# On Computing Probabilistic Explanations for Decision Trees

MARCELO ARENAS, Department of Computer Science and Institute for Mathematical and Computational Engineering, Faculty of Engineering, Catholic University of Chile, Chile and IMFD, Chile
PABLO BARCELÓ, Institite for Mathematical and Computational Engineering, Faculty of Engineering and Faculty of Mathematics, Catholic University of Chile, Chile, IMFD, Chile, and CENIA, Chile
ALEXANDER KOZACHINSKIY, CENIA, Chile
MIGUEL ROMERO, Department of Computer Science, Faculty of Engineering, Catholic University of Chile, Chile and CENIA, Chile
BERNARDO SUBERCASEAUX, School of Computer Science, Carnegie Mellon University, USA

Formal XAI (explainable AI) is a growing area that focuses on computing explanations with mathematical guarantees for the decisions made by ML models. Inside formal XAI, one of the most studied cases is that of explaining the choices taken by decision trees, as they are traditionally deemed as one of the most interpretable classes of models. Recent work has focused on studying the computation of *sufficient reasons*, a kind of explanation in which given a decision tree $T$ and an instance $\boldsymbol{x}$, one explains the decision $T(\boldsymbol{x})$ by providing a subset $\boldsymbol{y}$ of the features of $\boldsymbol{x}$ such that for any other instance $\boldsymbol{z}$ compatible with $\boldsymbol{y}$, it holds that $T(\boldsymbol{z}) = T(\boldsymbol{x})$, intuitively meaning that the features in $\boldsymbol{y}$ are already enough to fully justify the classification of $\boldsymbol{x}$ by $T$. It has been argued, however, that sufficient reasons constitute a restrictive notion of explanation. For such a reason, the community has started to study their probabilistic counterpart, in which one requires that the probability of $T(\boldsymbol{z}) = T(\boldsymbol{x})$ must be at least some value $\delta \in (0, 1]$, where $\boldsymbol{z}$ is a random instance that is compatible with $\boldsymbol{y}$. Our paper settles the computational complexity of $\delta$-sufficient-reasons over decision trees, showing that both (1) finding $\delta$-sufficient-reasons that are minimal in size, and (2) finding $\delta$-sufficient-reasons that are minimal inclusion-wise, are computationally intractable. By doing this, we answer two open problems originally raised by Izza et al. (2021), and extend the hardness of explanations for Boolean circuits presented by Wäldchen et al. (2021) to the more restricted case of decision trees. Furthermore, we present sharp non-approximability results under a widely believed complexity hypothesis. On the positive side, we identify structural restrictions of decision trees that make the problem tractable.

**JAIR Associate Editor:** Rafael Peñaloza

**JAIR Reference Format:**
Marcelo Arenas, Pablo Barceló, Alexander Kozachinskiy, Miguel Romero, and Bernardo Subercaseaux. 2025. On Computing Probabilistic Explanations for Decision Trees. *Journal of Artificial Intelligence Research* 83, Article 34 (August 2025), 44 pages. DOI: 10.1613/jair.1.17494

Authors' Contact Information: Marcelo Arenas, ORCID: 0000-0003-3678-1868, marenas@uc.cl, Department of Computer Science and Institute for Mathematical and Computational Engineering, Faculty of Engineering, Catholic University of Chile, Santiago, Metropolitan Region, Chile and IMFD, Santiago, Metropolitan Region, Chile; Pablo Barceló, ORCID: 0000-0003-2293-2653, pbarcelo@uc.cl, Institite for Mathematical and Computational Engineering, Faculty of Engineering and Faculty of Mathematics, Catholic University of Chile, Santiago, Metropolitan Region, Chile and IMFD, Santiago, Metropolitan Region, Chile and CENIA, Santiago, Metropolitan Region, Chile; Alexander Kozachinskiy, ORCID: 0009-0002-8690-3640, kozmath@proton.me, CENIA, Santiago, Metropolitan Region, Chile; Miguel Romero, ORCID: 0000-0002-2615-6455, mgromero@uc.cl, Department of Computer Science, Faculty of Engineering, Catholic University of Chile, Santiago, Metropolitan Region, Chile and CENIA, Santiago, Metropolitan Region, Chile; Bernardo Subercaseaux, ORCID: 0000-0003-2295-1299, bersub@cmu.edu, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.


DOI: 10.1613/jair.1.17494

## 1 Introduction

The level of trust that AI models generate in people has been repetitively linked to our ability of *explaining* the decision of said models [1], thus motivating the area of *explainable AI* (XAI) as an important aspect of the deployment of trustworthy artificial intelligence. A sub-area of explainability that has received considerable attention over the last years, showing quick progress in theoretical and practical terms, is that of *local* explanations, that is, explanations for the outcome of a particular input to an ML model after the model has been trained. Several queries and scores have been proposed to specify explanations of this kind. These include, for example, queries based on *prime implicants* [2] or *anchors* [3], which are parts of an instance that are sufficient to explain its classification, as well as scores that intend to quantify the impact of a single feature in the output of such a classification [4, 5].

A remarkable achievement of this area of research has been the development of *formal* notions of explainability. The benefits brought about by this principled approach have been highlighted in a very thorough way by a recent survey [6]. A prime example of this kind of approach is given by *"sufficient reasons"*, which are also known as prime implicant explanations [2] or abductive explanations [7].

Given an ML model $\mathcal{M}$ of dimension $n$ and a Boolean input instance $\boldsymbol{x} \in \{0,1\}^n$, a sufficient reason for $\boldsymbol{x}$ under $\mathcal{M}$ is a subset $\boldsymbol{y}$ of the features of $\boldsymbol{x}$, such that any instance $\boldsymbol{z}$ compatible with $\boldsymbol{y}$ receives the same classification result as $\boldsymbol{x}$ on $\mathcal{M}$. In more intuitive words, $\boldsymbol{y}$ is a sufficient reason for $\boldsymbol{x}$ under $\mathcal{M}$ if the features in $\boldsymbol{y}$ suffices to explain the output of $\mathcal{M}$ on $\boldsymbol{x}$. In the formal explainability approach, one then aims to find sufficient reasons $\boldsymbol{y}$ that satisfy one of the following optimality criteria:

(1) they are *minimum*, that is, there are no sufficient reasons with fewer features than $\boldsymbol{y}$, or
(2) they are *minimal*, that is, there are no sufficient reasons that are strictly contained in $\boldsymbol{y}$.

**Problem**. The XAI community has studied for which Boolean ML models the problem of computing (minimum or minimal) sufficient reasons is computationally tractable and for which it is computationally hard (see, e.g., [8, 9, 10]). It has been argued, however, that for practical applications sufficient reasons might be too *rigid*, as they are specified under worst-case conditions. That is, $\boldsymbol{y}$ is a sufficient reason for $\boldsymbol{x}$ under $\mathcal{M}$ if *every* "completion" of $\boldsymbol{y}$ is classified by $\mathcal{M}$ in the same way as $\boldsymbol{x}$. As several authors have noted already, there is a natural way in which this notion can be relaxed in order to become more suitable for real-world explainability tasks: Instead of asking for each completion of $\boldsymbol{y}$ to yield the same result as $\boldsymbol{x}$ on $\mathcal{M}$, we could allow for a small fragment of the completions of $\boldsymbol{y}$ to be classified differently than $\boldsymbol{x}$ [11, 12, 13]. More precisely, we would like to ensure that a random completion of $\boldsymbol{y}$ is classified as $\boldsymbol{x}$ with probability at least $\delta \in (0,1]$, a threshold that the recipient of the explanation controls. We call $\boldsymbol{y}$ a *$\delta$-sufficient reason for $\boldsymbol{x}$ under $\mathcal{M}$*.

The study of the cost of computing minimum $\delta$-sufficient reasons for expressive Boolean ML models based on propositional formulas was started by [11]. They show, in particular, that the decision problem of checking if $\boldsymbol{x}$ admits a $\delta$-sufficient reason of a certain size $k$ under a model $\mathcal{M}$, where $\mathcal{M}$ is specified as a CNF formula, is $\mathrm{NP}^{\mathrm{PP}}$-complete. This result shows that the problem is very difficult for complex models, at least in theoretical terms. Nonetheless, it leaves the door open for obtaining tractability results over simpler Boolean models, starting from those which are often deemed to be "easy to interpret", for example, *decision trees* [14, 15, 16]. In particular, the study of the cost of computing both minimum and minimal $\delta$-sufficient reasons for decision trees was initiated by [12, 17], but nothing beyond the fact that the problem lies in NP has been obtained. The work of [18] has shown that it is possible to obtain efficient algorithms that succeed with a certain probability, and that instead of finding a smallest (either cardinality or inclusion-wise) $\delta$-sufficient reason, find $\delta$-sufficient reasons that are small compared to the *average* size of $\delta$-sufficient reasons for the considered model.

*Decision Trees need to be explained.* When looking at small decision trees (e.g., ~15 leaves, as the example depicted in Figure 1) it is tempting to think that decision trees do not require algorithms to be explained, as a

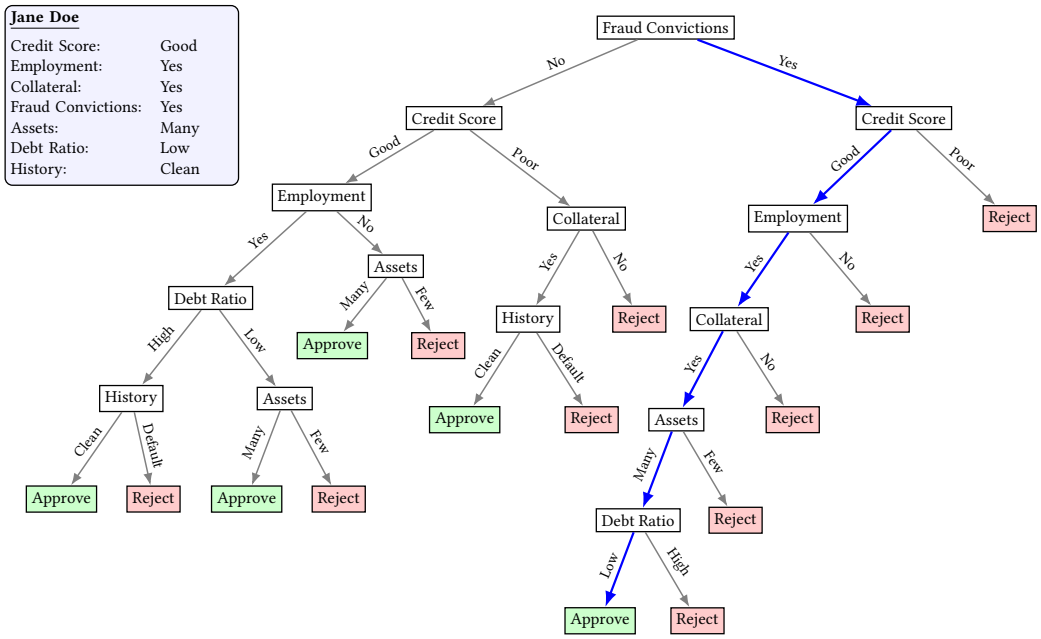



Fig. 1. Example of a decision tree for judging loan applicants with binary features. The decision path for a concrete applicant, Jane Done, is highlighted in blue.

human can readily look at the root-to-leaf path associated to any classification decision they want to understand. This view, however, has been repeatedly challenged by the formal XAI community as rather naïve [15, 19, 20]. Let us present a brief summary of some of the issues involved. First, practical training algorithms, with standard choices of hyper-parameters, often result in trees with hundreds or even thousands of nodes, which makes manual inspection significantly harder than in toy examples [21]. Second, "path explanations" (i.e., the set of features mentioned in the path that a given input instance takes along the decision tree) can be much larger than small sufficient reasons, as shown both in theory and practice [15, 19, 20]. The decision tree example presented in Figure 1 depicts a path explanation for applicant *Jane Doe* that includes 6 out of the 7 possible features. However, its subset of 5 features not including *Fraud Convictions* can be easily checked to be a (minimum) sufficient reason.[1] Not only is this reducing the size of the explanation by 1, but it also makes the explanation less misleading: we claim that Jane Doe's application is not approved *because* of her fraud convictions, but *despite* them, since her other features as an applicant are very positive. Checking our claim requires inspecting not only the decision path of Jane's application, but potentially the entire tree, to see that if she did not have fraud convictions, her application would have taken a different path but resulted in approval nonetheless. While in this particular toy example the feature names are suggestive enough to make rather obvious that the fraud conviction is not a good explanation for Jane's approval, we claim this illustrates how in a general case, with a large number of potentially opaque features, minimal explanations are needed to understand whether a classification is happening *despite* a given a feature or *partially because* of it. As it has been claimed by others in the literature (e.g., [15]), we believe this makes a compelling argument to reject the view that decision trees are immediately interpretable by looking

[1] As shown by [15], the difference in size between path explanations and minimal sufficient reasons can be arbitrarily large.

at their decision paths, and that to properly understand their decisions it might be necessary to run algorithms that obtain explanations that consider the globality of the tree.

**Our results**. In this paper, we provide an in-depth study of the complexity of the problem of minimum and minimal $\delta$-sufficient reasons for decision trees.

- *Hardness results:* We first show that, assuming P $\neq$ NP, neither of these problems can be solved in polynomial time. For the problem of computing minimum $\delta$-sufficient reasons over decision trees, the hardness holds even for each fixed $\delta \in (0, 1]$.
- *Non-approximability results:* Then we study the approximation properties of these problems, and establish that neither of them can be approximated by assuming a stronger complexity conjecture; namely, that NP problems cannot be solved in quasi-polynomial time. Although this assumption is stronger than P $\neq$ NP, it is still widely accepted. In fact, even the so-called ETH (*exponential time hypothesis*), stating that SAT cannot be solved just in sub-exponential time, is a standard complexity assumption now. Nevertheless, this means that our non-approximability results do not directly imply our hardness results, as the former rely on a stronger hypothesis. We mention that similar inapproximability results have been obtained by [11], but for Boolean circuits instead of decision trees, making our results much stronger (although their results are under the weaker assumption, P $\neq$ NP).
- *Tractability results:* Finally, we look at structural restrictions on decision trees that, at the same time, represent meaningful practical instances and ensure that the problems studied in the paper can be solved in polynomial time. The first such restriction is having a *bounded split number*, which intuitively means that there is a constant $c \geqslant 1$ such that, for every node $u$ in the decision tree $T$, the number of features that appear both in the subtree of $T$ rooted at $u$ and outside this subtree is bounded by $c$. We show that both problems studied in this paper, that is, computing minimum and minimal $\delta$-sufficient reasons over decision trees, can be solved in polynomial time for decision trees of bounded split number. The second restriction studied is *monotonicity*, which refers to the class of decision trees that, when presented with an instance $\boldsymbol{x} \in \{0, 1\}^n$ that is classified positively, also classify as positive every instance $\boldsymbol{x}'$ that is obtained from $\boldsymbol{x}$ by changing the value of some of its features from 0 to 1. We show that the problem of computing minimal $\delta$-sufficient reasons can be solved in polynomial time over monotone decision trees.

**Organization of the paper**. We start with preliminaries in Section 2. Our main results are presented, without proof, in Section 3. Proofs of our hardness results are in Section 4, of non-approximability results in Section 5, and of tractability results in Section 6. We finalize with closing remarks and open problems in Section 7.

## 2 Preliminaries

Let us begin by providing the necessary definitions and notation that will be used throughout the paper.

### 2.1 Models and Instances

An *instance* of dimension $n$, with $n \geqslant 1$, is a tuple $\boldsymbol{x} \in \{0, 1\}^n$. We use notation $\boldsymbol{x}[i]$ to refer to the $i$-th component of this tuple, or equivalently, its $i$-th feature. Moreover, we consider an abstract notion of a model of dimension $n$, and we define it as a Boolean function $\mathcal{M} : \{0, 1\}^n \to \{0, 1\}$. That is, $\mathcal{M}$ assigns a Boolean value to each instance of dimension $n$, so that we focus on binary classifiers with Boolean input features. Restricting inputs and outputs to be Boolean makes our setting cleaner while still covering several relevant practical scenarios. We use notation $\dim(\mathcal{M})$ for the dimension of a model $\mathcal{M}$.

## 2.2 Binary Decision Diagrams and Decision Trees

A *binary decision diagram* (BDD) of dimension $n$ is a rooted directed acyclic graph $\mathcal{M}$ with labels on edges and nodes such that:

- each leaf (a node with no outgoing edges) is labeled with **true** or **false**,
- each internal node (a node that is not a leaf) is labeled with a feature $i \in \{1, \ldots, n\}$,
- each internal node has two outgoing edges, one labeled 1 and the other one labeled 0.

Every instance $\boldsymbol{x} \in \{0,1\}^n$ defines a unique path $\pi_{\boldsymbol{x}} = u_1 \cdots u_k$ from the root $u_1$ to a leaf $u_k$ of $\mathcal{M}$ such that: if the label of $u_i$ is $j \in \{1, \ldots, n\}$, where $i \in \{1, \ldots, k-1\}$, then the edge from $u_i$ to $u_{i+1}$ is labeled with $\boldsymbol{x}[j]$. Moreover, the instance $\boldsymbol{x}$ is positive, denoted by $\mathcal{M}(\boldsymbol{x}) = 1$, if the label of $u_k$ is **true**; otherwise the instance $\boldsymbol{x}$ is negative, which is denoted by $\mathcal{M}(\boldsymbol{x}) = 0$. A BDD $\mathcal{M}$ is *free* if for every path from the root to a leaf, no two nodes on that path have the same label. A *decision tree* is simply a free BDD whose underlying directed acyclic graph is a rooted tree. When a computational problem has a decision tree as part of its input, we assume a standard encoding whose number of bits is polynomial in the number of nodes of the tree.

## 2.3 Partial Instances

In the following, a *partial instance* of dimension $n$ is a tuple $\boldsymbol{y} \in \{0, 1, \bot\}^n$. Intuitively, if $\boldsymbol{y}[i] = \bot$, then the value of the $i$-th feature is undefined. Notice that an instance is a particular case of a partial instance where all features are assigned value either 0 or 1. Given two partial instances $\boldsymbol{x}, \boldsymbol{y}$ of dimension $n$, we say that $\boldsymbol{y}$ is *subsumed* by $\boldsymbol{x}$, denoted by $\boldsymbol{y} \subseteq \boldsymbol{x}$, if for every $i \in \{1, \ldots, n\}$ such that $\boldsymbol{y}[i] \neq \bot$, it holds that $\boldsymbol{y}[i] = \boldsymbol{x}[i]$. That is, $\boldsymbol{y}$ is subsumed by $\boldsymbol{x}$ if it is possible to obtain $\boldsymbol{x}$ from $\boldsymbol{y}$ by replacing some undefined values. Moreover, we say that $\boldsymbol{y}$ is *properly subsumed* by $\boldsymbol{x}$, denoted by $\boldsymbol{y} \subsetneq \boldsymbol{x}$, if $\boldsymbol{y} \subseteq \boldsymbol{x}$ and $\boldsymbol{y} \neq \boldsymbol{x}$. For instance,

$$(0, \bot, 1, \bot) \subsetneq (0, 0, 1, \bot) \quad \text{and} \quad (0, \bot, 1, \bot) \not\subseteq (0, 0, 0, \bot).$$

Notice that a partial instance $\boldsymbol{y}$ can be thought of as a compact representation of the set of instances $\boldsymbol{z}$ such that $\boldsymbol{y}$ is subsumed by $\boldsymbol{z}$, where such instances $\boldsymbol{z}$ are called the *completions* of $\boldsymbol{y}$. We write $\textsc{Comp}(\boldsymbol{y})$ for the set of completions of $\boldsymbol{y}$.

## 2.4 Sufficient Reasons

Sufficient reasons are partial instances obtained by removing from an instance $\boldsymbol{x}$ components that do not affect the final classification. Formally, fix a dimension $n$. Given a decision tree $T$, an instance $\boldsymbol{x}$, and a partial instance $\boldsymbol{y}$ with $\boldsymbol{y} \subseteq \boldsymbol{x}$, we call $\boldsymbol{y}$ a *sufficient reason* for $\boldsymbol{x}$ under $T$ if $T(\boldsymbol{x}) = T(\boldsymbol{z})$ for every $\boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})$. In other words, the features of $\boldsymbol{y}$ that take value either 0 or 1 explain the decision taken by $T$ on $\boldsymbol{x}$, as $T(\boldsymbol{x})$ would not change if the remaining features (i.e., those that are undefined in $\boldsymbol{y}$) were to change in $\boldsymbol{x}$, thus implying that the classification $T(\boldsymbol{x})$ is a consequence of the features defined in $\boldsymbol{y}$.

We say that a sufficient reason $\boldsymbol{y}$ for $\boldsymbol{x}$ under $T$ is *minimal*, if it is minimal under the order induced by $\subseteq$, that is, if there is no sufficient reason $\boldsymbol{y}'$ for $\boldsymbol{x}$ under $T$ such that $\boldsymbol{y}' \subsetneq \boldsymbol{y}$. Also, we define a *minimum* sufficient reason for $\boldsymbol{x}$ under $T$ as a sufficient reason $\boldsymbol{y}$ for $\boldsymbol{x}$ under $T$ that maximizes the value $|\boldsymbol{y}|_\bot := |\{i \in \{1, \ldots, n\} \mid \boldsymbol{y}[i] = \bot\}|$.

It turns out that minimal sufficient reasons can be computed efficiently over decision trees.

**Proposition 2.1.** *There is a polynomial-time*[2] *algorithm that, given a decision tree $T$ and an instance $\boldsymbol{x}$ of the same dimension, computes a minimal sufficient reason for $\boldsymbol{x}$ under $T$.*

[2]Whenever we say a polynomial-time algorithm, the polynomial is with respect to the entire input size, which usually involves trees (whose size is polynomial in the number of nodes), partial instances (whose size is proportional to the dimension), and integers, which we assume are encoded in binary.

This proposition can be established by using a very simple algorithm, assuming a sub-routine to check whether a given partial instance is a sufficient reason (not necessarily minimal) of another given instance. As shown in Algorithm 2, the idea of the algorithm is as follows: start with a candidate answer $\boldsymbol{y}$ which is initially equal to $\boldsymbol{x}$, the instance to explain, and maintain the invariant that $\boldsymbol{y}$ is a sufficient reason for $\boldsymbol{x}$, while trying to remove defined components from $\boldsymbol{y}$ until no longer possible. It is not hard to see that one can check whether a partial instance $\boldsymbol{y}$ is a sufficient reason for an instance $\boldsymbol{x}$ in linear time over decision trees [22], which implies that Algorithm 2 runs in polynomial time over such models.

**Algorithm 2** Minimal Sufficient Reason

**Input:** Decision tree $T$ and instance $\boldsymbol{x}$, both of dimension $n$
**Output:** A minimal sufficient reason $\boldsymbol{y}$ for $\boldsymbol{x}$ under $T$
$\boldsymbol{y} \leftarrow \boldsymbol{x}$
**for** $i \in \{1, \ldots, n\}$ **do**
    $\hat{\boldsymbol{y}} \leftarrow \boldsymbol{y}$
    $\hat{\boldsymbol{y}}[i] \leftarrow \bot$
    **if** CheckSufficientReason$(T, \hat{\boldsymbol{y}}, \boldsymbol{x})$ **then**
        $\boldsymbol{y} \leftarrow \hat{\boldsymbol{y}}$
    **end if**
**end for**
**return** $\boldsymbol{y}$

This algorithm is well known (see e.g., [23]), and its correctness relies on the following simple result, tracing back to [24].

**Proposition 2.2.** *For any class of models $\mathfrak{C}$, if a partial instance $\boldsymbol{y}$ of dimension $n$ is a sufficient reason for an instance $\boldsymbol{x}$ under a model $\mathcal{M} \in \mathfrak{C}$, but not a minimal sufficient reason, then there is a partial instance $\hat{\boldsymbol{y}}$ which is equal to $\boldsymbol{y}$ except that for some $i \in \{1, \ldots, n\}$ we have that $\hat{\boldsymbol{y}}[i] = \bot$ and $\boldsymbol{y}[i] \neq \bot$, and such that $\hat{\boldsymbol{y}}$ is also a sufficient reason for $\boldsymbol{x}$ under $\mathcal{M}$.*

The following theorem shows a stark contrast between the complexity of computing minimal and minimum sufficient reasons over decision trees.

Theorem 2.3 ([9]). *Assuming* P $\neq$ NP*, there is no polynomial-time algorithm that, given a decision tree $T$ and an instance $\boldsymbol{x}$ of the same dimension, computes a minimum sufficient reason for $\boldsymbol{x}$ under $T$.*

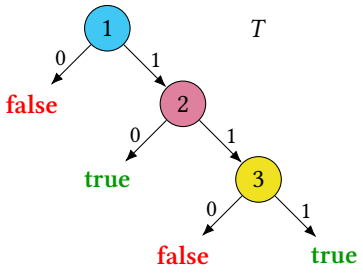
## 2.5 Probabilistic Sufficient Reasons

Arguably, the notion of sufficient reason is a natural notion of explanation for the result of a classifier. However, such a concept imposes a severe restriction by asking all completions of a partial instance to be classified in the same way. To overcome this limitation, a probabilistic generalization of sufficient reasons was proposed by [11] and [12]. More precisely, this notion allows to settle a confidence $\delta \in (0, 1]$[3] on the fraction of completions of a partial instance that yield the same classification.

**Definition 2.4** (Probabilistic sufficient reasons)**.** *Given a value $\delta \in (0, 1]$, a* $\delta$-sufficient reason *($\delta$-SR for short) for an instance $\boldsymbol{x}$ under a decision tree $T$ is a partial instance $\boldsymbol{y}$ such that $\boldsymbol{y} \subseteq \boldsymbol{x}$ and*

$$\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = T(\boldsymbol{x}) \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \;:=\; \frac{|\{\boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}) \mid T(\boldsymbol{z}) = T(\boldsymbol{x})\}|}{2^{|\boldsymbol{y}|_\bot}} \;\geqslant\; \delta.$$

[3] Given rational numbers $a, b$ with $a \leqslant b$, we use notation $[a, b]$ for the set of rational numbers $x$ such that $a \leqslant x \leqslant b$ (and $[a, b)$ for the set $\{x \in \mathbb{Q} \mid a \leqslant x < b\}$).




| | |
|---|---|
| $p(\emptyset) = \frac{3}{8}$ | $p(\{1,2\}) = \frac{1}{2}$ |
| $p(\{1\}) = \frac{3}{4}$ | $p(\{1,3\}) = 1$ |
| $p(\{2\}) = \frac{1}{4}$ | $p(\{2,3\}) = \frac{1}{2}$ |
| $p(\{3\}) = \frac{1}{2}$ | $p(\{1,2,3\}) = 1$ |

Fig. 2. The decision tree $T$ and the values $p(X)$ from Example 2.5.

Minimal and minimum $\delta$-sufficient reasons are defined analogously as in the case of minimal and minimum sufficient reasons.

**Example 2.5.** Consider the decision tree $T$ over features $\{1,2,3\}$ shown in Figure 2 and the input instance $\boldsymbol{x} = (1,1,1)$. Notice that $T(\boldsymbol{x}) = 1$. For each $X \subseteq \{1,2,3\}$, we show the probability $p(X) := \Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}_X)]$, where $\boldsymbol{y}_X$ is the partial instance that is obtained from $\boldsymbol{x}$ by fixing $\boldsymbol{y}_X[i] = \perp$ for each $i \notin X$. We observe, for instance, that $\boldsymbol{x}$ itself is neither a minimum nor a minimal 1-SR for $\boldsymbol{x}$ under $T$, as $\boldsymbol{y}_{\{1,3\}} = (1,\perp,1)$ is also a 1-SR. In turn, $\boldsymbol{y}_{\{1,3\}}$ is both a minimal and a minimum 1-SR for $\boldsymbol{x}$ under $T$. The partial instance $\boldsymbol{y}_{\{1,3\}}$ is not, however, a minimal or a minimum $3/4$-SR for $\boldsymbol{x}$ under $T$, as $\boldsymbol{y}_{\{1\}} = (1,\perp,\perp)$ is also a $3/4$-SR. □

**Example 2.6.** Consider the decision tree $T$ over features $\{1,2,3\}$ shown in Figure 3 and the input instance $\boldsymbol{x} = (1,1,1)$. Notice that $T(\boldsymbol{x}) = 1$. Exactly as in Example 2.5, we display as well the probabilities $p(X)$ for each $X \subseteq \{1,2,3\}$. Interestingly, this example illustrates that Proposition 2.2 does not hold when $\delta < 1$. Indeed, consider that $\boldsymbol{y}_{\{1,2,3\}}$ is a $5/8$-SR which is not minimal, as $\boldsymbol{y}_\emptyset$ is also a $5/8$-SR, but if we remove any single feature from $\boldsymbol{y}_{\{1,2,3\}}$, we obtain a partial instance which is not a $5/8$-SR. □

As illustrated on Example 2.6, it is not true in general that if $\boldsymbol{y}' \subseteq \boldsymbol{y}$ then

$$\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = T(\boldsymbol{x}) \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}')] \leqslant \Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = T(\boldsymbol{x}) \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})],$$

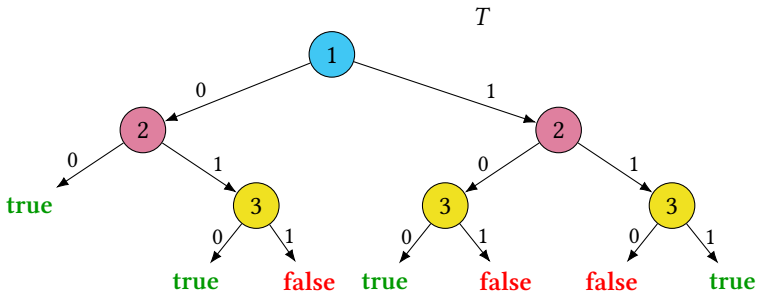
which means that standard algorithms for finding minimal sets holding monotone predicates (such as Algorithm 2) cannot be used to compute minimal $\delta$-SRs. In fact, we will show later that no polynomial time algorithm exists for computing minimal $\delta$-SRs (at least under complexity assumptions).

## 2.6 Computing Probabilistic Sufficient Reasons

The problems of computing minimum and minimal $\delta$-SR on decision trees were defined and left open by [12, 17]. These problems are formally defined as follows.

| | |
|---|---|
| PROBLEM: : | Compute-Minimum-SR |
| INPUT : | A decision tree $T$ of dimension $n$, an instance $\boldsymbol{x}$ of dimension $n$ and $\delta \in (0,1]$ |
| OUTPUT : | A minimum $\delta$-SR for $\boldsymbol{x}$ under $T$ |

| | |
|---|---|
| PROBLEM: : | Compute-Minimal-SR |
| INPUT : | A decision tree $T$ of dimension $n$, an instance $\boldsymbol{x}$ of dimension $n$ and $\delta \in (0,1]$ |
| OUTPUT : | A minimal $\delta$-SR for $\boldsymbol{x}$ under $T$ |




| | |
|---|---|
| $p(\emptyset) = \frac{5}{8}$ | $p(\{1,2\}) = \frac{1}{2}$ |
| $p(\{1\}) = \frac{1}{2}$ | $p(\{1,3\}) = \frac{1}{2}$ |
| $p(\{2\}) = \frac{1}{2}$ | $p(\{2,3\}) = \frac{1}{2}$ |
| $p(\{3\}) = \frac{1}{2}$ | $p(\{1,2,3\}) = 1$ |

Fig. 3. The decision tree $T$ and the values $p(X)$ from Example 2.6.

## 3 Main Results

In this section, we summarize the main results presented in this paper regarding the computation of probabilistic sufficient reasons over decision trees.

### 3.1 Hardness Results

We first establish that, if P ≠ NP, neither Compute-Minimum-SR nor Compute-Minimal-SR can be solved in polynomial time. We first consider the problem Compute-Minimum-SR, and in fact, prove a stronger result by considering the family of problems $\delta$-Compute-Minimum-SR which are obtained from Compute-Minimum-SR when $\delta \in (0, 1]$ is assumed to be fixed. More precisely, we obtain as a corollary of Theorem 2.3 that 1-Compute-Minimum-SR cannot be solved efficiently. Moreover, a non-trivial modification of the proof of this theorem shows that this negative result continues to hold for every fixed $\delta \in (0, 1]$.

Theorem 3.1. *Fix any value for $\delta \in (0, 1]$. Then, assuming that* P ≠ NP*, there is no polynomial-time algorithm for the* $\delta$-Compute-Minimum-SR *problem.*

Let us now look at the problem Compute-Minimal-SR. When $\delta = 1$, this problem can be solved in polynomial time as stated in Theorem 2.3. However, it was conjectured by [12] that assuming P ≠ NP, this positive behavior does not extend to the general problem Compute-Minimal-SR, in which $\delta$ is an input confidence parameter. Our main result confirms that this conjecture is correct.

Theorem 3.2. *Assuming that* P ≠ NP*, the* Compute-Minimal-SR *problem cannot be solved in polynomial time.*

The proof of this theorem heavily relies on the following result, which we believe to be of independent interest. The result establishes that the following problem over formulas in CNF, which we call Minimal-Expected-Clauses, is NP-hard. The problem Minimal-Expected-Clauses is defined as follows. Let $\varphi$ be a CNF formula over variables $X = \{x_1, \ldots, x_n\}$. Partial assignments of the variables in $X$, as well as the notions of subsumption and completions over them, are defined in the same way as for partial instances over features. For a partial assignment $\mu$ over

$X$, we denote by $E(\varphi, \mu)$ the expected number of clauses of $\varphi$ satisfied by a random completion of $\mu$. We then consider the following problem for fixed $k \geqslant 2$ (recall that a $k$-CNF formula is a CNF formula where each clause has at most $k$ literals):

| | |
|---|---|
| PROBLEM : | $k$-Minimal-Expected-Clauses |
| INPUT : | $(\varphi, \sigma)$, for $\varphi$ a $k$-CNF formula and $\sigma$ a partial assignment |
| OUTPUT : | Yes, if there is a $\mu \subsetneq \sigma$ such that $E(\varphi, \mu) \geqslant E(\varphi, \sigma)$ and No otherwise |

Theorem 3.3. *$k$-Minimal-Expected-Clauses is NP-hard for every $k \geqslant 2$.*

## 3.2 Non-Approximability Results

By assuming a stronger complexity assumption, namely, NP $\not\subseteq$ QP, where QP consists of all problems that have quasi-polynomial time algorithms, we can actually show stronger negative results that imply the non-approximability of the problems studied in this paper. Intuitively, we show that it is hard to distinguish cases when a very good sufficient reason with almost all variables left unfixed exists from cases when in all mildly good sufficient reasons just a few variables are left unfixed.

More precisely, for Compute-Minimum-SR we show the following:

Theorem 3.4. *Assuming that NP $\not\subseteq$ QP, for every $\varepsilon > 0$ there is no polynomial-time algorithm that, given a decision tree $T$ of dimension $n$ and an input $\boldsymbol{x} \in \{0,1\}^n$, distinguishes between the following two cases:*

- *there exists an $(1-\varepsilon)$-sufficient reason $\boldsymbol{y}$ for $\boldsymbol{x}$ under $T$ with $|\boldsymbol{y}|_\perp \geq n - n^\varepsilon$.*
- *there exists no $\varepsilon$-sufficient reason $\boldsymbol{y}$ for $\boldsymbol{x}$ under $T$ with $|\boldsymbol{y}|_\perp \geq n^\varepsilon$.*

We now address the Compute-Minimal-SR problem. We obtain the following inapproximability result.

Theorem 3.5. *Assuming that NP $\not\subseteq$ QP, for every $\varepsilon \in (0,1)$, there exists no polynomial-time algorithm that, given a decision tree $T$ of dimension $n$ and an input $\boldsymbol{x} \in \{0,1\}^n$, distinguishes between the following two cases:*

- *the only $(1/2+\varepsilon)$-sufficient reason for $\boldsymbol{x}$ under $T$ is $\boldsymbol{x}$ itself.*
- *there exists a $(1-\varepsilon)$-sufficient reason for $\boldsymbol{x}$ under $T$ other than $\boldsymbol{x}$.*

## 3.3 Tractability Results

We finally study restrictions on decision trees that lead to polynomial time algorithms for Compute-Minimum-SR or Compute-Minimal-SR, hence avoiding the general intractability results shown in the previous section. We identify two such restrictions: *bounded split number* and *monotonicity*.

*3.3.1 Bounded Split Number.* Let $T$ be a decision tree of dimension $n$. For a set $U$ of nodes of $T$, we denote by $\mathcal{F}(U)$ the set of features from $\{1, \ldots, n\}$ labeling the nodes in $U$. For each node $u$ of $T$, we denote by $N_u^{\downarrow}$ the set of nodes appearing in $T_u$, that is, the subtree of $T$ rooted at $u$. On the other hand, we denote by $N_u^{\uparrow}$ the set of nodes of $T$ minus the set of nodes $N_u^{\downarrow}$. We define the *split number* of the decision tree $T$ to be

$$\max_{\text{node } u \text{ in } T} \left| \mathcal{F}\left(N_u^{\downarrow}\right) \cap \mathcal{F}\left(N_u^{\uparrow}\right) \right|.$$

Intuitively, the split number of a decision tree $T$ is a measure of the interaction (number of common features) between the subtrees of the form $T_u$ and their exterior. A small split number allows us to essentially treat each subtree $T_u$ independently (in particular, the left and right subtrees below any node), which in turn leads to efficient algorithms for the problems Compute-Minimum-SR and Compute-Minimal-SR.

Theorem 3.6. *Both problems Compute-Minimum-SR and Compute-Minimal-SR can be solved in polynomial time for decision trees with split number at most $c$, where $c \geqslant 1$ is any fixed integer.*

*3.3.2 Monotonicity.* Classifiers that are *monotone* have been studied in the context of XAI as they often present tractable cases for different explanations, as shown in [10]. The computation of minimal sufficient reasons for monotone models was known to be in polynomial time [24]. We show that this is also the case for computing minimal $\delta$-SRs under a mild assumption on the class of models.

Let us define the ordering $\preceq$ for instances in $\{0,1\}^n$ as follows:

$$\boldsymbol{x} \preceq \boldsymbol{z} \quad \text{iff} \quad \boldsymbol{x}[i] \leqslant \boldsymbol{z}[i], \text{ for all } i \in \{1,\ldots,n\}.$$

We can now define monotonicity as follows. A model $\mathcal{M}$ of dimension $n$ is said to be *monotone* if for every pair of instances $\boldsymbol{x}, \boldsymbol{z} \in \{0,1\}^n$, it holds that

$$\boldsymbol{x} \preceq \boldsymbol{z} \implies \mathcal{M}(\boldsymbol{x}) \leqslant \mathcal{M}(\boldsymbol{z}).$$

We now prove that the problem of computing minimal probabilistic sufficient reasons can be solved in polynomial time for any class $\mathfrak{C}$ of monotone Boolean models for which the problem of counting positive completions can be solved efficiently. Formally, the latter problem is defined as follows: given a model $\mathcal{M} \in \mathfrak{C}$ of dimension $n$ and a partial instance $\boldsymbol{y} \in \{0,1,\bot\}^n$, compute

$$|\{\boldsymbol{x} \in \textsc{Comp}(\boldsymbol{y}) \mid \mathcal{M}(\boldsymbol{x}) = 1\}|.$$

We call this problem $\mathfrak{C}$-#Positive-Completions.

Theorem 3.7. *Let $\mathfrak{C}$ be a class of monotone Boolean models such that $\mathfrak{C}$-#Positive-Completions can be solved in polynomial time. Then the problem Compute-Minimal-SR can be solved in polynomial time over $\mathfrak{C}$.*

As a corollary, the computation of minimal probabilistic sufficient reasons can be carried out in polynomial time not only over monotone decision trees, but also over monotone and free BDDs.

**Corollary 3.8.** *The problem of computing minimal $\delta$-SRs can be solved in polynomial time over the class of monotone and free BDDs.*

## 4 Proofs of the Hardness Results

We start by stating an auxiliary lemma that we will need for Theorem 3.1.

**Lemma 4.1.** *Fix $\delta \in (0,1)$. Given as input an integer $n$ one can build in $n^{O(1)}$ time a decision tree $T_\delta$ of dimension $n$, such that*

$$\delta - \frac{1}{2^n} \leqslant \Pr_{\boldsymbol{z}}\left[T_\delta(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\bot^n)\right] \leqslant \delta,$$

*and moreover, there exists an instance $\boldsymbol{x}^\dagger$ for $T_\delta$ such that every partial instance $\boldsymbol{y} \subseteq \boldsymbol{x}^\dagger$ holds*

$$\Pr_{\boldsymbol{z}}\left[T_\delta(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})\right] \leqslant \Pr_{\boldsymbol{z}}\left[T_\delta(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\bot^n)\right] \leqslant \delta.$$

Let us now restate and prove Theorem 3.1 assuming Lemma 4.1, and after that we present the proof of Lemma 4.1.

Theorem 3.1 (Restated). *Fix any value for $\delta \in (0,1]$. Then, assuming that* P $\neq$ NP*, there is no polynomial-time algorithm for the $\delta$-*Compute-Minimum-SR *problem.*

Proof of Theorem 3.1. We prove that deciding whether a $\delta$-SR of size $k$ exists is NP-hard. We reduce from the case $\delta = 1$, proved NP-hard by [9]. We assume of course that $\delta < 1$, as otherwise the result is already known.

Let $(T, \boldsymbol{x}, k)$ be an input of the Minimum Sufficient Reason problem (i.e., $\delta = 1$), and let $n$ be the dimension of $T$ and $\boldsymbol{x}$. Assume without loss of generality that $T(\boldsymbol{x}) = 1$. If the given input of Minimum Sufficient Reason is

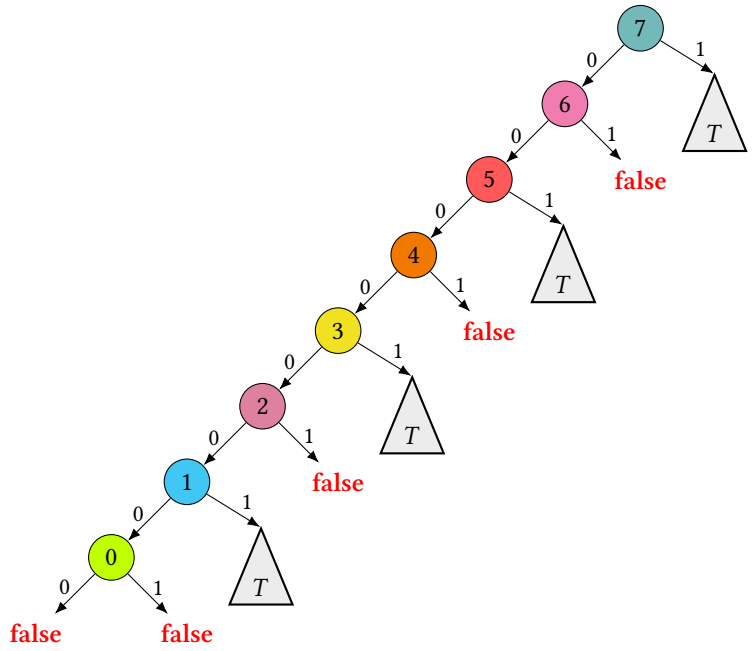



Fig. 4. Illustration of the construction of $F_\delta$ for $\delta = \frac{2}{3}$ and $n = 2$. Thus $2n + 3 + \lceil \log(1/\delta) \rceil = 8$ and $c = \lfloor \frac{2}{3} \cdot 2^8 \rfloor = 170 = 2^7 + 2^5 + 2^3 + 2^1$.

positive, then there is a partial instance $\boldsymbol{y} \subseteq \boldsymbol{x}$ with $|\boldsymbol{y}|_\perp \geqslant n - k$ such that $\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] = 1$, and otherwise for every partial instance $\boldsymbol{y} \subseteq \boldsymbol{x}$ with $|\boldsymbol{y}|_\perp \geqslant n - k$ it holds that

$$\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 0 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \geqslant \frac{1}{2^n}.$$

Let us build a tree $F_\delta$ with $3n+3+\lceil \log(1/\delta) \rceil$[4] variables as follows. First build $T_\delta$ of dimension $2n+3+\lceil \log(1/\delta) \rceil$ by using Lemma 4.1, and then replace every true leaf of $T_\delta$ by a copy of $T$. Assume the $2n+3+\lceil \log(1/\delta) \rceil$ variables of $T_\delta$ are disjoint from the $n$ variables that appear in $T$, and thus $F_\delta$ has the proposed number of variables. An example of the construction of $F_\delta$ is illustrated in Figure 4.

Define

$$\delta' \coloneqq \Pr_{\boldsymbol{z}}\left[T_\delta(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}\left(\perp^{2n+2+\lceil \log(1/\delta) \rceil}\right)\right],$$

and recall that $|\delta' - \delta| \leqslant \frac{1}{2^{2n+3+\lceil \log(1/\delta) \rceil}}$. Now, let us build a final decision tree $T^\star$ with $4n + k + 4 + \lceil \log(1/\delta') \rceil + \lceil \log(1/\delta) \rceil$ variables as follows. Create $\ell \coloneqq n + k + 1 + \lceil \log(1/\delta') \rceil$ vertices, labeled $r_i$ for $i \in \{1, \ldots, \ell\}$, and assume these labels are disjoint from the ones used in $F_\delta$. Let $r_1$ be the root of $T^\star$, and for each $i \in \{1, \ldots, \ell - 1\}$, connect vertex labeled with $r_i$ to vertex labeled with $r_{i+1}$ using a 0-edge. The 0-edge from vertex labeled $r_\ell$ goes towards a leaf labeled with **true**. The 1-edge from every vertex $r_i$ goes towards the root of a different copy of $F_\delta$. Note that this construction, illustrated in Figure 5, takes polynomial time. Now, consider the instance $\boldsymbol{x}^\star$ that is defined (i) exactly as $\boldsymbol{x}$ for the variables of $T$, (ii) exactly as in the instance $\boldsymbol{x}^\dagger$ coming from Lemma 4.1 for the variables of $T_\delta$ in $F_\delta$, and (iii) with all variables $r_i$ set to 0. Note that $T^\star(\boldsymbol{x}^\star) = 1$. Now we prove both directions of the reduction separately. Assume first that the instance $(T, \boldsymbol{x}, k)$ is a positive instance for Minimum Sufficient

[4]We use notation $\log(x)$ to refer to the logarithm in base 2 of $x$

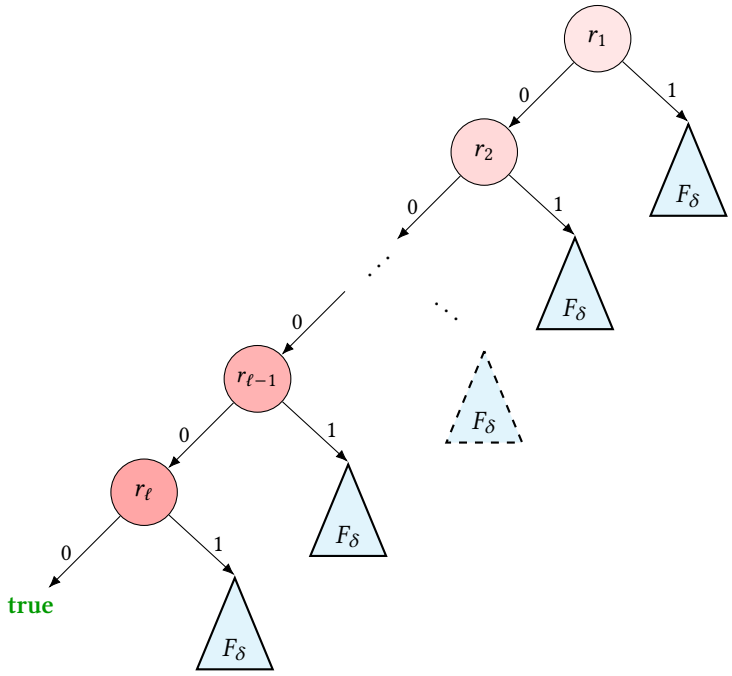



Fig. 5. Illustration of the construction of $T^\star$. Recall that $\ell = n + k + 1 + \lceil \log(1/\delta') \rceil$.

Reason. Then we claim that there is a $\delta$-SR for $T^\star$ of size at most $k$. Indeed, let $\boldsymbol{y} \subseteq \boldsymbol{x}$ be a sufficient reason for $\boldsymbol{x}$ under $T$ with at most $k$ defined components. Then consider the partial instance $\boldsymbol{y}^\star \subseteq \boldsymbol{x}^\star$, that is only defined in the components where $\boldsymbol{y}$ is defined. Now let us study $\Pr_{\boldsymbol{z}}[T^\star(\boldsymbol{z}') = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}^\star)]$. The probability that $\boldsymbol{z}$ ends up in the true leaf on the 0-edge from vertex $r_\ell$ is $\frac{1}{2^\ell}$. In any other case, $\boldsymbol{z}$ takes a path that goes into a copy of $F_\delta$, where its probability of acceptance is $\delta' \geqslant \delta - \frac{1}{2^{2n+3+\lceil \log(1/\delta) \rceil}}$ because of Lemma 4.1 and using that $\boldsymbol{y}^\star$ is undefined for all the variables of $T_c$. These two facts imply that

$$\Pr_{\boldsymbol{z}}[T^\star(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}^\star)] \geqslant \frac{1}{2^\ell} + \delta - \frac{1}{2^{2n+3+\lceil \log(1/\delta) \rceil}}.$$

Now consider that

$$\begin{aligned} \delta &\leqslant \delta' + \frac{1}{2^{2n+3+\lceil \log(1/\delta) \rceil}} = \delta' + \frac{1}{2^{2n+3}} \cdot \frac{1}{2^{\lceil \log(1/\delta) \rceil}} \\ &\leqslant \delta' + \frac{1}{2^{2n+3}} \cdot \frac{1}{2^{\log(1/\delta)}} = \delta' + \frac{\delta}{2^{2n+3}}, \end{aligned}$$

from where $\delta\left(1-\frac{1}{2^{2n+3}}\right) \leqslant \delta'$, and thus

$$\begin{aligned}
\log(1/\delta) &\geqslant \log(1/\delta') + \log\left(1-\frac{1}{2^{2n+3}}\right) \\
&= \log(1/\delta') - \log\left(\frac{2^{2n+3}}{2^{2n+3}-1}\right) \\
&= \log(1/\delta') - 2n - 3 + \log(2^{2n+3}-1) \\
&\geqslant \log(1/\delta') - 2n - 3 + 2n + 2 && \text{(using } 2^{2n+3}-1 \geqslant 2^{2n+2}\text{)} \\
&= \log(1/\delta') - 1.
\end{aligned}$$

From this we obtain that

$$\begin{aligned}
\ell &= n + k + 1 + \lceil \log(1/\delta') \rceil \\
&\leqslant 2n + 1 + \lceil \log(1/\delta') \rceil \\
&\leqslant 2n + 1 + \log(1/\delta') + 1 \\
&\leqslant 2n + 3 + \log(1/\delta) \\
&\leqslant 2n + 3 + \lceil \log(1/\delta) \rceil,
\end{aligned}$$

which allows us to conclude that

$$\Pr_{\boldsymbol{z}}[T^\star(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}^\star)] \geqslant \frac{1}{2^\ell} + \delta - \frac{1}{2^{2n+3+\lceil \log(1/\delta) \rceil}} \geqslant \delta.$$

On the other hand, if $(T, \boldsymbol{x}, k)$ is a negative instance for Minimum Sufficient Reason, consider any partial instance $\boldsymbol{y}^\star \subseteq \boldsymbol{x}^\star$ with at most $k$ defined components, and note that by hypothesis we have that $\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}^\star)] \leqslant 1 - \frac{1}{2^n}$. This implies, together with the second part of Lemma 4.1, that

$$\Pr_{\boldsymbol{z}}[F_\delta(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}^\star)] \leqslant \delta\left(1-\frac{1}{2^n}\right),$$

and thus subsequently

$$\Pr_{\boldsymbol{z}}[T^\star(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}^\star)] \leqslant \delta\left(1-\frac{1}{2^n}\right) + \frac{1}{2^{\ell-k}},$$

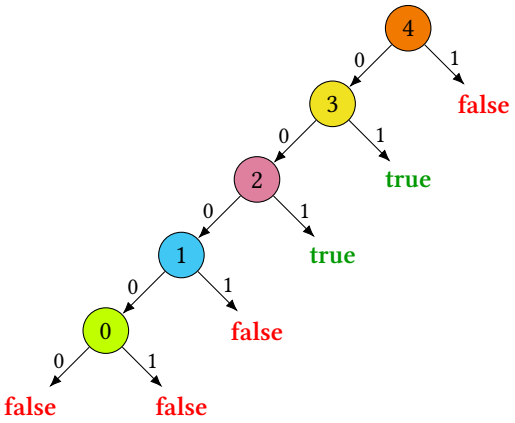



Fig. 6. Example of $T_c$, the tree constructed in the proof of Lemma 4.1, for $n = 5$ and $\delta = \frac{2}{5}$. In this case $c = 12 = 2^2 + 2^3$. Note that $\Pr_{\boldsymbol{z}}[T_c(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\bot^n)] = \frac{c}{2^n} = \frac{12}{32}$, and $\left|\frac{2}{5} - \frac{12}{32}\right| = \frac{1}{40} < \frac{1}{32}$.

by using that with at most $k$ defined components in $T^\star$, the probability of reaching the true leaf across the 0-edge from $r_\ell$ is at most $\frac{1}{2^{\ell-k}}$. To conclude, note that

$$\begin{aligned}
\Pr_{\boldsymbol{z}}[T^\star(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}^\star)] &\leqslant \delta\left(1 - \frac{1}{2^n}\right) + \frac{1}{2^{\ell-k}} \\
&= \delta - \frac{\delta}{2^n} + \frac{1}{2^{n+1+\lceil\log(1/\delta')\rceil}} \\
&\leqslant \delta - \frac{\delta}{2^n} + \frac{1}{2^{n+1+\log(1/\delta')}} \\
&= \delta + \frac{(\delta' - \delta) - \delta}{2^{n+1}} \\
&\leqslant \delta + \frac{\frac{1}{2^{2n+3+\lceil\log(1/\delta)\rceil}} - \delta}{2^{n+1}} \\
&\leqslant \delta + \frac{\frac{\delta}{2^{2n+3}} - \delta}{2^{n+1}} \\
&= \delta + \delta\left(\frac{-1 + \frac{1}{2^{2n+3}}}{2^{n+1}}\right) \\
&< \delta.
\end{aligned}$$

We have thus concluded that $\boldsymbol{y}^\star$ is a $\delta$-SR for $\boldsymbol{x}^\star$ over $T^\star$ if and only if $(T, \boldsymbol{x}, k)$ is a positive instance of Minimum Sufficient Reason, which completes our reduction. □

Now we settle our debt by proving Lemma 4.1, the only part missing from the previous proof.

Proof of Lemma 4.1. Let $c = \lfloor \delta 2^n \rfloor$, and note that $\delta - \frac{1}{2^n} \leqslant \frac{c}{2^n} \leqslant \delta$, from where $|\delta - \frac{c}{2^n}| \leqslant \frac{1}{2^n}$. This implies that we can prove the first part of the lemma by building in polynomial time a tree $T_c$ over $n$ variables, that has exactly $c$ different positive instances, as then its probability of accepting a random completion of $\bot^n$ will be exactly $\frac{c}{2^n}$. Note as well that $c < 2^n$ as $\delta < 1$.

As a first step, let us write $c$ in binary, obtaining

$$c = \alpha_0 2^0 + \alpha_1 2^1 + \cdots + \alpha_{n-1} 2^{n-1},$$

with $\alpha_i \in \{0, 1\}$ for each $i$. Then to build $T_c$ start by creating $n$ vertices, labeled 0 through $n-1$. These $n$ labels are the variables of $T_c$. For each $i \in \{1, \ldots, n-1\}$, connect vertex labeled $i$ to vertex labeled $i-1$ with a 0-edge, making vertex labeled $n-1$ the root of $T_c$. Then, for each vertex with label $i \in \{0, \ldots, n-1\}$, set its 1-edge towards a leaf with label **true** if $\alpha_i = 1$, and towards a leaf with label **false** if $\alpha_i = 0$. The 0-edge of vertex labeled 0 goes towards a leaf with label **false**. Now let us count how many different positive instances does $T_c$ have. We can do this by summing over all true leaves of $T_c$. Each true leaf comes from a 1-edge from a vertex labeled $i \in \{0, \ldots, n-1\}$. For every $i \in \{0, \ldots, n-1\}$, if the vertex labeled $i$ has a true leaf when following its 1-edge, then the number of instances reaching that true leaf is exactly $2^i$, as the variables whose value is not determined by the path to that leaf are those with labels less than $i$, which are exactly $i$ variables. Therefore, the number of different positive instances of $T_c$ along a 1-edge is the sum of $2^i$ for every $i$ such that $\alpha_i = 1$, which is exactly $c$. An example is given in Figure 6. This concludes the proof of the first part of the lemma as the construction is clearly polynomial in $n$. For the second part, let us build $\boldsymbol{x}^\dagger$ by setting $\boldsymbol{x}^\dagger[i] = 1 - \alpha_i$ for every $i \in \{0, \ldots, n-1\}$. In the example presented in Figure 6, we would build

$$\boldsymbol{x}^\dagger = \big(1, \quad 1, \quad 0, \quad 0, \quad 1\big).$$

We now prove that for any $\boldsymbol{y} \subseteq \boldsymbol{x}^\dagger$, it holds that

$$\Pr_{\boldsymbol{z}}\left[T_c(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})\right] \leqslant \Pr_{\boldsymbol{z}}\left[T_c(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\bot^n)\right].$$

We do this via a finite induction argument by strengthening our induction hypothesis; for $i \in \{0, \ldots, n-1\}$, let $T_c^i$ be the sub-tree of $T_c$ rooted at the vertex labeled $i$, and let us claim that for every $i \in \{0, \ldots, n-1\}$ we have that

$$\Pr_{\boldsymbol{z}}\left[T_c^i(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})\right] \leqslant \Pr_{\boldsymbol{z}}\left[T_c^i(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\bot^n)\right],$$

which implies what we want to show when taking $i = n-1$. The base case of the induction is when $i = 0$, in which case the claim trivially holds as if $\boldsymbol{y}[0] = \bot$ we have equality, and if $\boldsymbol{y}[0] \neq \bot$ then by construction

$$\Pr_{\boldsymbol{z}}\left[T_c^i(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})\right] = 0.$$

For the inductive case, let $i > 0$, and proceed by cases; if $\boldsymbol{y}[i] = \bot$, then by letting $t_i \in \{0, 1\}$ be an indicator variable for whether the leaf across the 1-edge from vertex $i$ is labeled **true** we have that

$$\begin{aligned}
\Pr_{\boldsymbol{z}}\left[T_c^i(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})\right] &= \frac{1}{2}t_i + \frac{1}{2}\Pr_{\boldsymbol{z}}\left[T_c^{i-1}(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})\right] \\
&\leqslant \frac{1}{2}t_i + \frac{1}{2}\Pr_{\boldsymbol{z}}\left[T_c^{i-1}(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\bot^n)\right] \\
&= \Pr_{\boldsymbol{z}}\left[T_c^i(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\bot^n)\right],
\end{aligned}$$

where the inequality has used the inductive hypothesis. On the other hand, if $\boldsymbol{y}[i] = 1$, that implies $\boldsymbol{x}^\dagger[i] = 1$ and thus $\alpha_i = 0$, which means the leaf across the 1-edge from vertex $i$ is labeled with **false**, and thus

$$Pr_{\boldsymbol{z}}\left[T_c^i(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})\right] = 0,$$

which trivially satisfies the claim. For the last case, if $\boldsymbol{y}[i] = 0$, then $\boldsymbol{x}^\dagger[i] = 0$ and thus $\alpha_i = 1$, which means the leaf across the 1-edge from vertex $i$ is labeled with **true**. Therefore, we have

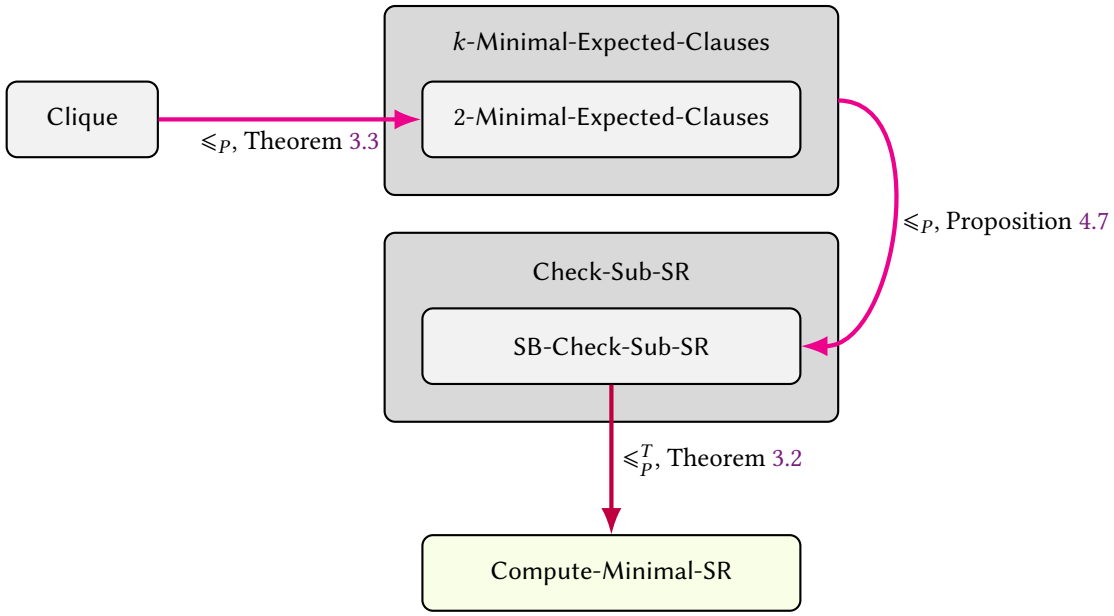



Fig. 7. Roadmap for the proof of Theorem 3.2. Arrows denote polynomial-time many-to-one reductions ($\leqslant_P$) or Turing reductions ($\leqslant_P^T$).

$$\begin{aligned}
\Pr_z\left[T_c^i(z) = 1 \mid z \in \textsc{Comp}(y)\right] &= \Pr_z\left[T_c^{i-1}(z) = 1 \mid z \in \textsc{Comp}(y)\right] \\
&\leqslant \Pr_z\left[T_c^{i-1}(z') = 1 \mid z \in \textsc{Comp}(\bot^n)\right] \\
&\leqslant \frac{1}{2} + \frac{1}{2}\Pr_z\left[T_c^{i-1}(z) = 1 \mid z \in \textsc{Comp}(\bot^n)\right] && \text{(as } \Pr[\cdot] \leqslant 1\text{)} \\
&= \Pr_z\left[T_c^{i}(z) = 1 \mid z \in \textsc{Comp}(\bot^n)\right].
\end{aligned}$$

This completes the induction argument, and thus we conclude the proof of the lemma. □

Our goal now is to prove Theorem 3.2. Figure 7 presents a roadmap of the proof with its corresponding chain of reductions. Following the roadmap, we need first Theorem 3.3, which we restate and prove next.

Theorem 3.3 (Restated). $k$-Minimal-Expected-Clauses *is* NP*-hard for every* $k \geqslant 2$.

Proof of Theorem 3.3. We start with some simple observations regarding $E(\varphi, \mu)$, for a CNF formula $\varphi$ and a partial assignment $\mu$. By linearity of expectation, we can write $E(\varphi, \mu)$ as the sum

$$E(\varphi, \mu) = \sum_{C \text{ clause of } \varphi} \text{Prob}_{C,\mu}, \tag{4.2}$$

where $\text{Prob}_{C,\mu}$ is the probability that a random completion of $\mu$ satisfies the clause $C$. In turn, the probabilities $\text{Prob}_{C,\mu}$ can be easily computed as:

- $\text{Prob}_{C,\mu} = 1$, if there is a positive literal $x$ in $C$ with $\mu(x) = 1$ or there a negative literal $\neg x$ in $C$ with $\mu(x) = 0$.

- $\mathsf{Prob}_{C,\mu} = 1 - \frac{1}{2^\eta}$, where $\eta$ is the number of literals in $C$ of the form $x$ or $\neg x$ with $\mu(x) = \bot$.

Note that for an assignment $\sigma$, $E(\varphi, \sigma)$ is simply the number of clauses of $\varphi$ satisfied by $\sigma$.

We reduce from the Clique problem. Recall this problem asks, given a graph $G$ and an integer $k \geqslant 3$, whether there is a *clique* of size $k$, that is, a set $K$ of $k$ vertices such that there is an edge between any pair of distinct vertices from $K$. Let $G$ be a graph and $k \geqslant 3$. We can assume without loss of generality that $k$ is odd and the degree of every vertex $x$ of $G$, denoted by $\deg(x)$, is at least $k-1$; if $k$ is even we can consider the equivalent instance given by the graph $G'$ that extends $G$ with a fresh node connected via an edge with all the other nodes and $k' = k+1$. On the other hand, we can iteratively remove vertices of degree less than $k-1$ as those cannot be part of any clique of size $k$. Starting from the Clique instance $(G, k)$ we will construct an instance $(\varphi, \sigma)$ for 2-Minimal-Expected-Clauses as follows. The variables of $\varphi$ are the nodes of $G$. For each variable $x$ we have the following clauses in $\varphi$:

- A clause $A_x = (\neg x)$. This clause $A_x$ is repeated $\deg(x) - \frac{k-1}{2}$ times in $\varphi$. Note this quantity is always a positive integer since $k$ is odd.
- A set of clauses $\mathcal{B}_x = \{(x \vee \neg y_1), \ldots, (x \vee \neg y_{\deg(x)})\}$, where $y_1, \ldots, y_{\deg(x)}$ are the neighbors of $x$ in $G$. Each clause in $\mathcal{B}_x$ appears only once in $\varphi$.

Additionally, for each set $\{x, y\}$ where $x \neq y$ and $\{x, y\}$ is *not* an edge in $G$, we have a clause $Z_{x,y} = (x \vee y)$ repeated $4e$ times in $\varphi$, where $e$ is the number of edges in $G$.

We define the assignment $\sigma$ such that $\sigma(x) = 1$, for all variables $x$ of $\varphi$.

For an arbitrary partial assignment $\mu$ to the variables of $\varphi$, with $\mu \subseteq \sigma$, we define

$$\mathsf{utility}_{\varphi,\sigma}(\mu) \coloneqq E(\varphi, \mu) - E(\varphi, \sigma).$$

In particular, the instance $(\varphi, \sigma)$ is a Yes-instance for 2-Minimal-Expected-Clauses if and only if there is $\mu \subsetneq \sigma$ with $\mathsf{utility}_{\varphi,\sigma}(\mu) \geqslant 0$. By equation (4.2), we can write

$$\mathsf{utility}_{\varphi,\sigma}(\mu) = \sum_{C \text{ clause of } \varphi} \mathsf{utility}_{\varphi,\sigma}(\mu, C),$$

where $\mathsf{utility}_{\varphi,\sigma}(\mu, C)$ is defined as:

$$\mathsf{utility}_{\varphi,\sigma}(\mu, C) \coloneqq \mathsf{Prob}_{C,\mu} - \mathsf{Prob}_{C,\sigma}.$$

Now observe that we have:

$$\mathsf{Prob}_{C,\sigma} = \begin{cases} 0 & \text{if } C = A_x \text{ for some variable } x \\ 1 & \text{if } C \in \mathcal{B}_x \text{ for some variable } x \text{ or } C = Z_{x,y} \text{ for some set } \{x, y\} \end{cases}$$

On the other hand, for the probability $\mathsf{Prob}_{C,\mu}$ we have the following:

(1) Assume $C = A_x = (\neg x)$ for some variable $x$. Then $\mathsf{Prob}_{C,\mu}$ is
   (a) $\frac{1}{2}$, if $\mu(x) = \bot$ (and hence $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = \frac{1}{2}$), and
   (b) 0 otherwise (then $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = 0$).

(2) Suppose $C = (x \vee \neg y) \in \mathcal{B}_x$ for some variable $x$. Then $\mathsf{Prob}_{C,\mu}$ is
   (a) $\frac{3}{4}$ if $\mu(x) = \bot$ and $\mu(y) = \bot$ (and hence $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = -\frac{1}{4}$),
   (b) $\frac{1}{2}$ if $\mu(x) = \bot$ and $\mu(y) = 1$ (and then $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = -\frac{1}{2}$), and
   (c) 1 otherwise (then $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = 0$).

(3) Suppose $C = Z_{x,y} = (x \vee y)$ for some set $\{x, y\}$. Then $\mathsf{Prob}_{C,\mu}$ is
   (a) $\frac{3}{4}$ if $\mu(x) = \bot$ and $\mu(y) = \bot$ (and hence $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = -\frac{1}{4}$), and
   (b) 1 otherwise (then $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = 0$).

We now show the correctness of our construction, that is, $G$ has a clique of size $k \geqslant 3$ if and only if $(\varphi, \sigma)$ is a Yes-instance for the problem 2-Minimal-Expected-Clauses.
($\Rightarrow$) Suppose $G$ has a clique $K$ of size $k \geqslant 3$. Let $\mu$ be the partial assignment that sets $\mu(x) = \bot$ if $x \in K$ and $\mu(x) = 1$ if $x \notin K$. Note that $\mu \subsetneq \sigma$. We claim that $\mathsf{utility}_{\varphi,\sigma}(\mu) = 0$ and hence $(\varphi, \sigma)$ is a Yes-instance. Let $C$ be a clause in $\varphi$. If $C$ is of the form $Z_{x,y}$, then $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = 0$. Indeed, by construction, $\{x, y\}$ is not an edge, and since $K$ is a clique, then $\mu(x) = 1$ or $\mu(y) = 1$. This means we are always in case 3(b) above. If $x \notin K$ and $C$ is of the form $A_x$ or belongs to $\mathcal{B}_x$, then $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = 0$, since $\mu(x) = 1$ and hence we fall either in case 1(b) or 2(c) above. It follows that $\mathsf{utility}_{\varphi,\sigma}(\mu)$ is the sum of the utilities of all the clauses involved with variables $x \in K$. That is:

$$\mathsf{utility}_{\varphi,\sigma}(\mu) = \sum_{x \in K} \left( \deg(x) - \frac{k-1}{2} \right) \mathsf{utility}_{\varphi,\sigma}(\mu, A_x) + \sum_{x \in K} \sum_{C \in \mathcal{B}_x} \mathsf{utility}_{\varphi,\sigma}(\mu, C). \tag{4.3}$$

Take $x \in K$. Then $\mathsf{utility}_{\varphi,\sigma}(\mu, A_x) = \frac{1}{2}$ as $\mu(x) = \bot$, and then case 1(a) applies. On the other hand, for a clause $C \in \mathcal{B}_x$ we have two cases:

- $C = (x \vee \neg y)$ for $y \in K$. In this case, $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = -\frac{1}{4}$ as we are in case 2(a) above.
- $C = (x \vee \neg y)$ for $y \notin K$. In this case, $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = -\frac{1}{2}$ as we are in case 2(b) above.

Moreover, note that the first case occurs exactly for $k - 1$ clauses in $\mathcal{B}_x$, as $x$ has precisely $k - 1$ neighbors in the clique $K$. The second case occurs exactly for $\deg(x) - (k - 1) \geqslant 0$ clauses in $\mathcal{B}_x$. Replacing in equation (4.3), we obtain:

$$\begin{aligned} \mathsf{utility}_{\varphi,\sigma}(\mu) &= \sum_{x \in K} \left( \frac{k-1}{4} + \frac{\deg(x)}{2} - \frac{k-1}{2} \right) + \left( -\frac{k-1}{4} - \frac{\deg(x)}{2} + \frac{k-1}{2} \right) \\ &= 0. \end{aligned}$$

We conclude that $(\varphi, \sigma)$ is a Yes-instance.
($\Leftarrow$) Suppose now that there is a partial assignment $\mu$, with $\mu \subsetneq \sigma$ and $\mathsf{utility}_{\varphi,\sigma}(\mu) \geqslant 0$. Let $K$ be the set of variables $x$ such that $\mu(x) = \bot$. For $x \notin K$ and $C = A_x$ or $C \in \mathcal{B}_x$, we have $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = 0$, as we are in cases 1(b) or 2(c) above. Then we can write:

$$\begin{aligned} \mathsf{utility}_{\varphi,\sigma}(\mu) = \sum_{x \in K} \left[ \left( \deg(x) - \frac{k-1}{2} \right) \mathsf{utility}_{\varphi,\sigma}(\mu, A_x) + \sum_{C \in \mathcal{B}_x} \mathsf{utility}_{\varphi,\sigma}(\mu, C) \right] \\ + \sum_{\{x,\, y\} \text{ non-edge}} 4e \left( \mathsf{utility}_{\varphi,\sigma}(\mu, Z_{x,y}) \right). \end{aligned} \tag{4.4}$$

We claim that $|K| \geqslant k$. Towards a contradiction, suppose $|K| = \ell < k$. Since $\mathsf{utility}_{\varphi,\sigma}(\mu, Z_{x,y}) \leqslant 0$ for every pair $\{x, y\}$, the last term in equation (4.4) is $\leqslant 0$, and then:

$$\mathsf{utility}_{\varphi,\sigma}(\mu) \leqslant \sum_{x \in K} \left( \frac{k-1}{2} + \deg(x) - (k-1) \right) \mathsf{utility}_{\varphi,\sigma}(\mu, A_x) + \sum_{x \in K} \sum_{C \in \mathcal{B}_x} \mathsf{utility}_{\varphi,\sigma}(\mu, C). \tag{4.5}$$

Take $x \in K$. Following the same argument as before, we have that

$$\mathsf{utility}_{\varphi,\sigma}(\mu, A_x) = \frac{1}{2},$$

and for a clause $C \in \mathcal{B}_x$ we have the two cases:

- $C = (x \vee \neg y)$ for $y \in K$, and $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = -\frac{1}{4}$.
- $C = (x \vee \neg y)$ for $y \notin K$, and $\mathsf{utility}_{\varphi,\sigma}(\mu, C) = -\frac{1}{2}$.

Let say the first case occurs precisely for $r$ clauses from $\mathcal{B}_x$. Then:

$$\sum_{C \in \mathcal{B}_x} \mathsf{utility}_{\varphi,\sigma}(\mu, C) = -\frac{r}{4} - \frac{\deg(x) - r}{2} = \frac{r}{4} - \frac{\deg(x)}{2}. \tag{4.6}$$

Note that $r \leqslant \ell - 1$ and from equation (4.6) we obtain (recall $\ell < k$):

$$\sum_{C \in \mathcal{B}_x} \mathsf{utility}_{\varphi,\sigma}(\mu, C) \leqslant \frac{\ell - 1}{4} - \frac{\deg(x)}{2} < \frac{k-1}{4} - \frac{\deg(x)}{2}.$$

Replacing in equation (4.5), we obtain:

$$\begin{aligned}
\mathsf{utility}_{\varphi,\sigma}(\mu) &< \sum_{x \in K} \left[ \left( \frac{k-1}{4} + \frac{\deg(x)}{2} - \frac{k-1}{2} \right) + \frac{k-1}{4} - \frac{\deg(x)}{2} \right] \\
&= \sum_{x \in K} \left[ \frac{\deg(x)}{2} - \frac{k-1}{4} + \frac{k-1}{4} - \frac{\deg(x)}{2} \right] \\
&= 0.
\end{aligned}$$

We conclude that $\mathsf{utility}_{\varphi,\sigma}(\mu) < 0$ which is a contradiction. Hence $|K| \geqslant k$.

Finally, we show that $K$ is a clique. By contradiction, assume there is a pair $\{\tilde{x}, \tilde{y}\}$ such that $\tilde{x} \neq \tilde{y}$, $\tilde{x}, \tilde{y} \in K$ and $\{\tilde{x}, \tilde{y}\}$ is not an edge in $G$. Then there is a clause $Z_{\tilde{x},\tilde{y}}$ which is repeated $e$ times in $\varphi$. Since $\mu(\tilde{x}) = \bot$ and $\mu(\tilde{y}) = \bot$, we have $\mathsf{utility}_{\varphi,\sigma}(\mu, Z_{\tilde{x},\tilde{y}}) = -\frac{1}{4}$, as we are in case 3(a) above. As $\mathsf{utility}_{\varphi,\sigma}(\mu, Z_{x,y}) \leqslant 0$ for all pairs $\{x, y\}$, we obtain:

$$\sum_{\{x,\, y\} \text{ non-edge}} 4e \left(\mathsf{utility}_{\varphi,\sigma}(\mu, Z_{x,y})\right) \leqslant 4e \left(\mathsf{utility}_{\varphi,\sigma}(\mu, Z_{\tilde{x},\tilde{y}})\right) \leqslant -e$$

For $x \in K$, since $\mathsf{utility}_{\varphi,\sigma}(\mu, C) \leqslant 0$, for all $C \in \mathcal{B}_x$, we have $\sum_{C \in \mathcal{B}_x} \mathsf{utility}_{\varphi,\sigma}(\mu, C) \leqslant 0$ and thus

$$\sum_{x \in K} \sum_{C \in \mathcal{B}_x} \mathsf{utility}_{\varphi,\sigma}(\mu, C) \leqslant 0.$$

On the other hand, for $x \in K$, we have $\mathsf{utility}_{\varphi,\sigma}(\mu, A_x) = \frac{1}{2}$. Combining all this with equation (4.4) we obtain:

$$\begin{aligned}
\mathsf{utility}_{\varphi,\sigma}(\mu) &\leqslant \sum_{x \in K} \left( \frac{\deg(x)}{2} - \frac{k-1}{4} \right) - e \\
&< \sum_{x \in K} \frac{\deg(x)}{2} - e \qquad \text{(since } k \geqslant 3\text{)} \\
&\leqslant \sum_{x \text{ in } G} \frac{\deg(x)}{2} - e \\
&= 0.
\end{aligned}$$

We conclude that $\mathsf{utility}_{\varphi,\sigma}(\mu) < 0$, and thus obtain a contradiction. Therefore, $G$ contains a clique of size $k$. □

We now continue towards the proof of Theorem 3.2. The next step is to show that the following decision problem is NP-hard.

| | |
|---|---|
| PROBLEM : | Check-Sub-SR |
| INPUT : | $(T, \boldsymbol{y})$, for a decision tree $T$ of dimension $n$, and partial instance $\boldsymbol{y} \in \{0, 1, \bot\}^n$ |
| OUTPUT : | Yes, if there is a partial instance $\boldsymbol{y}' \subsetneq \boldsymbol{y}$ such that $\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}')] \geqslant \Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})]$, and No otherwise |

**Proposition 4.7.** Check-Sub-SR *is* NP*-hard.*

Proof. We reduce from 2-Minimal-Expected-Clauses. Let $(\varphi, \sigma)$ be an instance of 2-Minimal-Expected-Clauses. Let $m$ be the number of clauses of $\varphi$ and assume that $\varphi$ has $n$ variables $x_1, \ldots, x_n$. Without loss of generality we assume that $m$ is a power of 2. Define a decision tree $T$ of dimension $n + m - 1$ as follows. Start with a *perfect* binary tree $S$ of depth $\log_2 m$, that is, each internal node has two children, and each leaf is at depth $\log_2 m$. In particular, $S$ has $m$ leaves and $m - 1$ internal nodes. All the internal nodes of $S$ are labeled with a different fresh feature from $\{n + 1, \ldots, n + m - 1\}$. For each clause $C$ in $\varphi$, pick a different leaf $\ell_C$ of $S$. It is easy to see that for each clause $C$ we can define a decision tree $S_C$ over the features $\{1, \ldots, n\}$ such that for every assignment $\mu : \{x_1, \ldots, x_n\} \to \{0, 1\}$ to the variables of $\varphi$, the corresponding instance $\boldsymbol{x} \in \{0, 1\}^n$ where $\boldsymbol{x}[i] = \mu(x_i)$ satisfies that $S_C(\boldsymbol{x}) = 1$ if and only if $\mu$ satisfies $C$. The decision tree $T$ is obtained from $S$ by identifying for each clause $C$, the leaf $\ell_C$ with the root of the decision tree $S_C$.

For any partial assignment $\mu : \{x_1, \ldots, x_n\} \to \{0, 1, \bot\}$ for $\varphi$, we denote by $\boldsymbol{y}_\mu$ the partial instance of dimension $n + m - 1$ such that $\boldsymbol{y}_\mu[i] = \mu(x_i)$ for every $i \in \{1, \ldots, n\}$ and $\boldsymbol{y}_\mu[i] = \bot$ for every $i \in \{n + 1, \ldots, n + m - 1\}$. The output of the reduction is $(T, \boldsymbol{y}_\sigma)$. Observe that the transformation from $\mu$ to $\boldsymbol{y}_\mu$ is a bijection between the sets $\{\mu \mid \mu \subseteq \sigma\}$ and $\{\boldsymbol{y}_\mu \mid \boldsymbol{y}_\mu \subseteq \boldsymbol{y}_\sigma\}$. By construction, for any partial assignment $\mu \subseteq \sigma$, we have that

$$\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}_\mu)] \;=\; \frac{E(\varphi, \mu)}{m}.$$

Hence $(\varphi, \sigma)$ is a Yes-instance of 2-Minimal-Expected-Clauses if and only if $(T, \boldsymbol{y}_\sigma)$ is a Yes-instance for the Check-Sub-SR problem. □

*Remark.* We can assume that the instance $(T, \boldsymbol{y}_\sigma)$ constructed in the proof of Proposition 4.7 satisfies

$$\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}_\sigma)] > \frac{1}{2}.$$

Indeed, the above probability is simply $\frac{E(\varphi,\sigma)}{m}$, where $m$ is the number of clauses of $\varphi$. On the other hand, from the proof of Theorem 3.3, we can choose $\sigma$ such that $\sigma(x_i) = 1$ for every variable $x_i \in \{x_1, \ldots, x_n\}$ of $\varphi$. It follows that $E(\varphi, \sigma)$ is simply the number of clauses satisfied by $\sigma$, which are all the clauses in $\mathcal{B}_x$ for some variable $x$, and all the clauses of the form $Z_{x,y}$. Note that the total number of clauses from the sets $\mathcal{B}_x$ is greater that the total number clauses of the form $A_x$, and hence $\frac{E(\varphi,\sigma)}{m} > \frac{1}{2}$. Indeed, there are $\deg(x)$ clauses in $\mathcal{B}_x$, and summing over all the variables $x$, we obtain $2e$, where $e$ are the number of edges in the graph $G$. On the other hand, each clause $A_x$ is repeated $\frac{k-1}{2} + \deg(x) - (k - 1) = \deg(x) - \frac{k-1}{2}$ times. Taking the sum over all the variables $x$, we obtain $2e - n\left(\frac{k-1}{2}\right) < 2e$. This property will be useful for the proof of Theorem 3.2, which we recall next.

Theorem 3.2 (Restated). *Assuming that* P $\neq$ NP*, the* Compute-Minimal-SR *problem cannot be solved in polynomial time.*

We show that if Compute-Minimal-SR admits a polynomial time algorithm, then Check-Sub-SR is in P, which contradicts the assumption that P $\neq$ NP. In particular, we show a Turing-reduction from a variant of

Check-Sub-SR to Compute-Minimal-SR, thus establishing that the latter cannot be solved in polynomial time unless P = NP.

For the sake of readability, given a partial instance $\boldsymbol{y}$, in this proof we use notation $\boldsymbol{z} \sim \mathbb{U}(\boldsymbol{y})$ to indicate that $\boldsymbol{z}$ is generated uniformly at random from the set $\textsc{Comp}(\boldsymbol{y})$. For instance, we obtain the following simplification by using this terminology:

$$\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \;=\; \Pr_{\boldsymbol{z} \sim \mathbb{U}(\boldsymbol{y})}[T(\boldsymbol{z}) = 1]$$

We require a particular kind of hard instances for the Check-Sub-SR to make our reduction work. In particular, we now define the notion of *strongly-balanced* inputs, which intuitively captures the idea that defined features in a partial instance $\boldsymbol{y}$ appear at the same depth in different branches of the decision tree $T$. To make this definition precise, consider that every path $\pi$ from the root to a leaf in a decision tree can be identified with a sequence of labels $s_\pi$ corresponding to the labels of the nodes of $\pi$, where the last label of $\pi$ is either **true** or **false**. We use notation $s_\pi[i]$ for the $i$-th label in the sequence $s_\pi$. With this notation, we can introduce the following definition.

**Definition 4.8.** *Given a decision tree $T$ of dimension $d$ and $\boldsymbol{y} \in \{0, 1, \bot\}^n$ a partial instance, we say that the pair $(T, \boldsymbol{y})$ is strongly-balanced if*

$$\Pr_{\boldsymbol{z} \sim \mathbb{U}(\boldsymbol{y})}[T(\boldsymbol{z}) = 1] \;>\; \frac{1}{2},$$

*and there exists $k \in \mathbb{N}$ such that for every root-to-leaf path $\pi$ in $T$, the sequence $s_\pi$ satisfies*

$$\boldsymbol{y}[s_\pi[i]] = \bot \iff i \leqslant k.$$

If $(T, \boldsymbol{y})$ is strongly-balanced, then there exists a unique value $k \in \mathbb{N}$ that satisfies the second condition of the definition. We denote this value by $u(T, \boldsymbol{y})$. In particular, if $\boldsymbol{y} \in \{0, 1\}^n$, then $(T, \boldsymbol{y})$ is strongly-balanced and $u(T, \boldsymbol{y}) = 0$.

Now let us define the following problem.

| | |
|---|---|
| PROBLEM : | SB-Check-SUB-SR |
| INPUT : | $(T, \boldsymbol{y})$, for $T$ a decision tree of dimension $n$ and $\boldsymbol{y} \in \{0, 1, \bot\}^n$ a partial instance, where $(T, \boldsymbol{y})$ is strongly-balanced. |
| OUTPUT : | Yes, if there is a partial instance $\boldsymbol{y}' \subsetneq \boldsymbol{y}$ such that $\Pr_{\boldsymbol{z} \sim \mathbb{U}(\boldsymbol{y}')}[T(\boldsymbol{z}) = 1] \geqslant \Pr_{\boldsymbol{z} \sim \mathbb{U}(\boldsymbol{y})}[T(\boldsymbol{z}) = 1]$, and No otherwise. |

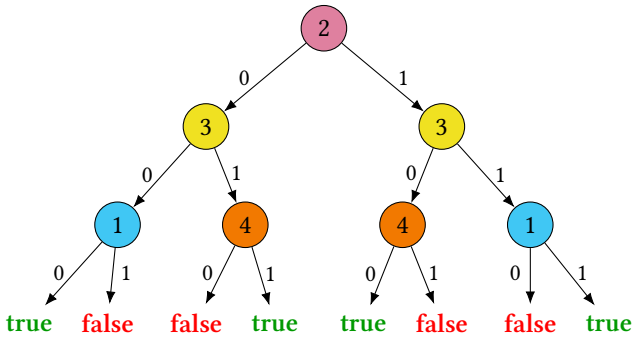
One can now check that the proof of Proposition 4.7 directly proves NP-hardness for this problem, and thus we can reduce from it to prove hardness for the computation variant. Indeed, the first part of the definition of strongly-balanced follows from the remark given after the proof of Proposition 4.7. The second part follows from the fact that the construction in the proof of Proposition 4.7 starts with a perfect binary tree.

We can now start proving Theorem 3.2, which requires a few auxiliary claims that for the sake of readability will be assumed within this proof, and then we will conclude this section by proving each claim individually.

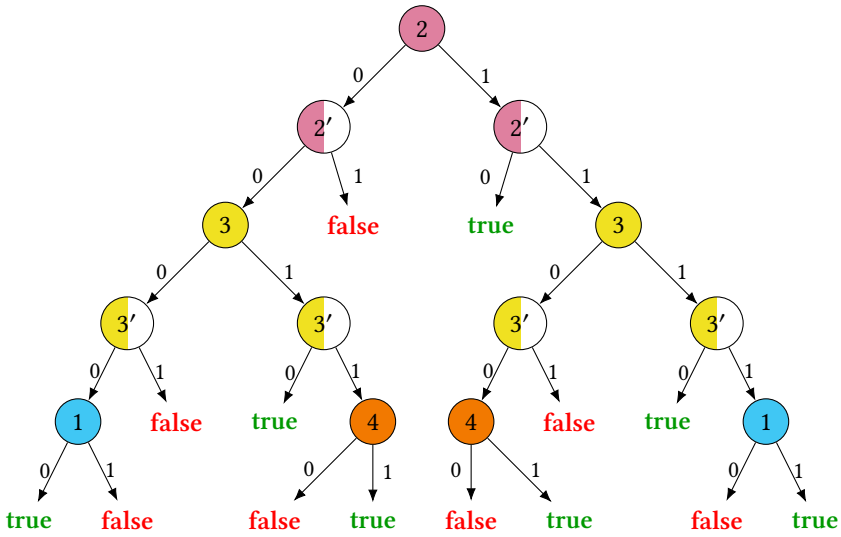
Proof of Theorem 3.2. Assume an instance $(T, \boldsymbol{y})$ for the SB-Check-Sub-SR problem. Let us enumerate the features in $T$ as $1, \ldots, n$. Also, let $S$ be the set of features defined in $\boldsymbol{y}$, that is, $\boldsymbol{y}[i] \neq \bot \iff i \in S$. We first build a decision tree $T'$ of dimension $2n - |S|$, with the following features:

(1) Create a feature $i$ for $i \in S$.
(2) For every $i \in \{1, \ldots, n\} \setminus S$ create features $i$ and $i'$.

Note that this amounts to the promised number of features. We build $T'$ through a recursive process $\mathcal{R}$ defined next. First, note that any decision tree can be described inductively as either a **true**/**false** leaf, or a tuple $(r, L, R)$, where $r$ is the root node, $L$ is a decision tree whose root is the left child of $r$, and $R$ is a decision tree whose root




(a) Original decision tree $T$.




(b) Resulting decision tree $T' = \mathcal{R}(T)$.

Fig. 8. Illustration of the recursive process $\mathcal{R}$ over an example where $\boldsymbol{y} = (0, \perp, \perp, 0)$. Note that the pair $(T, \boldsymbol{y})$ is strongly-balanced.

is the right child of $r$. We can now define $\mathcal{R}$ as a recursive procedure that when called with argument $\tau$ proceed as follows:

(1) If $\tau$ is a leaf, then simply return $\tau$.
(2) If $\tau = (r, L, R)$, and node $r$ is labeled with feature $i \in S$, then simply return

$$(r,\ \mathcal{R}(L),\ \mathcal{R}(R)).$$

(3) If $\tau = (r, L, R)$, and node $r$ is labeled with feature $i \in \{1, \ldots, n\} \setminus S$, then return the following decision tree:

$$(r,\ (u, \mathcal{R}(L), \textbf{false}),\ (v, \textbf{true}, \mathcal{R}(R)))\,,$$

where nodes $u$ and $v$ are new nodes, both labeled with feature $i'$.

As anticipated, $T' = \mathcal{R}(T)$. An example illustrating the process is depicted in Figure 8. Now we create a tree $T''$ of dimension $2n - |S| + m$ that on top of the previous features incorporates features $b_j$ for each $j \in \{1, \ldots, m\}$, where $m$ is an integer we specify later on. In order to construct $T''$, we start by defining $\boldsymbol{y}_0$ as the partial instance of dimension $2n - |S| + m$ such that $\boldsymbol{y}_0[i] = \boldsymbol{y}[i]$ for every $i \in S$ and

$$\boldsymbol{y}_0\left[b_j\right] = 0, \quad \forall j \in \{1, \ldots, m\},$$

with the remaining components of $\boldsymbol{y}_0$ being left undefined. Let $T_{\boldsymbol{y}_0}$ be a tree of dimension $2n - |S| + m$ that accepts exactly the completions of $\boldsymbol{y}_0$; this can be trivially done by creating a tree that accepts exactly the features that are defined in $\boldsymbol{y}_0$, and then observing that when running an instance whose feature space is a superset of this, then the instance is accepted if and only if it is a completion of $\boldsymbol{y}_0$. Now let $T_1$ be a tree of dimension $2n + |S| + m$ that implements the following Boolean formula:

$$\phi = \sum_{j=1}^{m} b_j \geqslant 2.$$

**Claim 4.9.** *Decision tree $T_1$, implementing the function $\phi$, can be constructed in polynomial time.*

Now, let us build an instance $\boldsymbol{x}$ of dimension $2n - |S| + m$ as follows. For each $i \in S$ let $\boldsymbol{x}[i] = \boldsymbol{y}[i]$, thus ensuring $\boldsymbol{x}$ is a completion of $\boldsymbol{y}$. Then for each $j \in \{1, \ldots, m\}$ let $\boldsymbol{x}[b_j] = 0$, and finally for each $i \in \{1, \ldots, n\} \setminus S$, let $\boldsymbol{x}[i] = 0$ and $\boldsymbol{x}[i'] = 1$. For example, if $\boldsymbol{y} = \big(0, \quad \bot, \quad \bot, \quad 1\big)$, and $m = 3$ then

$$\boldsymbol{x} = \big(0, \quad 0, \quad 1, \quad 0, \quad 1, \quad 1, \quad 0, \quad 0, \quad 0\big),$$

where the features $b_j$ have been placed at the end of the vector.

Let $\boldsymbol{y}^\star$ be the partial instance of dimension $2n - |S| + m$ such that $\boldsymbol{y}^\star[i] = \boldsymbol{y}[i]$ for every $i \in S$, and undefined otherwise. Let us abuse notation and assume now that $T'$ has dimension $2n - |S| + m$, even though it only explicitly uses the first $2n - |S|$ features, as this would make it compatible with other decision trees and instance we have constructed. Finally, let $T^\star$ be the decision tree defined as

$$T^\star = T_{\boldsymbol{y}_0} \cup (T' \cap T_1),$$

and note that the union and intersection of decision trees can be computed in polynomial time through a standard algorithm (see e.g., [25]).

Let us now define

$$\delta \coloneqq \Pr_{\boldsymbol{z} \sim \mathbb{U}(\boldsymbol{y}^\star)} \left[(T' \cap T_1)(\boldsymbol{z}) = 1\right].$$

We now claim that the result of Compute-Minimal-SR$(T^\star, \boldsymbol{x}, \delta)$ is different from $\boldsymbol{y}^\star$ if and only if $(T, \boldsymbol{y})$ is a positive instance of SB-Check-SUB-SR. But before we can prove this, we need some intermediary tools and claims that we develop next.

Let us start by distinguishing two kinds of leaves of $T'$. Let us say that the leaves of $T'$ introduced in step 1 of the recursive procedure $\mathcal{R}$ are *natural*, while those introduced in step 3 are *artificial*. We denote by $\mathcal{N}$ the set of natural leaves of $T'$ and by $\mathcal{A}$ the set of artificial leaves of $T'$. Moreover, let $\mathcal{N}_t, \mathcal{N}_f$ represent the **true** and **false** natural leaves, and define $\mathcal{A}_t, \mathcal{A}_f$ analogously. We also use $T'_{\downarrow \boldsymbol{z}}$ to denote the leaf where instance $\boldsymbol{z}$ ends when evaluated over tree $T'$. With this notation, $T'(\boldsymbol{z}) = 1$ is equivalent to $T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}_t \cup \mathcal{N}_t$. We now make the following claims.

**Claim 4.10.** *For every partial instance $\boldsymbol{y}'^\star \subseteq \boldsymbol{y}^\star$, it holds that*

$$\Pr_{\boldsymbol{z} \sim \mathbb{U}(\boldsymbol{y}'^\star)} \left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}_t \mid T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}\right] = \Pr_{\boldsymbol{z} \sim \mathbb{U}(\boldsymbol{y}'^\star)} \left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}_f \mid T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}\right] = \frac{1}{2}.$$

**Claim 4.11.** *Given a partial instance $\boldsymbol{y}' \subseteq \boldsymbol{y}$, we define $\boldsymbol{y}'^{\star}$ as the partial instance of dimension $2n - |S| + m$ that matches $\boldsymbol{y}'$ on its defined features, and holds $\boldsymbol{y}'^{\star}[i] = \boldsymbol{y}'^{\star}[i'] = \perp$ for every feature $i$ such that $\boldsymbol{y}'[i] = \perp$. Then it holds that*

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}')}\left[T(\boldsymbol{z}) = 1\right] \;=\; \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{N}_t \mid T'_{\downarrow \boldsymbol{z}} \in \mathcal{N}\right].$$

**Claim 4.12.** *By choosing $m \geqslant \max\{2u(T, \boldsymbol{y}) + 2n, 9\}$, we have that*

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[(T' \cap T_1)(\boldsymbol{z}) = 1\right] \;>\; \frac{1}{2}.$$

**Claim 4.13.** *For every partial instance $\boldsymbol{y}'^{\star} \subseteq \boldsymbol{y}^{\star}$, it holds that*

$$\begin{aligned}
\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}\right] &= \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}\right], \\
\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{N}\right] &= \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{N}\right].
\end{aligned}$$

With these claims we can finally prove the reduction is correct. That is, we show that the result of the procedure Compute-Minimal-SR$(T^{\star}, \boldsymbol{x}, \delta)$ is different from $\boldsymbol{y}^{\star}$ if and only if $(T, \boldsymbol{y})$ is a positive instance of SB-Check-SUB-SR.

($\Rightarrow$) Assume the result of Compute-Minimal-SR$(T^{\star}, \boldsymbol{x}, \delta)$ is some partial instance $\boldsymbol{y}'^{\star}$ different from $\boldsymbol{y}^{\star}$. Note immediately that it is not possible that $\boldsymbol{y}^{\star} \subsetneq \boldsymbol{y}'^{\star}$ as by definition of $\delta$, we have that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T^{\star}(\boldsymbol{z}) = 1\right] \;\geqslant\; \delta,$$

which would contradict the minimality of $\boldsymbol{y}'^{\star}$. Let us first prove that $\boldsymbol{y}'^{\star} \subseteq \boldsymbol{y}^{\star}$. As a first step, we show that $\boldsymbol{y}'^{\star}[i] = \boldsymbol{y}'^{\star}[i'] = \perp$ for every $i \notin S$. We do this by way of contradiction, assuming first that $\boldsymbol{y}'^{\star}[i] = 0$ or $\boldsymbol{y}'^{\star}[i'] = 1$ for some $i \notin S$, and exposing how either case generates a contradiction.[5]

(1) If there is some feature $i \notin S$ such that $\boldsymbol{y}'^{\star}[i] = 0$, let us define $\boldsymbol{y}^{\dagger}$ to be equal to $\boldsymbol{y}'^{\star}$ except that $\boldsymbol{y}^{\dagger}[i] = \perp$. This means that $\boldsymbol{y}^{\dagger} \subsetneq \boldsymbol{y}'^{\star}$. Moreover, let $\boldsymbol{y}_1^{\dagger}$ be equal to $\boldsymbol{y}'^{\star}$ except that $\boldsymbol{y}_1^{\dagger}[i] = 1$. We now show that $\boldsymbol{y}^{\dagger}$ is also a valid output of the computation problem, which contradicts the minimality of $\boldsymbol{y}'^{\star}$. Indeed, given that $(T_{\boldsymbol{y}_0} \cap T_1)(\boldsymbol{z}) = 0$ for every instance $\boldsymbol{z}$, it holds that

$$\begin{aligned}
\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\dagger})}\left[T^{\star}(\boldsymbol{z}) = 1\right] &= \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\dagger})}\left[T_{\boldsymbol{y}_0}(\boldsymbol{z}) = 1\right] + \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\dagger})}\left[(T' \cap T_1)(\boldsymbol{z}) = 1\right] \\
&= \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T_{\boldsymbol{y}_0}(\boldsymbol{z}) = 1\right] + \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\dagger})}\left[(T' \cap T_1)(\boldsymbol{z}) = 1\right],
\end{aligned}$$

and we can then observe that the events $T'(\boldsymbol{z}) = 1$ and $T_1(\boldsymbol{z}) = 1$ are independent, thus implying that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\dagger})}\left[(T' \cap T_1)(\boldsymbol{z}) = 1\right] \;=\; \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\dagger})}\left[T'(\boldsymbol{z}) = 1\right] \cdot \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\dagger})}\left[T_1(\boldsymbol{z}) = 1\right].$$

By construction of $\boldsymbol{y}^{\dagger}$, we also have that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\dagger})}\left[T_1(\boldsymbol{z}) = 1\right] = \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T_1(\boldsymbol{z}) = 1\right].$$

[5] Recall that $\boldsymbol{x}[i] = 0$ and $\boldsymbol{x}[i'] = 1$, so if $\boldsymbol{y}'^{\star}[i] \neq \perp$ or $\boldsymbol{y}'^{\star}[i'] \neq \perp$, then $\boldsymbol{y}'^{\star}[i] = 0$ or $\boldsymbol{y}'^{\star}[i'] = 1$ as $\boldsymbol{y}'^{\star} \subseteq \boldsymbol{x}$.

Thus, it now suffices to show that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[T'(\boldsymbol{z})=1] \geqslant \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'(\boldsymbol{z})=1],$$

as this implies by definition of $\boldsymbol{y}'^\star$, $\boldsymbol{y}^\dagger$ and $\boldsymbol{y}_1^\dagger$ that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\dagger)}[T'(\boldsymbol{z})=1] \geqslant \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^\star)}[\,T'(\boldsymbol{z})=1],$$

which in turn implies by the previous discussion that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\dagger)}\left[T^\star(\boldsymbol{z})=1\right] \;\geqslant\; \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^\star)}\left[T^\star(\boldsymbol{z})=1\right] \;\geqslant\; \delta,$$

and leads to a contradiction.

In order to prove that $\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[T'(\boldsymbol{z})=1] \geqslant \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'(\boldsymbol{z})=1]$, we consider two cases, either $\boldsymbol{y}'^\star[i']=1$ or $\boldsymbol{y}'^\star[i']=\bot$.

(a) If $\boldsymbol{y}'^\star[i']=1$, then we have that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[T'(\boldsymbol{z})=1] \geqslant \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'(\boldsymbol{z})=1],$$

as for every node $u$ in $T'$ labeled with $i$, any completion $\boldsymbol{z}$ of $\boldsymbol{y}'^\star$ that goes through $u$ is rejected by construction (landing in an artificial **false** leaf), and for paths of $T'$ that do not go through a node labeled $u$ there is no difference between completions of $\boldsymbol{y}'^\star$ and those for $\boldsymbol{y}_1^\dagger$.

(b) If $\boldsymbol{y}'^\star[i']=\bot$, then we have that for every node $u$ in $T'$ which corresponds to $(u,L,R)$ according to the recursive definition of a decision tree, and is labeled with $i$, the probability of acceptance of a random completion $\boldsymbol{z}$ conditioned on its path going through $u$, which is denoted by $\boldsymbol{z}\rightsquigarrow u$, is as follows:

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'(\boldsymbol{z})=1 \mid \boldsymbol{z}\rightsquigarrow u] = \frac{1}{2}\cdot\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^\star)}[L(\boldsymbol{z})=1 \mid \boldsymbol{z}\rightsquigarrow u],$$

while on the other hand we have

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[T'(\boldsymbol{z})=1 \mid \boldsymbol{z}\rightsquigarrow u] \;=\; \frac{1}{2}+\frac{1}{2}\cdot\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[R(\boldsymbol{z})=1 \mid \boldsymbol{z}\rightsquigarrow u].$$

By considering that

$$\frac{1}{2} \;\geqslant\; \frac{1}{2}\cdot\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^\star)}[L(\boldsymbol{z})=1 \mid \boldsymbol{z}\rightsquigarrow u],$$

we obtain

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[T'(\boldsymbol{z})=1 \mid \boldsymbol{z}\rightsquigarrow u] \;\geqslant\; \frac{1}{2}\cdot\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^\star)}[L(\boldsymbol{z})=1 \mid \boldsymbol{z}\rightsquigarrow u],$$

from where

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[T'(\boldsymbol{z})=1 \mid \boldsymbol{z}\rightsquigarrow u] \;\geqslant\; \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'(\boldsymbol{z})=1 \mid \boldsymbol{z}\rightsquigarrow u].$$

Therefore, by noting that $z \rightsquigarrow u_1$ and $z \rightsquigarrow u_2$ are disjoint events if $u_1, u_2$ are different nodes of $T'$ with label $i$, and defining $N(T', i)$ as the set of nodes of $T'$ with label $i$, we have that

$$\begin{aligned}
\Pr_{z\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[T'(z)=1] &= \sum_{u\in N(T',i)} \Pr_{z\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[T'(z)=1 \mid z \rightsquigarrow u] \cdot \Pr_{z\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[z \rightsquigarrow u] \\
&\geqslant \sum_{u\in N(T',i)} \Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'(z)=1 \mid z \rightsquigarrow u] \cdot \Pr_{z\sim\mathbb{U}(\boldsymbol{y}_1^\dagger)}[z \rightsquigarrow u] \\
&= \sum_{u\in N(T',i)} \Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'(z)=1 \mid z \rightsquigarrow u] \cdot \Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[z \rightsquigarrow u] \\
&= \Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'(z)=1]
\end{aligned}$$

This concludes the proof of this case.

(2) It remains to analyze the case when $\boldsymbol{y}'^\star[i] = \perp$ and $\boldsymbol{y}'^\star[i'] = 1$, which can be proved in the same way as the previous case $\boldsymbol{y}'^\star[i] = 0$ and $\boldsymbol{y}'^\star[i'] = \perp$.

After this case analysis, we can safely assume that $\boldsymbol{y}'^\star[i] = \boldsymbol{y}'^\star[i'] = \perp$ for every $i \notin S$. We now show that $\boldsymbol{y}'^\star[b_j] = \perp$ for every $j \in \{1, \ldots, m\}$. To see this, consider that in general it could be that $\boldsymbol{y}'^\star$ *forces* a certain number $k$ of features $b_j$ to get value 0, meaning that there is a set $K \subseteq \{1, \ldots, m\}$ with $|K| = k$ such that $\boldsymbol{y}'^\star[b_j] = 0$ for $j \in K$. We argue that $k = 0$. Let us start by arguing that $k \leqslant m - 2$. Indeed, assume expecting a contradiction that $k > m - 2$, then by definition

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T_1(z)=1] = 0,$$

and thus

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}\left[T^\star(z)=1\right] = \Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}\left[T_{\boldsymbol{y}_0}(z)=1\right],$$

but

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}\left[T_{\boldsymbol{y}_0}(z)=1\right] \leqslant \frac{1}{2},$$

as $\boldsymbol{y}'^\star$ cannot be a superset of $\boldsymbol{y}^\star$, and thus at least one feature $i$ of $\boldsymbol{y}^\star$ is undefined in $\boldsymbol{y}'^\star$, and the event $z[i] = \boldsymbol{y}^\star[i]$, which happens with probability $\frac{1}{2}$, is required for $T_{\boldsymbol{y}_0}(z) = 1$. But by definition of $\delta$, if $\boldsymbol{y}'^\star$ is the output of the computational problem, then its probability of acceptance is at least

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}\left[(T' \cap T_1)(z)=1\right],$$

and this probability is greater than $\frac{1}{2}$ (Claim 4.12), and thus we have a contradiction. We now safely assume $k \leqslant m - 2$ and thus $m - k \geqslant 2$. Observe that as at least one component of $\boldsymbol{y}^\star$ is undefined in $\boldsymbol{y}'^\star$, we have

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}\left[T_{\boldsymbol{y}_0}(z)=1\right] \leqslant \frac{1}{2}\cdot\left(\frac{1}{2}\right)^{m-k},$$

and thus

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}\left[(T' \cap T_1)(z)=1\right] \geqslant \delta - \frac{1}{2}\cdot\left(\frac{1}{2}\right)^{m-k},$$

which, considering that

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T_1(z)=1] = 1 - \left(\frac{1}{2}\right)^{m-k} - (m-k)\left(\frac{1}{2}\right)^{m-k},$$

implies that

$$\Pr_{\boldsymbol{y}\sim\mathbb{U}(\boldsymbol{z}'^{\star})}\left[T'(\boldsymbol{z})=1\right] \geqslant \frac{\delta-\frac{1}{2}\cdot\left(\frac{1}{2}\right)^{m-k}}{1-\left(\frac{1}{2}\right)^{m-k}-(m-k)\left(\frac{1}{2}\right)^{m-k}},$$

as $T'(\boldsymbol{z})=1$ and $T_1(\boldsymbol{z})=1$ are independent events. We now show that the right side of the previous equation is greater than $\delta$. Indeed, for ease of notation set $r \coloneqq (m-k+1)$ and note that the right side can be rewritten as

$$\frac{\delta-\left(\frac{1}{2}\right)^{r}}{1-r\left(\frac{1}{2}\right)^{r-1}}.$$

Now consider that as $m-k \geqslant 2$ we have that $r \geqslant 3$ and thus $2^{r-1} > r$, which implies $r\left(\frac{1}{2}\right)^{r-1} < 1$, and thus the denominator of the previous equation is positive, implying that what we want to show is equivalent to

$$\delta-\left(\frac{1}{2}\right)^{r} > \delta\left(1-r\left(\frac{1}{2}\right)^{r-1}\right),$$

which is in turn equivalent to

$$\delta r\left(\frac{1}{2}\right)^{r-1} > \left(\frac{1}{2}\right)^{r},$$

but as by Claim 4.12 we have $\delta > \frac{1}{2}$, and $r \geqslant 3$, the previous equation is trivially true. We have therefore showed that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'(\boldsymbol{z})=1\right] > \delta.$$

Now let $\boldsymbol{y}^{\ominus}$ be the partial instance such that $\boldsymbol{y}^{\ominus}[i] = \boldsymbol{y}'^{\star}[i]$ for every $i \in S$, and is undefined in all other features. Note that $\boldsymbol{y}^{\ominus} \subseteq \boldsymbol{y}'^{\star}$ and also $\boldsymbol{y}^{\ominus} \subseteq \boldsymbol{y}^{\star}$. If $\boldsymbol{y}^{\ominus} = \boldsymbol{y}'^{\star}$, then $\boldsymbol{y}'^{\star} \subseteq \boldsymbol{y}^{\star}$ which is what we are hoping to prove. So we now assume $\boldsymbol{y}^{\ominus} \subsetneq \boldsymbol{y}'^{\star}$ expecting a contradiction. Note that $T'$ does not use the $b_j$ features at all, and therefore we have that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[T'(\boldsymbol{z})=1\right] = \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'(\boldsymbol{z})=1\right] > \delta.$$

We use this to prove that $\boldsymbol{y}^{\ominus}$ would have been a valid outcome of the computing problem, thus contradicting the minimality of $\boldsymbol{y}'^{\star}$. Indeed,

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[T^{\star}(\boldsymbol{z})=1\right] > \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[T'(\boldsymbol{z})=1\right]\cdot\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[T_1(\boldsymbol{z})=1\right],$$

and note that as

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[T'(\boldsymbol{z})=1\right] > \delta,$$

it must be the case that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[T'(\boldsymbol{z})=1\right] \geqslant \delta+\left(\frac{1}{2}\right)^{2n-|S|},$$

as only $2n-|S|$ features appear as labels in $T'$ and, thus, the completion probability of any partial instance over $T'$ must be an integer multiple of $\left(\frac{1}{2}\right)^{2n-|S|}$. Now let us abbreviate $2n-|S|$ as $\ell$ and choose $m \geqslant 2\ell$. We thus have

that

$$\begin{aligned}
\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[T^{\star}(\boldsymbol{z})=1\right] &\geqslant \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[T'(\boldsymbol{z})=1\right]\cdot\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[T_1(\boldsymbol{z})=1\right]\\
&\geqslant \left(\delta+\left(\frac{1}{2}\right)^{\ell}\right)\left(1-(m+1)\left(\frac{1}{2}\right)^{m}\right)\\
&\geqslant \left(\delta+\left(\frac{1}{2}\right)^{\ell}\right)\left(1-(2\ell+1)\left(\frac{1}{2}\right)^{2\ell}\right)\\
&= \delta-\delta(2\ell+1)\left(\frac{1}{2}\right)^{2\ell}+\left(\frac{1}{2}\right)^{\ell}-(2\ell+1)\left(\frac{1}{2}\right)^{3\ell}\\
&\geqslant \delta-(2\ell+1)\left(\frac{1}{2}\right)^{2\ell}+\left(\frac{1}{2}\right)^{\ell}-(2\ell+1)\left(\frac{1}{2}\right)^{2\ell}\\
&= \delta-(4\ell+2)\left(\frac{1}{2}\right)^{2\ell}+\left(\frac{1}{2}\right)^{\ell}\\
&= \delta+\left(\frac{1}{2}\right)^{\ell}\left(1-(4\ell+2)\left(\frac{1}{2}\right)^{\ell}\right),
\end{aligned}$$

where the last parenthesis is positive for $\ell \geqslant 5$, which can be assumed without loss of generality as $2n-|S|<5$ requires $n<5$ (since $|S|\leqslant n$), and thus the original instance of the decision problem would have constant size. We have thus concluded that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[T^{\star}(\boldsymbol{z})=1\right] \geqslant \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\ominus})}\left[(T'\cap T_1)(\boldsymbol{z})=1\right] \geqslant \delta, \tag{4.14}$$

thus showing that $\boldsymbol{y}^{\ominus}$ is a valid outcome for the computing problem, which contradicts the minimality of $\boldsymbol{y}'^{\star}$. This in turn implies that $\boldsymbol{y}'^{\star}=\boldsymbol{y}^{\ominus}$, and thus subsequently that $\boldsymbol{y}'^{\star}\subseteq\boldsymbol{y}^{\star}$. Let us now show how by combining Claims 4.10, 4.11 and 4.13, we can conclude the forward direction entirely. Indeed, note that the trivial equality

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T_1(\boldsymbol{z})=1\right]=\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T_1(\boldsymbol{z})=1\right]$$

implies that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'(\boldsymbol{z})=1\right] \leqslant \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'(\boldsymbol{z})=1\right], \tag{4.15}$$

as we already have proved that $\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[(T'\cap T_1)(\boldsymbol{z})=1\right]\geqslant\delta$ by Equation 4.14 and the fact that $\boldsymbol{y}^{\ominus}=\boldsymbol{y}'^{\star}$, and we have that $\delta=\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\star})}[(T'\cap T_1)(\boldsymbol{z})=1]$. We can now use Claims 4.10, 4.11 and 4.13 together

with Equation 4.15 to conclude that

$$
\begin{aligned}
\Pr_{z\sim\mathbb{U}(\boldsymbol{y})}[T(z)=1] &= \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'_{\downarrow z}\in\mathcal{N}_t \mid T'_{\downarrow z}\in\mathcal{N}] && \text{(Claim 4.11)}\\
&= \frac{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'_{\downarrow z}\in\mathcal{N}_t]}{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'_{\downarrow z}\in\mathcal{N}]}\\
&= \frac{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'_{\downarrow z}\in\mathcal{N}_t]}{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{N}]} && \text{(Claim 4.13)}\\
&= \frac{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'(z)=1]-\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'_{\downarrow z}\in\mathcal{A}_t]}{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{N}]}\\
&= \frac{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'(z)=1]-\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'_{\downarrow z}\in\mathcal{A}_t \mid T'_{\downarrow z}\in\mathcal{A}]\cdot\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'_{\downarrow z}\in\mathcal{A}]}{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{N}]}\\
&= \frac{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'(z)=1]-\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{A}_t \mid T'_{\downarrow z}\in\mathcal{A}]\cdot\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{A}]}{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{N}]}\\
& && \text{(Claims 4.10 and 4.13)}\\
&\leqslant \frac{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'(z)=1]-\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{A}_t \mid T'_{\downarrow z}\in\mathcal{A}]\cdot\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{A}]}{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{N}]} && \text{(Equation 4.15)}\\
&= \frac{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'(z)=1]-\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{A}_t]}{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{N}]}\\
&= \frac{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{N}_t]}{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{N}]}\\
&= \Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^\star)}[T'_{\downarrow z}\in\mathcal{N}_t \mid T'_{\downarrow z}\in\mathcal{N}]\\
&= \Pr_{z\sim\mathbb{U}(\boldsymbol{y}')}[T(z)=1], && \text{(Claim 4.11)}
\end{aligned}
$$

where $\boldsymbol{y}'$ is the partial instance of dimension $n$ such that $\boldsymbol{y}'[i]=\boldsymbol{y}'^\star[i]$ for every $i$ such that $\boldsymbol{y}'^\star[i]\neq\perp$, and $\boldsymbol{y}'$ is undefined in all other features. By this definition, $\boldsymbol{y}'\subsetneq\boldsymbol{y}$ as we had $\boldsymbol{y}'^\star\subsetneq\boldsymbol{y}^\star$ (because by assumption $\boldsymbol{y}'^\star\neq\boldsymbol{y}^\star$), and thus we have effectively proved that the instance $(T,z)$ is a positive instance of SB-Check-SUB-SR. This concludes the proof of the forward direction.

($\Leftarrow$) Assume the instance $(T,\boldsymbol{y})$ is a positive instance of SB-Check-SUB-SR and, thus, there exists some $\boldsymbol{y}'\subsetneq\boldsymbol{y}$ such that

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}')}[T(z)=1]\geqslant\Pr_{z\sim\mathbb{U}(\boldsymbol{y})}[T(z)=1]\,.$$

Define $\boldsymbol{y}'^\star$ of the dimension of $T^\star$ based on $\boldsymbol{y}'$ by setting $\boldsymbol{y}'^\star[i]=\boldsymbol{y}'[i]$ for every $i$ such that $\boldsymbol{y}'[i]\neq\perp$, and leave the rest of $\boldsymbol{y}'^\star$ undefined. Note that this definition immediately implies $\boldsymbol{y}'^\star\subsetneq\boldsymbol{y}^\star$. By Claim 4.11 the previous

equation implies that

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow z}\in\mathcal{N}_t \mid T'_{\downarrow z}\in\mathcal{N}\right] \geqslant \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z}\in\mathcal{N}_t \mid T'_{\downarrow z}\in\mathcal{N}\right],$$

which implies in turn that

$$\frac{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}[T'_{\downarrow z}\in\mathcal{N}_t]}{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}[T'_{\downarrow z}\in\mathcal{N}]} \geqslant \frac{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}[T'_{\downarrow z}\in\mathcal{N}_t]}{\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}[T'_{\downarrow z}\in\mathcal{N}]}.$$

By Claim 4.13 the denominators of the previous inequality are equal and, thus,

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow z}\in\mathcal{N}_t\right] \geqslant \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z}\in\mathcal{N}_t\right],$$

from where

$$\begin{aligned}\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow z}\in\mathcal{N}_t\right] + \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z}\in\mathcal{A}_t\right] &\geqslant \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z}\in\mathcal{N}_t\right] + \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z}\in\mathcal{A}_t\right]\\ &= \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'(z)=1\right].\end{aligned}$$

But combining Claims 4.10 and 4.13 we have that

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z}\in\mathcal{A}_t\right] = \Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow z}\in\mathcal{A}_t\right],$$

which when combined with the previous equation gives us

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'(z)=1\right] \geqslant \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'(z)=1\right],$$

and using again that

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T_1(z)=1\right] = \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T_1(z)=1\right],$$

we obtain that

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[(T'\cap T_1)(z)=1\right] \geqslant \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[(T'\cap T_1)(z)=1\right] = \delta.$$

Finally, by observing that

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T^{\star}(z)=1\right] \;\geqslant\; \Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[(T'\cap T_1)(z)=1\right] \;\geqslant\; \delta,$$

we have that $\boldsymbol{y}'^{\star}$ is a valid output for the computational problem, and given that it is a strict subset of $\boldsymbol{y}^{\star}$, the result of Compute-Minimal-SR$(T^{\star}, \boldsymbol{x}, \delta)$ cannot be equal to $\boldsymbol{y}^{\star}$. This concludes the backward direction, and with it the entire proof is complete. □

We now restate and prove each of the claims used in the preceding theorem.

**Claim 4.9** (Restated). *Decision tree $T_1$, implementing the function $\phi$, can be constructed in polynomial time.*

Proof of Claim 4.9. This proof can be easily done by a direct construction. Indeed, consider the following Boolean formulas:

$$f(x_1,\ldots,x_n) \coloneqq \sum_{i=1}^{n} x_i \geqslant 2,$$

$$g(x_1,\ldots,x_n) \coloneqq \sum_{i=1}^{n} x_i \geqslant 1 = \bigvee_{i=1}^{n} x_i.$$

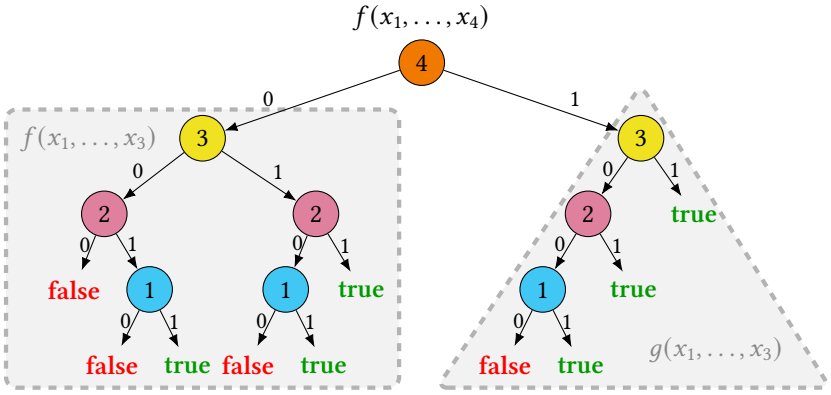



Fig. 9. Illustation of $T_1$ following the construction for Claim 4.9.

We then note that

$$f(x_1, \ldots, x_n) = [\neg x_n \wedge f(x_1, \ldots, x_{n-1})] \vee [x_n \wedge g(x_1, \ldots, x_{n-1})],$$

and thus we can build a decision tree for $f$ recursively as illustrated in Figure 9. Note that $g(x_1, \ldots, x_n)$ can be trivially implemented by a decision tree of size $O(n)$. Thus, the recursive equation characterizing the size $\alpha(n)$ of a decision tree for $f(x_1, \ldots, x_n)$ according to our construction is simply $\alpha(n) = 1 + \alpha(n-1) + O(n)$, from where we get $\alpha(n) \in O(n^2)$, thus concluding the proof of the claim.

□

**Claim 4.10** (Restated). *For every partial instance $\boldsymbol{y}'^{\star} \subseteq \boldsymbol{y}^{\star}$, it holds that*

$$\Pr_{z \sim \mathbb{U}(\boldsymbol{y}'^{\star})} \left[ T'_{\downarrow z} \in \mathcal{A}_t \mid T'_{\downarrow z} \in \mathcal{A} \right] = \Pr_{z \sim \mathbb{U}(\boldsymbol{y}'^{\star})} \left[ T'_{\downarrow z} \in \mathcal{A}_f \mid T'_{\downarrow z} \in \mathcal{A} \right] = \frac{1}{2}.$$

Proof of Claim 4.10. Observe that every leaf $\ell \in \mathcal{A}$ has a parent node $v$ in $T'$ labeled with some feature $i'$ whose parent node $u$ in $T'$ is labeled with feature $i$. Let $G(\ell) = u$ be said the grandparent of $\ell$, and assume that $G^{-1}(u) = \{\ell' \mid G(\ell') = u\}$. With this notation, we can split the set $\mathcal{A}$ as follows:

$$\mathcal{A} = \bigcup_{\text{node } u \text{ with label } i \notin S} \left\{ T'_{\downarrow z} \mid G\left(T'_{\downarrow z}\right) = u \right\},$$

where the union is actually disjoint. Thus, we have for every partial instance $\boldsymbol{y}'^{\star} \subseteq \boldsymbol{y}^{\star}$:

$$\begin{aligned} &\Pr_{z \sim \mathbb{U}(\boldsymbol{y}'^{\star})} \left[ T'_{\downarrow z} \in \mathcal{A}_t \mid T'_{\downarrow z} \in \mathcal{A} \right] \\ &\quad = \sum_{\substack{\text{node } u \\ \text{with label } i \notin S}} \Pr_{z \sim \mathbb{U}(\boldsymbol{y}'^{\star})} \left[ T'_{\downarrow z} \in \mathcal{A}_t \mid T'_{\downarrow z} \in \mathcal{A} \cap G^{-1}(u) \right] \cdot \Pr_{z \sim \mathbb{U}(\boldsymbol{y}'^{\star})} \left[ T'_{\downarrow z} \in \mathcal{A} \cap G^{-1}(u) \mid T'_{\downarrow z} \in \mathcal{A} \right]. \end{aligned} \tag{4.16}$$

Observe that

$$\Pr_{z \sim \mathbb{U}(\boldsymbol{y}'^{\star})} \left[ T'_{\downarrow z} \in \mathcal{A}_t \mid T'_{\downarrow w} \in \mathcal{A} \cap G^{-1}(u) \right] = \frac{1}{2},$$

since the event is equivalent to $z[i] = 1, z[i'] = 0$, and this is equally likely to the complement event $z[i] = 0, z[i'] = 1$, given that $\boldsymbol{y}'^{\star}[i] = \boldsymbol{y}'^{\star}[i'] = \perp$. Therefore,

$$\begin{aligned}\Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow z} \in \mathcal{A}_t \mid T'_{\downarrow z} \in \mathcal{A}\right] &= \sum_{\text{node } u \text{ with label } i\notin S} \frac{1}{2} \cdot \Pr_{z\in\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow z} \in \mathcal{A} \cap G^{-1}(u) \mid T'_{\downarrow z} \in \mathcal{A}\right] \\ &= \frac{1}{2} \cdot \sum_{\text{node } u \text{ with label } i\notin S} \Pr_{z\in\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow z} \in \mathcal{A} \cap G^{-1}(u) \mid T'_{\downarrow z} \in \mathcal{A}\right] \\ &= \frac{1}{2}.\end{aligned}$$

□

**Claim 4.11** (Restated). *Given a partial instance $\boldsymbol{y}' \subseteq \boldsymbol{y}$, we define $\boldsymbol{y}'^{\star}$ as the partial instance of dimension $2n-|S|+m$ that matches $\boldsymbol{y}'$ on its defined features, and holds $\boldsymbol{y}'^{\star}[i] = \boldsymbol{y}'^{\star}[i'] = \perp$ for every feature $i$ such that $\boldsymbol{y}'[i] = \perp$. Then it holds that*

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}')}\left[T(z) = 1\right] \;=\; \Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow z} \in \mathcal{N}_t \mid T'_{\downarrow z} \in \mathcal{N}\right].$$

Proof of Claim 4.11. Given that the resulting leaf is natural, for every node $u$ of $T'$ such that $u$ is labeled with feature $i \notin S$, the tuple $(u, L, R)$ was considered when constructing $T'$ and the path of $w$ in $T'$ goes through $u$, it must be the case that $z[i] = z[i'] = 0$ or $z[i] = z[i'] = 1$, as otherwise $T'_{\downarrow z} \in \mathcal{A}$. But these two alternatives are equally likely by definition of $T'$. Thus, by using a simple induction argument, for every leaf $\ell$ of $T$ with a corresponding natural leaf $\ell'$ of $T'$, it holds that

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}')}\left[T_{\downarrow z} = \ell\right] \;=\; \Pr_{z\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow z} \;=\; \ell' \mid T'_{\downarrow z} \in \mathcal{N}\right],$$

from where the claim immediately follows. □

**Claim 4.12** (Restated). *By choosing $m \geqslant \max\{2u(T, \boldsymbol{y}) + 2n, 9\}$, we have that*

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[(T' \cap T_1)(z) = 1\right] \;>\; \frac{1}{2}.$$

Proof of Claim 4.12. First, consider that for $T_1$ we have

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T_1(z) = 1\right] \;=\; 1 - \left(\frac{1}{2}\right)^m - m\left(\frac{1}{2}\right)^m = 1 - (m+1)\left(\frac{1}{2}\right)^m,$$

while on the other hand

$$\begin{aligned}\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'(z) = 1\right] &= \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z} \in \mathcal{A}_t\right] + \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z} \in \mathcal{N}_t\right] \\ &= \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z} \in \mathcal{A}_t \mid T'_{\downarrow z} \in \mathcal{A}\right] \cdot \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z} \in \mathcal{A}\right] \\ &\qquad + \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z} \in \mathcal{N}_t \mid T'_{\downarrow z} \in \mathcal{N}\right] \cdot \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z} \in \mathcal{N}\right] \\ &= \frac{1}{2} \cdot \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z} \in \mathcal{A}\right] + \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z} \in \mathcal{N}_t \mid T'_{\downarrow z} \in \mathcal{N}\right] \cdot \Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z} \in \mathcal{N}\right],\end{aligned}$$

where the last equality uses Claim 4.10. Let us now show that

$$\Pr_{z\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow z} \in \mathcal{N}_t \mid T'_{\downarrow z} \in \mathcal{N}\right] \;\geqslant\; \frac{1}{2} + \left(\frac{1}{2}\right)^n.$$

By Claim 4.11, this is the same as showing that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y})}[T(\boldsymbol{z})=1] \;\geqslant\; \frac{1}{2}+\left(\frac{1}{2}\right)^{n},$$

which is guaranteed by the definition of the SB-Check-SUB-SR problem, as we know that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y})}[T(\boldsymbol{z})=1] \;>\; \frac{1}{2},$$

and also that $\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y})}[T(\boldsymbol{z})=1]$ must be of the form $\left(\frac{k}{2^n}\right)$ with $k\in\mathbb{N}$, given that $n$ is the dimension of $T$.

Now, consider that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'_{\downarrow\boldsymbol{z}}\in\mathcal{A}] \;=\; 1-\left(\frac{1}{2}\right)^{u(T,\boldsymbol{y})}.$$

Notice that this holds because $T$ is strongly-balanced, so falling into a natural leaf in $T'$ requires going through $u(T,\boldsymbol{y})$ layers without choosing an artificial leaf, which happens with probability $\frac{1}{2}$ at each layer. Thus, we have that

$$\begin{aligned}\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'(\boldsymbol{z})=1] \;&\geqslant\; \frac{1}{2}\left(1-\left(\frac{1}{2}\right)^{u(T,\boldsymbol{y})}\right)+\left(\frac{1}{2}+\left(\frac{1}{2}\right)^{n}\right)\left(\frac{1}{2}\right)^{u(T,\boldsymbol{y})}\\ &=\; \frac{1}{2}+\left(\frac{1}{2}\right)^{n+u(T,\boldsymbol{y})}.\end{aligned}$$

Moreover, given that $T'$ and $T_1$ impose restrictions over disjoint sets of features, we have that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'(\boldsymbol{z})=1 \mid T_1(\boldsymbol{z})=1] \;=\; \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'(\boldsymbol{z})=1].$$

Putting together all the previous results, we obtain that

$$\begin{aligned}\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\star)}[(T'\cap T_1)(\boldsymbol{z})=1] &= \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'(\boldsymbol{z})=1 \mid T_1(\boldsymbol{z})=1]\cdot\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\star)}[T_1(\boldsymbol{z})=1]\\
&= \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\star)}[T'(\boldsymbol{z})=1]\cdot\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^\star)}[T_1(\boldsymbol{z})=1]\\
&\geqslant \left(\frac{1}{2}+\left(\frac{1}{2}\right)^{n+u(T,\boldsymbol{y})}\right)\left(1-(m+1)\left(\frac{1}{2}\right)^{m}\right)\\
&= \frac{1}{2}-(m+1)\left(\frac{1}{2}\right)^{m+1}+\left(\frac{1}{2}\right)^{n+u(T,\boldsymbol{y})}-(m+1)\left(\frac{1}{2}\right)^{m+n+u(T,\boldsymbol{y})}\\
&\geqslant \frac{1}{2}-(m+1)\left(\frac{1}{2}\right)^{m}+\left(\frac{1}{2}\right)^{n+u(T,\boldsymbol{y})}\\
&\geqslant \frac{1}{2}-\left(\frac{1}{2}\right)^{m-\lceil\log(m+1)\rceil}+\left(\frac{1}{2}\right)^{n+u(T,\boldsymbol{y})}.\end{aligned}$$

But we are assuming $m\geqslant\max\{2n+2u(T,\boldsymbol{y}),9\}$, which implies that $m\geqslant 2n+2u(T,\boldsymbol{y})$. Hence, we have that

$$m-\lceil\log(m+1)\rceil > n+u(T,\boldsymbol{y}),$$

as $m-\lceil\log(m+1)\rceil > \frac{m}{2}$ since $m\geqslant 9$. We conclude that

$$\frac{1}{2}-\left(\frac{1}{2}\right)^{m-\lceil\log(m+1)\rceil}+\left(\frac{1}{2}\right)^{n+u(T,\boldsymbol{y})} \;>\; \frac{1}{2},$$

from which the claim follows. □

**Claim 4.13** (Restated). *For every partial instance $\boldsymbol{y}'^{\star} \subseteq \boldsymbol{y}^{\star}$, it holds that*

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}\right] = \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}\right],$$
$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}\left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{N}\right] = \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\star})}\left[T'_{\downarrow \boldsymbol{z}} \in \mathcal{N}\right].$$

Proof of Claim 4.13. We only need to prove that

$$\Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}'^{\star})}[T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}] = \Pr_{\boldsymbol{z}\sim\mathbb{U}(\boldsymbol{y}^{\star})}[T'_{\downarrow \boldsymbol{z}} \in \mathcal{A}].$$

As shown in the proof of Claim 4.12, this follows from the strongly-balanced property of $T$. □

## 5 Proofs of Non-approximability Results

Both Theorem 3.4 and Theorem 3.5 (whose statements we will recall shortly) are consequences of the hardness of a related problem: we are given a decision tree $T$, and we are allowed to fix an arbitrary subset of the variables to 1. Our goal is to maximize the probability that $T$ outputs 1 under this fixation. Note that fixing to 0 is not allowed, otherwise, we could just find an arbitrary input on which $T$ is 1 (unless $T$ is always 0) and fix all the variables according to this input. If $T$ is equal to 1 on the input of all 1s, the optimal solution is fixing all variables to 1, but of course this will not happen in hard instances. For this problem, we obtain the following hardness result for its promise version.

Theorem 5.1. *Unless* QP = NP, *for every $\kappa > 0$ there exists no polynomial-time algorithm that, given a decision tree $T$ over $n$ variables, distinguishes between the following two cases:*

- *there exists a partial instance $\boldsymbol{y} \in \{1, \bot\}^n$ such that $\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \geq 1 - \kappa$;*
- *there exists no partial instance $\boldsymbol{y} \in \{1, \bot\}^n$ such that $\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \geq \kappa$.*

We first derive Theorem 3.4 and Theorem 3.5 from this result, and then we prove it separately.

Theorem 3.4 (Restated). *Assuming that* NP $\not\subseteq$ QP, *for every $\varepsilon > 0$ there is no polynomial-time algorithm that, given a decision tree $T$ of dimension $n$ and an input $\boldsymbol{x} \in \{0,1\}^n$, distinguishes between the following two cases:*

- *there exists an $(1-\varepsilon)$-sufficient reason $\boldsymbol{y}$ for $\boldsymbol{x}$ under $T$ with $|\boldsymbol{y}|_\bot \geq n - n^\varepsilon$.*
- *there exists no $\varepsilon$-sufficient reason $\boldsymbol{y}$ for $\boldsymbol{x}$ under $T$ with $|\boldsymbol{y}|_\bot \geq n^\varepsilon$.*

Proof of Theorem 3.4. For every $\varepsilon > 0$, we provide a polynomial-time reduction from the problem from Theorem 5.1 with $\kappa = \varepsilon/2$ to the problem from Theorem 3.4 with parameter $\varepsilon$. Let $\Delta$ be any positive integer integer larger than $1/\varepsilon$. Define $m = (n + 2\Delta)^\Delta$. Given a decision tree $T$ over $n$ variables, consider a decision tree $T_1$ over $n + m$ variables, defined as follows:

$$T_1(x_1, \ldots, x_{n+m}) = T(x_1, \ldots, x_n) \vee (x_{n+1} \wedge x_{n+2} \wedge \ldots \wedge x_{n+m}).$$

Note that $T_1$ can be constructed in polynomial time from $T$ (to every leaf of $T$ that outputs 0 attach a tree computing the conjunction $x_{n+1} \wedge x_{n+2} \wedge \ldots \wedge x_{n+m}$). Let $e^k$ denote a Boolean vector of dimension $k$ where all coordinates are 1s. It remains to establish two claims.

- If there exists a partial instance $\boldsymbol{y} \in \{1, \bot\}^n$ such that $\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \geq 1 - \kappa$, then there exists a $(1-\varepsilon)$-sufficient reason $\boldsymbol{y}_1$ for $e^{n+m}$ under $T_1$ such that $|\boldsymbol{y}_1|_\bot \geq (n+m) - (n+m)^\varepsilon$.
- If there exists no partial instance $\boldsymbol{y} \in \{1, \bot\}^n$ such that $\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})] \geq \kappa$, then there exists no $\varepsilon$-sufficient reason $\boldsymbol{y}_1$ for $e^{n+m}$ under $T_1$ such that $|\boldsymbol{y}_1|_\bot \geq (n+m)^\varepsilon$.

Let us start with the first claim. First, note that $T(e^{n+m}) = 1$. Let $\boldsymbol{y}_1$ be simply an extension of $\boldsymbol{y}$ to the variables $x_{n+1}, \ldots, x_{n+m}$ by letting them be undefined. Thus,

$$|\boldsymbol{y}_1|_\perp \geq m = (n+m) - n = (n+m) - (n^{1/\varepsilon})^\varepsilon \geq (n+m) - (n+m)^\varepsilon$$

where the last inequality is because $m = (n+2\Delta)^\Delta \geq n^{1/\varepsilon}$. It remains to show that $\boldsymbol{y}_1$ is a $(1-\varepsilon)$-sufficient reason for $e^{n+m}$ under $T_1$. This is because by sampling a random completion of $\boldsymbol{y}_1$ just in variables $x_1, \ldots, x_n$, we already make $T$ to be equal to 1 with probability $1 - \kappa = 1 - \varepsilon/2 > 1 - \varepsilon$. But if $T$ is equal to 1, then $T_1$ is also equal to 1.

Now, let us show the second claim. Assume that $\boldsymbol{y}_1$ is an $\varepsilon$-sufficient reason for $e^{n+m}$ under $T_1$. Our goal is to show that $|\boldsymbol{y}_1|_\perp < (n+m)^\varepsilon$. Consider a uniformly random completion $\mathbf{z}$ of $\boldsymbol{y}_1$. On one hand, $\mathbf{z}$ makes $T_1$ equal to 1 with probability at least $\varepsilon$. On the other hand, variables that are defined in $\boldsymbol{y}_1$ are all equal to 1 (as $\boldsymbol{y}_1$ is a sufficient reason for the input of all 1s), meaning that $\mathbf{z}$ makes $T$ equal to 1 with probability less than $\kappa = \varepsilon/2$. Hence, a uniformly random completion of $\boldsymbol{y}_1$ makes the conjuction $x_{n+1} \wedge x_{n+2} \wedge \ldots \wedge x_{n+m}$ true with probability larger than $\varepsilon/2$. On the other hand, the probability of making this conjunction true is 2 to the power of (minus the number of variables among $x_{n+1}, \ldots, x_{n+m}$ that are undefined in $\boldsymbol{y}_1$). The number of such variables is at least $|\boldsymbol{y}_1|_\perp - n$, which gives us:

$$\varepsilon/2 \;<\; 2^{n-|\boldsymbol{y}_1|_\perp},$$

from where we get an upper bound on $|\boldsymbol{y}_1|_\perp$:

$$|\boldsymbol{y}_1|_\perp \;<\; n + \log_2(2/\varepsilon) \;\leq\; n + \log_2(2\Delta) \leq n + 2\Delta \;=\; m^{1/\Delta} \;\leq\; (n+m)^\varepsilon.$$

This finishes the proof. □

Theorem 3.5 (Restated). *Assuming that* NP $\not\subseteq$ QP*, for every* $\varepsilon \in (0,1)$*, there exists no polynomial-time algorithm that, given a decision tree* $T$ *of dimension* $n$ *and an input* $\boldsymbol{x} \in \{0,1\}^n$*, distinguishes between the following two cases:*

- *the only* $(1/2+\varepsilon)$*-sufficient reason for* $\boldsymbol{x}$ *under* $T$ *is* $\boldsymbol{x}$ *itself.*
- *there exists a* $(1-\varepsilon)$*-sufficient reason for* $\boldsymbol{x}$ *under* $T$ *other than* $\boldsymbol{x}$*.*

Proof of Theorem 3.5. Given $\varepsilon > 0$, we establish a polynomial-time reduction from the problem given in Theorem 5.1 with $\kappa = \varepsilon$ to the problem from Theorem 3.5 with parameter $\varepsilon$. Suppose we are given a decision tree $T$ of dimension $n$, and we want to distinguish the case when there exists a fixation of some variables to 1s under which the probability of having 1 in the output of $T$ is at least $1-\varepsilon$, from the case when all such fixations give probability less than $\varepsilon$. First, we check if $T(1, \ldots, 1) = 1$, and if so, we do not need any reduction. From now on, assume that $T(1, \ldots, 1) = 0$. We define a tree

$$T_1(x_1, \ldots, x_n) = T(x_1, \ldots, x_n) \vee (x_1 \wedge x_2 \wedge \ldots \wedge x_n)$$

and we consider the input $e^n = (1, \ldots, 1)$ to it, on which $T_1$ is equal to 1 because of the conjunction. If we can fix some variables to 1s such that $T$ is equal to 1 with probability at least $1-\varepsilon$, then this fixation is a $(1-\varepsilon)$-sufficient reason for $x$ under $T_1$, just because $T_1$ is 1 whenever $T$ is. This sufficient reason cannot be $e^n$ itself because $T$ is 0 on the input of all 1s. Now, assume that under any fixation of some variables to 1s, the probability that $T$ is equal to 1 is lower than $\varepsilon$. Then for any fixation where at least one variable is not fixed, the probability of $T_1$ to be equal to 1 is lower than $1/2+\varepsilon$, because the conjunction $(x_1 \wedge x_2 \wedge \ldots x_n)$ is equal to 1 with probability at most $1/2$ then. This conludes the proof as then there is no $(1/2+\varepsilon)$-sufficient reason for $e^n$ under $T_1$ other than $e^n$. □

Proof of Theorem 5.1. To simplify the presentation, within this proof we employ the following notation:

$$T(\boldsymbol{y}) = \Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})]$$

for a decision tree $T$ of dimension $n$ and for a partial instance $\boldsymbol{y} \in \{1, \perp\}^n$.

We use a standard probability concentration inequality due to Hoeffding [26].

**Proposition 5.2.** *Let us assume that $X_1, X_2, \ldots, X_n$ are independent Bernoulli random variables. Denote $S_n = X_1 + \ldots + X_n$. Then for every $\delta > 0$ we have:*

$$\Pr\left[\frac{S_n}{n} \geq \frac{\mathbb{E}S_n}{n} + \delta\right] \leq \exp\{-2\delta^2 n\}.$$

We reduce from the *1-in-k exact hitting set problem* (1-in-E$k$HS) problem. It is a version of the SAT problem where each clause contains exactly $k$ variables (without negations) and a clause is satisfied if and only if there is exactly one variable in it which is set to 1. We use the following result of Guruswami and Trevisan [27, Lemma 13]:

Theorem 5.3. *For every $\delta > 0$ there exists $k$ such that it is NP-hard to distinguish satisfiable instances of 1-in-E$k$HS from instances of 1-in-E$k$HS for which it is impossible to satisfy more than a fraction $(1/e + \delta)$ of clauses.*

We fix any $k$ such that it is NP-hard to distinguish satisfiable instances of 1-in-E$k$HS from instances of 1-in-E$k$HS for which it is impossible to satisfy more than a fraction $1/2$ of clauses. Now, fix $\kappa > 0$. To establish Theorem 5.1, we take any instance $\varphi$ of 1-in-E$k$HS with $n$ variables and $m$ clauses and in *quasi-polynomial* time construct a decision tree $T$ over $N$ variables such that

- if $\varphi$ is satisfiable, then there exists $\boldsymbol{y} \in \{1, \bot\}^N$ such that $T(\boldsymbol{y}) \geq 1 - \kappa$;
- if it is impossible to satisfy more than a fraction $1/2$ of clauses of $\varphi$, then there exists no $\boldsymbol{y} \in \{1, \bot\}^N$ such that $T(\boldsymbol{y}) \geq \kappa$.

Let $l$ be the smallest integral number such that $m \leq 2^{2l}$. Note that $l = \Theta(\log m)$. The most technical part of the proof is to construct in polynomial time an $O(\log m)$-depth decision tree $L$ over $n + 2l + 2$ variables such that for some absolute constant $c > 0$ we have:

- if $\varphi$ is satisfiable, then there exists $\boldsymbol{y} \in \{1, \bot\}^{n+2l+2}$ such that $L(\boldsymbol{y}) \geq 7/8$;
- if it is impossible to satisfy more than a fraction $1/2$ of clauses of $\varphi$, then there for every $\boldsymbol{y} \in \{1, \bot\}^{n+2l+2}$ we have $L(\boldsymbol{y}) \leq 7/8 - \delta$, where $\delta = \frac{c}{\sqrt{\ln(m)}}$.

We first describe how to "increase the gap" from ($7/8 - \delta$ vs. $7/8$) to ($\kappa$ vs. $1 - \kappa$). The construction of $L$ is given afterwards.

*Finishing the proof modulo the construction of $L$.* We define a decision tree $T$ as

$$K = \frac{2\ln(2/\kappa)}{\delta^2} = O(\log m)$$

"independent copies" of $L$. More specifically, we let $T$ be over $N = K(n + 2l + 2)$ variables. First, $T$ runs $L$ on the first $n + 2l + 2$ variables, then on the second $n + 2l + 2$ variables, and so on (in total, $K$ runs). In the end, it outputs 1 if and only if for at least a fraction $7/8 - \delta/2$ of runs the output of $L$ was 1.

Assume first that $\varphi$ is satisfiable. Then, by definition of $L$, there exists $\boldsymbol{y} \in \{1, \bot\}^{n+2l+1}$ such that $L(\boldsymbol{y}) \geq 7/8$. Repeat it $K$ times to obtain a partial input $Y \in \{1, \bot\}^N$ for $T$. We claim that $T(Y) \geq 1 - \kappa$. Indeed, a random completion of $Y$ can be generated as $K$ independent samples of a random completion of $\boldsymbol{y}$. The tree $T$ outputs 1 if and only if $L$ outputs 1 for at least a fraction $7/8 - \delta/2$ of the samples. On the other hand, the probability that $L$ outputs 1 on one of the samples is $L(\boldsymbol{y}) \geq 7/8$. Hence, the average fraction of samples on which $L$ outputs 1 is at least $7/8$. By Hoeffding's inequality (Proposition 5.2), the probability that $T$ outputs 1 is at least $1 - \exp\{-2(\delta/2)^2 \cdot K\} = 1 - \kappa/2 > 1 - \kappa$.

Now, assume that it is impossible to satisfy more than a fraction $1/2$ of clauses of $\varphi$. Then $L(\boldsymbol{y}) \leq 7/8 - \delta$ for all $\boldsymbol{y} \in \{1, \bot\}^{n+2l+2}$. We claim that $T(Y) < \kappa$ for every $Y \in \{1, \bot\}^N$. Let $\boldsymbol{y}_1$ be a restriction of $Y$ to the first $n + 2l + 2$ variables, $\boldsymbol{y}_2$ be a restriction of $Y$ to the second $n + 2l + 2$ variables, and so on. Generating a random completion of $Y$ is the same as independently sampling random completions of $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_K$. For each $i \in \{1, \ldots, K\}$, the

probability that $L$ outputs 1 on a random completion of $\boldsymbol{y}_i$ is $L(\boldsymbol{y}_i) \leq 7/8 - \delta$. Hence, the average fraction of $i \in \{1, \ldots, K\}$ such that $L$ outputs 1 on a random completion of $\boldsymbol{y}_i$ is at most $7/8 - \delta$. In turn, $T(Y)$ is the probability that the fraction of $i \in \{1, \ldots, K\}$ such that $L$ outputs 1 on a random completion of $\boldsymbol{y}_i$ is at least $7/8$. Hence, by Hoeffding's inequality, $T(Y)$ is bounded by $\exp\{-2(\delta/2)^2 \cdot K\} = \kappa/2 < \kappa$.

The depth of $T$ is (depth of $L$) $\times K = O(\log^2 m)$. Hence, the size of $T$ is quasi-polynomial. The time to construct $T$ from $L$ is also quasi-polynomial because it is polynomial in the size of $T$.

*Construction of $L$.* Let $x_1, \ldots, x_n$ denote variables of $\varphi$. The variables of $L$ will be denoted by $x_1, \ldots, x_n, y_1, \ldots, y_{2l+1}, z$.

First, $L$ asks the values of $y_1, \ldots, y_{2l+1}$. Now, we will call a binary word "fat" if it has more 1s than 0s, and "'thin" otherwise. If $y_1 \ldots y_{2l+1}$ is thin, $L$ outputs 1. To describe what $L$ does when $y_1 \ldots y_{2k+1}$ is fat, we first notice that in $\{0,1\}^{2k+1}$ exactly half of the words are fat and exactly half are thin. In particular, the number of fat words is $2^{2l}$. We fix any surjective mapping from the set of fat words in $\{0,1\}^{2l+1}$ to the set of clauses of $\varphi$. Such mapping exists because $m \leq 2^{2l}$ by our choice of $l$.

So, when $y_1 \ldots y_{2l+1}$ is fat, $L$ takes the clause $C$, assigned to $y_1 \ldots y_{2l+1}$, and runs a decision tree $L_C$, defined as follows. If $C$ is over variables $x_{i_1}, \ldots, x_{i_k}$, then $L_C$ is over $x_{i_1}, \ldots, x_{i_k}$ and $z$. The tree $L_C$ asks the values of all its variables. It outputs 1 if and only if either there is exactly one 0 among $x_{i_1}, \ldots, x_{i_k}$ or ($x_{i_1} = \ldots = x_{i_k} = 1$ and $z = 0$).

The depth of $L$ is $2l + 1 + k + 1 = O(l) = O(\log m)$, and it can be constructed from $\varphi$ in polynomial time.

Consider any partial input $\boldsymbol{p}$ to $L_C$ in which no coordinate is equal to 0 (i.e, we can only have variables that are fixed to 1 and undefined variables). We call $\boldsymbol{p}$ good for $C$ if $z$ is undefined in $\boldsymbol{p}$ and there is also exactly one undefined variable among $x_{i_1}, \ldots, x_{i_k}$ in $\boldsymbol{p}$. Otherwise, we call $\boldsymbol{p}$ bad for $C$.

**Lemma 5.4.** *If $\boldsymbol{p}$ is good for $C$, we have $L_C(\boldsymbol{p}) = 3/4$, and if $\boldsymbol{p}$ is bad for $C$, $L_C(\boldsymbol{p}) \leq 5/8$.*

Proof of Lemma 5.4. Assume first that $\boldsymbol{p}$ is good for $C$. Let $x_{i_j}$ be a variable among $x_{i_1}, \ldots, x_{i_k}$ which is undefined in $\boldsymbol{p}$. Consider a random completion of $\boldsymbol{p}$. If $x_{i_j}$ is 0 in this completion, then there is exactly one 0 among $x_{i_1}, \ldots, x_{i_k}$, and hence $L_C$ outputs 1. If $x_{i_j}$ is 1, $L_C$ outputs 1 with probability $1/2$, depending on whether $z = 0$. Overall, we get $L_C(\boldsymbol{p}) = (1/2) \cdot 1 + (1/2) \cdot (1/2) = 3/4$.

Now, consider the case when $\boldsymbol{p}$ is bad for $C$. Assume first that there are $t \geq 2$ undefined variables among $x_{i_1}, \ldots, x_{i_k}$ in $\boldsymbol{p}$. Consider a random completion of $\boldsymbol{p}$. By definition, $L_C$ outputs 1 on it only in the following two cases:

(a) There is exactly 1 undefined variable among $x_{i_1}, \ldots, x_{i_k}$ which is equal to 0 in our random completion;
(b) all undefined variables among $x_{i_1}, \ldots, x_{i_k}$ are equal to 1 and $z = 0$ in our random completion.

The probability of (a) is $t2^{-t}$. The probability of (b) is $2^{-t-1}$ if $z$ is undefined and 0 otherwise. Overall, we get $L_C(\boldsymbol{p}) \leq t2^{-t} + 2^{-t-1}$. This expression decreases in $t$, and for $t = 2$ it is equal to $5/8$.

Now, assume that the number of undefined variables among $x_{i_1}, \ldots, x_{i_k}$ is at most 1. If it is 1, then $z$ has to be fixed to 1 in $\boldsymbol{p}$ (otherwise, $\boldsymbol{p}$ is good for $C$). Then $L_C$ can only output 1 if the unique undefined variable among $x_{i_1}, \ldots, x_{i_k}$ is 0, and this happens with probability $1/2$ in a random completion of $\boldsymbol{p}$. That is, in this case, $L_C(\boldsymbol{p}) = 1/2$. Now, if all variables $x_{i_1}, \ldots, x_{i_k}$ are fixed to 1s in $\boldsymbol{p}$, then $L_C$ outputs 1 only if $z = 0$. This either has probability $1/2$ (if $z$ is undefined) or 0 (if $z$ is fixed to 1). Hence, in this case, $L_C(\boldsymbol{p}) \leq 1/2$. □

Assume first that $\varphi$ is satisfiable. Fix any satisfying assignment $\alpha$ to $\varphi$. We construct a partial input $\boldsymbol{p} \in \{1, \bot\}^{n+2l+2}$ to $L$ such that $L(\boldsymbol{p}) = 7/8$. We define $p$ on $x_1, \ldots, x_n$ as follows:

$$x_i = 0 \text{ in } \alpha \implies x_i = 1 \text{ in } \boldsymbol{p} \tag{5.5}$$

$$x_i = 1 \text{ in } \alpha \implies x_i = \bot \text{ in } \boldsymbol{p} \tag{5.6}$$

Variables $y_1, \ldots, y_{2l+1}, z$ are undefined in $\boldsymbol{p}$. Since $\alpha$ is a satisfying assignment to $\varphi$, for every clause $C$ there is exactly one variable in $C$ which is undefined in $\boldsymbol{p}$. This means that the restriction of $\boldsymbol{p}$ to variables of $L_C$ is

good in the sense of Lemma 5.4, for every clause of $C$. Thus, conditioned on the event that $y_1 \dots y_{2l+1}$ is fat, $L$ outputs 1 with probability $3/4$ on a random completion of $\boldsymbol{p}$. Now, variables $y_1, \dots, y_{2l+1}$ are undefined in $\boldsymbol{p}$, which means that they are all sampled independently uniformly at random in our completion. The probability of the event "$y_1 \dots y_{2l+1}$ is fat" is $1/2$, which contributes $(1/2) \cdot (3/4) = 3/8$ to $L(\boldsymbol{p})$. In turn, conditioned on the event "$y_1 \dots y_{2l+1}$ is thin" (which also happens with probability $1/2$), our tree always outputs 1. Overall, we get $L(\boldsymbol{p}) = 3/8 + 1/2 = 7/8$.

We now show that if it is impossible to satisfy more than a half of clauses of $\varphi$, then for every $\boldsymbol{p} \in \{1, \bot\}^{n+2l+2}$ we have $L(\boldsymbol{p}) \leq 7/8 - \delta$, where $\delta = c/\sqrt{\ln(m)}$ and $c > 0$ is some absolute constant. Again, we do so by taking a random completion of $\boldsymbol{p}$ and showing that the probability that $L$ outputs 1 on it is at most $7/8 - \delta$.

Assume first that $y_1, \dots, y_{2l+1}$ are all undefined in $\boldsymbol{p}$. We turn $\boldsymbol{p}$ into an assignment $\alpha$ to $\varphi$ by reversing (5.5–5.6):

$$x_i = 1 \text{ in } \boldsymbol{p} \implies x_i = 0 \text{ in } \alpha$$
$$x_i = \bot \text{ in } \boldsymbol{p} \implies x_i = 1 \text{ in } \alpha$$

Observe that if $\alpha$ does not satisfy a clause $C$, then the restriction of $\boldsymbol{p}$ to variables of $L_C$ is bad for $C$ in the sense of Lemma 5.4. Now, the number of unsatisfied clauses is at least $m/2$. Each of these clauses is assigned to some fat $y_1 \dots y_{2l+1}$. That is, for at least $m/2$ fat $y_1 \dots y_{2k+1}$, the probability that $L$ outputs 1 (conditioned on this fixation of $y_1 \dots y_{2k+1}$) is at most $5/8$. For any other fat $y_1 \dots y_{2k+1}$, this conditional probability is at most $3/4$. Overall, the contribution of fat $y_1 \dots y_{2k+1}$ to $L(\boldsymbol{p})$ does not exceed

$$\frac{m/2}{2^{2l+1}} \cdot (5/8) \; + \; \frac{2^{2l} - m/2}{2^{2l+1}} \cdot (3/4),$$

and the contribution of thin $y_1 \dots y_{2k+1}$ to $L(\boldsymbol{p})$ is, as before, $1/2$. Overall, we get

$$L(\boldsymbol{p}) \; \leq \; \frac{m/2}{2^{2l+1}} \cdot (5/8) \; + \; \frac{2^{2l} - m/2}{2^{2l+1}} \cdot (3/4) \; + \; 1/2 \; = \; 7/8 - \frac{m}{2^{2l+5}}.$$

It remains to recall that $l$ was chosen as the smallest integral number such that $m \leq 2^{2l}$. This means that $m \geq 2^{2(l-1)}$. This gives us $L(\boldsymbol{p}) \leq 7/8 - 1/128 < 7/8 - \delta$.

It remains to consider the case when in $\boldsymbol{p}$ at least one variable among $y_1, \dots, y_{2l+1}$ is fixed to 1. As before, the probability that $L$ outputs 1, conditioned on the event "$y_1, \dots, y_{2k+1}$ is fat", is at most $3/4$. And this probability is 1 conditioned on the event "$y_1, \dots, y_{2k+1}$ is thin". Now, however, since at least one variable among $y_1, \dots, y_{2k+1}$ is fixed to 1, the probability to be thin is slightly less than $1/2$, which makes $L(\boldsymbol{p})$ slightly less than $7/8$. More specifically,

$$\begin{aligned} L(\boldsymbol{p}) \; &\leq \; \Pr[y_1 \dots y_{2l+1} \text{ is thin}] + (3/4) \Pr[y_1 \dots y_{2l+1} \text{ is fat}] \\ &= \; 1 - (1/4) \Pr[y_1 \dots y_{2l+1} \text{ is fat}], \end{aligned}$$

where $y_1, \dots, y_{2l+1}$ are chosen from a random completion of $\boldsymbol{p}$. That is, now our goal is to lower bound $\Pr[y_1 \dots y_{2l+1} \text{ is fat}]$, assuming that at least one of $y_1, \dots, y_{2l+1}$ is fixed to 1, and the rest of them are chosen independently uniformly at random. The more variables we fix to 1, the larger becomes $\Pr[y_1 \dots y_{2l+1} \text{ is fat}]$. Hence, without loss of generality we may assume that exactly one variable is fixed to 1, say, $y_{2l+1}$. Then $y_1 \dots y_{2l+1}$

is fat if and only if there are at least $l$ ones among $y_1, \ldots, y_{2l}$. In conclusion, we get

$$\begin{aligned}
\Pr[y_1 \ldots y_{2l+1} \text{ is fat}] &\geq \frac{\binom{2l}{2l} + \ldots + \binom{2l}{l+1} + \binom{2l}{l}}{2^{2l}} \\
&= \frac{\frac{\binom{2l}{2l}+\binom{2l}{0}}{2} + \ldots + \frac{\binom{2l}{l+1}+\binom{2l}{l-1}}{2} + \frac{\binom{2l}{l}+\binom{2l}{l}}{2}}{2^{2l}} \\
&= \frac{\binom{2l}{2l} + \ldots + \binom{2l}{0}}{2 \cdot 2^{2l}} + \frac{\binom{2l}{l}}{2 \cdot 2^{2l}} = \frac{1}{2} + \Omega(1/\sqrt{l}) \\
&= \frac{1}{2} + \Omega(1/\sqrt{\ln m}).
\end{aligned}$$

From this, we get $L(\boldsymbol{p}) \leq 1 - (1/4) \cdot (1/2 + \Omega(1/\sqrt{\ln m})) = 7/8 - \Omega(1/\sqrt{\ln m})$, as required. □

## 6 Proofs of the Tractability Results

We now restate and proof our main positive results.

Theorem 3.6 (Restated). *Both problems Compute-Minimum-SR and Compute-Minimal-SR can be solved in polynomial time for decision trees with split number at most c, where $c \geqslant 1$ is any fixed integer.*

Proof of Theorem 3.6. It suffices to provide a polynomial time algorithm for Compute-Minimum-SR. (The same algorithm works for Compute-Minimal-SR as a minimum $\delta$-SR is in particular minimal.) In turn, using standard arguments, it is enough to provide a polynomial time algorithm for the following decision problem Check-Minimum-SR: Given a tuple $(T, \boldsymbol{y}, \delta, k)$, where $T$ is a decision tree of dimension $n$, $\boldsymbol{y} \in \{0, 1, \bot\}^n$ is a partial instance, $\delta \in (0, 1]$, and $k \geqslant 0$ is an integer, decide whether there is a partial instance $\boldsymbol{y}' \subseteq \boldsymbol{y}$ such that $n - |\boldsymbol{y}'|_\bot \leqslant k$ (i.e., $\boldsymbol{y}'$ has at most $k$ defined components) and

$$\Pr_{\boldsymbol{z}}[T(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}')] \geqslant \delta.$$

To solve Check-Minimum-SR over an instance $(T, \boldsymbol{y}, \delta, k)$, where $T$ has split number at most $c$, we apply dynamic programming over $T$ in a bottom-up manner. Let $Z \subseteq \{1, \ldots, n\}$ be the set of features defined in $\boldsymbol{y}$, that is, features $i$ with $\boldsymbol{y}[i] \neq \bot$. Those are the features we could eventually remove when looking for $\boldsymbol{y}'$. For each node $u$ in $T$, we solve a polynomial number of subproblems over the subtree $T_u$. We define

$$\mathsf{Int}(u) \coloneqq \mathcal{F}\left(N_u^{\downarrow}\right) \cap \mathcal{F}\left(N_u^{\uparrow}\right) \cap Z \qquad \mathsf{New}(u) \coloneqq \left(\mathcal{F}\left(N_u^{\downarrow}\right) \setminus \mathsf{Int}(u)\right) \cap Z.$$

In other words, $\mathsf{Int}(u)$ are the features appearing both inside and outside $T_u$, while $\mathsf{New}(u)$ are the features only inside $T_u$, that is, the new features introduced below $u$. Both sets are restricted to $Z$ as features not in $Z$ play no role in the process.

Each particular subproblem is indexed by a possible size $s \in \{0, \ldots, k\}$ and a possible set $J \subseteq \mathsf{Int}(u)$ with $|J| \leqslant s$, and the goal is to compute the quantity:

$$p_{u,s,J} \coloneqq \max_{\boldsymbol{y}' \in C_{u,s,J}} \Pr_{\boldsymbol{z}}[T_u(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}')],$$

where $C_{u,s,J}$ is the space of partial instances $\boldsymbol{y}' \subseteq \boldsymbol{y}$ with $n - |\boldsymbol{y}'|_\bot \leqslant s$ and such that $\boldsymbol{y}'[i] = \boldsymbol{y}[i]$ for $i \in J$ and $\boldsymbol{y}'[i] = \bot$ for $i \in \mathsf{Int}(u) \setminus J$. In other words, the set $J$ fixes the behavior on $\mathsf{Int}(u)$ (keep features in $J$, remove features in $\mathsf{Int}(u) \setminus J$) and hence the maximization occurs over choices on the set $\mathsf{New}(u)$ (which features are kept and which features are removed). The key idea is that $p_{u,s,J}$ can be computed inductively using the information already computed for the children $u_1$ and $u_2$ of $u$. Intuitively, this holds since the common features between $T_{u_1}$ and $T_{u_2}$ are at most $c$, which is a fixed constant, and hence we can efficiently synchronize the information stored for $u_1$ and $u_2$. Finally, to solve the instance $(T, \boldsymbol{y}, \delta, k)$ we simply check whether $p_{r,k,\emptyset} \geqslant \delta$, for the root $r$ of $T$.

Formally, let us define for a set $H \subseteq Z$, the partial instance $\boldsymbol{y}_H \in \{0,1,\perp\}^n$ such that $\boldsymbol{y}_H[i] = \boldsymbol{y}[i]$ for every $i \in H$, and $\boldsymbol{y}_H[i] = \perp$ for every $i \notin H$. In particular, $\boldsymbol{y}_H \subseteq \boldsymbol{y}$. Then we can write $p_{u,s,J}$ as

$$p_{u,s,J} = \max_{\substack{K \subseteq \mathsf{New}(u) \\ |K| \leqslant s - |J|}} \Pr_{\boldsymbol{z}}[T_u(\boldsymbol{z}) = 1 \mid \boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y}_{J \cup K})].$$

Let $u_1$ and $u_2$ be the children of $u$. We have that $\mathsf{New}(u)$ is the disjoint union of:

$$\mathsf{New}(u) = \mathsf{New}(u_1) \cup \mathsf{New}(u_2) \cup \mathsf{Sync}(u),$$

where $\mathsf{Sync}(u) := \mathsf{New}(u) \cap \left(\mathcal{F}\left(N^{\downarrow}_{u_1}\right) \cap \mathcal{F}\left(N^{\downarrow}_{u_2}\right)\right)$. In other words, the features in $\mathsf{Sync}(u)$ are the features that are in both $T_{u_1}$ and $T_{u_2}$ but not outside $T_u$. We conclude by explaining the computation of $p_{u,s,J}$. We consider the following cases:

(1) The feature $i$ labeling $u$ is in $J$. This means we have to keep feature $i$. If $\boldsymbol{y}[i] = 0$, then to compute $p_{u,s,J}$ we can simply look at $u_1$ (the left child). Note that $\mathsf{Int}(u_1)$ is the disjoint union of $\mathsf{Int}(u_1) \cap \mathsf{Int}(u)$ and $\mathsf{Sync}(u)$. Then

$$p_{u,s,J} = \max_{\substack{J' \subseteq \mathsf{Sync}(u) \\ |J'| \leqslant s - |\mathsf{Int}(u_1) \cap J|}} p_{u_1,s,(\mathsf{Int}(u_1) \cap J) \cup J'}.$$

This computation can be done in polynomial time as $\mathsf{Sync}(u) \leqslant c$ and then there are a constant number of possible $J' \subseteq \mathsf{Sync}(u)$. The case when $\boldsymbol{y}[i] = 1$ is analogous, taking $u_2$ instead of $u_1$.

(2) The feature $i$ labeling $u$ is either outside $Z$ or belongs to $\mathsf{Int}(u) \setminus J$. This means feature $i$ is undefined. Again, we have that $\mathsf{Int}(u_1)$ is the disjoint union of $\mathsf{Int}(u_1) \cap \mathsf{Int}(u)$ and $\mathsf{Sync}(u)$. Similarly, $\mathsf{Int}(u_2)$ is the disjoint union of $\mathsf{Int}(u_2) \cap \mathsf{Int}(u)$ and $\mathsf{Sync}(u)$. Then

$$p_{u,s,J} = \max_{\substack{J' \subseteq \mathsf{Sync}(u) \\ |J'| \leqslant s - |J|}} \max_{\substack{0 \leqslant s_1, s_2 \leqslant s \\ s_1 + s_2 \leqslant s - |J| - |J'|}} \frac{1}{2} \cdot p_{u_1,s_1,(\mathsf{Int}(u_1) \cap J) \cup J'} + \frac{1}{2} \cdot p_{u_2,s_2,(\mathsf{Int}(u_2) \cap J) \cup J'}.$$

Again, this can be done in polynomial time as $\mathsf{Sync}(u) \leqslant c$.

(3) Finally, the remaining case is that the feature $i$ labeling $u$ is in $\mathsf{New}(u)$. In that case we have the two possibilities: either we keep feature $i$ or we remove it. If $s - |J| = 0$, then the only possible choice is to remove the feature $i$, and hence $p_{u,s,J}$ is computed exactly as in case (2). If $s - |J| > 0$. Then we take the maximum between the cases when we keep feature $i$ and the case when we remove feature $i$. For the latter, $p_{u,s,J}$ is computed exactly as in case (2). For the former, we compute $p_{u,s,J}$ in a similar way as in case (1). More precisely, if $\boldsymbol{y}[i] = 0$, then:

$$p_{u,s,J} = \max_{\substack{J' \subseteq \mathsf{Sync}(u) \\ |J'| \leqslant s - 1 - |\mathsf{Int}(u_1) \cap J|}} p_{u_1,s-1,(\mathsf{Int}(u_1) \cap J) \cup J'}.$$

The case $\boldsymbol{y}[i] = 1$ is analogous.

This finishes the proof of the theorem. □

Theorem 3.7 (Restated). *Let $\mathfrak{C}$ be a class of monotone Boolean models such that $\mathfrak{C}$-#Positive-Completions can be solved in polynomial time. Then the problem Compute-Minimal-SR can be solved in polynomial time over $\mathfrak{C}$.*

Proof of Theorem 3.7. Let us first introduce some notation. For any partial instance $\boldsymbol{y}$ of dimension $n$ and integer $i \in \{1, \ldots, n\}$, if $\boldsymbol{y}[i] \neq \perp$ then we use $\boldsymbol{y} \setminus \{i\}$ to denote the partial instance $\boldsymbol{y}'$ that is exactly equal to $\boldsymbol{y}$ except for $\boldsymbol{y}'[i] = \perp$. Now, to prove Theorem 3.7, we will use the following lemma, which is a probabilistic counterpart to Proposition 2.2.

**Lemma 6.1.** *Let $\mathfrak{C}$ be a class of monotone models, $\mathcal{M} \in \mathfrak{C}$ a model of dimension $n$, and $\boldsymbol{x} \in \{0,1\}^n$ an instance. Consider any $\delta \in (0,1]$. Then if $\boldsymbol{y} \subseteq \boldsymbol{x}$ is a $\delta$-SR for $\boldsymbol{x}$ under $\mathcal{M}$ which is not minimal, then there is a partial instance $\boldsymbol{y} \setminus \{i\}$, for some $i \in \{1,\ldots,n\}$, that is also a $\delta$-SR for $\boldsymbol{x}$ under $\mathcal{M}$.*

After proving Lemma 6.1, we can easily prove Theorem 3.7. Indeed, it suffices to make a slight modification to Algorithm 2: instead of "CheckSufficientReason$(T, \hat{\boldsymbol{y}}, \boldsymbol{x})$", we need to "CheckProbSufficientReason$(T, \hat{\boldsymbol{y}}, \boldsymbol{x}, \delta)$", which is easy since computing the probability associated to a partial instance can be done in polynomial time [12].[6] Therefore, one can compute in polynomial time a minimal $\delta$-SR for $\boldsymbol{x}$ under $\mathcal{M}$. This finishes the proof of Theorem 3.7. □

We now prove Lemma 6.1, the key for Theorem 3.7.

Proof of Lemma 6.1. Note that, if $\mathcal{M}(\boldsymbol{x}) = 1$ then we can safely assume that for every $i$ where $\boldsymbol{y}[i] \neq \perp$ it holds that $\boldsymbol{y}[i] = 1$, as otherwise if $\boldsymbol{y}[i^\star] = 0$ for some $i^\star$, then the lemma trivially holds by setting $\boldsymbol{y}' = \boldsymbol{y} \setminus \{i^\star\}$ because of monotonicity. Similarly, if $\mathcal{M}(\boldsymbol{x}) = 0$ then we can safely assume that for every $i$ where $\boldsymbol{y}[i] \neq \perp$ it holds that $\boldsymbol{y}[i] = 0$.

As by hypothesis $\boldsymbol{y}$ is not minimal, there exists a $\delta$-SR $\boldsymbol{y}^\star \subsetneq \boldsymbol{y}$ that minimizes $|\boldsymbol{y}^\star|_\perp$. We prove that $|\boldsymbol{y}^\star|_\perp = |\boldsymbol{y}|_\perp + 1$, from where the lemma immediately follows.

Assume for the sake of a contradiction that $|\boldsymbol{y}^\star|_\perp > |\boldsymbol{y}|_\perp + 1$. Then, there must exist a feature $i^\star$ that $\boldsymbol{y}^\star[i^\star] = \perp \neq \boldsymbol{y}[i^\star]$, and such that $\boldsymbol{y}^\star \cup \{i^\star\} \neq \boldsymbol{y}$, where $\boldsymbol{y}^\star \cup \{i^\star\}$ is defined as

$$(\boldsymbol{y}^\star \cup \{i^\star\})[i] = \begin{cases} \boldsymbol{y}[i^\star] & \text{if } i = i^\star \\ \boldsymbol{y}^\star[i^\star] & \text{otherwise.} \end{cases}$$

Similarly, we denote $\boldsymbol{y}^\star \cup (i^\star \to \alpha)$, with $\alpha \in \{0,1\}$, the partial instance defined as

$$(\boldsymbol{y}^\star \cup (i^\star \to \alpha))[i] = \begin{cases} \alpha & \text{if } i = i^\star \\ \boldsymbol{y}^\star[i^\star] & \text{otherwise.} \end{cases}$$

We now claim that $\boldsymbol{y}^\star \cup \{i^\star\}$ is also a $\delta$-SR for $\boldsymbol{x}$ under $\mathcal{M}$, which contradicts the minimality of $\boldsymbol{y}^\star$, as $|\boldsymbol{y}^\star \cup \{i^\star\}|_\perp < |\boldsymbol{y}^\star|$. Let us denote by $C(\mathcal{M}, \boldsymbol{y})$ the number of completions $\boldsymbol{z} \in \textsc{Comp}(\boldsymbol{y})$ such that $\mathcal{M}(\boldsymbol{z}) = 1$. Now there are two cases, if $\mathcal{M}(\boldsymbol{x}) = 1$ then

$$\begin{aligned} C(\mathcal{M}, \boldsymbol{y}^\star) &= C(\mathcal{M}, \boldsymbol{y}^\star \cup (i^\star \to 0)) + C(\mathcal{M}, \boldsymbol{y}^\star \cup (i^\star \to 1)) \\ &\leqslant 2C(\mathcal{M}, \boldsymbol{y}^\star \cup (i^\star \to 1)), \end{aligned}$$

where the inequality holds because of monotonicity. This implies that

$$\frac{C(\mathcal{M}, \boldsymbol{y}^\star \cup (i^\star \to 1))}{2^{|\boldsymbol{y}^\star \cup (i^\star \to 1)|_\perp}} = \frac{C(\mathcal{M}, \boldsymbol{y}^\star \cup (i^\star \to 1))}{2^{|\boldsymbol{y}^\star|_\perp - 1}} \geqslant \frac{C(\mathcal{M}, \boldsymbol{y}^\star)}{2 \cdot 2^{|\boldsymbol{y}^\star|_\perp - 1}} \geqslant \delta,$$

which implies that $\boldsymbol{y}^\star \cup (i^\star \to 1)$ is also a $\delta$-SR (note that $\boldsymbol{y}^\star \cup (i^\star \to 1) = \boldsymbol{y}^\star \cup \{i^\star\}$ because of the initial observation), contradicting the minimality of $\boldsymbol{y}^\star$. Similarly, if $\mathcal{M}(\boldsymbol{x}) = 0$, then

$$\begin{aligned} C(\mathcal{M}, \boldsymbol{y}^\star) &= C(\mathcal{M}, \boldsymbol{y}^\star \cup (i^\star \to 0)) + C(\mathcal{M}, \boldsymbol{y}^\star \cup (i^\star \to 1)) \\ &\geqslant 2C(\mathcal{M}, \boldsymbol{y}^\star \cup (i^\star \to 0)), \end{aligned}$$

[6] This can be easily seen by observing that with dynamic programming we can easily count the number of "completions" of a partial instance with a given classification [9].

from where

$$\frac{2^{|\boldsymbol{y}^\star \cup (i^\star \to 0)|_\perp} - C(\mathcal{M}, \boldsymbol{y}^\star \cup (i^\star \to 0))}{2^{|\boldsymbol{y}^\star \cup (i^\star \to 0)|_\perp}} = \frac{2^{|\boldsymbol{y}^\star|_\perp - 1} - C(\mathcal{M}, \boldsymbol{y}^\star \cup (i^\star \to 0))}{2^{|\boldsymbol{y}^\star|_\perp - 1}}$$
$$\geqslant 1 - \frac{C(\mathcal{M}, \boldsymbol{y}^\star)}{2 \cdot 2^{|\boldsymbol{y}^\star|_\perp - 1}} \geqslant \delta,$$

thus implying that $\boldsymbol{y}^\star \cup (i^\star \to 0)$ is also a $\delta$-SR for $\boldsymbol{x}$ under $\mathcal{M}$, which again contradicts the minimality of $\boldsymbol{y}^\star$. □

## 7 Final Remarks and Open Problems

We have settled the complexity of two open problems in formal XAI, proving that both minimal and minimum size probabilistic explanations can be hard to compute, and even approximate, for decision trees. These results further support the idea that decision trees may not always be interpretable [9, 14, 28, 19], while being the first results to do so even in a probabilistic setting. Our study has focused on Boolean models, and moreover it is based on the assumption that features are independent and identically distributed, with probability 1/2. Naturally, our hardness results in this restricted setting imply hardness in more general settings, but we leave as future work the study of tractable cases (e.g., bounded split-number trees, monotone classes that allow counting) under general, potentially correlated, distributions.

The results proven in this paper suggest that minimal (or minimum) explanations might be hard to obtain in practice even for decision trees, especially in problems where the feature space has large dimension. A way to circumvent the limitations proven in our work is to relax the guarantee of minimality, or introduce some probability of error, as done in the work of [18]. More recently, the concept of $(\delta, \varepsilon)$-sufficient reasons has been proposed [29], but such a relaxation will generally not be helpful for tractability over decision trees given the inapproximability results from Theorems 3.4 and 3.5 A promising direction of future research is to better understand the kind of guarantees and settings in which is still possible to obtain tractability.

Finally, our work leaves open some interesting technical questions:

(1) What is the *parameterized* complexity of computing minimum $\delta$-SRs over decision trees, assuming that the parameter is the size of the explanation one is looking for? It is not hard to see that W[2] hardness follows from our proofs (i.e., they are parameterized reductions all the way to Set Cover), but membership in any class is fully open. Parameterized complexity might be of particular interest for this class of problems as explanations might reasonably be expected to be very small in practice.
(2) Does Theorem 3.1 continue to hold for monotone decision trees? That is, is it the case that computing minimum $\delta$-SRs over monotone decision trees is hard for every fixed $\delta \in (0, 1]$?
(3) Is it the case that the hardness of computation for minimal sufficient reasons holds for a *fixed* $\delta \in (0, 1]$? If so, does it hold for every fixed such a $\delta$, or only for some?
(4) Is it possible to show the non-approximability results in Theorems 3.4 and 3.5 under the more standard assumption that P ≠ NP?
(5) Is it possible to extend the positive behavior of decision trees with bounded split number to more powerful Boolean ML models such as free BDDs?

## Acknowledgements

Arenas and Barceló are funded by ANID - Millennium Science Initiative Program - Code ICN17002. Barceló, Kozachinskiy and Romero are supported by the National Center for Artificial Intelligence CENIA FB210017, Basal ANID. Kozachinskiy is supported by the ANID Fondecyt Iniciación grant 11250060. Subercaseaux is supported by the U.S. National Science Foundation under grant DMS-2434625.

## References


[1] A. B. Arrieta et al. 2020. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, (June 2020), 82–115. DOI: 10.1016/j.inffus.2019.12.012.

[2] A. Shih, A. Choi, and A. Darwiche. 2018. A symbolic approach to explaining bayesian network classifiers. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (IJCAI'18). AAAI Press, Stockholm, Sweden, 5103–5111. ISBN: 9780999241127.

[3] M. T. Ribeiro, S. Singh, and C. Guestrin. 2018. Anchors: high-precision model-agnostic explanations. In *AAAI*, 1527–1535.

[4] S. M. Lundberg and S.-I. Lee. 2017. A unified approach to interpreting model predictions. In *NeurIPS*, 4765–4774.

[5] T. Yan and A. D. Procaccia. 2021. If you like shapley then you'll love the core. In *AAAI*, 5751–5759.

[6] J. Marques-Silva and A. Ignatiev. 2022. Delivering trustworthy AI through Formal XAI. In *AAAI*.

[7] A. Ignatiev, N. Narodytska, N. Asher, and J. Marques-Silva. 2021. From contrastive to abductive explanations and back again. In *AIxIA 2020 – Advances in Artificial Intelligence*. Springer International Publishing, 335–355. DOI: 10.1007/978-3-030-77091-4_21.

[8] J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, and N. Narodytska. 2020. Explaining naive bayes and other linear classifiers with polynomial time and delay. In *NeurIPS*.

[9] P. Barceló, M. Monet, J. Pérez, and B. Subercaseaux. 2020. Model interpretability through the lens of computational complexity. In *NeurIPS*, 15487–15498.

[10] J. Marques-Silva, T. Gerspacher, M. C. Cooper, A. Ignatiev, and N. Narodytska. 2021. Explanations for monotonic classifiers. In *ICML*, 7469–7479.

[11] S. Wäldchen, J. MacDonald, S. Hauch, and G. Kutyniok. 2021. The computational complexity of understanding binary classifier decisions. *J. Artif. Intell. Res.*, 70, 351–387.

[12] Y. Izza, A. Ignatiev, N. Narodytska, M. C. Cooper, and J. Marques-Silva. 2021. Efficient explanations with relevant sets. *ArXiv*, abs/2106.00546.

[13] E. Wang, P. Khosravi, and G. V. den Broeck. 2021. Probabilistic sufficient explanations. In *IJCAI*. Z. Zhou, (Ed.), 3082–3088.

[14] Z. C. Lipton. 2018. The mythos of model interpretability. *Queue*, 16, 3, 31–57.

[15] Y. Izza, A. Ignatiev, and J. Marques-Silva. 2020. On explaining decision trees. (2020). DOI: 10.48550/arXiv.2010.11034.

[16] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. A. Specter, and L. Kagal. 2018. Explaining explanations: an overview of interpretability of machine learning. In *DSAA*. F. Bonchi, F. J. Provost, T. Eliassi-Rad, W. Wang, C. Cattuto, and R. Ghani, (Eds.), 80–89.

[17] Y. Izza, A. Ignatiev, N. Narodytska, M. C. Cooper, and J. Marques-Silva. 2022. Provably precise, succinct and efficient explanations for decision trees. (2022). DOI: 10.48550/arXiv.2205.09569.

[18] G. Blanc, J. Lange, and L.-Y. Tan. 2021. Provably efficient, succinct, and precise explanations. In *NeurIPS*. A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, (Eds.)

[19] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Lagniez, and P. Marquis. 2022. On the explanatory power of Boolean decision trees. *Data & Knowledge Engineering*, 142, (Nov. 2022), 102088. DOI: 10.1016/j.datak.2022.102088.

[20] Y. Izza, A. Ignatiev, and J. Marques-Silva. 2022. On tackling explanation redundancy in decision trees. *Journal of Artificial Intelligence Research*, 75, (Sept. 2022), 261–321. DOI: 10.1613/jair.1.13575.

[21] R. Gomes Mantovani, T. Horváth, A. L. D. Rossi, R. Cerri, S. Barbon Junior, J. Vanschoren, and A. C. P. L. F. d. Carvalho. 2024. Better trees: an empirical study on hyperparameter tuning of classification decision tree induction algorithms. *Data Mining and Knowledge Discovery*, 38, 3, (Jan. 2024), 1364–1416. DOI: 10.1007/s10618-024-01002-5.

[22] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J. Lagniez, and P. Marquis. 2021. On the computational intelligibility of boolean classifiers. In *KR*, 74–86.

[23] X. Huang, Y. Izza, A. Ignatiev, and J. Marques-Silva. 2021. On efficiently explaining graph-based classifiers. In *KR*, 356–367.

[24] J. Goldsmith, M. Hagen, and M. Mundhenk. 2005. Complexity of DNF and isomorphism of monotone formulas. In *MFCS*. Springer Berlin Heidelberg, 410–421. DOI: 10.1007/11549345_36.

[25] I. Wegener. 2000. *Branching Programs and Binary Decision Diagrams*. Society for Industrial and Applied Mathematics, (Jan. 2000). DOI: 10.1137/1.9780898719789.

[26] W. Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58, 301, 13–30.

[27] V. Guruswami and L. Trevisan. 2005. The complexity of making unique choices: approximating 1-in-k sat. In *Proceedings of the 8th international workshop on Approximation, Randomization and Combinatorial Optimization Problems, and Proceedings of the 9th international conference on Randamization and Computation: algorithms and techniques*, 99–110.

[28] M. Arenas, D. Báez, P. Barceló, J. Pérez, and B. Subercaseaux. 2021. Foundations of symbolic languages for model interpretability. In *NeurIPS*, 11690–11701.

[29] B. Subercaseaux, M. Arenas, and K. S. Meel. 2024. Probabilistic explanations for linear models. (2024). https://arxiv.org/abs/2501.00154 arXiv: 2501.00154 [cs.AI].

Received 27 October 2024; accepted 13 July 2025