Fixed-Parameter Tractability of Learning Small Decision Trees (full paper)

Anonymous author
Anonymous affiliation

Abstract

We consider the NP-hard problem of finding a smallest decision tree which represents a given partially defined Boolean formula. We establish fixed-parameter tractability of the problem with respect to the NLC-width of the instance. We formulate a dynamic programming procedure which utilizes the NLC-decomposition of the instance. For this to work, we establish a succinct representation of partial solutions, so that the space and time requirements of each dynamic programming step remain bounded in terms of the NLC-width.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms → Parameterized complexity and exact algorithms → Fixed parameter tractability

Keywords and phrases parameterized complexity, NLC-width, clique-width, rank-width, decision trees, partially defined Boolean formulas

1 Introduction

Decision trees have proved to be extremely useful tools for the describing, classifying, generalizing data [21, 26, 30]. In this paper, we consider decision trees for *classification instances (CIs)*, consisting of a finite set E of *examples* (also called *feature vectors*) over a finite set F of *features*. Each example $e \in E$ is a function $e : F \to \{0, 1\}$ which determines whether the feature f is true or false for e. Moreover, E is given as a partition $E^+ \uplus E^-$ into positive and negative examples. For instance, examples could represent medical patients and features diagnostic tests; a patient is positive or negative corresponding to whether they have been diagnosed with a certain disease or not. CIs are also called *partially* or *incompletely defined Boolean functions*, as we can consider the features as Boolean variables, and examples as truth assignments that evaluate to 0 (for positive examples) or 1 (for negative examples). CIs have been studied as a key concept for the logical analysis of data and in switching theory [4, 6, 5, 7, 8, 19, 24].

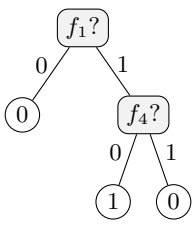
Because of their simplicity, decision trees are particularly attractive for providing interpretable models of the underlying CI, an aspect whose importance has been strongly emphasized over the recent years [10, 12, 16, 23, 25]. In this context, one prefers *small trees*, as they are easier to interpret and require fewer tests to make a classification. Small trees are also preferred in view of the parsimony principle (Occam's Razor) since small trees are expected to generalize better to new data [2]. However, finding a small decision tree, as formulated in the following decision problem, is NP-complete [18].

> Minimum Decision Tree Size (DTS): given a CI $E = E^+ \uplus E^-$ and an integer s, is there a decision tree with at most s nodes for E?

Given this complexity barrier, we propose a fixed-parameter algorithm for the problem, which exploits the input CI's hidden structure. We measure the presence of a hidden structure in a CI in terms of its *renamable treewidth*, which we define as follows. Borrowing from a similar notion for propositional CNF formulas [22], we define a *renaming* $r_X(E)$ of a CI E with respect to a set $X \subseteq F(E)$ to be the CI containing all the examples e_X for $e \in E$ with $e_X(f) = 1 - e(f)$ if $f \in X$ and $e_X(f) = e(f)$ if $f \notin X$. Clearly, the size of a smallest decision tree is the same for E and all its renamings. The *incidence treewidth* of a CI is the treewidth of the bipartite graph $G_I(E)$ whose vertices are the examples on one side and the features on the other, where an example e is adjacent with a feature f if and only if $e(f) = 1$. The *renamable treewidth* of E is now the minimum incidence treewidth over all possible renamings of E. Figure 1 shows a CI and a smallest decision tree for it, as well as the incidence graph of a renaming which minimizes the positive treewidth.

Key to our algorithm are new notions for succinctly representing decision trees that

E	f_1	f_2	f_3	f_4
$e_1 \in E^-$	0	0	1	0
$e_2 \in E^-$	0	0	1	1
$e_3 \in E^-$	0	1	1	0
$e_4 \in E^-$	1	1	0	0
$e_5 \in E^+$	1	0	0	1
$e_6 \in E^+$	1	0	1	1



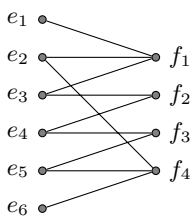


Figure 1 A CI $E = E^+ \uplus E^-$ with six examples and four features (left), a decision tree with 5 nodes that classifies E (middle), the incidence graph of the renaming $r_{\{f_2,f_4\}}(E)$, which has treewidth 2 (right).

correspond to subtrees of the incidence graph's tree decomposition. Based on that, we can carry out a dynamic programming (DP) procedure along the tree decomposition. However, before we can apply the DP algorithm, we first need to find a renaming of the given instance E which minimizes the incidence graph's treewidth. We achieve this by observing that the *rank-width* [29] of E's incidence graph is linear in its renamable treewidth. Thus, the rank-width is bounded by our parameter (renamable treewidth), and so we can (i) compute a rank decomposition through a known fpt-algorithm [17] and (ii) use a logical meta-theorem [15] to find a renaming $r_X(E)$ that minimizes the incidence treewidth.

While the DP approach using treewidth is quite well understood and can often be quite easily designed for problems on graphs (or more generally problems whose solutions can be represented in terms of the graph for which the tree decomposition is given), the same DP approach can become rather involved if applied to problems whose solutions have no or only minor resemblence to the graph for which one is given a tree decomposition. Probably the most prominent example for this is the celebrated result by Bodlaender [3], where he uses a DP approach on an approximate tree decomposition to compute the exact treewidth of a graph; here, the solutions are tree decompositions, which are complex structures that cannot easily be represented in terms of the graph. Other prominent examples include a DP approach to compute the exact treedepth [31] or clique-width [14] using an optimal tree decomposition. We face a similar problem, since solutions in our case are decision trees that do not bear any resemblence to the incidence graph for which we are given the tree decomposition. The main obstacle to overcome, therefore, is the design of the DP-records for our DP algorithm. That is, a record for a node b in a tree decomposition for the incidence graph of E needs to provide a compact representation of partial solutions, i.e. partial solutions in the sense that they represent the part of the solution for the whole instance E that corresponds to the sub-instance induced by all features and examples contained in the bags in the subtree of the tree decomposition rooted at the current node b. We overcome this obstacle in Section 4, where we also provide intuitive descriptions and motivation for the definition of the records (Subsection 4.1).

2 Preliminaries

2.1 Parameterized Complexity

We give some basic definitions of Parameterized Complexity and refer for a more in-depth treatment to other sources [9, 13]. Parameterized complexity considers problems in a two-dimensional setting, where a problem instance is a pair (I, k), where I is the main part and k is the parameter. A parameterized problem is *fixed-parameter tractable* if there exists a computable function f such that instances (I, k) can be solved in time $f(k)\|I\|^{O(1)}$.

2.2 Graphs, Treewidth, and Rank-width

We will assume that the reader is familiar with basic graph theory (see, e.g. [11, 1]). We consider (vertex and edge labelled) undirected graphs. Let $G = (V, E)$ be an undirected graph. We write $V(G) = V$ and $E(G) = E$ for the sets of vertices and edges of G, respectively. We denote an edge between $u \in V$ and $v \in V$ as $\{u, v\}$. For a set $V' \subseteq V$ of vertices we let $G[V']$ denote the graph induced by the vertices in V', i.e. $G[V']$ has vertex set V' and edge set $E \cap \{\, \{u, v\} \mid u, v \in V' \,\}$ and we let $G - V'$ denote the graph $G[V \setminus V']$. For a set $E' \subseteq E$ of edges we let denote $G - E'$ the graph with vertex set V and edge set $E \setminus E'$.

Node label control-width (NLC-width) is a graph parameter, defined as follows [33]: Let $k \in \mathbb{N}$ be a positive integer. A k-NLC-expression consists of a rooted subcubic tree such that:

1. Every leaf is labelled with a label $i \in \{1, \ldots, k\}$; this corresponds to a graph with a single vertex which has a label i.
2. Every non-leaf node with one child is labelled with a function $R : \{1, \ldots, k\} \to \{1, \ldots, k\}$. This corresponds to taking the labelled graph corresponding to the child and for every $i \in \{1, \ldots, k\}$, if a vertex has label i, changing this label to be $R(i)$ instead.
3. Every non-leaf node with two children is labelled with a $k \times k$ $\{0, 1\}$ matrix M. This corresponds to taking the disjoint union of the graphs corresponding to its children G_1 and G_2, and then for $i, j \in \{1, \ldots, k\}$, adding an edge from all vertices lablled i in G_1 to all vertices labelled j in G_2 if and only if $M_{i,j} = 1$.
4. The root node corresponds to the resulting labelled graph.

The NLC-width $NLC(G)$ of a graph G is the minimum k for which G has a k-NLC-expression. A k-NLC-expression is *nice* if every relabelling node has a function $R : \{1, \ldots, k\} \to \{1, \ldots, k\}$ such that for some $i, j \in \{1, \ldots, k\}$, $R(i) = j$ and $R(\ell) = \ell$ for all $\ell \in \{1, \ldots, k\} \setminus \{i\}$. Clearly, given a k-NLC-expression. a nice k-NLC-expression can be found in polynomial time.

Let x be a node in a k-NLC-expression tree of a graph G. We define $\chi(X)$ to be the set of vertices in $V(G)$ that correspond to leaves of the k-NLC-expression subtree rooted at x. By the definifion of a k-NLC-expression, if $s, t \in \chi(X)$ have the same label after applying node x and $u \in V(G) \setminus \chi(X)$, then s is adjacent to u if and only if t is. Furthermore, the k-NLC-expression subtree rooted at x is the graph induced by G on $\chi(X)$.

Computing the NLC-width of a graph is NP-hard [**?**].

However, it is sufficient to use the algorithm of Seymour and Oum [**?**], which returns a c-expression for some $c \leq 2^{3cw(G)+2} - 1$ in $O(n^9 \log n)$ time, or the later improvements of Oum [27] and Hliněný and Oum [**?**] that provide cubic-time algorithms which yield a c-expression for some $c \leq 8^{cw(G)} - 1$ and $c \leq 2^{cwG+1} - 1$, respectively.

For example, although computing the clique-width of a graph is known to be NP-hard in general [**?**],[1] the complexity of computing the clique-width is open even on very restricted graph classes, such as unit interval graphs (see [**?**] for some partial results). As another example, the reason that rank-width and clique-width are equivalent is because the inequalities $rw(G) \leq cw(G) \leq 2^{rw(G)+1} - 1$ hold for every graph G [**?**]. Of course, this suggests we cannot hope to compute a $cw(G)$-expression in polynomial time. However, it is sufficient to use the algorithm of Seymour and Oum [**?**], which returns a c-expression for some $c \leq 2^{3cw(G)+2} - 1$ in $O(n^9 \log n)$ time, or the later improvements of Oum [27] and Hliněný and Oum [**?**] that provide cubic-time algorithms which yield a c-expression for some $c \leq 8^{cw(G)} - 1$ and $c \leq 2^{cwG+1} - 1$, respectively.

For our tractability result for renamable treewidth, we will also require the notion of *rank-width*, denoted by $rw(G)$, which like treewidth, is another important complexity measure for graphs. In particular, we will need the following auxiliary results about rank-width.

▶ **Proposition 1** ([28]). *Let G be a graph. Then, $rw(G) \leq tw(G) + 1$.*

Let G be a bi-partite graph with partition (A, B), i.e. G is an undirected graph and $G[A]$ and $G[B]$ are independent sets. For a set $C \subseteq A \cup B$, we define the *local bi-complement of G w.r.t. C* as the (bi-partite) graph $\text{comp}(G, C)$ with partition (A, B) having an edge between

[1] It is also NP-hard to compute treewidth [**?**] and parameters equivalent to clique-width, such as NLC-width [**?**], rank-width (see [**?**, 27]) and boolean-width [**?**].

$a \in A$ and $b \in B$ if either $\{a, b\} \setminus C \neq \emptyset$ and $\{a, b\} \in E(G)$ or $\{a, b\} \subseteq C$ and $\{a, b\} \notin E(G)$. We will need the following proposition.

▶ **Proposition 2** ([20]). *Let G be a bipartite graph and $S \subseteq V(G)$. Then, $rw(\mathrm{comp}(G, S)) \leq 8rw(G)$.*

▶ **Proposition 3** ([17]). *Let $\omega \in \mathbb{N}$ and $n \geq 2$. For an n-vertex graph G, we can output a rank-decomposition of width at most ω or confirm that the rank-width of G is larger than ω in time $f(\omega) \cdot n^3$, where f is a computable function.*

2.3 Classification Problems

An *example* e is a function $e : \mathit{feat}(e) \to \{0, 1\}$ defined on a finite set $\mathit{feat}(e)$ of *features*. For a set E of examples, we put $\mathit{feat}(E) = \bigcup_{e \in E} \mathit{feat}(e)$. We say that two examples e_1, e_2 *agree* on a feature f if $f \in \mathit{feat}(e_1)$, $f \in \mathit{feat}(e_2)$ and $e_1(f) = e_2(f)$. If $f \in \mathit{feat}(e_1)$, $f \in \mathit{feat}(e_2)$ but $e_1(f) \neq e_2(f)$, we say that the examples *disagree on* f.

A *classification instance (CI)* (also called a *partially defined Boolean function* [19]) $E = E^+ \uplus E^-$ is the disjoint union of two sets of examples, where for all $e_1, e_2 \in E$ we have $\mathit{feat}(e_1) = \mathit{feat}(e_2)$. The examples in E^+ are said to be *positive*; the examples in E^- are said to be *negative*. A set X of examples is *uniform* if $X \subseteq E^+$ or $X \subseteq E^-$; otherwise X is *non-uniform*.

Given a CI E, a subset $F \subseteq \mathit{feat}(E)$ is a *support set* of E if any two examples $e_1 \in E^+$ and $e_2 \in E^-$ disagree in at least one feature of F. Finding a smallest support set, denoted by $\mathrm{MSS}(E)$, for a classification instance E is an NP-hard task [19, Theorem 12.2].

We define the *incidence graph* of E, denoted by $G_I(E)$, as the bipartite graph with partition $(E, \mathit{feat}(E))$ having an edge between an example $e \in E$ and a feature $f \in \mathit{feat}(e)$ if $f(e) = 1$.

Finally, we define a *renaming* $r_X(E)$ of a CI E with respect to a set $X \subseteq F(E)$ to be the CI containing all the examples e_X for $e \in E$ with $e_X(f) = 1 - e(f)$ if $f \in X$ and $e_X(f) = e(f)$ if $f \notin X$.

▶ **Observation 4.** *Let E be a CI and $X \subseteq \mathit{feat}(E)$. Then, E has a DT of size s if and only if $r_X(E)$ does.*

Proof. Let T be a DT for E and let T' be obtained from T by switching the left and right child of every node t of T with $\mathit{feat}(t) \in X$. Then T' is DT for $r_X(E)$ having the same size as T. ◀

2.4 Decision Trees

A *decision tree* (DT) (or *classification tree*) is a rooted tree T with vertex set $V(T)$ and arc set $A(T)$, where each non-leaf node (called a *test*) $v \in V(T)$ is labelled with a feature $\mathit{feat}(v)$, each non-leaf node v has exactly two out-going arcs, a *left arc* and a *right arc*, and each leaf is either a *positive* or a *negative* leaf. We write $\mathit{feat}(T) = \{\, v \in V(T) \mid \mathit{feat}(v) \,\}$.

Consider a CI E and a decision tree T with $\mathit{feat}(T) \subseteq \mathit{feat}(E)$. For each node v of T we define $E_T(v)$ as the set of all examples $e \in E$ such that for each left (right, respectively) arc (u, v) on the unique path from the root of T to v we have $e(\mathit{feat}(v)) = 0$ ($e(\mathit{feat}(v)) = 1$, respectively). T *correctly classifies* an example $e \in E$ if e is a positive (negative) example and $e \in E_T(v)$ for a positive (negative) leaf. We say that T *classifies* E (or simply that T is a DT for E) if T correctly classifies every example $e \in E$. See Figure 1 for an illustration of a CI, its incidence graph, and a DT that classifies E. The size of T is its number of nodes, i.e. $|V(T)|$. We consider the following problem.

Minimum Decision Tree Size (DTS)	
Input:	A classification instance E and an integer s.
Question:	Is there a decision tree of size at most s for E?

We now give some simple auxiliary lemmas that are required by our algorithm.

▶ **Lemma 5.** *Let A be a set of features of size a. Then the number of DTs of size at most s that use only features in A is at most a^{2s+1} and those can be enumerated in $\mathcal{O}(a^{2s+1})$ time.*

Proof. We start by counting the number of trees T with n nodes that can potentially underlie a DT with n nodes. Note that there is one-to-one correspondence between trees T that underlie a DT with n nodes and unlabelled rooted ordered binary trees with n nodes (where ordered refers to an ordering of the at most 2 child nodes). Since it is known that the number of unlabelled rooted ordered binary trees with n nodes is equal to the n-th Catalan number C_n and that those trees can be enumerated in $\mathcal{O}(C_n)$ time [32], we already obtain that we can enumerate all of the at most C_n possible trees T underlying a DT of size n in $\mathcal{O}(C_n)$ time. Therefore, there are at most sC_s possible trees of size at most s that can underlie a DT with at most s nodes and those can be enumerated in $\mathcal{O}(sC_s)$ time. It now remains to bound the number of possible feature assignments $\mathit{feat}(f)$ for these trees as well as the number of possibilities for the leave nodes that can be either labelled positive or negative. Since we can assume that $a \geq 2$, we obtain that the number of possible feature assignments (and labellings of leaf-nodes) of a tree T with n nodes is at most a^n. Taking everything together, we obtain that there are at most $sC_s a^s \leq s4^s a^s \leq a^{2s+1}$ many DTs of size at most s using only features in A and those can be enumerated in $\mathcal{O}(a^{2s+1})$ time. ◀

▶ **Lemma 6.** *Let A be a set of features of size a. There are at most $a^{2^{a+1}+3}$ inclusion-wise minimal DTs using only features in A and these can be enumerated in $\mathcal{O}(a^{2^{a+1}+3})$ time.*

Proof. Note that an inclusion-wise minimal DT T that uses only features in A has at most $2^a + 1$ nodes; this is because every feature appears at most once on every path T. Therefore, we obtain from Lemma 5 that the number of choices for T is at most $a^{2(2^a+1)+1} = a^{2^{a+1}+3}$ ◀

▶ **Lemma 7.** *Let E be a CI. Then one can decide whether E has a DT and if so output a DT of minimum size for E in time $\mathcal{O}((2^{|E|})^{4|E|-1})$.*

Proof. Note first that $|\mathit{feat}(E)| \leq 2^{|E|}$ since we can assume that E does not contain two equivalent features. Moreover, E has a DT if and only if $\mathit{feat}(E)$ is a support set, which can be checked in time $\mathcal{O}(|E|^2|\mathit{feat}(E)|)$ by checking, for every pair of positive and negative examples in E, whether there is a feature that distinguishes them. If this is not the case, we output **NO**, so assume that E has a DT. Note that any inclusion-wise minimal DT for E has at most $|E|$ leaves and therefore size at most $2|E| - 1$. We can therefore employ Lemma 5 to enumerate all inclusion-wise minimal potential DTs for E in time $\mathcal{O}((2^{|E|})^{2(2|E|-1)+1}) \in \mathcal{O}((2^{|E|})^{4|E|-1})$. For every such tree we then check whether it is indeed a DT for E and return a DT for E of minimum size found during this process. ◀

3 Introducing and Computing Renamable Treewidth

In this section, we are introducing our novel parameter called renamable treewidth. Let E be a CI. Then we could define the treewidth of E as the treewidth of the incidence graph $G_I(E)$. However, since $G_I(E)$ has an edge between every example $e \in E$ and feature $f \in \mathit{feat}(E)$ such that $e(f) = 1$, the treewidth of $G_I(E)$ depends very much on the concrete

values that the examples take for different features. For instance, consider the CI E with n examples $e_1, \ldots, e_n$ and n features $f_1, \ldots, f_n$ such that $e_i(f_j) = 0$ if and only if $i = j$, i.e. the matrix with features and examples is the complement of the identity matrix. Then the $G_I(E)$ is a bi-clique minus a perfect matching and therefore its treewidth is arbitrary large (i.e. to be exact it is easy to verify that its treewidth is $n - 1$). However, if we rename the values of all features, i.e. if we consider the CI $r_{feat(E)}(E)$, then its treewidth is 1 (since $G_I(r_{feat(E)}(E))$ is a perfect matching) even though E and $r_{feat(E)}(E)$ lead to equivalent instances of DTS (Observation 4). Therefore, it would be much better to consider the minimum treewidth of $G_I(r_X(E))$ over every possible choice for $X \subseteq feat(E)$. This leads us directly to our new notion *renamable treewidth* of a CI E, denoted by $rtw(E)$, which we define as $rtw(E) = \min_{X \subseteq feat(E)} tw(G_I(r_X(E)))$.

Unfortunately, unlike treewidth, where it is known that an optimal tree decomposition can be computed in fpt-time, this is not known for renamable treewidth. However, it is required for our algorithm to work. The remainder of this section is therefore devoted to a proof of the following theorem.

▶ **Theorem 8.** *Let E be a CI and ω and integer. Then, there is an algorithm that in time $o(\omega) \cdot n^3$ decides whether $rtw(E) \leq \omega$ and if so computes a subset $X \subseteq feat(E)$ such that $tw(G_I(r_X(E))) \leq \omega$, for some computable function o.*

Proof. We start by showing that $rw(G_I(E)) \leq 8rtw(E) + 1$. Let $X \subseteq feat(E)$ such that $tw(G_I(r_X(E))) = rtw(E)$. Then, $G_I(r_X(E))$ is the local bi-complement of $G_I(E)$ w.r.t. the set $C = X \cup E$. Therefore, we obtain from Proposition 2 that $rw(G_I(E)) \leq 8rw(G_I(r_X(E)))$, which because of Proposition 1 implies that $rw(G_I(E)) \leq 8rtw(E) + 1$, as required.

To compute X, we will make use of Proposition **??**. That is, we will construct an MSO formula $\varphi(X)$ such that:

(C1) $G_I(E) \models \varphi(X)$ if and only if X is a set of features of E such that $tw(G_I(r_X(E))) \leq \omega$,

(C2) $\varphi(X)$ can be constructed in time $g(\omega)$ and $|\varphi(X)| \leq g(\omega)$ for some computable function g.

Once we have $\varphi(X)$, we can compute X as follows:

- First we use Proposition 3 to decide in time $f(rw(G_I(E))) \leq f(8\omega + 1) \cdot n^3$ whether $rw(G_I(E)) \leq 8\omega + 1$. If this is not the case, then we know that $rtw(E) > \omega$ and we can reject.
- Otherwise, we construct the MSO formula $\varphi(X)$ in time $g(\omega)$ and use Proposition **??** to decide in time $f(rw(G_I(E)) + g(\omega)) \cdot n^3 \leq f(8\omega + 1 + g(\omega)) \cdot n^3$ whether there is a set X such that $tw(G_I(r_X(E))) \leq \omega$. If that is the case we return X and if not we correctly reject.

Therefore, it only remains to show how to construct $\varphi(X)$. Let Φ_ω be the MSO formula given by Lemma **??**. Moreover, let Φ'_ω be the MSO formula obtained from Φ_ω after replacing every occurrence of an atomic formula Exy with the formula $\varphi_E(x, y)$ given by:

$$(\neg Xx \wedge \neg Xy \rightarrow Exy) \wedge ((Xx \vee Xy) \wedge (\neg Ax \vee \neg Ay) \rightarrow \neg Exy) \wedge ((Ax \wedge Ay) \rightarrow \neg Ax);$$

here Ax is true if x is a feature vertex of $G_I(E)$. Note that the formula $\varphi_E(x, y)$ is true if and only if x is adjacent to y in $G_I(r_X(E))$. Then, $\varphi(X)$ is the formula $\exists X(\forall x Xx \rightarrow Ax) \wedge \Phi'_\omega$. It is straightforward to verify that $\varphi(X)$ satisfies (C1)–(C2). ◀

4 An FPT-Algorithm for NLC-width

In this section, we present our main result, i.e. we will show that DTS is fixed-parameter tractable parameterized by NLC-width.

▶ **Theorem 9.** DTS *is fixed-parameter tractable parameterized by NLC-width.*

▶ **Theorem 10.** *Let E be a CI, let (T, χ) be an NLC-expression decomposition of width k for $G_I(E)$, and let s be an integer. Then, deciding whether E has a DT of size at most s is fixed-parameter tractable parameterized by k.*

In principle, we will use a dynamic programming algorithm along the NLC-expression (T, χ) of $G_I(E)$ that computes a set of records for every node b of B in a bottom-up manner. Each record will represent an equivalence class of solutions (DTs) for the whole instance restricted to the examples and features represented by the current subtree rooted in b, i.e. the examples and features contained in $\chi(B_b)$. Before we continue with the formal notions and definitions required to define the records, we want to illustrate the main ideas and motivations. In what follows let (B, χ) be an NLC-width expression of $G_I(E)$ for the CI E of width k. For $b \in V(B)$, we write $\mathit{feat}(b)$ and $\mathit{exam}(b)$ for the sets $\chi(b) \cap \mathit{feat}(E)$ and $\chi(b) \cap E$, respectively. Similarly, we write $\mathit{feat}(B_b)$ and $\mathit{exam}(B_b)$ for the sets $\chi(B_b) \cap \mathit{feat}(E)$ and $\chi(B_b) \cap E$, respectively.

4.1 Description of the Main Ideas Behind the Algorithm

Consider a node b of B. To simplify the presentation, we will sometime refer to the features and examples in $\chi(B_b) \setminus \chi(b)$ as *forgotten* features and examples and we refer to the features and examples in $(\mathit{feat}(E) \cup E) \setminus \chi(B_b)$ as *future* features and examples. We start with some simple observations that follow immediately from the properties of tree decompositions.

▶ **Observation 11.** *(1) $e(f) = 0$ for every forgotten example $e \in \mathit{exam}(B_b) \setminus \mathit{exam}(b)$ and future feature $f \in \mathit{feat}(E) \setminus \mathit{feat}(B_b)$,*
(2) $e(f) = 0$ for every future example $e \in E \setminus \mathit{exam}(B_b)$ and forgotten feature $f \in \mathit{feat}(B_b) \setminus \mathit{feat}(b)$;

Proof. Towards showing (1), let e be an example in $\mathit{exam}(B_b) \setminus \mathit{exam}(b)$ and let f be a feature in $\mathit{feat}(E) \setminus \mathit{feat}(B_b)$. We claim that because (T, χ) is a tree decomposition of $G_I(E)$, the graph $G_I(E)$ cannot contain an edge between e and f, which implies that $e(f) = 0$. Suppose for a contradiction that this is not the case, i.e. $\{e, f\} \in E(G_I(E))$. Then, because of property (T1) of a tree decomposition, there must exist a node b' such that $e, f \in \chi(b')$. But then, if $b' \in V(B_b)$ we obtain that $f \notin \chi(b')$. Similarly, if $b' \in V(B \setminus B_b)$, we obtain that $e \notin \chi(b')$ since otherwise e would violate property (T2) of a tree decomposition. This completes the proof for (1); the proof for (2) is analogous. ◀

Informally, Observation 11 shows that forgotten examples cannot be distinguished by future features and future examples cannot be distinguished by forgotten features.

Consider a DT T for E and a node b of B. For a set W containing features and examples from E, we denote by $E[W]$ the sub-instance of E induced by the features and examples in W. Our aim is to obtain a compact representation (represented by records) of the partial solution for the sub-instance $E[\chi(B_b)]$ of E induced by the features and examples in $\chi(B_b)$ represented by T.

Intuitively, such a compact representation has to (1) represent a partial solution (DT) for the examples in $\mathit{exam}(B_b)$ and (2) retain sufficient information about the structure of T in order to decide whether it can be extended to a DT that also classifies the examples in $E \setminus \mathit{exam}(B_b)$.

For illustration purposes let us first consider the simplified case that $\mathit{exam}(b) = \emptyset$. Because of Observation 11 (1), this implies that every forgotten example goes to the left child of

any node t in T that is assigned a future feature. Therefore, under the assumption that $exam(b) = \emptyset$ the DT T' obtained from T after:

- removing the subtree T_r of T for every right child r of a node t of T with $feat(t) \in feat(E) \setminus feat(B_b)$ and replacing t with an edge from its parent in T to its left child in T

is a DT for $E[\chi(B_b)]$. Note that this means that under the rather strong assumption that $exam(b) = \emptyset$, the part of T that takes care of the sub-instance $E[\chi(B_b)]$ is itself a DT using only features in $feat(B_b)$; we will see later that unfortunately this is no longer the case if $exam(b) \neq \emptyset$. Note that even though T' is a DT for $E[B_b]$, it does not yet constitute a compact representation, since the number of features it uses in $feat(B_b) \setminus feat(b)$ is potentially unbounded. However, we obtain from Observation 11 (2) that every future example will end up in the left child of every node t of T' that is assigned a forgotten feature. This means that to decide whether T' can be extended to a DT for the whole instance, the nodes that are assigned forgotten features are not important. In fact, the only nodes in T' that can be important for the classification of future examples are the nodes that are assigned features in $feat(b)$. That is, it is sufficient to remember the DT T'' obtained from T' after:

- removing the subtree T_r of T' for every right child r of a node t of T' with $feat(t) \in feat(B_b) \setminus feat(b)$ and replacing t with an edge from its parent in T' to its left child in T'

Since the number of possible DT T'' is clearly bounded in terms of the number of features in $feat(b)$ (and therefore in terms of the treewidth of $G_I(E)$), this would already give us the compact representation that we are looking for. However, this only works in the case that $exam(b) = \emptyset$, which is clearly not the case in general.

So let us now consider the general case with $exam(b) \neq \emptyset$. The first difference now is that the part of T that takes care of the sub-instance $E[\chi(B_b)]$ is no longer a DT that only uses features in $feat(B_b)$. In fact, it could even be the case that $E[\chi(B_b)]$ does not have a DT, because there could exist examples in $exam(b)$ that can only be distinguished using the features in $feat(E) \setminus feat(B_b)$. This means that we have to allow our partial solution for $E[\chi(B_b)]$ to use future features. Fortunately, we do not need to know which exact future feature is used by our partial solution but it suffices to know that a future feature is used and how it behaves w.r.t. the examples in $exam(b)$; this is because Observation 11 (1) implies that a future feature is used in a partial solution only for the purpose of distinguishing examples in $exam(b)$. Moreover, because every forgotten example ends up in the left child of any node t of T that uses a future feature, we only need to remember the left child for those nodes. Also, we only need to remember occurrences of those nodes (using future features) if at least one example in $exam(b)$ ends up to in the right child of such a node; otherwise the node has no influence on the classification of examples in $exam(B_b)$. Finally, we cannot simply forget nodes that use forgotten features (as we could in the case that $exam(b) = 0$). This is because we need to know exactly where the examples in $exam(b)$ end up at. For instance, if such an example in $exam(b)$ ends up in the right child of a node using a future feature, we need to know that this is the case because this means that the example has to be classified in this place at a later stage of the algorithm. Nevertheless, we do not need to remember all occurrences of nodes using forgotten features, but only those for which there is at least one example in $exam(b)$ that ends up in the right child of the node. Similarly, we do not need to remember the exact forgotten feature that is used but only how it behaves towards the examples in $exam(b)$. In summary, we only need to remember the full information about the nodes of T that use a feature in $feat(b)$. For all other nodes, i.e. nodes that use either forgotten or future features, we only need to remember such a node, if at least one example in $exam(b)$ ends up in its right child. Moreover, even if this is the case, we only need to remember the following for such nodes:

- whether it uses a future or a forgotten feature and
- how it behaves w.r.t. the examples in $exam(b)$.

With these ideas in mind, we are now ready to provide a formal definition of the compact representation of the part of T that takes care of the sub-instance $E[\chi(B_b)]$.

4.2 Formal Definition of Records and Preliminary Results

Consider a node $b \in V(B)$. We start by introducing novel features that will model the behaviour of future features on the examples in $exam(b)$. For a subset of labels $B \in \{1, \dots, k\}$, we define the feature template f_B by setting $e(f_B) = 1$ if and only if $lab(e) \in B$ and $e(f_B) = 0$ otherwise.

Notes: In our DT, we have leaves labelled $+$ or $-$. We have can have actual features with labels in $\{1, \dots, k\}$ (because of the NLC-expression, we can choose any representative, and it will work the same as everything else). We can also have feature templates: there are at most 2^k of these, depending on whether they will send each label in $\{1, \dots, k\}$ to the right or to the left. Note: whenever we have a relabelling node, these templates will need to be mangled. Note: when we do a join node, we need to check which label actual features actually exist in each subtree.

Reduction algorithm: Notation: for $X \in \{1, \dots, k\}$, let $\overline{X} = \{1, \dots, k\} \setminus X$.

Suppose we have a DT such that some actual feature f occurs twice on a path from the root to the leaves, say f_1 is the instance closer to the root and f_2 is the other instance. If f_2 is in the left (resp. right) subtree of f_1, we delete f_2's right (resp. left) subtree. We then contract away the node f_2 (i.e. replace the subtree rooted at f_2 by the subtree rooted at f_2's unique remaining child. Now suppose we have a feature template B in our decision tree. Let $B_1, \dots, B_\ell$ be the sequence of feature templates on the path from the root to B in order (not including B, so B_ℓ). Let $B'_i = B_i$ if B is in the right sub-tree of B_i and let $B'_i = \overline{B_i}$ otherwise. If $\overline{B} \subseteq B'_1 \cup \dots \cup B'_\ell$, then we delete the subtree rooted at the left child of B and contract B away. If $B \subseteq \overline{B'_1} \cup \dots \cup \overline{B'_\ell}$, then we delete the subtree rooted at the right child of B and contract B away. If this procedure has been applied to a record exhaustively, we say that the record is *reduced*.

The *score* of a record at a node x is the number of actual features used in the DT for the nodes in $\chi(x)$ (disregarding any feature templates). A record is *complete* if it contains no feature templates.

▶ **Lemma 12.** *If there are k actual feature labels and ℓ future templates labels, then every reduced DT has height at most $k + \ell$. Futhermore, every path from the root to the leaves contains at most k actual feature labels and at most $\ell - 1$ future templates.*

Proof. Consider a path P of maximum length from the root to the leaves in a reduced DT. No actual feature appears more than once on this path, so the number of actual feature nodes on this path is at most k. Each feature template must divide the set of colours into two non-empty parts, so there can be at most $\ell - 1$ feature template nodes on this path. The path ends with a leaf node, so this gives a total of $k + \ell - 1 + 1 = k + \ell$ nodes, as required. ◀

▶ **Lemma 13.** *If there are k actual feature labels and ℓ future template labels, then there are at most TODO reduced DTs. Furthermore, these can be enumerated in $f(k + \ell)$-time for some computable function f.*

Proof. There are k actual feature labels and 2^ℓ possible future template labels and leaves can be either positive or negative. By Lemma 12, the tree has height at most $k + \ell$. Each

node of the decision tree could be a feature label, a future template, or a leaf (in which case it has no children), there are at most $k + 2^\ell + 2$ possible labels for each node of the tree. Since the height is at most $k + \ell$, there are at most $2^{k+\ell}$ nodes in the tree, so there are at most $(k + 2^\ell + 2)^{2^{k+\ell}}$ possible decision trees. ◀

Leaf nodes: Go though every possible reduced DT and set the score appropriately. If the leaf is an example node, make sure it gets sent to the correct $+$ or $-$. If the leaf is a feature node, set the score appropriately (no checkin necessary).

For join nodes: Enumerate every reduced DT T with feature labels $1, \ldots, k, 1', \ldots, k'$ and future feature labels $1, \ldots, k$. For each such DT, create the DT T_L as follows: if a feature has label $i' \in \{1', \ldots, k'\}$, replace it with the future feature B such that the join node joins the feature i' to precisely the labels in B. Then apply the reduction algorithm. Symmetrically, create the DT T_R. Finally, create the DT T_J as follows: replace every i' in the description of T with i, then reduce. If the score of T_J is not defined yet, or is larger than the sum of the scores of T_L and T_R, then set the score of T_J to the sum of the scores of T_L and T_R.

For relabel nodes: Relabel the description of the DT T. Note: if two labels are merged, and the future features disagree, then we ignore this record. There will be a set of still-used labels. Reduce over these labels only. Next, for each possible DT, check whether it reduces to the same thing, and set the score appropriately if so.

For root node: find minimum score over all reduced DTs with no future features.

Need to show: If there is a DT, then it crops up in our decomposition. If we get a score for some example, we can reconstruct a DT with that score.

At a join and connect node: We will have 2^{2k} node templates again, k actual features corresponing to the left child and k actual features corresponding to the right child. For each of these, we take our prototype record D, we create the left record D_L by replacing the k actual features from the right child with the relevant feature template and reducing; we create the right record D_R by replacing the k actual features from the left child with the relevant feature template and reducing. We then store the answer by replacing the prototype record with its reduction. As discussed with Sebastian, possibly don't even need to project when you relabel.

We denote by $\mathrm{TF}(b)$ the set of all features f_B for every B with $B \subseteq \mathit{exam}(b)$ and $B \neq \emptyset$ for which there is a feature $f \in \mathit{feat}(E)$ that agrees agrees with f_B on all examples in $\mathit{exam}(b)$. Moreover, we denote by $\mathrm{TF}_F(b)$ the set of all features f in $\mathrm{TF}(b)$ such that there is a feature in $\mathit{feat}(E) \setminus \mathit{feat}(B_b)$ that agrees with f on all examples in $\mathit{exam}(b)$. Similarly, we denote by $\mathrm{TF}_P(b)$ the set of all features f in $\mathrm{TF}(b)$ such that there is a feature in $\mathit{feat}(B_b) \setminus \mathit{feat}(b)$ that agrees with f on all examples in $\mathit{exam}(b)$.

We now introduce the notion of a DT-template that will allow use to employ DTs, where not all nodes have a right child. A *DT-template* for E is a DT T, where some nodes (which we call *template nodes*) are allowed not to have a right child and every example e of E that ends up in a leaf node is correctly classified; informally, this means that every example that would go to the right child of a template node is assumed to be automatically classified.

A *(b)-DT pattern T for $E[\chi(B_b)]$* is a DT-template for $E[\chi(B_b)]$ that can additionally use the features in $\mathrm{TF}_F(b)$ such that:

- a node $t \in V(T)$ is a template node of T if and only if $\mathit{feat}(t) \in \mathrm{TF}_F(b)$ and t does not occur in a subtree T_c for some right child c of a node t' with $\mathit{feat}(t') \in \mathit{feat}(B_b) \setminus \mathit{feat}(b)$.

The *skeleton* of an (b)-DT pattern T for $E[\chi(B_b)]$ is the pair (T_s, M), where:

- T_s is obtained from T after the application of the following steps:

(S1) For every node t of T with $feat(t) \notin feat(b)$ remove the subtree rooted at the right child of t in T. Let T' be the tree obtained from T after applying this procedure exhaustively.

(S2) Let P be an inclusion-wise maximal path in T' such that every inner node p of P is a non-leaf node and $feat(p) \in (feat(B_b) \setminus feat(b))$. Let S be the sub-sequence of all nodes p in P such that at least one example in $exam(b)$ goes to the right child of p, i.e. there is an example in $e \in exam(b) \cap E_T(p)$ with $e(feat(p)) = 1$. Moreover, let S' be obtained from S by replacing the feature $f = feat(s)$ assigned to every node s in S with the unique feature in $\mathrm{TF}_P(b)$ that agrees with f for all examples in $exam(b)$. We now replace P in T' with the sequence S', i.e. we replace the sub-path of P containing all non-leaf nodes p with $feat(p) \in (feat(B_b) \setminus feat(b))$ with a path defined by the sequence S'. Then T_s is the tree obtained from T' after exhaustively applying this procedure.

- M is the set of all nodes of T_s that have been added during (S2).

We call any pair (T_s, M) such that:

- T_s is a (b)-DT pattern using only features in $feat(b) \cup \mathrm{TF}(b)$,
- M is a subset of all nodes t in T_s with $feat(t) \in \mathrm{TF}_P(b)$ and
- $feat(t) \in \mathrm{TF}_F(b)$ for every node t of T_s with $feat(t) \in \mathrm{TF}(b)$ and $t \notin M$

a *skeleton* (b)*-DT pattern*. We say that (T_s, M) is *inclusion-wise minimal* if for every node t of T_s with $feat(t) \in \mathrm{TF}(b)$ there is at least one example $e \in E_{T_s}(t) \cap exam(b)$ with $e(feat(t)) = 1$.

We are now ready to define the records used by our dynamic programming algorithm.

A *record* for b is a pair $(\mathcal{S}, s)$, where:

- $\mathcal{S} = (S, M)$ is an inclusion-wise minimal skeleton (b)-DT pattern and
- s is an integer.

We say that a record $(\mathcal{S}, s)$ is *valid* for b if s is the smallest integer such that the CI $E[\chi(B_b)]$ has a (b)-DT pattern of size s, whose skeleton is $\mathcal{S}$. We denote by $\mathcal{R}(b)$ the set of all valid records for b.

We now provide some useful basic results about skeleton (b)-DT patterns. The first one concerns the size and number of features used by a skeleton (b)-DT pattern.

▶ **Lemma 14.** *Let (S, M) be an inclusion-wise minimal skeleton (b)-DT pattern. Then:*
(1) *S has size at most $2^\omega + 1$,*
(2) *$|feat(S)| \leq \omega$,*
(3) *the number of nodes t in S with $feat(t) \in \mathrm{TF}(b)$ is at most ω.*

Proof. We start with showing that S has at most $|exam(b)|$ nodes t with $feat(t) \in \mathrm{TF}(b)$. Consider a node t of S with $feat(t) \in \mathrm{TF}(b)$. Because, S is inclusion-wise minimal there must exist an example $e \in exam(b) \cap E_S(t)$ such that $e(feat(t)) = 1$. Therefore, S can contain at most $|exam(b)| \leq \omega$ nodes using features in $\mathrm{TF}(b)$, which shows (3).

Since the nodes of S use only features from $feat(b) \cup \mathrm{TF}(b)$ and we have just shown that the nodes in S use at most $|exam(b)|$ features in $\mathrm{TF}(b)$, we obtain that the nodes in S use at most $|feat(b)| + |exam(b)| \leq \omega$ features, showing (2).

Clearly, the size of S is maximum if it only uses features in $feat(b)$ and $|feat(b)| = \omega$. In this case the size of S is equal to the maximum size of a DT that uses only ω features. Because every feature can appear at most once on every path of S, we obtain that the maximum size of S is at most $2^\omega + 1$. ◀

The next lemma shows that the number of skeleton (b)-DT pattern and therefore the number of records for b can be upper bounded by a function depending only on ω.

▶ **Lemma 15.** *There are at most $\mathcal{O}(\omega^{3\cdot 2^\omega})$ inclusion-wise minimal skeleton (b)-DT patterns and these can be enumerated in time $\mathcal{O}(\omega^{3\cdot 2^\omega})$.*

Proof. Let (S, M) be an inclusion-wise minimal skeleton (b)-DT pattern for $E[\chi(B_b)]$. Then, $\mathit{feat}(S) \subseteq \mathit{feat}(b) \cup \mathrm{TF}(b)$. Moreover, because of Lemma 14 $|\mathit{feat}(S)| \leq \omega$ and therefore there are at most $(|\mathit{feat}(b) \cup \mathrm{TF}(b)|)^\omega \leq (\omega + 2^\omega)^\omega \leq (2^{\omega+1})^\omega$ possible sets A (each of size at most ω) of features such that $\mathit{feat}(S) \subseteq A$. Furthermore, we obtain from Lemma 6 that for every such set A, there are at most $(|A|^{2\cdot 2^{|A|}+1}) \leq \omega^{2^{\omega+1}+1}$ inclusion-wise minimal DTs using only the features in A. Since every inclusion-wise minimal skeleton (b)-DT pattern for $E[\chi(B_b)]$ can be obtained from at least one of those DTs, we obtain that there are at most $(2^{\omega+1})^\omega \omega^{2^{\omega+1}+1}$ choices for S. Finally, since there are at most 2^ω choices for M, we obtain that there are at most $2^\omega (2^{\omega+1})^\omega \omega^{2^{\omega+1}+1} \leq \omega^{3\cdot 2^\omega}$ choices for (S, M). ◀

4.3 Proof of the Main Result

It remains to show that the set $\mathcal{R}(b)$ of valid records can be computed for every node b of B in an efficient manner. Since we employ a bottom-up dynamic programming approach along the nodes of the nice tree decomposition (B, χ), it suffices to show how to compute the set of valid records for every of the four types of nodes of a nice tree decomposition; assuming that the set of valid records has already been computed for every child of b in B. Since introduce nodes (forgot nodes) can introduce (forget) either features or examples, we actually distinguish between six types of nodes.

▶ **Lemma 16 (leaf node).** *Let $b \in V(B)$ be a leaf node. Then, $\mathcal{R}(b)$ can be computed in $\mathcal{O}(1)$ time.*

Proof. Since $\chi(B_b) = \emptyset$, $\mathcal{R}(b)$ only contains the two records $(\mathcal{S}_0, 1)$ and $(\mathcal{S}_0, 1)$, where $\mathcal{S}_i = (S_i, \emptyset)$ and S_i is the DT consisting only of a negative leaf (if $i = 0$) and only of a positive leaf (if $i = 1$). ◀

▶ **Lemma 17 (feature introduce node).** *Let $b \in V(B)$ be an introduce node with child b_0 such that $\chi(b) \setminus \chi(b_0) = \{f\}$ for some feature $f \in \mathit{feat}(E)$. Then, $\mathcal{R}(b)$ can be computed in $\mathcal{O}(\omega^{15\cdot 2^\omega})$ time.*

Proof. Informally, $\mathcal{R}(b)$ contains a record $(\mathcal{S}, s)$ for every skeleton (b)-pattern such that: (1) the skeleton (b_0)-DT pattern obtained from $\mathcal{S}$ after removing the right child of every node t of S with $\mathit{feat}(t) = f$ has a record in $\mathcal{R}(b_0)$ and (2) for every node t of S with $\mathit{feat}(t) = f$, there is a (b_0)-DT pattern T^t for the examples in $E_S(t) \cap \{\, e(f) = 1 \mid e \in E \,\}$ (note that deciding this can be done efficiently because $E_S(t) \cap \{\, e(f) = 1 \mid e \in E \,\} \subseteq \mathit{exam}(b)$ and hence only at most ω examples need to be classified by T^t).

More formally, let $\mathcal{S} = (S, M)$ be an inclusion-wise minimal skeleton (b)-DT pattern. We say that $\mathcal{S}_0 = (S_0, M)$ is the (b_0)-*projection* of $\mathcal{S}$, if S_0 is obtained from S after exhaustively applying the following procedure: For every node t of S with $\mathit{feat}(t) = f$, remove the subtree rooted at the right child of t from S. Moreover, either:

(S1) if there is an example $e \in E_S(t) \cap \mathit{exam}(b)$ such that $e(f) = 1$, then set $\mathit{feat}(t)$ to the unique feature f_0 in $\mathrm{TF}_F(b_0)$ that agrees with f on the examples in $\mathit{exam}(b)$ (note that $f_0 \in \mathrm{TF}_F(b_0)$ because $f \in \mathit{feat}(E) \setminus \mathit{feat}(B_{b_0})$),

(S2) otherwise remove t from S and replace it by an edge from its parent to its left child in S.

We say that the (b_0)-projection has *difference* d if d is equal to the number of nodes removed from S in step (S2). Note that $\mathcal{S}_0$ is a skeleton (b_0)-DT pattern and if $\mathcal{S}$ has no node with $\mathit{feat}(t) = f$, then $\mathcal{S} = \mathcal{S}_0$.

Then, $\mathcal{R}(b)$ consists of all records $(\mathcal{S}, s)$, where $s = s'_0 + \sum_{t \in V(S) \wedge \mathit{feat}(t)=f} s_t$ such that $\mathcal{S} = (S, M)$ is an inclusion-wise minimal skeleton (b)-DT pattern satisfying:

(C1) $\mathcal{R}(b_0)$ contains a record $(\mathcal{S}_0, s_0)$ with $\mathcal{S}_0 = (S_0, M)$, where $\mathcal{S}_0$ is the (b_0)-projection of $\mathcal{S}$. We set $s'_0 = s_0 + d$, where d is the difference of the (b_0)-projection.

(C2) For every node t of S with $f = \mathit{feat}(t)$ and whose right child is c, there is a (b_0)-DT pattern for $E_c = E[(E_S(c) \cap \mathit{exam}(B_b)) \cup \mathit{feat}(B_{b_0})]$, whose skeleton is equal to $(S_c, M \cap V(S_c))$. We denote by s_t the smallest size of any such (b_0)-DT pattern for E_c.

We start with showing the correctness of the definition for $\mathcal{R}(b)$. That is, we need to show that a record $(\mathcal{S}, s)$ is in $\mathcal{R}(b)$ if and only if s is the smallest number of a (b)-DT pattern for $E[\chi(B_b)]$, whose skeleton is $\mathcal{S}$.

Towards showing the forward direction, let $(\mathcal{S}, s)$ with $\mathcal{S} = (S, M)$ be in $\mathcal{R}(b)$. Then, $\mathcal{S}$ is a skeleton (b)-DT pattern satisfying (C1) and (C2). Moreover, if S does not contain a node t with $\mathit{feat}(t) = f$, then (C1) implies that $(\mathcal{S}, s) \in \mathcal{R}(b_0)$. But then there is a (b_0)-DT pattern T for $E[\chi(B_{b_0})]$ of size s, whose skeleton is $\mathcal{S}$. Note that because $\mathcal{S}$ is a skeleton (b)-DT pattern all template nodes of T only use the feature f_0 if also $f_0 \in \mathrm{TF}_F(b)$. Let T' be obtained from T after changing $\mathit{feat}(t)$ for all non-template nodes t of T with $\mathit{feat}(t) = f_0$ to f. Then, f_0 and f agree on all examples in $\mathit{exam}(B_b)$ (because of Observation 11 (1)) and therefore T' is a (b)-DT pattern of minimum size for $E[\chi(B_b)]$, whose skeleton is $\mathcal{S}$, which implies the validity of the record $(\mathcal{S}, s)$.

So suppose that S contains at least one node t with $\mathit{feat}(t) = f$. Because of (C1), we obtain that $\mathcal{R}(b_0)$ contains the record $(\mathcal{S}_0, s_0)$ such that $\mathcal{S}_0 = (S_0, M)$ is the (b_0)-projection of $\mathcal{S}$. Consequently, there is a (b_0)-DT pattern T_0 of size s_0 for $E[\chi(B_{b_0})]$, whose skeleton is $\mathcal{S}_0$. Let T'_0 be obtained from T_0 after applying the following operations:

- For every non-template node t of T_0 with $\mathit{feat}(t) = f_0$, we set $\mathit{feat}(t) = f$; note that this has no influence on how the examples in $\mathit{exam}(B_b)$ are classified by T_0 because f and f_0 agree on all examples in $\mathit{exam}(B_b)$ due to Observation 11 (1).
- Informally, we revert the changes made to $\mathcal{S}$ in Steps (S1) and (S2). More formally, let $\alpha : V(S_0) \to V(T_0)$ be the mapping that assigns to every node of S_0 the unique node in T_0 it originates from. We revert (S1) and (S2) as follows:
  - Let A be the set of all nodes in S_0 whose feature assignment was changed from f to f_0 in Step (S1). Then, we change the feature assignment $\mathit{feat}(t)$ from f_0 to f for every node t of T_0 with $t \in \alpha(O)$ (where $\alpha(O) = \{\, \alpha(o) \mid o \in O \,\}$.
  - Let D be the set of all nodes of S removed in Step (S2). Then, for every such node $t \in D$, we do the following. W.l.o.g., we will assume that t has a parent p and a left child c in $\mathcal{S}$, since if this is not the case the constructions is analogous. We insert a new node t_0 into T_0 with $\mathit{feat}(t) = f$ anywhere on the path between $\alpha(p)$ and $\alpha(c)$ in T_0. For instance, if t is the left (right) child of p in $\mathcal{S}$, we will insert t_0 by making it a left (right) child of $\alpha(p)$ and by making the left (right) child of $\alpha(p)$ in T_0 the left child of t_0. Note that the exact position of where we insert t_0 does not matter because all examples in $\mathit{exam}(B_b)$ that will end up at t_0 will go to the left child of t_0.

Note that $|T'_0| = |T_0| + d = s'_0$ and all examples in $\mathit{exam}(B_b)$ still end up in the same place as they did in T_0. Furthermore, there is now a one-to-one correspondence β between nodes t of T'_0 with $\mathit{feat}(t) = f$ and nodes t' in S with $\mathit{feat}(t') = f$.

Now, for every node t of S with $\mathit{feat}(t) = f$, let T_t be the (b_0)-DT pattern of size s_t, whose skeleton is $(S_c, M \cap V(S_c))$ and whose existence is guaranteed by (C3). Finally, let T be obtained from T'_0 after inserting T_t below the right child of $\beta(t)$ for every t of S with $\mathit{feat}(t) = f$. Then, $|T| = |T'_0| + \sum_{t \in V(S) \wedge \mathit{feat}(t)=f} |T_t| = s'_0 + \sum_{t \in V(S) \wedge \mathit{feat}(t)=f} s_t = s$. Moreover, T is a (b)-DT pattern for $E[\chi(B_b)]$ whose skeleton is $\mathcal{S}$. Finally, the minimality of T follows from the minimality of T_0 and T_t for every t of S with $\mathit{feat}(t) = f$. Therefore, T witnesses the validity of $(\mathcal{S}, s)$.

Towards showing the backward direction, let T be a (b)-DT pattern for $E[\chi(B_b)]$ of minimum size (s), whose skeleton is $\mathcal{S} = (S, M)$. Then, $\mathcal{S}$ is an inclusion-wise minimal skeleton (b)-DT pattern.

Let O be the set of all nodes t of T with $\mathit{feat}(t) = f$ such that t occurs in a subtree T_c of T for some right child c of a node p with $\mathit{feat}(p) \in \mathit{feat}(B_b) \setminus \mathit{feat}(b)$. Recall that f_0 is the unique feature in $\mathrm{TF}_F(b_0)$ that agrees with f on the examples in $\mathit{exam}(b)$ and recall that (because of Observation 11 (2) applied to b_0), it holds that f_0 agrees with f on all examples in $\mathit{exam}(B_b)$. Therefore, the tree T' obtained from T after changing $\mathit{feat}(t)$ from f to f_0 for every node t in O is a (b)-DT pattern for $E[\chi(B_b)]$. Moreover, since we only changed $\mathit{feat}(t)$ for the nodes t in O, it holds that T and T' have the same skeleton.

Now consider the case that S does not contain a node t with $\mathit{feat}(t) = f$. Then, T' does not contain such a node and therefore T' is a (b_0)-DT pattern for $E[B_{b_0}]$ of minimum size, whose skeleton is $\mathcal{S}$. Moreover, because of its minimality it holds that $(\mathcal{S}, |T'| = s) \in \mathcal{R}(b_0)$. Since moreover the (b_0)-projection of $\mathcal{S}$ is equal to $\mathcal{S}$, we obtain that $\mathcal{S}$ satisfies (C1) with $s'_0 = |T'| = s$. Finally, since S does not contain a node t with $\mathit{feat}(t) = f$, we obtain that $\mathcal{S}$ (trivially) satisfies (C2).

Now consider the final case that S contains a node t with $\mathit{feat}(t) = f$.

Towards showing (C1), let $\mathcal{S} = (S_0, M)$ be the (b_0)-projection of $\mathcal{S}$ and let s'_0 and d be as described in (C1). Moreover, let T'' be obtained from T' in the same manner as S_0 is obtained from S. That is, let T'' be obtained from T' after exhaustively applying the following procedure:

For every node t of T' with $\mathit{feat}(t) = f$, remove the subtree rooted at the right child of t from T'. Moreover, either:

(S1) if there is an example $e \in E_{T'}(t) \cap \mathit{exam}(b)$ such that $e(f) = 1$, then set $\mathit{feat}(t)$ to f_0,

(S2) otherwise remove t from T' and replace it by an edge from its parent to its left child in T'.

Then, T'' does not contain a node t with $\mathit{feat}(t) = f$ and therefore T'' is a (b_0)-DT pattern for $E[B_{b_0}]$. Moreover, since T'' was obtained from T' in the same manner as S_0 from S, we obtain that the skeleton of T'' w.r.t. b_0 is equal to $\mathcal{S}_0$. Finally, because of the minimality of T, T'' witnesses the validity of the record $(\mathcal{S}_0, |T''| = s'_0) \in \mathcal{R}(b_0)$, which shows that $\mathcal{S}$ satisfies (C1).

Towards showing (C2), let t be a node in T' with $\mathit{feat}(t) = f$, whose right child is c and let t' be the corresponding node in S with right child c'. Then T'_c is a (b_0)-DT pattern for $E_{T'}(c) \cap \mathit{exam}(B_b)$ whose skeleton is $(S_{c'}, M \cap V(S_{c'}))$. Moreover, the minimality of T (and T') implies that T'_c is a (b_0)-DT pattern for $E_{T'}(c) \cap \mathit{exam}(B_b)$ of minimum size, whose skeleton is $(S_{c'}, M \cap V(S_{c'}))$, which shows that $\mathcal{S}$ satisfies (C2).

We are now ready to argue how to compute $\mathcal{R}(b)$. Because of Lemma 15, we can enumerate all of the at most $\mathcal{O}(\omega^{3 \cdot 2^\omega})$ inclusion-wise minimal skeleton (b)-DT patterns in time $\mathcal{O}(\omega^{3 \cdot 2^\omega})$. Now for each such pattern $\mathcal{S} = (S, M)$, we will check whether $\mathcal{S}$ satisfies (C1)–(C2) as follows.

To verify (C1), we first need to compute $\mathcal{S}_0$ from $\mathcal{S}$, which can be achieved in time $\mathcal{O}(|S|)$, which because of Lemma 14 is at most $\mathcal{O}(2^\omega)$. We then check whether $\mathcal{R}(b_0)$ contains a

record of the form $(\mathcal{S}_0, s_0)$, which can be achieved in constant time. Therefore, the total time to verify (C1) is at most $\mathcal{O}(2^\omega)$.

Towards verifying that $\mathcal{S}$ satisfies (C2), we first verify that every example in $E_c = E[(E_S(c) \cap \mathit{exam}(B_b)) \cup \mathit{feat}(B_{b_0})]$ is correctly classified by $(S_c, M \cap V(S_c))$. This is possible, because $E_c \subseteq \mathit{exam}(b)$, which follows from Observation 11 (1) and the fact that c is the right child of t with $\mathit{feat}(t) = f$. If this is not the case, we correctly reject $\mathcal{S}$, otherwise we need to compute a (b_0)-DT pattern for $E_c = E[(E_S(c) \cap \mathit{exam}(B_b)) \cup \mathit{feat}(B_{b_0})]$ of minimum size, whose skeleton is equal to $(S_c, M \cap V(S_c))$, for every node t of S with $\mathit{feat}(t) = f$.

Let t be a node of S with $\mathit{feat}(t) = f$ and consider an inclusion-wise minimal (b_0)-DT pattern T^t for $E_c = E[(E_S(c) \cap \mathit{exam}(B_b)) \cup \mathit{feat}(B_{b_0})]$, whose skeleton is equal to $(S_c, M \cap V(S_c))$. Recall that $(E_S(c) \cap \mathit{exam}(B_b) \subseteq \mathit{exam}(b)$. Therefore, T^t only has to correctly classify the examples in $\mathit{exam}(b)$ and moreover every node t' of T^t with $\mathit{feat}(t') \in \mathit{feat}(B_b) \setminus \mathit{feat}(b_b)$ must have at least one example $e \in \mathit{exam}(b)$ such that $e(\mathit{feat}(t')) = 1$. Therefore, the only difference between T^t and its skeleton $((S_c, M \cap V(S_c)))$ is that T^t contains a right child (and a subtree rooted at that right child) for every node $t' \in V(S_c)$ with $t' \in M$. Therefore, it suffices to compute a DT of minimum size for $E_S(t') \cap \{\, e \mid e(\mathit{feat}(t')) = 1 \,\} \subseteq \mathit{exam}(b)$ using only features in $\mathrm{TF}(b)$ for every node $t' \in V(S_c) \cap M$. Because $E_S(t') \cap \{\, e \mid e(\mathit{feat}(t')) = 1 \,\} \subseteq \mathit{exam}(b)$ this can be achieved using Lemma 7 in time $\mathcal{O}((2^\omega)^{4\omega-1})$. Note that if no such DT exists for some node $t' \in V(S_c) \cap M$, we correctly reject $\mathcal{S}$, otherwise we employ Lemma 7 for every (of the at most ω) nodes t' in $V(S_c) \cap M$ to compute the (b_0)-DT pattern for E_c of minimum size, whose skeleton is equal to $(S_c, M \cap V(S_c))$, in time $\mathcal{O}(\omega(2^\omega)^{4\omega-1}) \in \mathcal{O}((2^\omega)^{4\omega})$. Repeating the process for every (of the at most 2^ω) nodes t of S with $\mathit{feat}(t) = f$, we obtain $\mathcal{O}(2^\omega(2^\omega)^{4\omega}) \in \mathcal{O}((2^\omega)^{4\omega+1})$ as the total time required to verify (C2).

In conclusion, the total run-time of the procedure is equal to the number $\mathcal{O}(\omega^{3\cdot 2^\omega})$ of possible $\mathcal{S} = (S, M)$'s times the time required to verify (C1)–(C2), which is dominated by the time $\mathcal{O}((2^\omega)^{4\omega+1})$ required to verify (C3). Therefore, the total run-time of the procedure is at most $\mathcal{O}(\omega^{3\cdot 2^\omega}(2^\omega)^{4\omega+1}) \in \mathcal{O}(\omega^{15\cdot 2^\omega})$. ◀

▶ **Lemma 18 (example introduce node).** *Let $b \in V(B)$ be an introduce node with child b_0 such that $\chi(b) \setminus \chi(b_0) = \{e\}$ for some example $e \in E$. Then, $\mathcal{R}(b)$ can be computed in $\mathcal{O}(\omega^{4\cdot 2^\omega})$ time.*

Proof. Informally, $\mathcal{R}(b)$ contains all records $(\mathcal{S}, s)$, where $\mathcal{S}$ is a skeleton (b)-DT pattern that correctly classifies the new example e and after forgetting the information about the example e, $\mathcal{S}$ can be turned into a skeleton (b_0)-DT pattern, whose skeleton is in $\mathcal{R}(b_0)$.

More formally, $\mathcal{R}(b)$ consists of all records $(\mathcal{S}, s)$, where $s = s'_0$ such that $\mathcal{S} = (S, M)$ is an inclusion-wise minimal skeleton (b)-DT pattern satisfying:

(C1) The example e is correctly classified by S, i.e. either e ends up in a positive or negative leaf of S if e is positive or negative, respectively, or $e \in E_S(t)$ for some $t \in V(S)$ with $\mathit{feat}(t) \in \mathrm{TF}(b)$ and $e(\mathit{feat}(t)) = 1$. Note that one of these two cases must happen for e, because $e(f) = 0$ for every feature $f \in \mathit{feat}(B_b) \setminus \mathit{feat}(b)$, because of Observation 11 (2).

(C2) $\mathcal{R}(b_0)$ contains a record $(\mathcal{S}_0, s_0)$ with $\mathcal{S}_0 = (S_0, M)$, where S_0 is the skeleton (b_0)-DT pattern obtained from (S, M) after exhaustively applying the following procedure.

For every node t of S with $f = \mathit{feat}(t) \in \mathrm{TF}(b) \setminus \mathrm{TF}(b_0)$ either:

(S1) if there is an example $e_0 \in \mathit{exam}(b_0) \cap E_S(t)$ such that $e_0(f) = 1$, then change $\mathit{feat}(t)$ to the unique feature in $\mathrm{TF}(b_0)$ that agrees with f on the examples in $\mathit{exam}(b_0)$,

(S2) otherwise remove t from S and replace it by an edge from its parent to its left child in S.

Note that because $\mathcal{S} = (S, M)$ was inclusion-wise minimal at most one node of S is removed by (S2), i.e. the at most one node t of S with $f = \mathit{feat}(t) \in \mathrm{TF}(b) \setminus \mathrm{TF}(b_0)$, $e \in E_S(t)$ and $e(\mathit{feat}(t)) = 1$. Let s'_0 be equal to s_0 if no node was removed by (S2) and otherwise let $s'_0 = s_0 + 1$.

We first show that the definition of $\mathcal{R}(b)$ is correct. That is, we need to show that a record $(\mathcal{S}, s) \in \mathcal{R}(b)$ if and only if s is the smallest number of a (b)-DT pattern for $E[\chi(B_b)]$, whose skeleton is $\mathcal{S}$.

Towards showing the forward direction, let $(\mathcal{S}, s) \in \mathcal{R}(b)$ with $\mathcal{S} = (S, M)$. Then, $\mathcal{S}$ is an inclusion-wise minimal skeleton (b)-DT pattern satisfying (C1)–(C2). Let T_0 be a (b_0)-DT pattern for $E[\chi(B_{b_0})]$ of minimum size s_0, whose skeleton is equal to $\mathcal{S}_0 = (S_0, M)$; note that the existence of T_0 is guaranteed by (C2). Let T'_0 be obtained from T_0 after reversing the changes applied in (S1) and (S2). More formally, T'_0 is obtained from T_0 by changing the feature $\mathit{feat}(t)$ of every node t of T_0 whose corresponding node t' in S was changed in (S1) back to the feature of t'; note that this simply means that the old feature is replaced with a feature that only differs at the example e, where it is 1 instead of 0. Moreover, if there is a node t of S that was removed in (S2), then add a new node with feature $\mathit{feat}(t)$ into T_0 that is in-between the nodes in T_0 corresponding to the parent and the child of t in S. For instance, if t is the left (right) child of its parent p in S, then make the new node the left (right) child of the node corresponding to p in T_0 and make the original child the left child of the new node. If t had no parent in S, then simply make the node in T_0 corresponding to the left child of t in S to the left child of the new node. Then, T'_0 is a (b)-DT pattern for $E[\chi(B_b) \setminus \{e\}]$ of size s'_0, whose skeleton is $\mathcal{S}$. Moreover, because $\mathcal{S}$ satisfies (C1), T'_0 is also a (b)-DT pattern for $E[\chi(B_b)]$. Finally, T'_0 is of minimum size because so is T_0. Therefore, T'_0 is a witness for the validity of $(\mathcal{S}, s = s'_0)$.

Towards showing the backward direction, let T be a (b)-DT pattern for $E[\chi(B_b)]$ of minimum size, whose skeleton is equal to $\mathcal{S} = (S, M)$. Then, (S, M) clearly satisfies (C1). Towards showing that it also satisfies (C2), let T_0 be obtained from T as follows. Let A be the set of all nodes in T $\mathit{feat}(t) \in \mathrm{TF}(b) \setminus \mathrm{TF}(b_0)$. Note that because T is inclusion-wise minimal there is at most one node $a' \in A$ such that there is no example e_0 in $\mathit{exam}(b_0) \cap E_T(a')$ with $e_0(\mathit{feat}(a')) = 1$; i.e. if there is such a node then it must be that $e \in E_T(a')$ and $e(\mathit{feat}(a')) = 1$ and therefore the example e goes to the right child of a' in T. We distinguish two cases: (1) T contains such a node a' or (2) T does not contain such a node a'. In the first case, we replace a' in T with an edge between its parent and its left child. We then change $\mathit{feat}(a)$ to the unique feature in $\mathrm{TF}(b_0)$ that agrees with $\mathit{feat}(a)$ on the examples in $\mathit{exam}(B_{b_0})$ for every $a \in A \setminus \{a'\}$. In the second case, we simply change $\mathit{feat}(a)$ to the unique feature in $\mathrm{TF}(b_0)$ that agrees with $\mathit{feat}(a)$ on the examples in $\mathit{exam}(B_{b_0})$ for every $a \in A$. Then T_0 is a (b_0)-DT pattern for $E[\chi(B_{b_0})]$, whose skeleton is $\mathcal{S}_0$. Moreover, T_0 is minimal because so is T. Therefore, T_0 witnesses the existence of the record $(\mathcal{S}_0, |T_0|)$ in $\mathcal{R}(b_0)$, showing that $\mathcal{S}$ satisfies (C2).

We are now ready to argue how to compute $\mathcal{R}(b)$. Because of Lemma 15, we can enumerate all of the at most $\mathcal{O}(\omega^{3 \cdot 2^\omega})$ inclusion-wise minimal skeleton (b)-DT patterns in time $\mathcal{O}(\omega^{3 \cdot 2^\omega})$. Now for each such pattern $\mathcal{S} = (S, M)$, we will check whether $\mathcal{S}$ satisfies (C1)–(C2) as follows. We can verify (C1) by checking where e ends up in $\mathcal{S}$ in time $\mathcal{O}(|S|)$, which because of Lemma 14 is at most $\mathcal{O}(2^\omega)$. To check (C2), we first need to compute $\mathcal{S}_0$, which can be achieved in time $\mathcal{O}(|S|) \in \mathcal{O}(2^\omega)$. We then need to check whether $\mathcal{R}(b_0)$ contains a record $(\mathcal{S}_0, s_0)$, which can be achieved in constant time. Therefore, the total run-time of the procedure is $\mathcal{O}(\omega^{3 \cdot 2^\omega}(2^\omega)) \in \mathcal{O}(\omega^{4 \cdot 2^\omega})$. ◀

▶ **Lemma 19 (feature forget node).** *Let $b \in V(B)$ be a forget node with child b_0 such that $\chi(b_0) \setminus \chi(b) = \{f\}$ for some feature $f \in feat(E)$. Then, $\mathcal{R}(b)$ can be computed in $\mathcal{O}(|\mathcal{R}(b_0)|2^\omega)$ time.*

Proof. Informally, $\mathcal{R}(b)$ is obtained from $\mathcal{R}(b_0)$ by going over all records $(\mathcal{S}, s)$ in $\mathcal{R}(b_0)$ with $\mathcal{S} = (S, M)$ and transforming $\mathcal{S}$ from a skeleton (b_0)-DT pattern to a skeleton (b)-DT pattern by forgetting the information about the feature f. Here it is important to note that not every skeleton (b_0)-DT pattern can be transformed into a skeleton (b)-DT pattern, but only those where the subtree of the right child of every node t with $feat(t) = f$ is "complete" in the sense that it does not have template nodes. Moreover, since this procedure can lead to the same skeleton (b)-DT pattern obtained from many different skeleton (b_0)-DT patterns, one then needs to set the size to the minimum size for any such skeleton (b_0)-DT pattern.

More formally, we say that a record $(\mathcal{S}, s) \in \mathcal{R}(b_0)$ with $\mathcal{S} = (S, M)$ is complete for feature f if for every node t of S with $feat(t) = f$ and whose right child is c, it holds that $t' \in M$ for every node t' in S_c with $feat(t') \notin feat(b_0)$. In other words $\mathcal{S}$ is complete for f if every subtree rooted at a right child of a node t with $feat(t) = f$ does not contain any template nodes that are not in M.

We also define the (b)-projection of $\mathcal{S} = (S, M)$ with $(\mathcal{S}, s) \in \mathcal{R}(b_0)$ as the skeleton (b)-DT pattern obtained from $\mathcal{S}$ after exhaustively applying the following procedure:

For every node t of S with $feat(t) = f$ and right child c do the following:

(P1) remove the subtree S_c from S,

(P2) remove all nodes in $M \cap V(S_c)$ from M,

(P3) if there is no example $e \in exam(b)$ with $e(f) = 1$, then replace t in S with an edge from its parent to its left child. Otherwise, add t to M.

Then, $\mathcal{R}(b)$ is defined as the set of all records $(\mathcal{S}, s)$ such that s is the minimum integer such that there is a record $(\mathcal{S}', s) \in \mathcal{R}(b_0)$ that is complete for f and $\mathcal{S}$ is the (b)-projection of $\mathcal{S}'$.

We start with showing the correctness of the definition for $\mathcal{R}(b)$. That is, we need to show that a record $(\mathcal{S}, s)$ is in $\mathcal{R}(b)$ if and only if s is the smallest number of a (b)-DT pattern for $E[\chi(B_b)]$, whose skeleton is $\mathcal{S}$.

Towards showing the forward direction, let $(\mathcal{S}, s)$ be in $\mathcal{R}(b)$. By definition of $\mathcal{R}(b)$, we obtain that s is the smallest number such that there is a record $(rdt', s) \in \mathcal{R}(b_0)$ with $\mathcal{S}' = (S', M')$ that is complete for f such that $\mathcal{S}$ is the (b)-projection of $\mathcal{S}'$. Let $(\mathcal{S}', s) \in \mathcal{R}(b_0)$ and let T be the (b_0)-DT pattern for $E[\chi(B_{b_0})]$ of size s, whose skeleton is $\mathcal{S}'$; which exists because $(\mathcal{S}', s) \in \mathcal{R}(b_0)$. Because $\mathcal{S}'$ is complete for f, we obtain that T is also a (b)-DT pattern for $E[\chi(B_b)] = E[\chi(B_{b_0})]$. Moreover, because $\mathcal{S}$ is the (b)-projection of $\mathcal{S}'$, it follows that $\mathcal{S}$ is the skeleton of T. Therefore, T witnesses the validity of the record $(\mathcal{S}, s)$.

Towards showing the backward direction, let T be a (b)-DT pattern for $E[\chi(B_b)]$ of minimum size (s), whose skeleton is $\mathcal{S} = (S, M)$. Then, T is also a (b_0)-DT pattern for $E[\chi(B_{b_0})] = E[\chi(B_b)]$. Moreover, the skeleton $\mathcal{S}' = (S', M')$ of T (seen as a (b_0)-DT pattern) is complete for f and $\mathcal{S}$ is the (b)-projection of $\mathcal{S}'$. Therefore, T witnesses the validity of the record $(\mathcal{S}', s)$ for b_0 and hence $(\mathcal{S}', s) \in \mathcal{R}(b_0)$.

To compute $\mathcal{R}(b)$ we use an array with one entry for every possible (b)-pattern for $E[\chi(B_b)]$ that is initially empty. We then go over every record $(\mathcal{S}, s) \in \mathcal{R}(b_0)$ and do the following:

- Check whether $\mathcal{S}$ is complete for f. This can be achieved in time $\mathcal{O}(|\mathcal{S}|)$, which because of Lemma 14 is at most $\mathcal{O}(2^\omega)$.

- If $\mathcal{S}$ is not complete for f, we stop. Otherwise, we compute the (b)-projection $\mathcal{S}'$ of $\mathcal{S}$ in time $\mathcal{O}(|\mathcal{S}|) \in \mathcal{O}(2^\omega)$.
- We then check the currently stored record for $\mathcal{S}'$ in our array and update its size to s, if its current size is larger than s. This can be achieved in constant time.

Therefore, the total run-time is at most $\mathcal{O}(|\mathcal{R}(b_0)|2^\omega)$. ◀

▶ **Lemma 20 (example forget node).** *Let $b \in V(B)$ be a forget node with child b_0 such that $\chi(b_0) \setminus \chi(b) = \{e\}$ for some example $e \in E$. Then, $\mathcal{R}(b)$ can be computed in $\mathcal{O}(|\mathcal{R}(b_0)|2^\omega)$ time.*

Proof. Informally, we can obtain $\mathcal{R}(b)$ from $\mathcal{R}(b_0)$ by removing all records that do not classify the example e yet, i.e. where e ends up at the right child of a template node t with $\mathit{feat}(t) \in \mathrm{TF}_F(b_0)$. Moreover, for the records that we keep we have to replace features in $\mathrm{TF}(b_0) \setminus \mathrm{TF}(b)$ with the features in $\mathrm{TF}(b)$ that agrees with them on all examples in $\mathit{exam}(b)$.

More formally, we say that a record $(\mathcal{S}, s) \in \mathcal{R}(b_0)$ with $\mathcal{S} = (S, M)$ *classifies* e if there is no node t of S with $\mathit{feat}(t) \in \mathrm{TF}_F(b_0)$, $e \in E_S(t)$, and $e(\mathit{feat}(t)) = 1$. We say that a skeleton (b)-DT pattern $\mathcal{S} = (S, M)$ is the *(b)-projection* of a skeleton (b_0)-DT pattern $\mathcal{S}' = (S', M)$ if S is obtained from S' after changing $\mathit{feat}(t)$ to the unique feature in $\mathrm{TF}(b)$ that agrees with $\mathit{feat}(t)$ on all examples in $\mathit{exam}(b)$ for every node t of S' with $\mathit{feat}(t) \in \mathrm{TF}(b_0) \setminus \mathrm{TF}(b)$. Then, $\mathcal{R}(b)$ contains a record $(\mathcal{S}, s)$ for every record $(\mathcal{S}', s) \in \mathcal{R}(b_0)$ that classifies e such that $\mathcal{S}$ is the (b)-projection of $\mathcal{S}'$.

We start with showing the correctness of the definition for $\mathcal{R}(b)$. That is, we need to show that a record $(\mathcal{S}, s)$ is in $\mathcal{R}(b)$ if and only if s is the smallest number of a (b)-DT pattern for $E[\chi(B_b)]$, whose skeleton is $\mathcal{S}$.

Towards showing the forward direction, let $(\mathcal{S}, s)$ with $\mathcal{S} = (S, M)$ be in $\mathcal{R}(b)$. By definition of $\mathcal{R}(b)$, there is a record $(\mathcal{S}', s) \in \mathcal{R}(b_0)$ with $\mathcal{S}' = (S', M)$ such that $\mathcal{S}'$ classifies e and $\mathcal{S}$ is the (b)-projection of $\mathcal{S}'$.

Therefore, there is a (b_0)-DT pattern T' for $E[\chi(B_{b_0})]$ of size s, whose skeleton is $\mathcal{S}'$. Let T be obtained from T' in the same way that $\mathcal{S}$ is obtained from $\mathcal{S}'$, i.e. after changing $\mathit{feat}(t)$ for every t of T' with $\mathit{feat}(t) \in \mathrm{TF}(b_0) \setminus \mathrm{TF}(b)$ to the unique feature in $\mathrm{TF}(b)$ that agrees with $\mathit{feat}(t)$ on the examples in $\mathit{exam}(b)$. Then, because T' classifies e, we obtain that T is a (b)-DT pattern for $E[\chi(B_b)]$ of size s, which shows the validity of the record $(\mathcal{S}, s)$ for b.

Towards showing the backward direction, let T be a (b)-DT pattern for $E[\chi(B_b)]$ of minimum size (s), whose skeleton is $\mathcal{S} = (S, M)$. Then, T is also a (b_0)-DT pattern for $E[\chi(B_{b_0})] = E[\chi(B_b)]$. This is because $\mathrm{TF}_F(b) \subseteq \mathrm{TF}_F(b_0)$ and therefore $\mathit{feat}(t) \in \mathrm{TF}_F(b_0)$ for every template node t of T with $\mathit{feat}(t) \in \mathrm{TF}_F(b)$. Note that $\mathrm{TF}_F(b) \subseteq \mathrm{TF}_F(b_0)$ follows from Observation 11 (1) as follows. Since if $f \in \mathrm{TF}_F(b)$, then there is a feature $f' \in \mathit{feat}(E) \setminus \mathit{feat}(B_b)$ that agrees with f on all examples in $\mathit{exam}(b)$. Moreover, because $f' \in \mathit{feat}(E) \setminus \mathit{feat}(B_b)$ and $e \in \mathit{exam}(B_b) \setminus \mathit{exam}(b)$, it holds that $e(f') = 0$. Let $\mathcal{S}'$ be the skeleton of T (seen as a (b_0)-DT pattern). Then $\mathcal{S}'$ classifies e (because T every template node t of T satisfies $e(\mathit{feat}(t)) = 0$) and moreover, $\mathcal{S}$ is the (b)-projection of $\mathcal{S}'$. Finally, T witnesses the validity of the record $(\mathcal{S}', s)$ for b_0 and hence $(\mathcal{S}', s) \in \mathcal{R}(b_0)$.

To compute $\mathcal{R}(b)$, we simply need to go over all records $(\mathcal{S}, s)$ in $\mathcal{R}(b_0)$, check whether $\mathcal{S}$ classifies e and compute the (b)-projection of $\mathcal{S}$. Since both of the latter can be done in time $\mathcal{O}(|S|)$, which because of Lemma 14 is at most $\mathcal{O}(2^\omega)$, we obtain $|\mathcal{R}(b_0)|2^\omega$ as the total run-time of the procedure. ◀

▶ **Lemma 21 (join node).** *Let $b \in V(T)$ be a join node with children b_1 and b_2, where $\chi(b) = \chi(b_1) = \chi(b_2)$. Then, $\mathcal{R}(b)$ can be computed in $\mathcal{O}(\omega^{4 \cdot 2^\omega})$ time.*

Proof. $\mathcal{R}(b)$ contains a record $(\mathcal{S}, s)$ with $\mathcal{S} = (S, s)$ if $\mathcal{S}$ is a skeleton (b)-DT pattern and s is the smallest number such for every i with $i \in \{1, 2\}$, $\mathcal{R}(b_i)$ contains a record $(\mathcal{S}_i, s_i)$ with $\mathcal{S}_i = (S, M_i)$ such that $M_1 \cap M_2 = \emptyset$, $M_1 \cup M_2 = M$ and $s = s_1 + s_2 - |S|$.

We start with showing the correctness of the definition for $\mathcal{R}(b)$. That is, we need to show that a record $(\mathcal{S}, s)$ is in $\mathcal{R}(b)$ if and only if s is the smallest number of a (b)-DT pattern for $E[\chi(B_b)]$, whose skeleton is $\mathcal{S}$.

Towards showing the forward direction, let $(\mathcal{S}, s)$ with $\mathcal{S} = (S, M)$ be in $\mathcal{R}(b)$. By definition of $\mathcal{R}(b)$, S is a skeleton (b)-DT pattern and for every i with $i \in \{1, 2\}$, there is a $(\mathcal{S}_i, s_i) \in \mathcal{R}(b_i)$ with $\mathcal{S}_i = (S, M_i)$ such that $M_1 \cup M_2 = M$ and $s = s_1 + s_2 - |S|$. Let T_i be the (b_i)-DT pattern for $E[\chi(B_{b_i})]$ of size s_i, whose skeleton is $\mathcal{S}_i$.

We now show how to construct a (b)-DT pattern T for $E[\chi(B_b)]$ from T_1 and T_2, whose skeleton is $\mathcal{S}$. Let $\alpha_i : V(S) \to V(T_i)$ be the unique and injective mapping that maps every node of S to its corresponding node in T_i. In the beginning we set $T = S$ and throughout we let $\alpha : V(\mathcal{S}) \to T$ be the (unique) injective function that maps every node of S to its corresponding node in T. We then apply the following operations to T:

- Let r be the root of S and let $T_i^{\uparrow}[r]$ be the component of $T_i \setminus \alpha_i(r)$ containing the ancestor of $\alpha_i(r)$ in T_i. W.lo.g., we can assume that $T_i^{\uparrow}[r]$ are both non-empty, since the construction otherwise is analogous. Let $T^{\uparrow}[r]$ be the tree obtained from the disjoint union of $T_1^{\uparrow}[r]$ and $T_2^{\uparrow}[r]$ after identifying the root of $T_1^{\uparrow}[r]$ as the root of $T^{\uparrow}[r]$ and making the root of $T_2^{\uparrow}[r]$ the left child of the left-most leave in $T_1^{\uparrow}[r]$. We now add $T^{\uparrow}[r]$ to T by taking the disjoint union of both, making $\alpha(r)$ the left child of the left-most leave of $T^{\uparrow}[r]$ and identifying the root of $T^{\uparrow}[r]$ as the new root of T.
- Let p and c be two nodes in $\mathcal{S}$ such that p is the parent of c in $\mathcal{S}$. Moreover, let $T_i[p, c]$ be the component of $T_i \setminus \{\alpha_i(p), \alpha_i(c)\}$ containing the parent of $\alpha(c)$. W.l.o.g., we can assume that $T_i[p, c]$ is non-empty since the construction otherwise is analogous. Let $T[p, c]$ be the tree obtained from the disjoint union of $T_1[p, c]$ and $T_2[p, c]$ after making the root of $T_2[p, c]$ the left child of the left-most leaf of $T_1[p, c]$ and identifying the root of $T_1[p, c]$ as the root of $T[p, c]$. We now add $T[p, c]$ to T by taking the disjoint union of both, making the root of $T[p, c]$ the left respectively right child of $\alpha(p)$ (depending on whether c is the left or right child of p in $\mathcal{S}$), making $\alpha(c)$ the left child of the left-most leaf in $T[p, c]$.
- Let t be a node of $\mathcal{S}$ with $t \in M$. Because $M = M_1 \cup M_2$ and $M_1 \cap M_2 = \emptyset$, we obtain that $t \in M_i$ for some $i \in \{1, 2\}$. Let T_i^t be the subtree of T_i rooted at the right child of $\alpha_i(t)$. Then, we add T_i^t to T and make the root of T_i^t the right child of $\alpha(t)$ in T.

Note first that $|T| = s_1 + s_2 - |\mathcal{S}|$ because all nodes in $\mathcal{S}$ appear in both T_1 and T_2 and every other node of T appear only in one of T_1 or T_2. Moreover, it is straightforward to verify that the skeleton of T w.r.t. b is $\mathcal{S}$. We now show that T is a (b)-DT pattern for $E[\chi(B_b)]$. To see this consider an example $e \in \mathit{exam}(B_b)$. If $e \in \mathit{exam}(b)$, then there is nothing to show. Therefore, assume that $e \in \mathit{exam}(B_{b_i})$. Then, because of Observation 11 (2), we obtain that $e(f) = 0$ for every feature in $\mathit{feat}(B_{b_{3-i}})$ and therefore e ends up in the same leaf as it ended up in T_i, showing that it is correctly classified. Therefore, T witnesses the validity of the record $(\mathcal{S}, s)$.

Towards showing the backward direction, let T be a (b)-DT pattern for $E[\chi(B_b)]$ of minimum size (s), whose skeleton is $\mathcal{S} = (S, M)$. We start with some simple but important observations:

(O1) for every example e in $\mathit{exam}(B_b) \setminus \mathit{exam}(b)$ either $e \in \mathit{exam}(B_{b_1}) \setminus \mathit{exam}(b)$ or $e \in \mathit{exam}(B_{b_2}) \setminus \mathit{exam}(b)$,

(O2) for every node t of T with $\mathit{feat}(t) \in \mathit{feat}(B_b) \setminus \mathit{feat}(b)$, it holds that either $\mathit{feat}(t) \in$

$feat(B_{b_1}) \setminus feat(b)$ or $feat(t) \in feat(B_{b_2}) \setminus feat(b)$,

(O3) if t is a node of T with $feat(t) \in feat(B_{b_i}) \setminus feat(b)$, then $e(feat(t)) = 0$ for every example e in $exam(B_{b_{3-i}}) \setminus feat(b)$ (because of Observation 11 (2)).

For $i \in \{1, 2\}$, let T_i be obtained from T as follows. For every node t of T with $feat(t) \in feat(B_{b_{3-i}})$ and t does not appear in a subtree rooted at the right child of a node p in T with $feat(t) \notin feat(b)$, either:

- if there is an example $e \in E_T(t) \cap exam(b)$ with $e(feat(t)) = 1$, then change $feat(t)$ to the unique feature in $\mathrm{TF}_F(b_i)$ that agrees with $feat(t)$ on all examples in $exam(b)$. Note that $\mathrm{TF}(b_i)$ contains such a feature, because $feat(f) \in feat(E) \setminus feat(B_{b_i})$. Moreover, remove the subtree rooted at the right child of t from T.
- otherwise, replace t (and the subtree rooted at the right child of t in T) with an edge from t's parent to the left child of t.

Then, because of (O3), we obtain that T_i is a (b_0)-DT pattern for $E[\chi(B_{b_i})]$. Moreover, it is straightforward to verify that the skeleton of T_i w.r.t. b_i is $\mathcal{S} = (S, M_i)$ for some M_i such that $M_1 \cup M_2 = M$ and $M_1 \cap M_2 = \emptyset$. Note that the minimality of T implies the minimality of T_i and therefore $(\mathcal{S}_i, |T_i|) \in \mathcal{R}(b_i)$. Finally, it is easy to see that $|T_1| + |T_2| - |S| = |T| = s$ showing that $(\mathcal{S}, s) \in \mathcal{R}(b)$.

We are now ready to argue how to compute $\mathcal{R}(b)$. Because of Lemma 15, we can enumerate all of the at most $\omega^{3 \cdot 2^\omega}$ inclusion-wise minimal skeleton (b)-DT patterns in time $\mathcal{O}(\omega^{3 \cdot 2^\omega})$. For every such skeleton (b)-DT pattern $\mathcal{S} = (S, M)$, we then have to compute the minimum number s such that there are records $(\mathcal{S}_i, s_i) \in \mathcal{R}(b_i)$ with $\mathcal{S}_i = (S, M_i)$ such that $M_1 \cup M_2 = M$, $M_1 \cap M_2 = \emptyset$, and $s = s_1 + s_2 - |S|$. Note that since we assume that the records in $\mathcal{R}(b_i)$ are stored as an array indexed by $\mathcal{S}$, we can access all such records $(\mathcal{S}, s_i)$ in $\mathcal{R}(b_i)$ with $\mathcal{S} = (S, M_i)$ in constant time. Moreover, since there are 2^ω possible choices for M_i (because of Lemma 14), there are at most $2^{2\omega}$ pairs of records $(\mathcal{S}_1, s_1) \in \mathcal{R}(b_1)$ and $(\mathcal{S}_2, s_2) \in \mathcal{R}(b_2)$. Furthermore, for every such pair we only need to check that $M_1 \cup M_2 = M$ and $M_1 \cap M_2 = \emptyset$ and we need to compute $s_1 + s_2 + |S|$, all of which can be achieved in time $\mathcal{O}(\omega)$ (since $|M_i| \leq \omega$). Therefore, the total time required for one skeleton (b)-DT pattern $\mathcal{S} = (S, M)$ is at most $\mathcal{O}(2^{2\omega}\omega)$ and since there are at most $\omega^{3 \cdot 2^\omega}$ choices for $\mathcal{S}$, we obtain $\mathcal{O}(\omega^{3 \cdot 2^\omega} 2^{2\omega}\omega) \in \mathcal{O}(\omega^{4 \cdot 2^\omega})$ as the total run-time of the procedure. ◀

We are now ready to show our main result of this section.

Proof of Theorem ??. The algorithm computes the set of all valid records $\mathcal{R}(b)$ for every node b of B using a bottom-up dynamic programming algorithm starting at the leaves of B. It then decides whether the CI E has a DT of size at most s by checking whether $\mathcal{R}(r)$, for the root r of B, contains a record of the form $(\emptyset, s')$ with $s' \leq s$. The correctness of the algorithm follows from Lemmas 16, 17, 18, 19, 20, and 21.

The run-time of the algorithm is at most the number of nodes of B, which can be assumed to be bounded from above by $\mathcal{O}(\omega(|E| + |feat(E)|))$ (Proposition **??**), times the maximum time required to compute $\mathcal{R}(b)$ for any of the node types of a nice tree-decomposition, which is obtained for feature introduce node (Lemma 17) with a run-time of $\omega^{15 \cdot 2^\omega}$. Therefore, we obtain $\mathcal{O}(\omega(|E| + |feat(E)|)\omega^{15 \cdot 2^\omega})$ as the total run-time of the algorithm, which shows the theorem. ◀

5 Conclusion

We have initiated the study of the parameterized complexity of learning DTs from data. Our main tractability result provides novel insights into the structure of DTs and is based on

the novel decompositional paramater renamable treewidth that seems to be well suited to measure the complexity of input instances for the problem.

The problem of learning DTs comes in many variants and flavors, which opens up a wide range of new research directions to explore. For instance:

- What other (structural) parameters can be exploited to efficiently learn DTs? For instance, given the relation of renamable treewidth to rank-width: Is learning DTs of small size fixed-parameter tractable parameterized by the rank-width of $G_I(E)$?
- Instead of learning DTs of small size, one often wants to learn DTs of small height. Therefore, it is natural to ask whether our approach can be also used in this setting. While one can adapt our approach to obtain an XP-algorithm for learning DTs of small height parameterized by renamable treewidth, it is not clear to us whether the problem also allows for an fpt-algorithm.
- Can we extend our approach to CIs, where features range over an arbitrary domain? In this case, one usually still uses DTs that make binary decisions (i.e. whether a feature is smaller equal or larger than a given threshold). While it is relatively easy to see that our approach can be extended if the domain's size (for every feature) is bounded or used as an additional parameter, it is not clear what happens if the size of the domain is allowed to grow arbitrarily.

References

1 Jørgen Bang-Jensen and Gregory Gutin. *Digraphs*. Springer Monographs in Mathematics. Springer-Verlag London Ltd., London, second edition, 2009.

2 Christian Bessiere, Emmanuel Hebrard, and Barry O'Sullivan. Minimising decision tree size as combinatorial optimisation. In Ian P. Gent, editor, *Principles and Practice of Constraint Programming - CP 2009*, pages 173–187, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

3 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

4 Endre Boros, Yves Crama, Peter L. Hammer, Toshihide Ibaraki, Alexander Kogan, and Kazuhisa Makino. Logical analysis of data: classification with justification. *Ann. Oper. Res.*, 188(1):33–61, 2011.

5 Endre Boros, Vladimir Gurvich, Peter L. Hammer, Toshihide Ibaraki, and Alexander Kogan. Decomposability of partially defined Boolean functions. *Discr. Appl. Math.*, 62(1-3):51–75, 1995.

6 Endre Boros, Takashi Horiyama, Toshihide Ibaraki, Kazuhisa Makino, and Mutsunori Yagiura. Finding essential attributes from binary data. *Ann. Math. Artif. Intell.*, 39:223–257, 11 2003. doi:10.1023/A:1024653703689.

7 Endre Boros, Toshihide Ibaraki, and Kazuhisa Makino. Variations on extending partially defined Boolean functions with missing bits. *Information and Computation*, 180(1):53–70, 2003.

8 Yves Crama, Peter L. Hammer, and Toshihide Ibaraki. Cause-effect relationships and partially defined Boolean function. *Ann. Oper. Res.*, 16:299–326, 1988.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

10 Adnan Darwiche and Auguste Hirth. On the reasons behind decisions. In Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2020.

11 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer Verlag, New York, 2nd edition, 2000.

12 Finale Doshi-Velez and Been Kim. A roadmap for a rigorous science of interpretability. *CoRR*, abs/1702.08608, 2017. URL: http://arxiv.org/abs/1702.08608, arXiv:1702.08608.

13 Rodney G. Downey and Michael R. Fellows. *Fundamentals of parameterized complexity*. Texts in Computer Science. Springer Verlag, 2013.

14 Wolfgang Espelage, Frank Gurski, and Egon Wanke. Deciding clique-width for graphs of bounded tree-width. *J. Graph Algorithms Appl.*, 7(2):141–180, 2003.

15 Robert Ganian and Petr Hliněný. On parse trees and Myhill-Nerode-type tools for handling graphs of bounded rank-width. *Discr. Appl. Math.*, 158(7):851–867, 2010.

16 Bryce Goodman and Seth R. Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, 2017.

17 Petr Hliněný and Sang-il Oum. Finding branch-decompositions and rank-decompositions. *SIAM J. Comput.*, 38(3):1012–1032, 2008.

18 Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

19 Toshihide Ibaraki, Yves Crama, and Peter L. Hammer. *Partially defined Boolean functions*, page 511–563. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011.

20 Marcin Kaminski, Vadim V. Lozin, and Martin Milanic. Recent developments on graphs of bounded clique-width. *Discret. Appl. Math.*, 157(12):2747–2761, 2009.

21 Daniel T. Larose. *Discovering knowledge in data*. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, 2005. An introduction to data mining.

22 Harry R. Lewis. Renaming a set of clauses as a Horn set. *J. of the ACM*, 25(1):134–135, January 1978.

23 Zachary C. Lipton. The mythos of model interpretability. *Communications of the ACM*, 61(10):36–43, 2018.

24 E.J. McCluskey. *Introduction to the Theory of Switching Circuits*. Electrical and electronic engineering series. Princeton University series. McGraw-Hill, 1965.

25 Don Monroe. AI, explain yourself. *AI Communications*, 61(11):11–13, 2018. doi:10.1145/3276742.

26 Sreerama K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Min. Knowl. Discov.*, 2(4):345–389, 1998. doi:10.1023/A:1009744630224.

27 Sang-il Oum. Approximating rank-width and clique-width quickly. *ACM Transactions on Algorithms*, 5(1), 2008.

28 Sang-il Oum. Approximating rank-width and clique-width quickly. *ACM Trans. Algorithms*, 5(1):10:1–10:20, 2008.

29 Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.

30 J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. doi:10.1023/A:1022643204877.

31 Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 931–942. Springer, 2014.

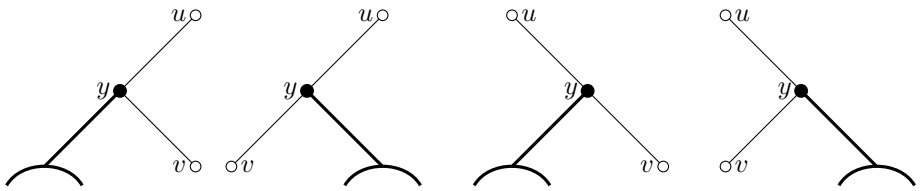
32 Richard Stanley and Eric W. Weisstein. Catalan number, from mathworld–a wolfram web resource, 2015.

33 Egon Wanke. k-NLC graphs and polynomial algorithms. *Discr. Appl. Math.*, 54(2-3):251–266, 1994. Efficient algorithms and partial k-trees.

NLC-width

We start off with some definitions. We say an edge is a *left (right) edge* of a subcubic rooted tree if it connects a non-leaf node with his left (resp. right) child. Let Y be a rooted subcubic tree and $S \in \{left, right\}$, then we say the pair (Y, S) is a *single pair* if the root of Y has at most one child and the side S indicates whether the edge from the root is either a left or right edge. Moreover, we say that (Y, S) is single pair in a subcubic rooted tree T if Y is a (downwards) maximal subtree of T and in Y the root can only have the S child. Note that when tree of a single pair is made of just a node, the side is not relevant.

Now we can define two operations on subcubic rooted trees and single pairs. We say that we *plug in* a single pair (Y, S) in a left (right) edge uv as follows: we make the root y of Y the left (right) child of u, $Y \setminus \{y\}$ to be the S subtree of y and v to be the $H \in \{left, right\} \setminus S$ child of y. See Figure 2 for the corresponding drawings. Note after a plug in of a single pair in an edge, the node v belongs in the same side of the subtree rooted at u as it was before the plug in.



(a) $(Y, left)$ left. **(b)** $(Y, right)$ left. **(c)** $(Y, left)$ right. **(d)** $(Y, right)$ right.

Figure 2 The drawings describe the plug in operation in the different four cases. The bold part highlight the single pair (Y, S).

Let (Y, S) be a single pair in a rooted subcubic tree T, then we *remove* (Y, S) from T as follows. Let y be the root of Y. If y is the root of T, then we obtain an empty tree. If y is a leaf node of T, then we obtain $T - y$. Otherwise let y be a non-root and non-leaf node, let u be the parent of y and v be the child of y that is not in $V(Y)$, then we consider the tree obtained from T after replacing y with v as the child of u and deleting Y. See Figure 3 for an example.

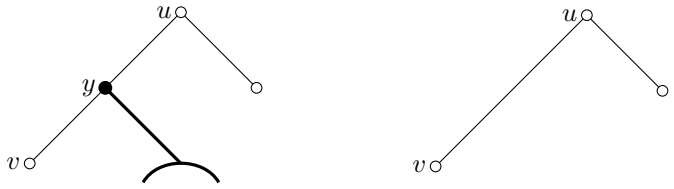


Figure 3 The drawing describe an example of the remove operation: a single pair $(Y, right)$ is removed from a subcubic rooted tree. The bold part highlight the single pair (Y, S).

It is clear from the four different plug in cases that if we want to plug in two pairs (Y, S) and (Y', S') on an edge uv such that the ancestor-descendant relationship is given, say y of Y has to be in the path from the root to y' of Y', then we can do these plug ins in any order but with some care. It is the same if we first plug in (Y, S) in the edge uv and then plug in (Y', S') in the edge yv or if we first plug in (Y', S') in the edge uv and then plug in (Y, S) in the edge uy'. See Figure 4 for the an example.

For a subset of labels $A \subseteq [k]$, we define the feature template f_A by setting $e(f_A) = 1$ if

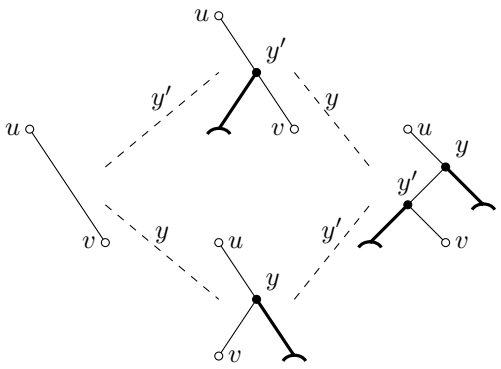


Figure 4 An example of plugging in two pairs $(Y, left)$ and $(Y', right)$ in a left edge uv.

and only if $lab(e) \in A$ and $e(f_A) = 0$ otherwise. With a small abuse of notation, we often identify the feature template f_A with the corresponding subset of labels A.

Suppose we have a DT such that some feature label i occurs twice on a path from the root to the leaves, say f_1 is the instance closer to the root and f_2 is the other instance. If f_2 is in the left (resp. right) subtree of f_1, we remove f_2's right (resp. left) subtree. In this case we say we have done an *actual removal*.

Suppose we have a feature template labelled A in our decision tree. Let $A_1, \ldots, A_\ell$ be the sequence of feature templates on the path from the root to A in order (not including A). Let $A'_i = A_i$ if A is in the right sub-tree of A_i and let $A'_i = \overline{A_i}$ otherwise. If $\overline{A} \subseteq A'_1 \cup \ldots \cup A'_\ell$, then we remove the subtree rooted at the left child of A. If $A \subseteq \overline{A'_1} \cup \ldots \cup \overline{A'_\ell}$, then we remove the subtree rooted at the right child of A. In this case we say we have done a *template removal*. If this procedure has been applied to a record exhaustively, we say that the DT is *reduced*.

To be short, for a DT T and a node v, we write $v \in T$ instead of $v \in V(T)$ and $v \notin T$ otherwise.

We now formally define two important operations. Given a DT T, we say that we *reduce* T if we exhaustively do actual removals and template removals. Call $r(T)$ the resulting DT.

Recall that in any DT T, every non-leaf node v has one of the following three contents: v is a real feature (without label), or v is a feature with a label, or v is a future feature with the corresponding subset of labels. A *relabelling* p for T is an assignment of contents of T as follows. Every feature is assigned to a feature with is either future, real or with a label. We say that we *relabel* the DT T via the relabelling p if for every node of T we apply the corresponding assignment and call $p(T)$ the resulting DT.

The following lemma shows that, after repeatedly applying it the necessary amount of times, to obtain a reduced DT after a sequence of relabels, it is safe to reduce at the end.

▶ **Lemma 22** (Relabelling Lemma). *Let T be a DT and p be relabelling of T. Then $(r \circ p \circ r)(T) = (r \circ p)(T)$.*

Proof. For every $v \in T$, we want to prove $v \in (r \circ p \circ r)(T) \Leftrightarrow v \in (r \circ p)(T)$.

$\Rightarrow$ Suppose there is a node $v \notin (r \circ p)(T)$. Since $v \in p(T)$, there is a set of ancestors of v in $p(T)$ that allows to remove v. Let A_v be the union of all the minimal set of ancestors of v in $p(T)$ that allows to remove v. If A_v is a set of ancestors of v in T that allows to reduce v then $v \notin r(T)$ and so $v \notin (r \circ p \circ r)(T)$. Otherwise let A'_v be the subset of A_v in $(p \circ r)(T)$. We conclude by noting that A'_v contains one of the minimal sets A_v is composed of and so $v \notin (r \circ p \circ r)(T)$.

$\Leftarrow$ Suppose there is a node $v \notin (r \circ p \circ r)(T)$. If $v \in (p \circ r)(T)$, there exists a set A_v of ancestors of v in $(p \circ r)(T)$ that allows to reduce v. Then A_v is a set of ancestors of v in $p(T)$ that allows to reduce v and so $v \notin (r \circ p)(T)$. If $v \notin (p \circ r)(T)$ then $v \notin r(T)$: there exists a set A_v of ancestors of v in T that allows to remove v. This means A_v is a set of ancestors of v in $p(T)$ that allows to remove v and so $v \notin (r \circ p)(T)$. ◀

We say that a DT T is a *real DT* if every non-leaf node is either a real feature or a future feature, whereas it is a *DT template* if it contains no real feature.

Let B be a rooted subcubic tree that corresponds to a k-NLC expression of the graph $G_I(E)$. For $b \in V(B)$, we write $feat(b)$ and $exam(b)$ for the sets of features and examples introduced at nobe b. We say that a real DT T is a DT for the node b if every real feature of T is an element of $feat(b)$ and every example in $exam(b)$ is correctly classified by T, i.e. if $e \in exam(b) \cap E^+$ then e ends in a leaf with a $+$ label and if $e \in exam(b) \cap E^-$ then e ends in a leaf with a $-$ label.

Given a real DT T and a node $b \in B$, often we want to perform a very specific composition of operations. Let p_b be the following relabelling of T: every real feature of T is assigned to a feature with the label given by the k-NLC expression at node b and every other feature is assigned to itself. Then the composition $r \circ p_b$ is called the *standard reduction* of T at node b. Given a DT T and a node $b \in B$, it is useful to give the following relabelling p'_b: every feature with a label is assigned to the real feature of that node. The relabelling p'_b is called the *real relabelling* of T at node b.

We say that a DT template T is a DT for the node b if there exits a real DT T' for b such that T is the standard reduction of T'. In this case we say that T' is the witness of T for b.

The *semantics* for a record are defined as follows. We say that a pair (T, s) is a *record* for the node $b \in B$ and we write $(T, s) \in \mathcal{R}(b)$, if T is a DT template for b and s is the minimum number of elements that have been deleted from a witness T' of T for b.

Now, it suffices to compute $\mathcal{R}(b)$ via leaf-to-root dynamic programming. The following four lemmas show how this can be achieved for all of the four types of nodes in a k-NLC expression tree B.

▶ **Lemma 23.** *Let $b \in V(B)$ be a leaf node. Then $\mathcal{R}(b)$ can be computed in time $\mathcal{O}()$.*

Proof. Let v be the vertex of $G_I(E)$ that corresponds to the leaf node b. This means either $v \in E$ or $v \in feat(E)$.

We have to enumerate all possible reduced DT templates T for b. It is enough to consider all reduced DT templates T of height at most $k + 1$ and discard those that are not DT templates for b; these are at most () by Lemma 13. We add the pair $(T, 0)$ to the set of records $\mathcal{R}(b)$.

Now we have to show the correctness of the construction for $\mathcal{R}(b)$, i.e. $(T, s) \in \mathcal{R}(b)$ if and only if s is the minimum number of elements that have been deleted from a witness T' of T for b.

We start with the forward direction. Let $(T, s) \in \mathcal{R}(b)$. By construction, we have that $s = 0$ and T is a DT template for b which is already reduced. Then T is trivially a witness of T for b.

Now we prove the backward direction. Let T be a reduced DT template such that 0 is the minimum number of elements that have been deleted from a witness T' of T for b. This means T' is obtained from T after the real relabelling at node b is applied: T is a DT template among the considered DTs above which leads to the fact that $(T, 0) \in \mathcal{R}(b)$. ◀

▶ **Lemma 24.** *Let $b \in V(B)$ be a join node. Then $\mathcal{R}(b)$ can be computed in time $\mathcal{O}()$.*

Proof. Let b_L and b_R be the left, resp. right, child of b in B: we may assume the labels for $feat(b_L)$ are in $[k]$ and the labels for $feat(b_R)$ are in $[k']$. Moreover, let M be the $k \times k$ $\{0,1\}$ matrix that represent the node b. Finally, for every label $i \in [k]$, let $A_i = \{j \in [k] \mid M_{i,j} = 1\}$.

We consider every reduced DT T for b with feature labels in $[k] \cup [k']$ and future feature labels in $\mathcal{P}([k])$; these are at most () by Lemma 13.

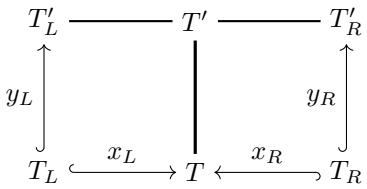
For every such DT T, we create a DT T_L as follows. Let p_* be the following relabelling: for every $i' \in [k']$, every feature with label i' is assigned to the future feature A_i. Then we apply the composition $r \circ p_*$ to T. In a symmetrical way we create a DT T_R. Let p'_* be the following relabelling: for every $i \in [k]$, every feature with label i is assigned to the future feature A_i. Then we apply the composition $r \circ p'_*$ to T.

Now we want to understand if there is a record in $\mathcal{R}(b_L)$ of the form (T_L, s_L) for some positive integer s_L and if there is a record in $\mathcal{R}(b_R)$ of the form (T_R, s_R) for some positive integer s_R: if the answer is yes in both cases, we add a record $(T, s_L + s_R)$ to $\mathcal{R}(b)$; otherwise we discard this option.

Now we have to show the correctness of the construction for $\mathcal{R}(b)$. We start with the forward direction. Let $(T, s) \in \mathcal{R}(b)$. By construction there exist records $(T_L, s_L) \in \mathcal{R}(b_L)$ and $(T_R, s_R) \in \mathcal{R}(b_R)$ such that T_L and T_R are obtained by the application of $r \circ p_*$ and $r \circ p'_*$ respectively to T and $s_L + s_R = s$.

By induction, for $H \in \{L, R\}$, we know that s_H is the minimum number of elements that have been deleted from a witness T'_H of T_H for b_H.

For $H \in \{L, R\}$, we define maps x_H and y_H as follows. Let $x_H : V(T_H) \to V(T)$ and $y_H : V(T_H) \to V(T'_L)$ be the functions that maps every node of T_H to the corresponding node in T and in T'_L and note that by constructions both these maps are injective.



Moreover, $V(T) \setminus Im(x_H)$ and $V(T'_H) \setminus Im(y_H)$ can be partitioned into subtrees that have been deleted after the application of $r \circ p_*$, $r \circ p'_*$ on T or of the standard reduction on T'_H: let $X^*_H = \{x^H_1, \ldots, x^H_\ell\}$ and $Y^*_H = \{y^H_1, \ldots, y^H_r\}$ be the set of roots of the above subtrees in $V(T) \setminus Im(x_H)$ and $V(T'_H) \setminus Im(y_H)$ respectively. In addition, for every i, let Y^H_i be the maximal subtree of T'_H rooted at y^H_i with no elements from $Im(y_H)$ and that does not contain any vertex from $Y^*_H \setminus \{y^H_i\}$; let (Y^H_i, S^H_i) the corresponding single pair. In a similar way, for every i, let X^H_i be the maximal subtree of T rooted at x^H_i with no elements from $Im(x_H)$ and that does not contain any vertex from $X^*_H \setminus \{x^H_i\}$; let (X^H_i, S^H_i) the corresponding single pair. Finally, for any i, let p^H_i be the shortest downwards path in T'_H that contains y^H_i and with both endpoints in $Im(y_H)$, say $y_H(v_1)$ and $y_H(v_2)$.

Claim 1: For every $H \in \{L, R\}$ and for every $i, j \in [r]$, the paths p^H_i and p^H_j are either edge disjoint or $p^H_i = p^H_j$.

Proof. If p^H_i and p^H_j are edge disjoint, then the statement is proven immediately. Suppose p^H_i and p^H_j share an edge. By minimality and the fact they are downwards paths, p^H_i and p^H_j share the endpoint towards the root. If they also share the other endpoint, then the statement is proven immediately. Suppose now their endpoints towards the leaves is different, say v_i and v_j, and consider the last edge those paths have in common in a descending order, say uv.

Without loss of generality, we can assume v_i belongs to the left branch of v and v_j belongs to the right branch of v. Note that $v \in V(T'_H) \setminus Im(y_H)$, or we get a contradiction due the minimality of p_i^H. Now we get the following contradiction: by construction, v_i and v_j are both elements of $Im(y_H)$ but at least one of them must be in $V(T'_H) \setminus Im(y_H)$ since it is an element of either Y_i^H or of Y_j^H. This proves Claim 1.

Now we consider the path q_i^H in T having endpoints $x_H(v_1)$ and $x_H(v_2)$.

Claim 2: For every $H \in \{L, R\}$ and for every $i \in [r]$, every internal vertex of q_i^H is an element of X_H^*.

Proof. Suppose that q_i^H has an internal vertex $t \notin X_H^*$. By definition, there exists a vertex $v \in V(T_H)$ such that $x_H(v) = t$. Since x_H is injective then $v \notin \{v_1, v_2\}$. Since y_H is injective $y_H(v) \notin \{y_H(v_1), y_H(v_2)\}$ and belongs to p_i, which contradicts the minimality of p_i. This proves Claim 2.

Before we describe how to obtain a witness T' of T for b, we must make an observation. We note that $Im(x_L) \cup Im(x_R) = V(T)$: the idea is that every node of T must originate from either T_L or T_R.

Now we are able to describe how to obtain a witness T' of T for b. For every $i \in [r]$, we consider the internal vertices $\{y_{i_1}, \ldots, y_{i_t}\}$ of p_i^L, that we recall belong to Y_L^*, in a descending order. In the path q_i^L, we plug in, in sequence, every single pair $(Y_{i_j}^L, S_{i_j}^L)$ rooted at y_{i_j} with $j \in [t]$ in the last edge of the path q_i^L. Note that, in the case an element of Y_L^* is present in more than one p_i^L, we plug in the corresponding single pair only once. Note also that whenever we plug in some single pair $(Y_{i_j}^L, S_{i_j}^L)$ in a DT, the tree $Y_{i_j}^L$ has real features and future features as nodes. Call this graph T^*. Now we do the same sequence of plug ins of the single pairs corresponding to the internal vertices of p_i^R in the last edge of the path q_i^R. Again, in the case an element of Y_R^* is present in more than one p_i^R, we plug in the corresponding single pair only once. Call the tree obtained in this way T'. Node that T' contains real features from $feat(b_L)$ and from $feat(b_R)$ and future features with labels in $\mathcal{P}([k])$.

To conclude this part of the proof we have to show two things: (i) T is obtained from T' after removing s vertices; (ii) T' is a real DT for b. We start proving (i): by construction T' is obtained from T after adding s_L elements from T'_L and s_R elements from T'_R, and so with $s_L + s_R = s$ more elements.

Before considering statement (ii), we consider the following relabelling p_+ of T': every real feature in $feat(b_R)$ is a assigned to a feature with its label at node b_R and every other feature is assigned to itself. The real DT T'_L can be obtained from T' by the application of the composition $r \circ p_* \circ p_+$.

Now we consider statement (ii). We show that given an example $e \in exam(b_L)$, e is correctly classified by T' and to do so we show that e ends in a leaf of T' that corresponds to the leaf where e ends in T'_L. Say that e goes along a path P of T'_L from the root to a leaf ℓ and let Q be the corresponding path T', i.e. the from the root r to ℓ (note that by construction ℓ is present in T' and is still a leaf). Let v be a node of Q, we can have the following different cases.

- v is a real feature from $feat(b_L)$: v is also present in T'_L as real feature;
- v is a real feature from $feat(b_R)$: v might not be present in T'_L due reductions but if it is present it is a future feature A_i for some $i \in [k]$;
- v is a future feature f_A: v might not be present in T'_L due reductions but if it is present it is still the same future feature A_i.

If v is present in T'_L then the behaviour of v on e in T'_L and in T' is the same. Suppose now v is a node of Q that is being reduced due his label and so it is not present in T'_L. This means there is a set of ancestors of v such that their labels allows to remove v and by construction v behaves on e like those ancestors. This proves e goes along Q and in particular it ends at leaf ℓ and so T' is a real DT for b_L. With symmetric construction, we show that T' is also a real DT for b_R.

Now we prove the backward direction. Let T be a reduced DT such that s is the minimum number of elements that have been deleted from a witness T' of T for b. In particular, we recall that T' is a real DT for b with actual feature labels in $[k] \cup [k']$ and future feature labels in $\mathcal{P}([k])$.

We create at real DT T'_L by the application of the composition $r \circ p_* \circ p_+$ to T'. By assumption T' is a real DT for b_L and by construction T'_L is a real DT for b_L. Denote with T_L the DT template obtained from T'_L by standard reduction and denote with s_L the number of nodes that have been deleted from T'_L to obtain T. By induction we have $(T_L, s_L) \in \mathcal{R}(b_L)$. Now we note that T_L is obtained from T after the application of the composition $r \circ p_*$. In a symmetric way, we construct T'_R, T_R and the record $(T_R, s_R) \in \mathcal{R}(b_R)$. Then $(T, s_L + s_R) \in \mathcal{R}(b)$. ◀

▶ **Lemma 25.** *Let $b \in V(B)$ be relabelling node. Then $\mathcal{R}(b)$ can be computed in time $\mathcal{O}()$.*

Proof. Let b_C be the unique child of b in B. Let R be the mapping of $[k]$ to itself that represent the node b. Moreover, since we are considering a *nice* NLC-expression we can assume R is the identity mapping, i.e. $R(\ell) = \ell$, for all values except for a unique element i of its domain, i.e. $R(i) = j$ for some $j \in [k] \setminus \{i\}$.

We say that a future feature A is *good* if it does not distinguish between i and j, that is $i \in A$ if and only if $j \in A$, and *bad* otherwise. Let (T_C, s_C) be an element of $\mathcal{R}(b_C)$. Let p'' the following relabelling of the DT template T_C: every feature with label i is assigned to label j and every future feature with label A is assigned to the future feature with label $A \setminus \{i\}$.

If T_C has a bad future feature then we do not take any other action. Suppose now T_C has only good future features; now let T be the DT template obtained from T_C after the application of the composition $r \circ p''$ and let s^* be the number of nodes that have been deleted from T_C to T.

If there is a record in $\mathcal{R}(b)$ of the form (T, s') for some integer $s' \leq s_C + s^*$ then we do not take any other action. If there is a record in $\mathcal{R}(b)$ of the form (T, s') for some integer $s' > s_C + s^*$ then we replace it with $(T, s_C + s^*)$. If there is no record in $\mathcal{R}(b)$ of the form (T, s') for some integer s' then we add $(T, s_C + s^*)$ to $\mathcal{R}(b)$.

Now we have to show the correctness of the construction for $\mathcal{R}(b)$, i.e. $(T, s) \in \mathcal{R}(b)$ if and only if s is the minimum number of elements that have been deleted from a witness T' of T for b.

We start with the forward direction. Let $(T, s) \in \mathcal{R}(b)$. By construction there exists a record $(T_C, s_C) \in \mathcal{R}(b_C)$ such that T is obtained from T_C after the application of $r \circ p''$ and let $s^* = s - s_C$. By induction s_C is the minimum amount of nodes that have been deleted from a witness T'_C of T_C for b_C. By construction we also know that every future feature of both T'_C and T_C is good.

Denote with T' the real DT obtained T'_C after the application of $r \circ p''$: note that this last reduction does not any node since every future feature of T'_C is good and there is no feature with label i. To conclude this part of the proof we have to show two things: (i) T is obtained from T' after removing s vertices; (ii) T' is a witness of T for b.

Before proving (i), we describe how T can be obtained from T'. Let p''' be the following relabelling of T': every real feature that contains j is assigned to the real feature $A \cup \{i\}$ and every other feature is assigned to itself. Then the application of the composition p''', the standard reduction and $r \circ p''$ to T' is exactly the standard reduction for T' which then result to the DT template T. By Lemma 22 the score of the standard reduction from T' to T is exactly $s_C + s^* = s$.

Now we consider statement (ii). First note that $exam(b) = exam(b_C)$. We show that a given example $e \in exam(b)$ is correctly classified by T'. Say that e goes along a path P of T'_C from the root to a leaf ℓ. We show e goes along the path P in T' as well: every real feature has not changed and so e behaves the same. Since every future feature of T'_C is good, then e behave the same on the corresponding future feature of T'.

Now we prove the backward direction. Let T be a reduced DT such that s is the minimum number of elements that have been deleted from a witness T' of B for b. In particular, we recall that real T' is a DT for b with real features and future feature labels in $\mathcal{P}([k] \setminus \{i\})$.

We create the real DT T'_C as the application of $r \circ p'''$ to T', the DT template T_C as the application of the standard reduction to T'_C. By construction we have $(T_C, s_C) \in \mathcal{R}(b_C)$, where s_C is the number of nodes that have been removed from T'_C to T_C. Note that T_C has only good future features. Finally we note that T is obtained from T_C by the application of $r \circ p''$. ◀