtt{R}}$), linear regression models (`LIN`$_{\mathtt{R}}$), as well as neural networks (`NN-SIGMOID` and `RNN-ReLU`).

Decision trees (`DT`). We define a *decision tree* (DT) as a directed acyclic graph representing a graphical model for a discrete function $f : \mathcal{X} \to \mathbb{R}$ in regression tasks, or $f : \mathcal{X} \to [c]$ for classification tasks (where $c \in \mathbb{N}$ is the number of classes). We assume that each input $\mathbf{x}_i$ can take a bounded number of discrete values, limited by some k. This graph encodes the function as follows:

1. Each internal node v is associated with a unique binary input feature from the set $\{1, \ldots, n\}$;

2. Every internal node v has up to k outgoing edges, corresponding to the values $[k]$ which are assigned to v;

3. In the DT, each variable is encountered no more than once on any given path α;

4. Each leaf node is labeled as one of the classes in $[c]$ (for classification tasks) or some $c \in \mathbb{R}$ for regression tasks.

Thus, assigning a value to the inputs $\mathbf{x} \in \mathcal{X}$ uniquely determines a specific path α from the root to a leaf in the DT. The function $f(\mathbf{x})$ is assigned either some $i \in [c]$ or some $i \in \mathbb{R}$ (depending on classification or regression tasks). The size of the DT, denoted as $|f|$, is measured by the total number of edges in its graph. To allow flexibility of the modeling of the DT, we permit different varying orderings of the input variables $\{1, \ldots, n\}$ across any two distinct paths, α and α'. This ensures that no two nodes along any single path α share the same label. We note that in some of the proofs provided in this work, we simplify them by assuming that $k := 2$ or in other words that each feature is defined over binary feature assignments rather than discrete ones. However, this assumption is only for the sake of simplifying the proofs, and our proofs hold also for a general k as well.

Decision tree ensembles (`ENS-DT`$_{\mathtt{R}}$, `ENS-DT`$_{\mathtt{C}}$). There are various well-known architectures for tree ensembles. While these models typically differ in their training processes, our work focuses on post-hoc interpretation, so we emphasize the distinctions in the inference phase rather than the training phase. Specifically, our analysis targets ensemble families that use weighted-voting methods during inference. This includes tree ensembles based on boosting, such as XGBoost. Additionally, in cases where all weights are equal, our formalization also covers majority-voting tree ensembles, such as those used in common bagging techniques like Random Forests. Our approach applies to both tree ensembles for regression tasks (denoted as `ENS-DT`$_{\mathtt{R}}$) and classification tasks (denoted as `ENS-DT`$_{\mathtt{C}}$). We make this distinction explicitly, as we will demonstrate that the complexity results differ for classification and regression ensembles.

Conceptually, an ensemble tree model $\mathcal{T} \in$ `ENS-DT`$_{\mathtt{R}}$ is constructed as a linear combination of DTs. Formally, $\mathcal{T}$ is parametrized by the tuple $\{T_i\}_{i \in [m]}, \{w_i\}_{i \in [m]}$ where $\{T_i\}_{i \in [m]}$ is a collection of DTs (referred to as an ensemble), and $\{w_i\}_{i \in [m]}$ is a set of real numbers. The model $\mathcal{T}$ is used for regression tasks and the function that it computes is given as:

$$f_{\mathcal{T}}(x_1, \ldots, x_n) := \sum_{i=1}^{m} w_i \cdot f_{T_i}(x_1, \ldots, x_n) \tag{6}$$

Aside from regression tasks, ensemble trees are also commonly used for classification tasks. In this context, each decision tree $\mathcal{T}$ is a classification tree (as defined earlier for DTs) that assigns a class label $i \in [c]$. By using the formulation from Equation 7, we can limit the following sum to only those decision trees relevant to class i, and obtain a corresponding weight $f_{\mathcal{T}}^i$. The class assigned by the ensemble will be the one with the highest value of $f_{\mathcal{T}}^i$. It's worth noting that if all weights are taken to be equal, this mirrors majority voting classification, as seen in random forest classifiers. Furthermore, since our proofs in this context only address *hardness*, for simplicity, we assume the number of classes $c = 2$, meaning the random forest classifier is binary. Clearly, the hardness results for this case also apply to the more general multiclass scenario. In this specific case, the given formalization can be restated as:

$$f_{\mathcal{T}}(x_1, \ldots, x_n) := \text{step}(\sum_{i=1}^{m} w_i \cdot f_{T_i}(x_1, \ldots, x_n)) \tag{7}$$

where the step function is defined as: $\text{step}(\mathbf{x}) = 1 \iff \mathbf{x} \geq 0$. We refer to the class of classification tree ensembles as `ENS-DT`$_C$.

Neural Networks (`NN-SIGMOID`, `RNN-ReLU`). Here, we define the general neural network architecture used throughout this work, followed by a specific definition of the (non-recurrent) sigmoidal neural network (`NN-SIGMOID`) referenced in the paper. We note that the definition of recurrent neural networks with ReLU activations (`RNN-ReLU`) mentioned in the paper was provided in the main text, and hence it is not explicitly redefined here. We denote a neural network f with $t-1$ hidden layers ($f^{(j)}$, where j ranges from 1 to $t-1$), and a single output layer ($f^{(t)}$). We observe that we can assume a single output layer since $f^{(t)}$ will yield a value in $\mathbb{R}$ for regression tasks, or it will be applied for binary classification (we will later justify why assuming binary classification is sufficient, and the correctness results will extend to multi-class classification as well). The layers of f are defined recursively — each layer $f^{(j)}$ is computed by applying the activation function $\sigma^{(j)}$ to the linear combination of the outputs from the previous layer $f^{(j-1)}$, the corresponding weight matrix $W^{(j)}$, and the bias vector $b^{(j)}$. This is represented as:

$$f^{(j)} := \sigma^{(j)}(f^{(j-1)}W^{(j)} + b^{(j)}) \tag{8}$$

Where $f^{(j)}$ is computed for each j in $\{1, \ldots, t\}$. The neural network includes t weight matrices ($W^{(1)}, \ldots, W^{(t)}$), t bias vectors ($b^{(1)}, \ldots, b^{(t)}$), and t activation functions ($\sigma^{(1)}, \ldots, \sigma^{(t)}$). The function f is defined to output $f := f^{(t)}$. The initial input layer $f^{(0)}$ serves as the model's input. The dimensions of the biases and weight matrices are specified by the sequence of positive integers $d_0, \ldots, d_t$. We specifically focus on weights and biases that are rational numbers, represented as $W^{(j)} \in \mathbb{Q}^{d_{j-1} \times d_j}$ and $b^{(j)} \in \mathbb{Q}^{d_j}$, which are parameters optimized during training. Clearly, it holds that $d_0 = n$. For regression tasks, the output of f (i.e., d_t) will be in $\mathbb{R}$. For classification tasks, we assume multiple outputs, where typically a softmax function is applied to choose the class $i \in [c]$ with the highest value. However, since the proofs for both `NN-SIGMOID` and `RNN-ReLU` are *hardness* proofs, we assume for simplicity that in classification cases, we use a binary classifier, i.e., $c = 2$. Thus, hardness results will clearly apply to the multi-class setting as well. Therefore, for binary classification, it follows that $d_0 = n$ and $d_t = 1$.

The activation functions $\sigma^{(i)}$ that we focus on in this work are either the ReLU activation function, defined as $\text{ReLU}(x) := \max(0, x)$ (used for the RNN-ReLU model), or the sigmoid activation function, defined as $\text{Sigmoid} := \frac{1}{1+e^{-x}}$. In the case of our binary classification assumption, we assume that the last output layer can either use a sigmoid function or (without loss of generality) a step function for the final layer activation. Here, we denote $\text{step}(x) = 1 \iff x \geq 0$.

Linear Regression Models (`LIN`$_R$). A linear regression model corresponds to a single-layer neural network in the regression context, as defined in the formalization above, where $t = 1$. The linear model is described by the function $f(\mathbf{x}) := (\mathbf{w} \cdot \mathbf{x}) + b$, with $b \in \mathbb{Q}$ and $\mathbf{W} \in \mathbb{Q}^{n \times d_1}$. Furthermore, we introduce an alternative renormalization of linear regression models, which will be helpful for the technical developments in Section 5. Specifically, a linear regression model over the finite set $\mathbb{D} := [m_1] \times \ldots [m_n]$, where $\{m_i\}_{i \in [n]}$ represents a set of integers, can be parametrized as $\langle \{w_{i,d}\}_{i \in [n], d \in \mathbb{D}_i}, b \rangle$, where $\{w_{i,d}\}_{i \in [n], d \in m_i}$ is a collection of rational numbers and $b \in \mathbb{Q}$. The function computed by f is given as:

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{d \in \mathbb{D}_i} w_{i,d} \cdot I(x_i = d) + b$$

3.3 Distributions

In this subsection, we will formally define various distributions relevant to this work, which have been referenced throughout the main paper or in the appendix. We note that the definition of hidden Markov models (`HMM`), the

most significant class of distributions examined in this study, is not included here as it was already provided in the main text.

Independent Distributions (`IND`). The family `IND` is the most elementary family of distributions based on the assumption of probabilistic independence of all random variables (RVs) involved in the model. Formally, given some set of discrete values $[k]$, we can describe a probability function $p : [n] \times [k] \to [0, 1]$. For example, $p(1, 2) = \frac{1}{2}$, implies that the probability of feature $i = 1$, to be set to the value $k = 2$ is $\frac{1}{2}$. Then we can define $\mathcal{D}_p$ as an independent distribution over $\mathcal{X}$ iff:

$$\mathcal{D}_p(\mathbf{x}) := \Big(\prod_{i \in [n], \mathbf{x}_i = j} p(i, j) \Big) \tag{9}$$

It is evident that the uniform distribution is a specific instance of $\mathcal{D}_p$, obtained by setting $p(i, j) := \frac{1}{|k|}$ for every $i \in [n]$ and $j \in [k]$.

Empirical Distributions (`EMP`). The empirical distribution provides a practical way of estimating probabilities from a finite dataset. Given a set of M samples, each represented as a vector in $\{0, 1\}^N$, the empirical distribution assigns a probability to each possible vector x in the space. This probability is simply the proportion of samples in the dataset that are equal to x. Formally, for a dataset $\mathcal{D} = \{x_1, \ldots, x_M\}$, the empirical distribution $P_{\mathcal{D}}(x)$ is defined as the frequency of occurrences of the vector x in the dataset, normalized by the total number of samples, i.e.:

$$P_{\mathcal{D}}(x) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} I(x_i = x)$$

Naive Bayes Model (`NB`). A naive Bayes model is a latent probabilistic model that involves $n + 1$ random variables (RVs), denoted as $(X_1, \ldots, X_n, Y)$, where $\{X_i\}_{i \in [n]}$ represent the n observed RVs, and Y is an unobserved (latent) RV. The key probabilistic assumption of naive Bayes models is that the observed RVs are conditionally independent given the value of the latent variable Y. More formally, a model $M \in \overrightarrow{\text{NB}}$ over n RVs is specified by the parameters $\langle \pi, \{P_i\}_{i \in [n]} \rangle$, where:

- π is a probability distribution over the domain value of the latent variable Y ($dom(Y)$),
- For $i \in [n]$, $P_i \in \mathbb{R}^{n \times dom(Y)}$ is a stochastic matrix.

The marginal probability distribution computed by M is given as:

$$P_M(x_1, \ldots x_n, y) = \pi(y) \prod_{i=1}^{n} P_i[x_i, y]$$

Markovian Distributions (`MARKOV`). A (stationary) Markovian distribution $M \in \text{MARKOV}$ over an alphabet Σ is represented by the tuple α, T where π is a probability distribution over Σ and T is a stochastic matrix in $\mathbb{R}^{|\Sigma| \times |\Sigma|}$ [4]. A Markovian model M computes a probability distribution over Σ^{∞}. The probability of generating a given sequence $w \in \Sigma^*$ as a prefix by a Markovian model $M = \pi, T$ is given as:

$$P_M^{(|w|)}(w) = \pi[w_1] \cdot \prod_{i=2}^{|w|} |w| T[w_{i-1}, w_i]$$

where for a given integer n, $P_M^{(n)}$ designates the probability distribution over Σ^n interpreted as the probability of generating a prefix of length n.

Analogous to HMMs, one can define a family of models representing the non-sequential counterpart of Markovian models, which we'll refer to as $\overrightarrow{\text{MARKOV}}$. A model $\overrightarrow{M} \in \overrightarrow{\text{MARKOV}}$ defines a probability distribution over Σ^n for $n \geq 1$, and parameterized by the tuple $\pi, \alpha, \{T_i\}_{i \in [n]}$, where:

- π is a permutation from $[n]$ to $[n]$.

---

[4] A matrix $A \in \mathbb{R}^{n \times m}$ is said to be stochastic if each row vector corresponds to a probability distribution over $[m]$.

- α defines a probability distribution over $[n]$ (also called the initial state vector),
- For each $i \in [n]$, T_i is a stochastic matrix over $\mathbb{R}^{|\Sigma| \times |\Sigma|}$

The procedure of generating the tuple $(x_1, \ldots, x_n)$ (where $x_i \in \Sigma$) by $\overrightarrow{\mathsf{M}}$ can be described recursively as follows:

1. **Generation of the first element of the sequence.** Generate $x_{\pi(1)}$ with probability $\alpha[x_{\pi(1)}]$.

2. **Generation of the (i+1)-th element.** For $i \in [n-1]$, the probability of generating the element $x_{\pi(i+1)}$ given that $x_{\pi(i)}$ is generated is equal to $T[x_{\pi(i)}, x_{\pi(i+1)}]$

4 The Tractability of computing SHAP for the class of WAs: Proofs of Intermerdiary Results

In this section of the appendix, we provide detailed proofs of intermediary mathematical statements to prove the tractability of computing different SHAP variants on the class of WAs (Theorem 1). Specifically, we shall provide proofs of three intermediary results:

1. Proposition 1 that states the computational efficiency of implementing the projection operation.

2. Lemma 2, which demonstrates how the computation of both local and global Interventional and Basleine SHAP for the family of WAs under distributions modeled by HMMs can be reduced to performing operations over N-Alphabet WAs.

3. Proposition 2, which asserts that the construction of N-Alphabet WAs can be achieved in polynomial time, thus enabling the polynomial-time algorithmic construction for both `LOC-I-SHAP(WA,HMM)` and `GLO-I-SHAP(WA,HMM)`.

4.1 Terminology and Technical Background

The proof of Theorem 1 will rely on certain technical tools that were not introduced in the main paper. This initial section is devoted to providing the technical background upon which the proofs of various results presented in the rest of this section rely.

The Kronecker product. The Kronecker product between $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{k \times l}$, denoted $A \otimes B$, is a matrix in $\mathbb{R}^{(n \cdot k) \times (m \cdot l)}$ constructed as follows

$$A \otimes B = \begin{bmatrix} a_{1,1} \cdot B & a_{1,2} \cdot B & \ldots & a_{1,m} \cdot B] \\ a_{2,1} \cdot B & a_{2,2} \cdot B & \ldots & a_{2,m} \cdot B] \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} \cdot B & a_{n,2} \cdot B & \ldots & a_{n,m} \cdot B \end{bmatrix}$$

where, for $(i,j) \in [n] \times [m]$, $a_{i,j}$ corresponds to the element in the i-th row and the j-th column of A. A property of the Kronecker product of matrices that will be utilized in several proofs in the appendix is the *mixed-product* property:

Property 1. *Let A, B, C, D be four matrices with compatible dimensions. We have that:*

$$(A \cdot B) \otimes (C \cdot D) = (A \otimes C) \cdot (B \otimes D)$$

N-Alphabet Deterministic Finite Automata (N-Alphabet DFAs). In subsection 2, we shall employ a sub-class of N-Alphabet WAs more adapted to model binary functions (i.e. functions whose output domain is $\{0,1\}$)in the proof of Proposition 2. The class of N-Alphabet DFAs can be seen as a generalization of the classical family of Deterministic Finite Automata for the multi-alphabet case. N-Alphabet DFAs are formally defined as follows:

Definition 1. *A N-Alphabet DFA A is represented by a tuple $\langle Q, q_{init}, \delta, F \rangle$ where:*

- *Q is a finite set corresponding to the state space,*

(a) A 1-Alphabet DFA:
$\Sigma_1 = \{a, b, c\}$

(b) A 2-Alphabet DFA:
$\Sigma_1 = \{a, b, c\}, \ \Sigma_2 = \{0, 1\}$

(b) A 3-Alphabet DFA :
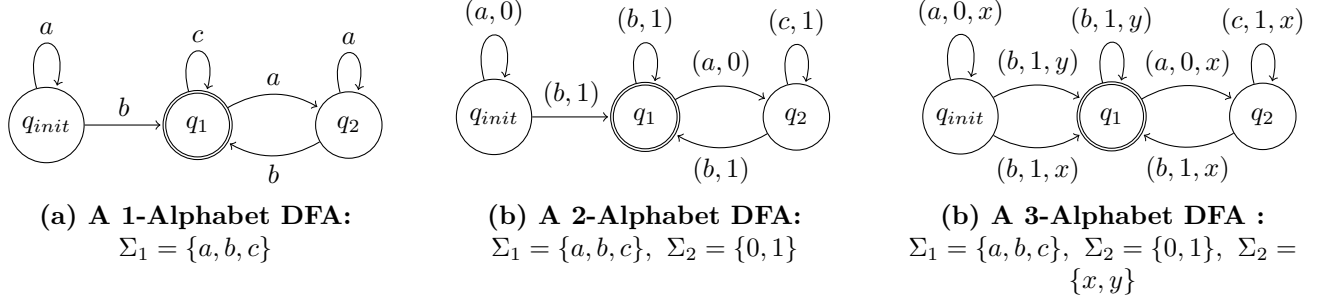$\Sigma_1 = \{a, b, c\}, \ \Sigma_2 = \{0, 1\}, \ \Sigma_2 = \{x, y\}$

Figure 1: A graphical representation of N-Alphabet DFAs. Nodes corresponding to final states are represented by double circles.

- $q_{init} \in Q$ is called the initial state.
- δ, called the transition function, is a partial map from $Q \times \Sigma_1 \times \ldots \times \Sigma_N$ to Q
- $F \subseteq Q$ is called the final state set

Figure 1 illustrates the graphical representation of some N-Alphabet DFAs. Analogous to N-Alphabet WAs, we shall use the terminology *DFA*, instead of 1-Alphabet DFA for $N = 1$.

To show how N-Alphabet DFAs compute (binary) functions, we need to introduce the notion of a *path*. For a N-Alphabet DFA $A = \langle Q, q_{init}, \delta, F \rangle$ over $\Sigma_1 \times \ldots \times \Sigma_N$, a valid path in A is a sequence $P = (q_1, \sigma_1^{(1)}, \ldots \sigma_1^{(N)}) \ldots (q_L, \sigma_1^{(L)}, \ldots \sigma_L^{(N)}) q_{L+1}$ in $(Q \times \Sigma_1 \times \ldots \times \Sigma_N)^* \times Q$ such that for any $i \in [L]$, $\delta(q_i, (\sigma_i^{(1)}, \ldots \sigma_i^{(N)})) = q_{i+1}$. Given this definition of a valid path, the N-Alphabet A for a given tuple of sequences $(w^{(1)}, \ldots, w^{(N)}) \in \Sigma_1^* \times \ldots \times \Sigma_N^*$ such that $|w^{(1)}| = \ldots = |w^{(N)}| = L$ if and only if there exists a valid path $(q_1, w_1^{(1)}, \ldots w_1^{(N)})(q_2, w_2^{(1)}, \ldots, w_2^{(N)}) \ldots (q_L, w_L^{(1)}, \ldots, w_L^{(N)}) q_{L+1}$ such that $q_1 = 1$ and $q_{L+1} \in F$. For instance, the sequence *abab* of the 1-Alphabet DFA in Figure 1 is labeled by 1. Indeed, the valid path $(q_{init}, a)(q_{init}, b)(q_1, a)(q_2, b)q_1$ satisfies these conditions.

Linear Algebra operations over N-Alphabet WAs. The development of polynomial-time algorithms for computing various SHAP variants for the class of WAs in section 2 of the main paper is based on two operations: the projection and the Kronecker product operations. In the following section of the appendix, we will provide a proof of the tractability of their construction.

In addition to these two operators, linear algebra operations over N-Alphabet WAs have also been implicitly utilized in the construction. The closure of 1-Letter WAs under linear algebra operations is a well-established result in the WA literature (Droste et al., 2009). For completeness, we offer a brief discussion below on how linear algebra operations can be extended to handle multiple alphabets, including their construction and the associated complexity results:

- *The addition operation:* Given two N-Alpphabet WAs $T = \langle \alpha, \{A_{\sigma_1, \ldots, \sigma_N}\}_{(\sigma_1, \ldots, \sigma_N) \in \Sigma_1 \times \ldots \times \Sigma_N}, \beta \rangle$ and $T' = \langle \alpha', \{A'_{\sigma_1, \ldots, \sigma_N}\}_{(\sigma_1, \ldots, \sigma_N) \in \Sigma_1 \times \ldots \times \Sigma_N}, \beta' \rangle$ over $\Sigma_1 \times \ldots \times \Sigma_N$, the N-Alphabet WA, denoted $T + T'$, that computes the function:

$$f_{T+T'}(w^{(1)}, \ldots, w^{(N)}) = f_T(w^{(1)}, \ldots, w^{(N)}) + f_{T'}(w^{(1)}, \ldots, w^{(N)})$$

is parametrized as follows:

$$\left\langle \begin{pmatrix} \alpha \\ \alpha' \end{pmatrix}, \left\{ \begin{pmatrix} A_{\sigma_1, \ldots, \sigma_N} & \mathbf{O} \\ \mathbf{O} & A'_{\sigma_1, \ldots, \sigma_N} \end{pmatrix} \right\}_{(\sigma_1, \ldots, \sigma_N) \in \Sigma_1 \times \ldots \times \Sigma_N}, \begin{pmatrix} \beta \\ \beta' \end{pmatrix} \right\rangle$$

The running time of the addition operation is $O(|\Sigma|^N \cdot (\texttt{size}(T) + \texttt{size}(T')))$. The size of the resulting N-Alphabet WA is equal to $O(\texttt{size}(T) + \texttt{size}(T'))$.

- *Multiplication by a scalar.* Let $T = \langle \alpha, \{A_{\sigma_1, \ldots, \sigma_N}\}_{(\sigma_1, \ldots, \sigma_N) \in \Sigma_1 \times \ldots \times \Sigma_N}, \beta \rangle$ be an N-Alphabet WA over $\Sigma_1 \times \ldots \times \Sigma_N$, and a real number $C > 0$, the N-Alphabet WA, denoted $C \cdot T$ that computes the function $f_{C \cdot T}(w^{(1)}, \ldots, w^{(N)}) = C \cdot f_T(w^{(1)}, \ldots, w^{(N)})$ is parametrized as: $\langle C \cdot \alpha, \{A_{\sigma_1, \ldots, \sigma_N}\}_{(\sigma_1, \ldots, \sigma_N) \in \Sigma_1 \times \ldots \times \Sigma_N}, \beta \rangle$.

It is easy to see that the construction of the N-Alphabet WA $C \cdot T$ runs in $O(1)$ time, and has size equal to the size of T.

Table 2: Operations on N-Alphabet WAs, along with their time complexity and output size. The "In 1" and "In 2" (respectively "Out") columns indicate the number of alphabets in the input N-Alphabet WAs for each operation. The "Time" column specifies the time complexity of executing the operation, and the "Output size" column denotes the size of the resulting N-Alphabet WA after the operation is applied. By convention, a value of 0 in the "Output" and "Output size" columns indicates a scalar result.

	In 1	In 2	Out	Time	Output size
Addition $(+)$	N	N	N	$O(\max_{i \in [N]} |\Sigma_i|^N \cdot (\texttt{size}(\text{in}_1) + \texttt{size}(\text{in}_2)))$	$O(\texttt{size}(\text{in}_1) + \texttt{size}(\text{in}_2))$
Scalar Multiplication	N	0	N	$O(1)$	$O(\texttt{size}(\text{in}_1))$
Π_0	1	$-$	0	$O(|\Sigma_1| \cdot \texttt{size}(\text{in}_1)^2 \cdot n)$	0
Π_1	1	1	0	$O(|\Sigma_1| \cdot (|\texttt{size}(\text{in}_1) \cdot \texttt{size}(\text{in}_2))^2 \cdot n)$ [5]	0
Π_i $(i \geq 2)$	1	N	$N-1$	$O(\max_{i \in [N]} |\Sigma_i|^N \cdot \texttt{size}(\text{in}_1) \cdot \texttt{size}(\text{in}_2))$	$O(\texttt{size}(\text{in}_1) \cdot \texttt{size}(\text{in}_2))$
$\otimes$	N	N	N	$O(\max_{i \in [N]} |\Sigma_i|^N \cdot \texttt{size}(\text{in}_1) \cdot \texttt{size}(\text{in}_2))$	$O(\texttt{size}(\text{in}_1) \cdot \texttt{size}(\text{in}_2))$

A summary of the running time complexity and the size of outputted WAs by all operations over N-Alphabet encountered in this work can be found in Table 2.

4.2 Proof of proposition 1

Recall the statement of Proposition 1:

Proposition. *Assume that $N = O(1)$. Then, the projection and the Kronecker product operations between N-Alphabet WAs can be computed in polynomial time.*

The following result provides an implicit construction of these two operators which implicitly induces the result of Proposition 1:

Proposition 1. *Let N be an integer, and $\{\Sigma_i\}_{i \in [N]}$, a collection of finite alphabets. We have:*

1. *The projection operation: Fix an integer $i \in [N]$. Let $A = \langle \alpha, \{A_\sigma\}_{\sigma \in \Sigma}, \beta \rangle$ be a WA over Σ_i, $T = (\alpha', \{A'_{\sigma_1,\dots,\sigma_N}\}_{(\sigma_1,\dots,\sigma_N) \in \Sigma_1 \times \dots \times \Sigma_N}, \beta')$ be an N-Alphabet WA over $\Sigma_1 \times \dots \times \Sigma_N$, and A be a WA over Σ_i. The projection of A over T at index i, denoted $\Pi_i(A, T)$, is parametrized as:*

$$\Pi_i(A, T) := \langle \Sigma_1 \times \dots \times \Sigma_{i-1} \times \Sigma_i \times \dots \times \Sigma_N, \alpha \otimes \alpha',$$
$$\{ \sum_{\sigma_i \in \Sigma_i} A_{\sigma_i} \otimes A'_{\sigma_1,\dots,\sigma_{i-1},\sigma_{i+1},\dots,\sigma_N} \}_{(\sigma_1,\dots,\sigma_{i-1},\sigma_i,\dots,\sigma_N) \in \Sigma_1 \times \dots \times \Sigma_{i-1} \times \Sigma_{i+1} \times \dots \times \Sigma_N}, \beta \otimes \beta' \rangle$$

2. *The Kronecker product operation: Let $T = \langle \alpha, \{A_{\sigma_1,\dots,\sigma_N}\}_{(\sigma_1,\dots,\sigma_N) \in \Sigma_1 \times \dots \times \Sigma_N}, \beta \rangle$ and $T' = \langle \alpha', \{A'_{\sigma_1,\dots,\sigma_N}\}_{(\sigma_1,\dots,\sigma_N) \in \Sigma_1 \times \dots \times \Sigma_N}, \beta' \rangle$ be two N-Alphabet WAs over $\Sigma_1 \times \dots \times \Sigma_N$. The Kronecker product between T and T', $T \otimes T'$, is parametrized as:*

$$T \otimes T' = \langle \alpha \otimes \alpha', \sum_{\sigma \in \Sigma} A_{\sigma_1,\dots,\sigma_N} \otimes A'_{\sigma_1,\dots,\sigma_N}, \beta \otimes \beta' \rangle$$

Proof. Let N be an integer, and $\{\Sigma_i\}_{i \in [N]}$, a collection of finite alphabets.

1. For the projection operation: Fix $i \in [N]$. Let $T = \langle \alpha', \{A'_{\sigma_1,\dots,\sigma_N}\}_{(\sigma_1,\dots,\sigma_N) \in \Sigma_1 \times \dots \times \Sigma_N}, \beta' \rangle$ be an N-Alphabet WA over $\Sigma_1 \times \dots \times \Sigma_N$, and let A be a WA over Σ_i. Let there be some $(w^{(1)}, \dots, w^{(N)}) \in \Sigma_1^* \times \dots \Sigma_N^*$, such that $|w^{(1)}| = \dots = |w^{(N)}| = L$. We have:

$$f_{\Pi_i(A,T)}(w^{(1)}, \ldots, w^{(i-1)}, w^{(i+1)}, w^{(N)}) = \sum_{w \in \Sigma_i^L} f_A(w) \cdot f_T(w^{(1)}, \ldots, w^{(i-1)}, w, w^{(i+1)}, w^{(N)})$$

$$= \sum_{w \in \Sigma_i^L} \left(\alpha^T \cdot \prod_{j=1}^{L} A_{w_j} \cdot \beta \right) \cdot \left(\alpha'^T \cdot \prod_{j=1}^{L} A'_{w_j^{(1)}, \ldots, w_j^{(i)}, \ldots w_j^{(N)}} \cdot \beta' \right)$$

$$= \sum_{w \in \Sigma_i^L} (\alpha \otimes \alpha')^T \cdot \left[\prod_{j=1}^{L} A_{w_j} \otimes A'_{w_j^{(1)}, \ldots, w_j^{(i)} \ldots w_j^{(N)}} \right] \cdot (\beta \otimes \beta')$$

$$= (\alpha \otimes \alpha')^T \cdot \prod_{j=1}^{L} \left(\sum_{\sigma \in \Sigma_i} A_\sigma \otimes A'_{w_j^{(1)}, \ldots, \sigma, w_j^{(N)}} \right) \cdot (\beta \otimes \beta')$$

where the third equality is obtained using the mixed-product property of the Kronecker product between matrices.

2. The Kronecker product operation: Let $T = \langle \alpha, \{A_{\sigma_1, \ldots, \sigma_N}\}_{(\sigma_1, \ldots, \sigma_N) \in \Sigma_1 \times \ldots \times \Sigma_N}, \beta \rangle$ and $T' = \langle \alpha', \{A'_{\sigma_1, \ldots, \sigma_N}\}_{(\sigma_1, \ldots, \sigma_N) \in \Sigma_1 \times \ldots \times \Sigma_N}, \beta' \rangle$ be two N-Alphabets WA over $\Sigma_1 \times \ldots \times \Sigma_N$. Let $(w^{(1)}, \ldots, w^{(N)}) \in \Sigma_1^* \times \ldots \Sigma_N^*$ such that $|w^{(1)}| = \ldots = |w^{(N)}| = L$. We have:

$$f_{T \otimes T'}(w^{(1)}, \ldots, w^{(N)}) = f_T(w^{(1)}, \ldots, w^{(N)}) \cdot f_{T'}(w^{(1)}, \ldots, w^{(N)})$$

$$= (\alpha^T \cdot \prod_{j=1}^{L} A_{w_j^{(1)}, \ldots w_j^N} \cdot \beta) \cdot (\alpha'^T \cdot \prod_{j=1}^{L} A'_{w_j^{(1)}, \ldots w_j^N} \cdot \beta')$$

$$= (\alpha \otimes \alpha')^T \cdot \prod_{j=1}^{L} \left(A_{w_j^{(1)}, \ldots w_j^N} \otimes A'_{w_j^{(1)}, \ldots w_j^N} \right) \cdot (\beta \otimes \beta')$$

where the last equality is obtained using the mixed-product property of the Kronecker product between matrices.

$\square$

4.3 Proof of Lemma 2

In this segment, we provide the proof of the main lemma of section 2:

Lemma. *Fix a finite alphabet Σ. Let f be a WA over Σ, and consider a sequence $(w, w^{reff}) \in \Sigma^* \times \Sigma$ (representing an input and a basline $\boldsymbol{x}, \boldsymbol{x}^{reff} \in \mathcal{X}$) such that $|w| = |w^{reff}|$. Let $i \in [|w|]$ be an integer, and $\mathcal{D}_P$ be a distribution modeled by an HMM over Σ. Then:*

$$\phi_i(f, w, i, \mathcal{D}_P) =$$
$$\Pi_1(A_{w,i}, \Pi_2(\mathcal{D}_P, \Pi_3(f, T_{w,i}) - \Pi_3(f, T_w)));$$
$$\Phi_i(f, i, n, \mathcal{D}_P) =$$
$$\Pi_0(\Pi_2(\mathcal{D}_P, A_{i,n} \otimes \Pi_2(\mathcal{D}_P, \Pi_3(f, T_i) - \Pi_3(f, T))));$$
$$\phi_b(f, w, i, w^{reff}) =$$
$$\Pi_1(A_{w,i}, \Pi_2(f_{w^{reff}}, \Pi_3(f, T_{w,i}) - \Pi_3(f, T_w)));$$
$$\Phi_b(f, i, n, w^{reff}, \mathcal{D}_P) =$$
$$\Pi_0(\Pi_2(\mathcal{D}_P, A_{i,n} \otimes \Pi_2(f_{w^{reff}}, \Pi_3(f, T_i) - \Pi_3(f, T))))$$

where:

- $A_{w,i}$ *is a 1-Alphabet WA over $\Sigma_\#$ implementing the uniform distribution over coalitions excluding the feature i (i.e., $f_{A_{w,i}} = \mathcal{P}_i^w$);*
- T_w *is a 3-Alphabet WA over $\Sigma_\# \times \Sigma \times \Sigma$ implementing the function: $g_w(p, w', u) := I(\boldsymbol{do}(p, w', w) = u)$.*
- $T_{w,i}$ *is a 3-Alphabet WA over $\Sigma_\# \times \Sigma \times \Sigma$ implementing the function: $g_{w,i}(p, w', u) := I(\boldsymbol{do}(\boldsymbol{swap}(p, w_i, i), w', w) = u)$.*

- T is a *4-Alphabet* WA *over* $\Sigma_\# \times \Sigma \times \Sigma \times \Sigma$ *given as:* $g(p, w', u, w) := g_w(p, w', u)$.

- T_i is a *4-Alphabet* WA *over* $\Sigma_\# \times \Sigma \times \Sigma \times \Sigma$ *given as:* $g_i(p, w', u, w) := g_{w,i}(p, w', u)$.

- $A_{i,n}$ is a *2-Alphabet* WA *over* $\Sigma_\# \times \Sigma$ *implementing the function:* $g_{i,n}(p, w) := I(p \in \mathcal{L}_i^w) \cdot \mathcal{P}_i^w(p)$, *where* $|w| = |p| = n$.

- $f_{w^{reff}}$ is an HMM *such that the probability of generating* w^{reff} *as a prefix is equal to* 1.

Proof. We will prove the complexity results specifically for the cases involving either local or global *Interventional* SHAP. The corresponding proof for local and global Baseline SHAP can be derived by following the same approach as in the interventional case, with the sole modification of replacing $\mathcal{D}_p$ with the HMM that models the empirical distribution induced by the reference instance w^{reff}.

Fix a finite alphabet Σ. Let f be a WA over Σ, $w \in \Sigma^*$ a sequence, $i \in [|w|]$ an integer, and $\mathcal{D}_P$ an HMM. Define $A_{w,i}$ as a WA and T_w, $T_{w,i}$ as two 3-Alphabet WAs as stated in the lemma. We divide the following proof into two parts. The first part addresses the local interventional SHAP version, while the second part covers the global version.

1. For *local Interventional SHAP* we have that:

$$\phi_i(f, w, i, \mathcal{D}_P) = \mathbb{E}_{p \sim \mathcal{P}_i^w}\left[V_I(w, \texttt{swap}(p, w_i, i), \mathcal{D}_P) - V_I(w, p, \mathcal{D}_P)\right]$$

$$= \sum_{p \in \Sigma_\#^{|w|}} f_{A_{w,i}}(p) \left[\sum_{w' \in \Sigma^w} \mathcal{D}_P(w') \cdot \left[f(\texttt{do}(\texttt{swap}(p, w'_i, i), w', w)) - f(\texttt{do}(p, w', w)) \right] \right] \quad (10)$$

Note that for any $p \in \Sigma_\#^{|w|}$ and $(w', u) \in \Sigma^{|w|} \times \Sigma^{|w|}$, we have:

$$f(\texttt{do}(\texttt{swap}(p, w'_i, i), w', w)) = \sum_{u \in \Sigma^{|w|}} f(u) \cdot g_{w,i}(p, w', u) = f_{\Pi_3(f, T_{w,i})}(p, w') \quad (11)$$

and,

$$f(\texttt{do}(p, w', w)) = \sum_{u \in \Sigma^{|w|}} f(u) \cdot g_w(p, w', w) = f_{\Pi_3(f, T_w)}(p, w') \quad (12)$$

where $g_{w,i}$ and g_w are defined implicitly in the body of the lemma statement.

By plugging equations (11) and (12) in Equation (10), we obtain:

$$\phi_i(f, w, i, \mathcal{D}_P) = \sum_{p \in \Sigma_\#^{|w|}} f_{A_{w,i}}(p) \left[\sum_{w' \in \Sigma^{|w|}} \mathcal{D}_P(w') \cdot \left[f_{\Pi_3(M, T_{w,i})}(p, w') - f_{\Pi_3}(M, T_w)(p, w') \right] \right]$$

To ease exposition, we employ the symbol $\tilde{T}$ to refer to the intermediary 2-Alphabet WA over $\Sigma_\# \times \Sigma$ defined as:

$$\tilde{T} \stackrel{\text{def}}{=} \Pi_3(M, T_{w,i}) - \Pi_3(M, T_w)$$

Then, we have:

$$\phi_i(f, w, i, \mathcal{D}_P) = \sum_{p \in \Sigma_\#^{|w|}} f_{A_{w,i}}(p) \left[\mathcal{D}_P(w') \cdot f_{\tilde{T}}(p, w') \right]$$

$$= \sum_{p \in \Sigma_\#^{|w|}} f_{A_{w,i}}(p) \cdot f_{\Pi_2(\mathcal{D}_P, \tilde{T})}(p)$$

$$= \Pi_1(A_{w,i}, \Pi_2(\mathcal{D}_P, \tilde{T}))$$

2. For *Global Interventional SHAP*, the proof follows the same structure as that of the Local version. We hence have that:

$$\phi_i(f, i, n, P) = \sum_{w \in \Sigma^n} \mathcal{D}_P(w) \cdot \phi_i(f, w, i, \mathcal{D}_P)$$

$$= \sum_{w \in \Sigma^n} \mathcal{D}_P(w) \sum_{p \in \Sigma_\#}^{|w|} f_{A_{w,i}}(p) \left[\sum_{w' \in \Sigma^w} \mathcal{D}_P(w') \cdot [f(\texttt{do}(\texttt{swap}(p, w'_i, i), w', w)) - f(\texttt{do}(p, w', w))] \right]$$

$$\tag{13}$$

Note that for any $p \in \Sigma_\#^n$ and $(w', u, w) \in \Sigma^n \times \Sigma^n \times \Sigma^n$, we have:

$$f(\texttt{do}(\texttt{swap}(p, w'_i, i), w', w)) = \sum_{u \in \Sigma^n} f(u) \cdot g_i(p, w', u, w) = f_{\Pi_3(M, T_i)}(p, w', w) \tag{14}$$

and,

$$f(\texttt{do}(p, w', w)) = \sum_{u \in \Sigma^{|w|}} f(u) \cdot g(p, w', u, w) = f_{\Pi_3(M, T)}(p, w', w) \tag{15}$$

where g_i and g are functions defined implicitly in the lemma statement.

By, again, plugging equations (14) and (15) into the equation 13, we obtain:

$$\phi_i(f, i, n, P) = \sum_{w \in \Sigma^n} \mathcal{D}_P(w) \cdot \sum_{p \in \Sigma_\#^n} f_{A_{i,n}}(p, w) \left[\sum_{w' \in \Sigma^n} \mathcal{D}_P(w') \cdot f_{\tilde{T}}(p, w', w) \right]$$

$$= \sum_{w \in \Sigma^n} \mathcal{D}_P(w) \cdot \sum_{p \in \Sigma_\#^n} f_{A_{i,n}}(p, w) \cdot f_{\Pi_2(\mathcal{D}_P, \tilde{T})}(p, w)$$

$$= \sum_{w \in \Sigma^n} \mathcal{D}_P(w) \sum_{p \in \Sigma_\#^n} f_{A_{i,n} \otimes \Pi_2(\mathcal{D}_P, \tilde{T})}(p, w)$$

$$= \sum_{p \in \Sigma_\#^n} f_{\Pi_2(\mathcal{D}_P, A_{i,n} \otimes \Pi_2(\mathcal{D}_P, \tilde{T}))}(p)$$

$$= \Pi_0 \left(\Pi_2(\mathcal{D}_P, A_{i,n} \otimes \Pi_2(\mathcal{D}_P, \tilde{T})) \right)$$

$\square$

4.4 Proof of proposition 2

In this segment, we shall provide a constructive proof of all machines defined implicitly in Lemma 2. Formally, we shall prove the following:

Proposition. *The N-Alphabet WAs $A_{w,i}$, T_w, $T_{w,i}$, T, T_i, $A_{i,n}$ and the HMM $f_{w^{\text{reff}}}$ can be constructed in polynomial time with respect to $|w|$ and $|\Sigma|$.*

We split the proof of this proposition into four sub-sections. The first sub-section is dedicated to the construction of $A_{w,i}$ and $A_{i,n}$. The second sub-section is dedicated to the construction of $T_{w,i}$ and T_i. The third sub-section is dedicated to the construction of T_i and T. The final subsection treats the construction of $f_{w^{reff}}$. The running time of all these constructions as well as the size of their respective outputs are summarized in Table 3.

4.4.1 The construction of $A_{w,i}$ and $A_{i,n}$

Recall $A_{w,i}$ is a WA over $\Sigma_\#$ that implements the probability distribution:

$$\mathcal{P}_i^{(w)}(p) \stackrel{\text{def}}{=} \frac{1}{|w|} \sum_{k=1}^{|w|} \mathcal{P}_{i,k}^{(w)}(p) \tag{16}$$

Table 3: The summary of complexity results (in terms of the length of the sequence to explain w and the size of the alphabet $|\Sigma|$) for the construction of models in proposition 2

	Running Time complexity	**Output Size**	**Output's Alphabet size (N)**
$A_{w,i}$	$O(|w|^3)$	$O(|w|^3)$	1
$T_{w,i}$	$O(|\Sigma|^3 \cdot |w|)$	$|w|$	3
T_w	$O(|\Sigma|^3 \cdot |w|)$	$|w|$	3
T_i	$O(|w|)$	$O(|w|)$	4
T	$O(1)$	$O(1)$	4
$A_{i,n}$	$O(|\Sigma|^2 \cdot |w|^4)$	$O(|w|^4)$	2
$f_{w^{reff}}$	$O(|w|)$	$O(|w|)$	1

where $\mathcal{P}_{i,k}^{(w)}$ is the uniform distribution over patterns belonging to the following set:

$$\mathcal{L}_{i,k}^{(w)} \stackrel{\text{def}}{=} \{p \in \Sigma_{\#}^{|w|} : \ w \in L_p \wedge |p|_{\#} = k \wedge p_i = \#\}$$

The 2-Alphabet WA $A_{i,n}$ can be seen as a global version of $A_{w,i}$ where the sequence w becomes a part of the input of the automaton. Next, we shall provide the construction for $A_{w,i}$. The construction of $A_{i,n}$ is somewhat similar to that of $A_{w,i}$ and will be discussed later on in this section.

The construction of $A_{w,i}$. Algorithm 2 provides the pseudo-code for constructing $A_{w,i}$. By noting that the target probability distribution $\mathcal{P}_i^{(w)}$ is a linear combination of the set of functions $\mathcal{F} = \{\mathcal{P}_{i,k}^{(w)}\}$ (Equation 16), the strategy of our construction consists at iteratively constructing a sequence of WAs $\{A_{i,k}^{(w)}\}_{k \in [[w]]}$, each of which implements a function $F \in \mathcal{F}$. Thanks to the closure of WAs under linear algebra operations (addition, and multiplication with a scalar) and the tractability of implementing them, one can recover the target WA $A_{w,i}$ using linear combinations of the collection of WAs $\mathcal{A} = \{A_{i,k}^{(w)}\}_{k \in [[w]]}$.

Algorithm 2 Construction of $A_{w,i}$

Input: A sequence $w \in \Sigma^*$, An integer $i \in [[w]]$
Output: A WA $A_{w,i}$
 1: Initialize $A_{w,i}$ as the empty WA
 2: **for** $k = 1 \ldots |w|$ **do**
 3:   Construct a DFA $\bar{A}_{i,k}^{|w|}$ that accepts the language $\mathcal{L}_{i,k}^{(w)}$
 4:   $A_{w,i} \leftarrow A_{w,i} + \frac{1}{|w| \cdot |\mathcal{L}_{i,k}^{|w|}|} \cdot \bar{A}_{i,k}^{|w|}$
 5: **end for**
 6: **return** $A_{w,i}$

The missing link to complete the construction is to show how each element in the set $\mathcal{A}$ is constructed. For a given $(i,k) \in [[w]]^2$, the strategy consists at constructing a DFA that accepts the language $\mathcal{L}_{i,k}^{(w)}$ (denoted $\bar{A}_{i,k}^{|w|}$ in Line 3 of Algorithm 2)). Since the function implemented by $A_{i,k}^{(w)}$ represents the uniform probability distribution over patterns in $\mathcal{L}_{i,k}^{(w)}$, then $A_{i,k}^{(w)}$ can be recovered by simply normalizing the DFA $\bar{A}_{i,k}^{(w)}$ by the quantity $\frac{1}{|\mathcal{L}_{i,k}^{(w)}|}$.

Given a sequence $w \in \Sigma^*$ and $(i,k) \in [[w]]^2$, the construction of the DFA, $\bar{A}_{i,k}^{(w)}$, is given as follows:

- **The state space:** $Q = [[w] + 1] \times \{0, \ldots, |w|\}$ (For a given state $(l,e) \in [[w]] \times [[w]]$, the element l tracks the position of the running pattern, and e computes the number of $\{\#\}$ symbols of the running pattern.

- **Initial State:** $(1, 0)$

- **The transition function:** For a given state $(l,e) \in [[w]] \times \{0, \ldots, |w|\}$ and a symbol $\sigma \in \Sigma_{\#}$, we have:

$$\delta((l,e), \sigma) = \begin{cases} (l+1, e+1) & \text{if } \sigma = \# \\ (l+1, e) & \text{if } \sigma \in \Sigma \wedge w_l = \sigma \wedge l \neq i \end{cases}$$

- **The final state:** $F = (|w| + 1, k)$

Complexity. The complexity of constructing $\bar{A}_{i,k}^{(w)}$ for a given $k \in [|w|]$ is equal to $O(|w|^2)$. Consequently, taking into account the iterative procedure in Algorithm 2, this latter algorithm runs in $O(|w|^3)$. In addition, the size of the resulting $A_{w,i}$ is also $O(|w|^3)$.

The construction of $A_{i,n}$. As previously noted, the 2-Alphabet WA can be interpreted as the global equivalent of $A_{w,i}$. Formally, for an integer $n > 0$ and $i \in [n]$, the 2-Alphabet WA $A_{i,n}$ realizes the function $g_{i,n}$ over $\Sigma_\# \times \Sigma$ as defined below:

$$g_{i,n}(p, w) = I(p \in \mathcal{L}_i^w) \cdot \mathcal{P}_i^w(p) \tag{17}$$

In light of Equation (17), the construction of $A_{i,n}$ aligns to the following three step procedure:

1. Construct a 2-Alphabet DFA $A'_{w,i}$ over $\Sigma_\# \times \Sigma$ that accepts the language $I(p \in \mathcal{L}_i^w)$

2. Construct a 2-Alphabet WA $\tilde{A}_{w,i}$ over $\Sigma_\# \times \Sigma$ such that: $f_{\tilde{A}_{w,i}}(p, w) = f_{A_{w,i}}(p)$ (Note that $f_{\tilde{A}_{w,i}}$ is independent of its second argument)

3. Return $A'_{w,i} \otimes A_{w,i}$

The derivation of the 2-Alphabet WA $\tilde{A}_{w,i}$ from $A_{w,i}$, whose construction is detailed in Algorithm 2, is straightforward. It remains to demonstrate how to construct the 2-Alphabet DFA $A'_{w,i}$ (Step 1).

The construction of $A'_{w,i}$. Recall that $\mathcal{L}_i^{(w)} \overset{\text{def}}{=} \bigcup_{k=1}^{|w|} \mathcal{L}_{i,k}^{(w)}$. Constructing a 2-Alphabet DFA that accepts the language $I(p \in \mathcal{L}_i^w)$ is relatively simple: It involves verifying the conditions $w \in L_p$ and $p_i = \#$ for the running sequence $(p, w) \in \Sigma_\#^* \times \Sigma^*$. The construction is provided as follows:

- **The state space:** $Q = [|w|]$

- **The initial state:** The state 1

- **The transition function:** For a state $q \in Q$ and $(\sigma, \sigma') \in \Sigma_\# \times \Sigma$, then we have that $\delta(q, (\sigma, \sigma')) = q + 1$ holds if and only if the predicate:

$$(q \neq i \wedge (\sigma = \#) \vee (\sigma = \sigma')) \vee (q = i \wedge \sigma = \#) \tag{18}$$

  is true.

  In essence, the predicate defined in Equation (18) captures the idea that, for a given position q in the current pair of sequences $(p, w) \in \Sigma_\#^* \times \Sigma^*$, either $w_q \in L_{p_q}$ when $q \neq i$ (ensuring the condition $w \in L_p$), or $p_q = \#$ when $q = i$ (ensuring the condition $p_i = \#$).

- **The final state:** The state $|w|$

Complexity. The construction of $A'_{w,i}$ requires $O(|w|)$ running time, and its size is equal to $O(|w|)$. The running time complexity for the construction of $A_{i,n}$ is dictated by the application of the Kronecker product between the 2-Alphabet WAs $A'_{w,i} \otimes A_{w,i}$ (Step 3 of the procedure outlined above). Given the complexity of computing the Kronecker product between N-Alphabet WAs (see Table 2) and the sizes of $A'_{w,i}$ and $A_{w,i}$, the overall time complexity is equal to $O(|\Sigma|^2 \cdot |w|^4)$. The size of $A_{i,n}$ is equal to $O(|w|^4)$.

4.4.2 The construction of $T_{w,i}$ and T_w

The constructions of $T_{w,i}$ and T_w are highly similar. Consequently, this section will mainly concentrate on the complete construction of $T_{w,i}$, as it introduces an additional challenge with the inclusion of the `swap` operation in the function implemented by this 3-Alphabet WA. A brief discussion on the construction of T_i will follow at the end of this section, based on the approach used for $T_{w,i}$.

Recall that $T_{w,i}$ is a 3-Alphabet DFA over $\Sigma_\# \times \Sigma \times \Sigma$ that implements the function:

$$g_{w,i}(p, w', u) = I(\text{do}(\text{swap}(p, w_i, i), w', w) = u)$$

for a triplet $(p, w', u) \in \Sigma_\#^* \times \Sigma^* \times \Sigma^*$, for which $|p| = |w'| = |u|$.

To ease exposition, we introduce the following predicate:

$$\Phi(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \stackrel{\text{def}}{=} (\sigma_1 = \# \wedge \sigma_3 = \sigma_2) \vee (\sigma_1 \neq \# \wedge \sigma_3 = \sigma_4) \tag{19}$$

where $(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \in \Sigma_\# \times \Sigma \times \Sigma \times \Sigma$.

The construction of $T_{w,i}$ is given as follows:

- **The state space:** $Q = [|w| + 1]$
- **The initial state:** $q_{init} = 1$
- **The transition function:** For a state $q \in Q$, and a tuple of symbols $(\sigma_1, \sigma_2, \sigma_3) \in \Sigma_\# \times \Sigma \times \Sigma$, we have $\delta(q, (\sigma_1, \sigma_2, \sigma_3)) = q + 1$ if and only if the predicate:

$$[q \neq i \wedge \Phi(\sigma_1, \sigma_2, \sigma_3, w_q)] \vee [q = i \wedge \sigma_3 = w_q] \tag{20}$$

  is true.
- **The final state:** $|w| + 1$.

Note. As previously mentioned, the construction of the 3-Alphabet WA, T_w, is quite similar to that of $T_{w,i}$. To obtain a comparable construction of T_w, one can easily adjust the predicate defined in equation 20 to $\Phi(\sigma, w_q, \sigma_3, \sigma_4)$.

4.4.3 The construction of T_i and T.

The 4-Alphabet WAs T_i and T can be seen as the global counterparts of $T_{w,i}$ and T_w, respectively. In addition, their constructions are somewhat simpler than the latter.

The construction of T_i. For a given $i \in \mathbb{N}$, the 4-Alphabet WA T_i is of size i, and constructed as follows:

- **The state space:** $Q = [i + 1]$
- **The initial state:** $q_{init} = 1$
- **The transition function:** For a state $q \in Q$, and $(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \in \Sigma_\# \times \Sigma \times \Sigma \times \Sigma$, we have:

$$\delta(q, (\sigma_1, \sigma_2, \sigma_3, \sigma_4)) = \begin{cases} q + 1 & \text{if } [q < i \wedge \Phi(\sigma_1, \sigma_2, \sigma_3, \sigma_4)] \vee [q = i \wedge \sigma_3 = \sigma_4] \\ i + 1 & \text{if } q = i + 1 \wedge \Phi(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \end{cases}$$

- **The final state:** $i + 1$

The construction of T. The 4-Alphabet WA T is an automaton with a single state, formally defined as follows:

- **The state space:** $Q = \{1\}$,
- **The initial state:** $q_{init} = 1$.
- **The transition function:** For $(\sigma_1, \sigma_2, \sigma_3, \sigma_4) \in \Sigma_\# \times \Sigma \times \Sigma \times \Sigma$, we have that: $\delta(1, (\sigma_1, \sigma_2, \sigma_3, \sigma_4)) = 1$ holds, if and only if the predicate $\Phi(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ is true.

4.4.4 The construction of $f_{w^{ref}}$

Given a sequence $w^{ref} \in \Sigma^*$, the machine $f_{w^{ref}}$ is defined as an HMM such that the probability of generating the sequence w^{ref} as a prefix is equal to 1. The set of HMMs that meet this condition is infinite. In our case, we opt for an easy construction of an HMM, denoted $f_{w^{ref}}$, belonging to this set.

The state space of $f_{w^{ref}}$ is given as $Q = [|w_{ref}| + 1]$ states. Similarly to the constructions above, each state is associated with a position within the emitted sequence of the HMM. Its functioning mechanism can be described using the following recursive procedure:

- The HMM starts from the state $q = 1$ with probability 1. The probability of emitting the symbol w_1^{ref} and transitioning to the state $q = 2$ is equal to 1.

- For $q \in [|w^{ref}|]$, the probability of generating the symbol w_q^{ref} and transitioning to the state $q + 1$ is equal to 1.

When the HMM reaches the state $|w^{ref}| + 1$, it generates a random symbol in Σ and remains at state $|w^{ref}| + 1$ with probability 1. One can readily verify (by a straightforward induction argument) that this HMM generates infinite sequences prefixed by the sequence w^{ref} with probability 1.

Complexity. The running time of this construction is equal to $O(|w^{ref}|)$, and the size of the obtained HMM $f_{w^{ref}f}$ is equal to $O(|w^{ref}|)$.

5 From Sequential Models to Non-Sequential Models: Reductions and Inter-inclusions

In Section 2.2 of the main paper, we demonstrated how the tractability result for computing various SHAP variants for WAs extends to several non-sequential models, including Ensemble Trees for Regression and Linear Regression Models (Theorem 2). In this section, we will present a complete and rigorous proof of this connection, along with additional theoretical insights into the relationships between these models and distributions. Specifically:

1. In the first subsection, we establish the proof of Theorem 2. This proof is based on several reductions, including those from linear regression models and ensemble trees to weighted automata, as well as from empirical distributions to the family $\overrightarrow{\texttt{HMM}}$.

2. In the second subsection, we present additional reductions that, while not explicitly used to prove the complexity results stated in the article, showcase the expressive power of the HMM class in modeling various families of distributions relevant to SHAP computations. These include distributions represented by Naive Bayes models and Markovian distributions. Moreover, we will highlight certain complexity results, listed in Table 1, that were not directly mentioned in the main text but are illuminated by these reduction findings.

5.1 Proof of Theorem 2

Recall the statement of Theorem 2:

Theorem. *Let* $\mathbb{S} := \{\texttt{LOC}, \texttt{GLO}\}$, $\mathbb{V} := \{\texttt{B}, \texttt{I}\}$, $\mathbb{P} := \{\texttt{EMP}, \overrightarrow{\texttt{HMM}}\}$, *and* $\mathbb{F} := \{\texttt{DT}, \texttt{ENS-DT}_{\texttt{R}}, \texttt{Lin}_{\texttt{R}}\}$. *Then, for any* $\texttt{S} \in \mathbb{S}$, $\texttt{V} \in \mathbb{V}$, $\texttt{P} \in \mathbb{P}$, *and* $\texttt{F} \in \mathbb{F}$ *the problem* $\texttt{S-V-SHAP}(\texttt{F}, \texttt{P})$ *can be solved in polynomial time.*

The proof of this theorem will rely on four inter-model polynomial reductions:

Claim. *The following statements hold true:*

1. $\overrightarrow{\texttt{HMM}} \preceq_{\texttt{P}} \texttt{HMM}$
2. $\texttt{EMP} \preceq_{\texttt{P}} \overrightarrow{\texttt{HMM}}$
3. $\texttt{ENS-DT}_{\texttt{R}} \preceq_{\texttt{P}} \texttt{WA}$
4. $\texttt{LIN}_{\texttt{R}} \preceq_{\texttt{P}} \texttt{WA}$

Theorem 2 is a direct consequence of the following four claims. This section is structured into four parts, each dedicated to proving one of these claims. Without loss of generality and to simplify the notation, we will assume throughout that all models operate on binary inputs. However, it is important to note that all of our results can be extended to the setting where each input is defined over k discrete values.

5.1.1 Claim 1: $\overrightarrow{\texttt{HMM}}$ is polynomially reducible to $\texttt{HMM}$

Consider a model $\overrightarrow{\mathrm{M}} = \langle \pi, \alpha, \{T_i\}_{i \in [n]}, \{O_i\}_{i \in [n]} \rangle$, where the size of $\overrightarrow{\mathrm{M}}$ is denoted as m. This model $\overrightarrow{\mathrm{M}}$ encodes a probability distribution over the hypercube $\{0, 1\}^n$. The goal is to prove that a model $M \in \texttt{HMM}$, which satisfies the following condition:

$$P_{\overrightarrow{\mathrm{M}}}(x_1, \ldots, x_n) = P_{\mathrm{M}}^{(n)}(x_1, \ldots, x_n), \tag{21}$$

can be constructed in polynomial time with respect to the size of $\overrightarrow{\text{M}}$. Intuitively, this condition asserts that the probability of generating a sequence $x = x_1, \ldots, x_n$ of length n using M is the same as the probability of generating $(x_1, \ldots, x_n)$ using $\overrightarrow{\text{M}}$.

The construction aims to build an HMM M that simulates $\overrightarrow{\text{M}}$ up to position n. Afterward, M transitions into a dummy hidden state where it stays stuck permanently, emitting symbols uniformly at random. To simulate $\overrightarrow{\text{M}}$ within the support $\{0, 1\}^M$, M must keep track of both the state reached by $\overrightarrow{\text{M}}$ after emitting a prefix $x_{\pi 1}, \ldots x_{\pi j}$ for a given $j \in [n]$, and the position reached in the sequence. Once the n-th symbol has been emitted, the HMM M transitions into a dummy state, which emits symbols uniformly at random and remains in that state indefinitely. The detailed construction is provided as follows:

- **The state space:** $Q = [M] \times [n + 1]$
- **State initialization:** The HMM M begins generating from the state $(i, 1)$ for $i \in [M]$ with a probability of $\alpha(i)$.
- **The transition dynamics:** For $(i, k, j) \in [M]^2 \times [N]$, the HMM M transitions from state (i, j) to state $(k, j + 1)$ with a probability of $T_j[i, k]$. Additionally, for any $i \in [M]$, we have $T[(i, n + 1), (i, n + 1)] = 1$ (meaning the HMM remains in the dummy state $n + 1$ indefinitely).
- **Symbol emission:** At a state $(i, j) \in [M] \times [n]$, the probability of emitting the symbol σ is equal to $O_{\pi(i)}[i, \sigma]$. On the other, for any state $(i, n + 1)$ where $i \in [M]$, the HMM M generates the symbol σ uniformly at random.

5.1.2 Claim 2: EMP is polynomially reducible to $\overrightarrow{\text{HMM}}$

The purpose of this subsection is to demonstrate that the class of Empirical distributions can be polynomially reduced to the HMM family. This claim has been referenced multiple times throughout the main article, where it played a key role in deriving complexity results for computing SHAP variants across various ML models, including decision trees, linear regression models, and tree ensemble regression models. Specifically, it shows that computing these variants under distributions modeled by HMMs is at least as difficult as computing them under empirical distributions.

The strategy for reducing empirical distributions to those modeled by the $\overrightarrow{\text{HMM}}$ family involves two steps:

1. The sequentialization step, which aims to transform the vectors in $\mathcal{D}$ into a dataset of binary sequences, referred to as $\text{SEQ}(\mathcal{D})$.

2. The construction of a model in $\overrightarrow{\text{HMM}}$ that encodes the empirical distribution induced by $\text{SEQ}(\mathcal{D})$.

We will now provide a detailed explanation of each step:

Step 1 (The sequentialization step): The sequentialization step is fairly straightforward and has been utilized in Marzouk and De La Higuera (2024) to transform decision trees into equivalent WAs. The sequentialization operation, denoted as $\text{SEQ}(., .)$, is defined as follows:

$$\text{SEQ}(\overrightarrow{x}, \pi) = x_{\pi(1)} \ldots x_{\pi(N)}$$

where π is a permutation. Essentially, the $\text{SEQ}(., .)$ operation transforms a given vector in $\{0, 1\}^N$ into a binary string, with the order of the feature variables determined by the permutation π. From here on, without loss of generality, we assume π to be the identity permutation.

Step 2 (The construction of the HMM): Applying the sequentialization step to the dataset $\mathcal{D}$ results in a *sequentialized* dataset consisting of M binary strings, denoted as $\text{SEQ}(\mathcal{D})$. The goal of the second step in the reduction is to construct a model in $\overrightarrow{HMM}$ that models the empirical distribution induced by $\text{SEQ}(\mathcal{D})$. While this construction is well-known in the literature of Grammatical Inference (refer to La Higuera book on Grammatical Inference), we will provide the details of this construction below.

For a given sequence w, we denote the number of occurences of w as a prefix in the dataset $P(\mathcal{D})$ as:

$$N(w) \stackrel{\text{def}}{=} \#|\{x \in \text{SEQ}(\mathcal{D}) : \exists(s) \in \{0, 1\}^* : x = ws\}|$$

We also define the set of prefixes appearing in $\texttt{SEQ}(\mathcal{D})$ as the set of all sequences $w \in \{0,1\}^*$ such that $N(w) \neq 0$. This set shall be denoted as $\mathrm{P}(\mathcal{D})$. Note that the set P is prefix-closed [6].

A (stationary) model in $\overrightarrow{\texttt{HMM}}$ that encodes the empirical distribution induced by the dataset $\texttt{SEQ}(\mathcal{D})$ is a probabilistic finite state automaton parametrized as follows:

1. **The permutation:** The identity permutation.

2. **The state space:** $Q = \mathrm{P}(\mathcal{D})$.

3. **The initial state vector:** Given $\sigma \in \{0,1\}$, the probability of starting from the state $\sigma \in \mathrm{P}(\mathcal{D})$ is equal to $\frac{N(\sigma)}{|\mathcal{D}|}$.

4. **The transition dynamics:** For a state $w\sigma \in Q$, where $w \in Q$ [7] and $\sigma \in \Sigma$, the probability of transitioning from the state w to the state $w\sigma$ is equal to $\frac{N(w\sigma)}{N(w)}$.

5. **The symbol emission:** For any state $w\sigma \in Q$, where $w \in Q$ and $\sigma \in \{0,1\}$, the probability of generating the symbol σ from the state $w\sigma$ is equal to 1.

One can readily prove that the resulting model computes the empirical distribution induced by $\mathrm{P}(\mathcal{D})$. Indeed, by construction, the probability of generating a binary sequence w by the model is equal to the probability of generating the sequence of states $w_1, w_{1:2}, \ldots, w_{1:n-1}w$. The probability of generating this state sequence is equal to:

$$\frac{N(w_1)}{|\mathcal{D}|} \prod_{i=1}^{|w|} \frac{N(w_{1:i+1})}{N(w_{1:i})} = \frac{N(w)}{|\mathcal{D}|}$$

5.1.3 Claim 3: $\texttt{DT}$ and $\texttt{ENS-DT}_\mathrm{R}$ are polynomially reducible to $\texttt{WA}$

The construction of an equivalent WA for either a decision tree or a tree ensemble used in regression tasks (i.e., a model in $\texttt{DT}$ or a model in $\texttt{ENS-DT}_\mathrm{R}$) is built upon the following result provided in Marzouk and De La Higuera (2024) for the $\texttt{DT}$ family:

Proposition 2. *There exists a polynomial-time algorithm that takes as input a decision Tree $T \in \texttt{DT}$ over the binary feature set $X = \{X_1, \ldots, X_n\}$ and outputs a WA A such that:*

$$f_A(x_1 \ldots x_n) = f_T(x_1, \ldots, x_n)$$

Since an ensemble of decision trees used for regression tasks is a linear combination of decision trees, and the family of WAs is closed under linear combination operations, with these operations being implementable in polynomial time as demonstrated in section 4.1, Proposition 2 consequently implies the existence of a polynomial-time algorithm that produces a WA equivalent to a given tree ensemble model used for regression.

Complexity. The algorithm for constructing a WA equivalent to a given DT T runs in $O(|T|)$, where $|T|$ denotes the number of edges in the DT T. The size of the resulting WA is $O(|T|)$ (See Marzouk and De La Higuera (2024) for more details). Consequently, the overall algorithm for constructing an equivalent regression tree ensemble consisting of m DTs $\{T_i\}_{i \in [m]}$ operates in $O(m \cdot \max_{i \in [m]} |T_i|)$. The size of the ensemble is also $O(m \cdot \max_{i \in [m]} |T_i|)$.

5.1.4 Claim 4: $\texttt{Lin}_\mathrm{R}$ is polynomially reducible to $\texttt{WA}$.

Let $M := \langle \{w_{i,d}\}_{i \in [n], d \in \mathbb{D}_i}, b \rangle$ be a linear regression model over the finite set $\mathbb{D} = [m_1] \times \ldots [m_n]$ where $\{m_i\}_{i \in [n]}$, computing the function (as defined earlier):

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{d \in \mathbb{D}_i} w_{i,d} \cdot I(x_i = d) + b$$

---

[6]A set of sequences S is prefix-closed if the set of prefixes of any sequence in S is also in S

[7]Thanks to the prefix-closedness property of the set $\mathrm{P}(\mathcal{D})$, if $w\sigma$ is in Q, then w is also in Q.

For the purposes of our reduction to WAs, we will assume that: $m_1 = \ldots = m_n = m$. The conversion of a linear regression model M over the input space $\mathbb{D} = [m_1] \times \ldots [m_n]$ to an equivalent one over $[m]^n$ can be done as follows: We set m to be $\max_{i \in [n]} m_i$, and the parameters $\{\tilde{w}_{i,d}\}_{i \in [n], d \in [m]}$ for the new model are set as:

$$\tilde{w}_{i,d} = \begin{cases} w_{i,d} & \text{if } d \leq m_i \\ 0 & \text{otherwise} \end{cases}$$

Observe that this procedure operates in $O(\max_{i \in [n]} m_i)$ time.

Reduction strategy. For a given linear regression model $M := \langle \{w_{i,d}\}_{i \in [n], d \in [m]}, b \rangle$ over $[m]^n$, we construct a WA over the alphabet $\Sigma = [m]$ with $n+1$ states. The initial state vector has value 1 for state 1, and 0 for all other states. For $i \in [n]$, the transition from state i to state $i+1$ is assigned a weight of $w_{i,\sigma}$ when processing the symbol σ from this state. Figure 2 illustrates an example of the graphical representation of the resulting WA from a given linear regression tree.

The additive intercept parameter b can be incorporated into this constructed model by simply adding the resulting WA from the procedure described above to a trivial single-state WA that outputs the value b for all binary strings (see Section 4.1 for more details on the addition operation between two WAs).
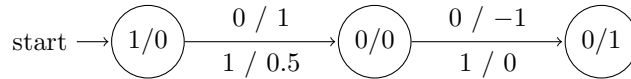

Figure 2: A construction of a WA equivalent to a linear regression over 2 binary features with the following weights: $w_{1,0} = 1$, $w_{1,1} = 0.5$, $w_{2,0} = -1$, $w_{2,1} = 0$. The notation x/y within the state nodes represents the initial weight of the state (x) and the final weight of the state (y).

5.2 Additional reduction results and implications on SHAP computation

Some of the complexity results presented in Table 1 were not explicitly discussed in the main article. Instead, additional polynomial-time reductions between Hidden Markov Models and other families of distributions are needed to derive them. For completeness, we include in the first part of this subsection proofs of these reduction relationships. Following that, we outline the implications of these relationships on the complexity of certain SHAP configurations shown in Table 1.

5.2.1 Additional reduction results

An interesting aspect of Hidden Markov Models is that they provide a unifying probabilistic modeling framework encompassing several families of distributions discussed in the literature on SHAP computation. Specifically, the class `MARKOV` (and, *de facto*, the family `IND`, which is trivially a subclass of `MARKOV`) as well as `NB` (more formally, `NB` $\preceq_P \overrightarrow{\text{HMM}}$) can be polynomially reduced to the class of Hidden Markov Models.

Claim 1: `NB` is polynomially reducible to $\overrightarrow{\text{HMM}}$. Let $M \in$ `NB` be a model over n binary features with a hidden variable Y that takes values from the discrete set $[m]$. The goal is to construct, in polynomial time, a model $M' \in \overrightarrow{\text{HMM}}$ that is equivalent to M, i.e.:

$$\forall (x_1, \ldots, x_n) \in \{0,1\}^n : \quad P_M(x_1, \ldots, x_n) = P_{M'}(x_1, \ldots, x_n)$$

A key observation underlying the reduction is that a naive Bayes Model can be viewed as a specific instance of a model in the family $\overrightarrow{\text{HMM}}$, where the state space coincides with the domain of the hidden variable in the naive Bayes model. The distinct feature of this model is that it remains in the same state throughout the entire generation process, with this state being determined at the initialization phase according to the probability distribution.

Formally, let $M = \langle \pi, \{P_i\}_{i \in [n]} \rangle$ be a naive bayes model over n observed RVs such that the domain value of its hidden state variable Y is equal to $[m]$. The construction of a model $M' \in \overrightarrow{\text{HMM}}$ is given as follows:

- **The permutation:** The identity permutation,

- **The state space:** $[m]$.

- **The state initialization:** M' starts generating from the state $i \in [m]$ with probability equal to $\pi[m]$.

- **The transition dynamics:** M' transitions from a given state j to the same state j with probability equal to 1.

- **The symbol emission:** At position i, the model M' emits the symbol $\sigma \in \{0, 1\}$ from the state j with probability equal to P_i.

Claim 2: MARKOV (resp. $\overrightarrow{\text{MARKOV}}$) is polynomially reducible to HMM (resp. $\overrightarrow{\text{HMM}}$). The class HMM (resp. $\overrightarrow{\text{HMM}}$) can be viewed as a generalization of the class MARKOV (resp. $\overrightarrow{\text{MARKOV}}$) for handling partially observable stochastic processes: Markovian distributions represent fully observable processes, whereas HMMs represent partially observable ones. Converting a Markovian distribution into a distribution modeled by an HMM involves encoding the Markovian dynamics into the hidden state dynamics of the HMM. The HMM then trivially emits the corresponding symbol of the reached hidden state at each position.

For completeness, the following provides a formal description of how MARKOV is polynomially reducible to HMM. The derivation of a polynomial-time reduction from $\overrightarrow{\text{MARKOV}}$ to $\overrightarrow{\text{HMM}}$ can be achieved using a similar construction. Let $M := \langle \pi, T \rangle$ be a model in MARKOV over the alphabet Σ. The description of a model $M' \in \text{HMM}$, equivalent to M, is obtained as follows:

- **The state space:** The alphabet Σ

- **The state initialization:** M' starts by generating a state $\sigma \in \Sigma$ with probability equal to $\pi(\sigma)$.

- **The transition dynamics:** From a hidden state $\sigma \in \Sigma$, the probability of transitioning to the state $\sigma' \in \Sigma$ is equal to $T[\sigma, \sigma']$.

- **The symbol emission:** For any symbol $\sigma \in \Sigma$, M' emits a symbol $\sigma \in \Sigma$ from the hidden state σ with probability equal to 1.

5.2.2 Corollaries on SHAP computational problems

The reduction relationships among independent distributions, Markovian distributions, and those modeled by HMMs (i.e., $\text{IND} \preceq_P \text{MARKOV} \preceq_P \text{HMM}$ and $\overrightarrow{\text{IND}} \preceq_P \overrightarrow{\text{MARKOV}} \preceq_P \overrightarrow{\text{HMM}}$), as well as the relationship between empirical and HMM-modeled distributions (i.e., $\text{EMP} \preceq_P \overrightarrow{\text{HMM}} \preceq_P \text{HMM}$), naturally leads to a corollary regarding the relationships between different SHAP variants:

Corollary 5. *For any family of sequential (resp. non-sequential) models $\mathbb{M}$ (resp. $\overrightarrow{\mathbb{M}}$), $\mathbb{S} = \{\text{LOC}, \text{GLO}\}$, we have:*

$$\mathbb{S} - \mathbb{V} - \text{SHAP}(\mathbb{M}, \text{IND}) \preceq_P \mathbb{S} - \mathbb{V} - \text{SHAP}(\mathbb{M}, \text{MARKOV}) \preceq_P \mathbb{S} - \mathbb{V} - \text{SHAP}(\mathbb{M}, \text{HMM}) \tag{22}$$

$$\mathbb{S} - \mathbb{V} - \text{SHAP}(\overrightarrow{\mathbb{M}}, \overrightarrow{\text{IND}}) \preceq_P \mathbb{S} - \mathbb{V} - \text{SHAP}(\overrightarrow{\mathbb{M}}, \overrightarrow{\text{MARKOV}}) \preceq_P \mathbb{S} - \mathbb{V} - \text{SHAP}(\overrightarrow{\mathbb{M}}, \overrightarrow{\text{HMM}}) \tag{23}$$

$$\mathbb{S} - \mathbb{V} - \text{SHAP}(\mathbb{M}, \text{EMP}) \preceq_P \mathbb{S} - \mathbb{V} - \text{SHAP}(\mathbb{M}, \text{HMM}) \; ; \; \mathbb{S} - \mathbb{V} - \text{SHAP}(\overrightarrow{\mathbb{M}}, \text{EMP}) \preceq_P \mathbb{S} - \mathbb{V} - \text{SHAP}(\overrightarrow{\mathbb{M}}, \overrightarrow{\text{HMM}}) \tag{24}$$

This corollary, along with previous complexity results for computing conditional SHAP values (Van den Broeck et al., 2022; Huang and Marques-Silva, 2024b; Arenas et al., 2023), and the fact that conditional and interventional SHAP variants coincide under independent distributions (Sundararajan and Najmi, 2020), presents a wide range of complexity results on SHAP computational problems that can be derived as corollaries:

Corollary 6. *All of the following complexity results hold:*

1. *For the class of models $\{\text{DT}, \text{ENS-DT}_{\text{R}}, \text{LIN}_{\text{R}}\}$: The computational problems $\mathbb{S} - \mathbb{V} - \text{SHAP}(\overrightarrow{\mathbb{M}}, \overrightarrow{\text{IND}})$ for $\mathbb{V} \in \{\text{B}, \text{V}\}$ can be solved in polynomial time. The computational problem $\text{GLO} - \text{C} - \text{SHAP}(\overrightarrow{\mathbb{M}}, \overrightarrow{\text{IND}})$ can be solved in polynomial time. The computational problem $\text{GLO} - \text{C} - \text{SHAP}(\overrightarrow{\mathbb{M}}, \overrightarrow{\text{HMM}})$ is #P-Hard.*

2. *For* WA*: The computational problems* $\mathbb{S}-\mathbb{V}-\text{SHAP}(\text{WA},\mathbb{P})$ *for* $\mathbb{S} \in \{\text{LOC},\text{GLO}\}$, $\mathbb{P} \in \{\text{IND},\text{EMP}\}$, *and* $\mathbb{V} \in \{\text{B},\text{V}\}$ *can be solved in polynomial time. The computational problem* $\text{GLO}-\text{C}-\text{SHAP}(\text{WA},\text{IND})$ *can be solved in polynomial time. The computational problem* $\text{LOC}-\text{C}-\text{SHAP}(\text{WA},\text{EMP})$ *is NP-Hard. The computational problem* $\text{LOC}-\text{C}-\text{SHAP}(\text{WA},\text{HMM})$ *is #P-Hard.*

3. *For* ENS-DT_{C}*: The computational problem* $\text{LOC}-\text{C}-\text{SHAP}(\text{ENS-DT}_{\text{C}},\text{EMP})$ *is NP-Hard. The computational problem* $\text{LOC}-\text{C}-\text{SHAP}(\text{ENS-DT}_{\text{C}},\overrightarrow{\text{HMM}})$ *is #P-Hard.*

4. *For* NN-SIGMOID*: The computational problem* $\text{LOC}-\text{C}-\text{SHAP}(\text{NN-SIGMOID},\overrightarrow{\text{HMM}})$ *is NP-Hard.*

5. *For* RNN-ReLU*: The computational problem* $\text{LOC}-\text{C}-\text{SHAP}(\text{RNN-ReLU},\overrightarrow{\text{IND}})$ *is NP-Hard.*

Proof. We will explain the results of each part of the corollaries separately:

1. For the class of models $\{\text{DT},\text{ENS-DT}_{\text{R}},\text{LIN}_{\text{R}}\}$: The complexity results for these model families under independent distributions, for both baseline and interventional SHAP, are derived from our findings on the tractability of these models under HMM-modeled distributions and our proof that $\text{EMP} \preceq_P \overrightarrow{\text{HMM}}$. The complexity results for the global and conditional forms under independent distributions stem from our tractability results for interventional SHAP in this setup, as well as the fact that interventional and conditional SHAP coincide under independent distributions (Sundararajan and Najmi, 2020). Lastly, the #P-Hardness of the conditional variant under HMM-modeled distributions follows from the hardness of generating this form of explanation under Naive Bayes modeled distributions (Van den Broeck et al., 2022), and our proof that $\overrightarrow{\text{NB}} \preceq_P \overrightarrow{\text{HMM}}$.

2. For WA, the tractability results for local and global baseline, as well as interventional SHAP under independent and empirical distributions, follow from our primary complexity findings for this family of models, which demonstrate tractability over HMM-modeled distributions, and our proof that both $\text{EMP} \preceq_P \text{HMM}$ and $\text{IND} \preceq_P \text{HMM}$. The tractability of global interventional SHAP under independent distributions also follows from these results, along with the fact that interventional and conditional SHAP coincide under independent distributions (Sundararajan and Najmi, 2020). The NP-hardness of conditional SHAP under empirical distributions is derived from the complexity of this setting for decision tree classifiers (Van den Broeck et al., 2022) and our proof that $\text{DT} \preceq_P \text{WA}$. Finally, the #P-hardness of conditional SHAP under HMM-modeled distributions results from the hardness of this setting for decision trees, as discussed earlier, and our proof that $\text{DT} \preceq_P \text{WA}$.

3. For ENS-DT_{C}: The complexity results for local conditional SHAP under empirical distributions are derived from the hardness results provided for decision trees under Naive Bayes-modeled distributions (Van den Broeck et al., 2022), along with the facts that $\text{DT} \preceq_P \text{ENS-DT}_{\text{C}}$ and $\overrightarrow{\text{NB}} \preceq_P \text{EMP}$. The result on the complexity of conditional SHAP under hidden Markov distributions follows as a corollary of the results in Huang and Marques-Silva (2024b) regarding the hardness of computing conditional SHAP under independent distributions, combined with our result showing that $\text{IND} \preceq_P \text{EMP}$.

4. For NN-SIGMOID: The complexity results for conditional SHAP under HMM distributions follow as a corollary of the hardness results presented in (Van den Broeck et al., 2022) for computing conditional SHAP values for sigmoidal neural networks under independent distributions, along with our proof that $\text{IND} \preceq_P \overrightarrow{\text{HMM}}$.

5. For RNN-ReLU: The complexity results for conditional SHAP under independent distributions stem from our main complexity result for this family of models, which established the hardness for interventional SHAP. Additionally, interventional and conditional SHAP coincide under independent distributions (Sundararajan and Najmi, 2020).

$\square$

6 On the NP-Hardness of computing local Interventional SHAP for RNN-ReLus under independent distributions

The objective of this segment is to prove Lemma 3 from section 3.1 of the main paper. This lemma played a crucial role in proving the primary result of this section, which demonstrates the intractability of the problem LOC-I-SHAP(RNN-ReLu, IND). The lemma is stated as follows:

Lemma. *The problem* EMPTY(RNN-ReLu) *is NP-Hard.*

The proof is performed by reduction from the closest string problem (CSP). Formally, the CSP problem is given as follows:

• **Problem:** CSP
Instance: A collection of strings $S = \{w_i\}_{i \in [m]}$, whose length is equal to n, and an integer $k > 0$.
Output: Does there exist a string $w' \in \Sigma^n$, such that for any $w_i \in S$, we have $d_H(w_i, w') \leq k$?

where $d_H(.,.)$ is the Hamming distance given as: $d_H(w, w') := \sum_{i=1}^{[\|w\|]} 1_{w_j}(w'_j)$.

The CSP problem is known to be NP-Hard (Li et al., 2002). Our reduction approach involves constructing, in polynomial time, an RNN that accepts only closest string solutions for a given input instance (S, k) of the CSP problem. Consequently, the CSP produces a *Yes* answer for the input instance if and only if the constructed RNN-ReLU is empty on the support Σ^n, which directly leads to the result of Lemma 3. We divide the proof of this reduction strategy into two parts:

• Given an arbitrary string w and an integer $k > 0$, we present a construction of an RNN-ReLu that simulates the computation of the Hamming distance for w. The outcome, indicating whether the Hamming distance exceeds the threshold k, is encoded in the activation of a specific neuron within the constructed RNN-ReLu. This procedure will be referred to as CONSTRUCT(w, k).

• Concatenate the RNN-ReLUs generated by the procedure CONSTRUCT$(.,.)$ into a single unified RNN cell. Then, assign an appropriate output matrix that accomplishes the desired objective of the construction.

The procedure CONSTRUCT(w, k). Let $w \in \{0, 1\}^n$ be a fixed reference string and $k > 0$ be an integer. The procedure CONSTRUCT(w, k) returns an RNN cell of dimension $|w| + 1$ that satisfies the properties discussed earlier. The parametrization of CONSTRUCT(w, k) is defined as follows:

• **The initial state vector.** $h_{init} = [1, 0 \ldots, 0]^T \in \mathbb{R}^{n+1}$.

• **The transition matrix.** he transition matrix is dependent on k and is expressed as:

$$W_k = \begin{pmatrix} 0 & 0 & . & . & . & 0 & 0 \\ 1 & 0 & . & . & . & 0 & 0 \\ 0 & 1 & . & . & . & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & . & . & . & 0 & -k \\ 0 & 0 & .. & . & . & 0 & 1 \end{pmatrix} \in \mathbb{R}^{(n+1) \times (n+1)} \tag{25}$$

• **The embedding vectors.** Embedding vectors are dependent on the reference string w. To prevent any confusion, we will use the superscript w for indexing. The embedding vectors are constructed as follows: For a symbol $\sigma \in \{0, 1\}$ and $l \in [n]$, we define $v_\sigma^{w_i}[l] = 1$ if $w_l \neq \sigma$. All other elements of v_σ^w are set to 0.

The following proposition formally demonstrates that this construction possesses the desired property:

Proposition 3. *Let $w \in \{0, 1\}^n$ be an arbitrary string, and $k > 0$ be an arbitrary integer. The procedure* CONSTRUCT(w, k) *outputs an RNN cell $\langle h_{init}, W_k, \{v_\sigma^{(w)}\}_{\sigma \in \{0,1\}} \rangle$, which satisfies the following properties:*

1. *For any string $w' \in \{0, 1\}^s$, where $s < n$, we have that $h_w[s] = d_H(w, w')$,*

2. *For a string $w' \in \{0, 1\}^n$, it holds that:*

$$h_{w'}[n] = \text{ReLu}(d_H(w, w') - k) \tag{26}$$

Proof. We individually prove the two defined requirements as follows:

1. The proof proceeds by induction on s. For the base case, when $s = 1$, we observe that for any symbol $\sigma \in \Sigma$, it holds that $h_\sigma = v_\sigma^{(w)}$. By construction, $v_\sigma^{(w)}[1] = 1_{w_1}(\sigma) = d_H(\sigma, w_1)$. Assume the proposition holds for $s < n - 2$. We now prove it for $s + 1$. Let $w' \in \Sigma^s$ and $\sigma \in \Sigma$. Then, we have

$$
\begin{aligned}
h_{w'\sigma}[s+1] &= \text{ReLu}(W[:,s+1]^T \cdot h_{w'} + v_\sigma[s+1]) \\
&= \text{ReLu}(h_{w'}[s] + v_\sigma[s+1]) \\
&= d_H(w_{1:s}, w') + 1_{w_{s+1}}(\sigma) \\
&= d_H(w'\sigma, w_{1:s+1})
\end{aligned}
$$

2. Let $w" = w'\sigma$ be a string of length n. From the first part of the proposition, we have $h_{w'}[n-1] = d(w', w_{1:(n-1)})$. Consequently,

$$
\begin{aligned}
h_{w"}[n] &= \text{ReLu}(W[:,n]^T \cdot h_{w'} + v_\sigma[n]) \\
&= \text{ReLu}(h_{w'}[n-1] - k + 1_{w_n}(\sigma)) \\
&= \text{ReLu}(d_H(w, w^i) - k)
\end{aligned}
$$

$\square$

The key property emphasized by Proposition 3 is stated in Equation 26. It demonstrates that the n-th neuron of the constructed RNN cell encodes the Hamming distance between the reference string w and the input string w'. Notably, the activation value of this neuron is 0 if and only if $d_H(w, w') \leq k$; otherwise, it is at least 1.

Concatenation and Output Matrix instantiation. Let (S, k) be an instance of the closest string problem. The final RNN cell will be formed by concatenating the RNN cells $\langle h_{init}, W_k, \{v_\sigma^{w^{(i)}}\}_{i\in[|S|]}\rangle$, where the set $\{w^{(i)}\}_{i\in|S|}$ consists of elements from S. The concatenation of two RNN cells, such as $\langle h_1, W_1, v_\sigma^{(1)}\rangle$ and $\langle h_1, W_2, v_\sigma^{(2)}\rangle$, produces a new cell defined as $\left\langle \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}, \begin{pmatrix} W_1 & 0 \\ 0 & W_2 \end{pmatrix}, \{\begin{pmatrix} v_\sigma^1 \\ v_\sigma^2 \end{pmatrix}\}_{\sigma\in\Sigma}\right\rangle$. [8]

We concatenate all RNN-ReLUs produced by the $\texttt{CONSTRUCT}(.,.)$ procedure on the instances $\{(w, k)\}_{w\in S}$. The output matrix of the resulting RNN-ReLU, denoted as $O \in \mathbb{R}^{(n+1)\cdot|S|}$, is selected such that for any set of vectors $\{h_i\}_{i\in[|S|]}$ in $\mathbb{R}^{n+1}$, the following holds:

$$
O^T \cdot \begin{bmatrix} h_1 & \cdots & h_s \end{bmatrix} = \sum_{i\in[|S|]} -h_i[n] + \frac{1}{2} \cdot h_1[n+1]
$$

Under this setting, it is important to note that, given the properties of the RNN-ReLu cell produced by the procedure $\texttt{CONSTRUCT}(.,.)$ (Proposition 3), and the fact that the activation value of the $(n+1)$-neuron is always equal to 1 by design, the output of the constructed RNN-ReLu on an input sequence w' is 1 if and only if for all $w \in S$, we have $d_H(w, w') \leq k$. As a result, the constructed RNN-ReLu is empty on the support n if and only if there is no string w' such that $d_H(w, w') \leq k$ for all w.

7 The problem $\texttt{LOC-B-SHAP(NN-SIGMOID)}$, $\texttt{LOC-B-SHAP(RNN-ReLu)}$ and $\texttt{LOC-B-SHAP(ENS-DT}_{\texttt{c}})$ are Hard: Proofs of Intermediary Results

This section of the appendix is devoted to presenting the proofs of the mathematical results discussed in subsection 3.2 regarding the local Baseline SHAP problem for different model families. The structure is as follows:

1. The first subsection (Subsection 7.1) presents the proof of the intractability of the $\texttt{LOC-B-SHAP(NN-SIGMOID)}$ problem, established through a reduction from the dummy player problem in WMGs.

2. The second subsection (Subsection 7.2) presents the proof of the NP-hardness of computing Local B-SHAP for the tree ensemble classifiers, achieved via a reduction from the 3SAT problem.

---

[8]Although the concatenation operation is non-commutative, the order according to which we perform the concatenation operator does not affect the result of the reduction.

7.1 Reducing the dummy player problem of WMGs to `LOC-B-SHAP(NN-SIGMOID)` and `LOC-B-SHAP(RNN-ReLu)`

The purpose of this segment is to establish Proposition 5 from the main paper. Let us first restate the proposition:

Proposition. *There exist two polynomial-time algorithms, which are defined as follows:*

1. *For* `NN-SIGMOID`*, there exists a polynomial-time algorithm that takes as input a WMG G and a player i in G and returns a sigmoidal neural network f_G over $\{0,1\}^N$, $(x, x^{ref}) \in \{0,1\}^N$ and $\epsilon \in \mathbb{R}$ such that:*

$$\text{The player } i \text{ is not dummy} \iff \phi_b(f_G, i, x, x^{ref}) > \epsilon$$

2. *For* `RNN-ReLu`*, there exists a polynomial-time algorithm that takes as input a WMG G and a player i in G and returns a sigmoidal neural network f_G over $\{0,1\}^N$, $(x, x^{ref}) \in \{0,1\}^N$ and $\epsilon \in \mathbb{R}$ such that:*

$$\text{The player } i \text{ is not dummy} \iff \phi_b(f_G, i, x, x^{ref}) > 0$$

We divide the remainder of this segment into two parts. The first part focuses on the family of sigmoidal neural networks (`NN-SIGMOID`), and the second part addresses the class of RNN-ReLus (`RNN-ReLu`).

7.1.1 The case of `LOC-B-SHAP(NN-SIGMOID)`

The intractability of the problem `LOC-B-SHAP(NN-SIGMOID)` is obtained by reduction from the dummy player problem of WMGs. The remainder of this segment will be dedicated to prove the following:

Proposition 4. *There exists a polynomial-time algorithm that takes as input a WMG $G = \langle N, \{n_j\}_{j \in [N]}, q \rangle$, and a player i in G, and returns a sigmoidal neural network f_G over $\mathbb{R}^N$, two vectors $(x, x^{ref}) \in \mathbb{R}^N \times \mathbb{R}^N$, and a scalar $\epsilon > 0$ such that:*

$$\text{The player } i \text{ is dummy} \iff \texttt{LOC-B-SHAP}(f_G, x, i, x^{ref}) \leq \epsilon$$

Algorithm 3 Reduction of the `DUMMY` problem to `LOC-B-SHAP(NN-SIGMOID)`

Input: A WMG $G = \langle N, \{n_j\}_{j \in [N]}, q \rangle$, $i \in [N]$
Output: A Sigmoidal Neural Network σ over $\mathbb{R}^N$, an integer $i \in [N]$, $(x, x^{ref}) \in \mathbb{R}^2$, and $\epsilon \geq 0$
1: $x \leftarrow [1, \ldots, 1]$
2: $x^{ref} \leftarrow [0, \ldots, 0]$
3: $C_N \leftarrow N \cdot \binom{N-1}{\lfloor \frac{N-1}{2} \rfloor}!$
4: $\epsilon \leftarrow \frac{1}{1+C_N}$
5: Construct the Sigmoidal Neural Network f, parameterized by:

$$f_G(x) = \sigma(2 \cdot \log(\frac{1-\epsilon}{\epsilon}) \cdot (x - q + \frac{1}{2}))$$

6: **return** $\langle f_G, \ i, \ x, \ x^{(ref)}, \ \epsilon \rangle$

The pseudo-code of the (polynomial-time) algorithm whose existence is implicitly stated in proposition 4 is provided in Algorithm 3. In the following, we provide the proof of the correctness of this reduction. In the remainder of this segment, we fix an input instance $I = \langle G, i \rangle$ of the `DUMMY` problem where $G = \langle N, \{n_j\}_{j \in [N]}, q \rangle$ is a WMG, and $i \in [N]$ is a player in G. And, we use the notation $\langle f_G, i, x, x^{ref}, \epsilon \rangle$ to represent the output of Algorithm 3. To ease exposition, we also introduce the following parameter which depends on the number of players N (appearing in Line 3 of Algorithm 3):

$$C_N \stackrel{\text{def}}{=} N \cdot \binom{\lfloor \frac{N-1}{2} \rfloor}{N-1}!$$

Note that by setting the instance to explain x to $\begin{pmatrix} 1 \\ \ldots \\ 1 \end{pmatrix}$, and the reference instance x^{ref} to $\begin{pmatrix} 0 \\ \ldots \\ 0 \end{pmatrix}$, the constructed function f_G (Line 5 of Algorithm 3) computes the following quantity for any coalition of players $S \subseteq [N] \setminus \{i\}$:

$$f_G(x_S; x_{\bar{S}}^{ref}) = \sigma\left(2\log(\frac{1-\epsilon}{\epsilon}) \cdot \log(N) \cdot (\sum_{j\in S} n_j - q + \frac{1}{2})\right)$$

where $\mathbf{x}_S = (x_S; x_{\bar{S}}^{ref}) \in \{0,1\}^n$ is such that $x_i = 1$ if $i \in S$, 0 otherwise.

The correctness of the reduction proposed in Algorithm 3 is a result of the following two claims:

Claim 1. *If the player i is dummy then for any $S \subseteq [N] \setminus \{i\}$ it holds that:*

$$f_G(x_{S\cup\{i\}}; x_{S\cup\{i\}}^{ref}) - f_G(x_S; x_{\bar{S}^{ref}}) \leq \epsilon \tag{27}$$

Claim 2. *If the player i is not dummy then there must exist some $S_d \subseteq [N] \setminus \{i\}$, such that:*

$$f_G(x_{S_d\cup\{i\}}; x_{S_d\cup\{i\}}^{ref}) - f_G(x_{S_d}; x_{\bar{S}_d}) > 1 - \epsilon \tag{28}$$

The result of proposition 4 is a corollary of claims 1 and 2. Before proving these two claims, we will first prove that, provided claims 1 and 2 are true, then the proposition 4 holds. To prove this, we will consider two separate cases:

- **Case 1 (The player i is dummy):** In this case, we show that if the player i is dummy and claim 1 holds, then:
$$\phi_b(f_G, i, x, x^{ref}) \leq \epsilon$$
.

  Assume claim 1 holds, we have:

$$\phi_b(f_G, i, x, x^{ref}) = \sum_{S\subseteq[n]\setminus\{i\}} \frac{|S|! \cdot (N - |S| - 1)!}{N!} \cdot \left[f_G(x_{S\cup\{i\}}; x_{S\cup\{i\}}^{ref}) - f_G(x_S; x_{\bar{S}^{ref}})\right]$$
$$\leq \sum_{S\subseteq[n]\setminus\{i\}} \frac{|S|! \cdot (N - |S| - 1)!}{N!} \cdot \epsilon$$
$$= \epsilon$$

- **Case 2 (The player i is not dummy):** In this case, we show that if the player i is not dummy and Claim 2 holds, then:
$$\phi_b(f_G, i, x, x^{ref}) > \epsilon$$

  Assume claim 2 holds, we have:

$$\phi_b(f_G, i, x, x^{ref}) = \sum_{S\subseteq[n]\setminus\{i\}} \frac{|S|! \cdot (n - |S| - 1)!}{n!} \cdot \left[f_G(x_{S\cup\{i\}}; x_{S\cup\{i\}}^{ref}) - f_G(x_S; x_{\bar{S}^{ref}})\right]$$
$$> \frac{|S_d|! \cdot (N - |S_d| - 1)!}{N!} \cdot (1 - \epsilon)$$
$$= \frac{1}{N \cdot \binom{|S_d|}{N-1}!} \cdot (1 - \epsilon)$$
$$> \frac{1}{N \cdot \binom{\lfloor\frac{N-1}{2}\rfloor}{N-1}!} \cdot (1 - \epsilon)$$
$$> \frac{1}{C_N} \cdot (1 - \epsilon) = \frac{1}{C_N} \cdot (1 - \frac{1}{1+C_N}) = \frac{1}{1+C_N} = \epsilon$$

  where the third inequality follows from the fact that for any $N \in \mathbb{N}$, $k \in [N]$, we have $\binom{N}{k}! \leq \binom{N}{\lfloor\frac{N}{2}\rfloor}!$.

The above argument indicates that proving claims 1 and 2 is sufficient to establish the desired result in this section (Proposition 4). What remains is to show that claims 1 and 2 indeed hold.

The following simple technical lemma leverages some properties of the sigmoidal function to prove that the constructed sigmoidal function f_G in ALgorithm 3 verifies the desired properties of both these claims:

Lemma 1. *The constructed function f_G in Algorithm 3 satisfies the following conditions:*

1. *If $x \leq q - 1$, we have: $f_G(x; N, q) \leq \epsilon$*

2. *If $x > q$, we have: $f_G(x; N, q) \geq 1 - \epsilon$*

Proof. By the property of monotonicity of the sigmoidal function and its symmerty around 0 (thus, assuming $q = 0$ w.l.o.g), it's sufficient to prove that for $x = -1$, we have that: $f_G(-1; N, 0) \leq \epsilon$. By simple calculation of $f_G(-1, N, 0)$ using the parametrization of the function f_G, one can obtain the result. $\qquad\square$

Now we are ready to prove claims 1 and 2.

Proof of Claim 1. Assume that player i is a dummy. Fix an arbitrary coalition $S \subseteq [N] \setminus \{i\}$. We now need to demonstrate that condition (27) holds for S. Since player i is a dummy, there are two possible cases: either both S and $S \cup \{i\}$ are winning, or neither of them are.

• **Case 1 (The winning case: $v_G(S) = 1$ and $v_G(S \cup \{i\}) = 1$):** In this case, we have both $\sum_{j \in S} n_j \geq q$ and $\sum_{j \in S} n_j + n_i \geq q$. Consequently, by Lemma 1, both $f_G(\sum_{j \in S \cup \{i\}} n_j; N, q)$ and $f_G(\sum_{j \in S} n_j; N, q)$ lie in the interval $(1 - \epsilon, 1)$. Thus, we have that:

$$f_G(x_{S \cup \{i\}}; x_{\overline{S \cup \{i\}}}^{ref}) - f_G(x_S; x_{\bar{S}}^{ref}) \leq 1 - (1 - \epsilon) = \epsilon$$

• **Case 2 (The non-winning case: $v_G(S) = 0$ and $v_G(S \cup \{i\}) = 0$):** The proof for this case mimicks the one of the former case. In this case, we have both $\sum_{j \in S} n_j \leq q - 1$ and $\sum_{j \in S} n_j + n_i \leq q - 1$. Consequently, by Lemma 1, both $g(\sum_{j \in S \cup \{i\}} n_j; N, q)$ and $g(\sum_{j \in S} n_j; N, q)$ lies in the interval $[0, \epsilon)$. Thus, we have:

$$f_G(x_{S \cup \{i\}}; x_{\overline{S \cup \{i\}}}^{ref}) - f_G(x_S; x_{\bar{S}^{ref}}) \leq \epsilon$$

Proof of Claim 2. Assume that the player i is not dummy. Then, there must exist a coalition $S_d \subset [N] \setminus \{i\}$ such that $\sum_{j \in S \cup \{i\}} n_j \geq q$ (a winning coalition), and $\sum_{j \in S} n_j \leq q - 1$ (A losing coalition). By proposition 1, we have then: $f_G(x_{S \cup \{i\}}; x_{\overline{S_d \cup \{i\}}}) \in (1 - \epsilon, 1)$ and $f_G(x_{S_d}; x_{\bar{S}_d}) \in (0, \epsilon)$. Consequently, we have:

$$f_G(x_{S \cup \{i\}}; x_{\overline{S_d \cup \{i\}}}) - f_G(x_{S_d}; x_{\bar{S}_d}) \geq 1 - \epsilon$$

7.1.2 The case `LOC-B-SHAP(RNN-ReLu)`

In this subsection, we shall provide the details of the construction of a polynomial-time algorithm that takes as input an instance $\langle G, i \rangle$ of the Dummy problem of WMGs and outputs an input instance of `LOC-B-SHAP(RNN-ReLu)` $\langle f_G, i, x, x^{ref} \rangle$ such that:

$$\text{The player i is dummy} \iff \phi_b(f_G, i, x, x^{ref}) > 0$$

where f_G is a RNN-ReLu.

Similar to the case of Sigmoidal neural networks, the main idea is to construct a RNN-ReLu that simulates a given WMG. However, contrary to sigmoidal neural networks, the constructed RNN-ReLu perfectly simulates a WMG:

Proposition 5. *There exists a polynomial time algorithm that takes as input a WMG $G = \langle N, \{n_j\}_{j \in [N]}, q \rangle$ and outputs an RNN-ReLu such that:*

$$\forall x \in \{0, 1\}^N : f_G(x) = v_G(S_x)$$

where: $S_x \stackrel{\text{def}}{=} \{j \in [N] : x_j = 1\}$

Proof. Fix a WMG $G = < N, \{n_j\}_{j \in [N]}, q >$. The idea of constructing a RNN-ReLu that satisfies the property implicitly stated in Proposition 5 consists at maintaining the sum of votes of players participating in the coalition in the hidden state vector of the RNN during the forward run of the RNN.

The dynamics of a RNN-ReLu of size $N + 2$ that performs this operation is given as follows:

1. **Initial state:** $h_{init} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$

2. **Transition function:** for $j \in \{1, \ldots, N - 1\}$

$$h[j + 1] = \begin{cases} h[j] + n_j & \text{if } x_j = 1 \quad \text{(Add the vote of the player } j \text{ and store it in neuron j+1 (i.e. } h[j + 1])) \\ h[j] & \text{if } x_j = 0 \quad \text{(Ignore the vote of player j as he/she is not part of the coalition)} \end{cases}$$

$$h[N + 2] = h[N + 2]$$

3. **The output layer:** For a given hidden state vector $\mathbb{R}^{N+1}$

$$y = I(h[N + 1] - q \cdot h[N + 2] \geq 0)$$

In other words, the output vector $O = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ -q \end{pmatrix}$

The dynamics of this RNN-ReLu is designed in such a way that for any $j \in [N]$ the value of the element $h[j + 1]$ of the hidden state vector stores the cumulative votes of participants represented by the input sequence $x_{1:j}$ (The first j symbols of the input sequence corresponding tp the players $\{1, \ldots, j\}$ in the game). Consequently, when $j = N + 1$, the voting power of all players in the coalition is stored in $h[N + 1]$. The output layer then outputs 1 if the $h[N + 1] \geq q$, otherwise it's equal to 0. $\qquad \square$

7.2 The problem LOC-B-SHAP(ENS-DT$_\text{C}$) is NP-Hard

This segment is dedicated to proving the remaining point of Theorem 4, which states the NP-Hardness of LOC-B-SHAP(ENS-DT$_\text{C}$). As noted in the definition of ENS-DTC in Appendix 3, since we are focusing on a *hardness* proof, we can, for simplicity, assume that the weights associated with each tree are equal, giving us a classic majority voting setting. Additionally, we will assume that the number of classes $c := 2$, meaning we have a binary classifier. Clearly, proving hardness for this setting will establish hardness for the more general setting as well. Essentially, these assumptions provide us with a simple random forest classifier for boolean classification. For simplicity, we will henceforth denote this problem as LOC-B-SHAP(ENS-DT$_c$) rather than the more general LOC-B-SHAP(ENS-DT$_\text{C}$). Proving NP-Hardness for the former will also establish it for the latter. As mentioned earlier, this problem is reduced from the classical 3-SAT problem, a widely known NP-Hard problem.

Reduction strategy. The reduction strategy is illustrated in Algorithm 4. For a given input CNF formula Ψ over n boolean variables $X = \{X_1, \ldots, X_n\}$ and m clauses, the constructed random forest is a model whose set of input features contains the set X, with an additional feature denoted X_{n+1} added for the sake of the reduction. The resulting random forest comprises a collection of $2m - 1$ decision trees, which can be categorized into two distinct groups:

- $\mathcal{T}_\Psi$: A set of m decision trees, each corresponding to a distinct clause in the input CNF formula. For a given clause C in Ψ, the associated decision tree is constructed to assign a label 1 to all variable assignments that satisfy the clause C while also ensuring that $x_{n+1} = 1$. It is simple to verify that such a decision tree can be constructed in polynomial time relative to the size of the input CNF formula

- $\mathcal{T}_{null}$: This set consists of $m - 1$ copies of a trivial null decision tree. A null decision tree assigns a label of 0 to all input instances.

Algorithm 4 Reduction of the `SAT` problem to `LOC-B-SHAP(ENS-DT`$_c$`)`

Input: A CNF Formula Φ of m clauses over $X = \{X_1, \ldots, X_n\}$
Output: An input instance of `LOC-B-SHAP(ENS-DT`$_c$`)`: $\langle \mathcal{T}, i, x, x^{ref} \rangle$
 1: $x \leftarrow [1, \ldots, 1]$
 2: $x^{ref} \leftarrow [0, \ldots, 0]$
 3: $i \leftarrow n + 1$
 4: $\mathcal{T} \leftarrow \emptyset$
 5: **for** $j \in [1, m]$ **do**
 6:    Construct a Decision Tree T_j that assigns a label 1 to variable assignments satisfying the formula: $C_j \wedge x_{n+1}$

 7:    $\mathcal{T} \leftarrow \mathcal{T} \cup \{T\}_j$
 8: **end for**
 9: Construct a null decision tree T_{null} that assigns a label 0 to all variable assignments
10: Add $m - 1$ copies of T_{null} to $\mathcal{T}$
11: **return** $\langle \mathcal{T}, i, , x, x^{ref} \rangle$

The next proposition provides a property of the Random Forest classifier resulting from the construction. This property shall be leveraged in Lemma 4 to yield the main result of this section:

Proposition 6. *Let Ψ be an arbitrary CNF formula over n boolean variables, and let $\mathcal{T}$ be the ensemble of decision trees outputted by Algorithm 4 for the input Ψ. We have:*

$$f_{\mathcal{T}}(x_1, \ldots, x_n, x_{n+1}) = \begin{cases} 1 & \text{if } x_{n+1} = 1 \wedge (x_1, \ldots, x_n) \models \Psi \\ 0 & \text{otherwise} \end{cases}$$

Proof. Fix an arbitrary CNF formula over n boolean variables, and let $(x_1, \ldots, x_n)$ be an arbitrary variable assignment. If $x_{n+1} = 0$, the decision trees in the set $\mathcal{T}$ assign the label 0, by construction. Consequently, $f_{\mathcal{T}}(x) = 0$.

For now, we assume that $x_{n+1} = 1$, if $x \models \Psi$, then x is assigned the label 1 for all decision trees in $\mathcal{T}$. Consequently, $f_{\mathcal{T}}(x) = 1$. On the other hand, if x is not satisfied by Ψ, then there exists at least one decision tree in $\mathcal{T}$, say T_j, that assigns a label 0 to x. Consequently, x is assigned a label 0 by at least m decision trees, i.e., for all decision trees in $\mathcal{T}_{null} \cup \{T_j\}$. □

Leveraging the result of Proposition 6, the following lemma yields immediately the result of NP-Hardness of the decision problem associated to `LOC-B-SHAP(ENS-DT`$_c$`)`:

Lemma 2. *Let Ψ be an arbitrary CNF formula of n variables, and $\langle \mathcal{T}, n+1, x, x^{ref} \rangle$ be the output of Algorithm 4 for the input Ψ. We have:*

$$\phi_b(f_{\mathcal{T}}, n + 1, x, x^{ref}) > 0 \iff \exists x \in \{0,1\}^n : x \models \Psi$$

Proof. Let Φ be an arbitrary CNF formula of n variables, and $\langle \mathcal{T}, n+1, x, x^{ref} \rangle$ be the output of Algorithm 4.

By proposition 6, we note that:

$$\forall \mathbf{x}_n = (x_1, \ldots, x_n) \in \{0,1\}^n : [f_{\mathcal{T}}(\mathbf{x}_n, 1) - f_{\mathcal{T}}(\mathbf{x}_n, 0) = 0 \iff \mathbf{x}_n \text{ doesn't satisfy } \Phi]$$

Combining this fact with the following two facts yiels the result of the lemma:

1. The Baseline SHAP score $\phi_b(f_{\mathcal{T}}, n + 1, x, x^{ref})$ is expressed as a linear combination of positive weights of the terms $\{f_{\mathcal{T}}(\mathbf{x}_n, 1) - f_{\mathcal{T}}(\mathbf{x}_n, 1)\}_{\mathbf{x}_n \in \{0,1\}^n}$
2. According to Proposition6, we obtain:

$$\forall \mathbf{x}_n \in \{0,1\}^n : f_{\mathcal{T}}(\mathbf{x}_n, 1) - f_{\mathcal{T}}(\mathbf{x}_n, 0) = f_{\mathcal{T}}(\mathbf{x}_n, 1) \geq 0$$

□

8   Generalized Complexity Relations of SHAP Variants (Proof of Proposition 7)

In this section, we present the proof for Proposition 7 from the main paper. Let us first restate the proposition:

Proposition. *Let $\mathcal{M}$ be a class of models and $\mathcal{P}$ a class of probability distributions such that* EMP $\preceq_P \mathcal{P}$. *Then,* LOC-B-SHAP*$(\mathcal{M}) \preceq_P$* GLO-B-SHAP*$(\mathcal{M}, \mathcal{P})$ and* LOC-B-SHAP*$(\mathcal{M}) \preceq_P$* LOC-I-SHAP*$(\mathcal{M}, \mathcal{P})$.*

Proof. (1) LOC-B-SHAP$(\mathcal{M}) \preceq_P$ GLO-B-SHAP$(\mathcal{M}, \mathcal{P})$.

This result is derived directly by observing that:

$$\mathbf{\Phi}_b(f, i, x^{ref}, P_x) = \Phi_b(f, i, x, x^{ref})$$

where P_x is the empirical distribution induced by the input instance x.

(2) LOC-B-SHAP$(\mathcal{M}) \preceq_P$ LOC-I-SHAP$(\mathcal{M}, \mathcal{P})$: The result is obtained straightforwardly by noting that:

$$\phi_i(f, i, x, P_{x^{ref}}) = \phi_b(f, i, x, x^{ref})$$

where $P_{x^{ref}}$ is the empirical distribution induced by the reference instance x^{ref}. $\qquad\square$