Lower Bounds for QBFs of Bounded Treewidth

Johannes K. Fichte
TU Dresden
Germany
johannes.fichte@tu-dresden.de

Markus Hecher
TU Wien
Austria
hecher@dbai.tuwien.ac.at

Andreas Pfandler
TU Wien
Austria
pfandler@dbai.tuwien.ac.at

Abstract

The problem of deciding the validity (QSat) of quantified Boolean formulas (QBF) is a vivid research area in both theory and practice. In the field of parameterized algorithmics, the well-studied graph measure treewidth turned out to be a successful parameter. A well-known result by Chen [9] is that QSat when parameterized by the treewidth of the primal graph and the quantifier rank of the input formula is fixed-parameter tractable. More precisely, the runtime of such an algorithm is polynomial in the formula size and exponential in the treewidth, where the exponential function in the treewidth is a tower, whose height is the quantifier rank. A natural question is whether one can significantly improve these results and decrease the tower while assuming the Exponential Time Hypothesis (ETH). In the last years, there has been a growing interest in the quest of establishing lower bounds under ETH, showing mostly problem-specific lower bounds up to the third level of the polynomial hierarchy. Still, an important question is to settle this as general as possible and to cover the whole polynomial hierarchy. In this work, we show lower bounds based on the ETH for arbitrary QBFs parameterized by treewidth and quantifier rank. More formally, we establish lower bounds for QSat and treewidth, namely, that under ETH there cannot be an algorithm that solves QSat of quantifier rank i in runtime significantly better than *i*-fold exponential in the treewidth and polynomial in the input size. In doing so, we provide a reduction technique to compress treewidth that encodes dynamic programming on arbitrary tree decompositions. Further, we describe a general methodology for a more fine-grained analysis of problems parameterized by treewidth that are at higher levels of the polynomial hierarchy. Finally, we illustrate the usefulness of our results by discussing various applications of our results to problems that are located higher on the polynomial hierarchy, in particular, various problems from the literature such as projected model counting problems.

LICS '20, July 8–11, 2020, Saarbrücken, Germany
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7104-9/20/07.
DOI 10.1145/3373718.3394756

CCS Concepts: • **Theory of computation** → **Parameterized complexity and exact algorithms**; **Complexity theory and logic**; *Complexity classes*; • **Mathematics of computing** → **Graph algorithms**; *Trees*; • **Computing methodologies** → *Knowledge representation and reasoning*.

Keywords: Parameterized Complexity, ETH, Lower Bounds, Quantified Boolean Formulas, Treewidth, Pathwidth

ACM Reference Format:
Johannes K. Fichte, Markus Hecher, and Andreas Pfandler. 2020. Lower Bounds for QBFs of Bounded Treewidth. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany*. ACM, New York, NY, USA, 15 pages.

Acknowledgments. This work has been supported by the Austrian Science Fund (FWF), Grants P26696, P30930-N35, P32830, and Y698, and the German Science Fund (DFG), Grant HO 1294/11-1. Hecher is also affiliated with the University of Potsdam, Germany. We would like to thank the reviewers as well as Stefan Szeider and Stefan Woltran for their support. Special appreciation goes to Michael Morak for early discussions.

1 Introduction

Treewidth, which was introduced specifically for graph problems by Robertson and Seymour in a series of papers [50–54], is a popular parameter in the community of parameterized complexity [13, 16, 30] and according to Google Scholar mentioned in 20,000 results (queried on April 27, 2020). Treewidth is a combinatorial invariant that renders a large variety of NP-complete or #P-complete graph problems tractable [6, 10]. Among these problems are for example deciding whether a graph has a Hamiltonian cycle, whether a graph is 3-colorable, or determining the number of perfect matchings of a graph [12]. Still, treewidth has also been widely employed for important applications that are defined on more general input structures such as Boolean satisfiability (Sat) [55] and constraint satisfaction (CSP) [14, 31]. Even problems that are located "beyond NP" such as probabilistic inference [46], problems in knowledge representation and reasoning [18, 33, 49] as well as deciding the validity (QSat) of quantified Boolean formulas (QBF) can be turned tractable using treewidth; for QSat we also parameterize by the number of alternating quantifier blocks (quantifier rank) [9]. However, QSat remains intractable when parameterized by treewidth alone [1], which is established using a particular fragment of path decompositions for QBF. QSat is also known as the prototypical problem for the polynomial hierarchy in descriptive complexity [34, 36]. Indeed, an encoding in QBF allows the characterization of problems on certain levels of the hierarchy using results by Fagin [23]. This has, for instance, been done for reasoning problems [19–21].

The meta results on treewidth are the well-known *Courcelle's theorem* [11] and its logspace version [22], which states that whenever one can encode a problem into a formula in monadic second order logic (MSO), then the problem can be decided in time linear in the input size and some function in the treewidth. While Courcelle's theorem provides a full framework for classifying problems concerning the existence of a tractable algorithm, its practical application is limited due to potentially huge constants, and the exponential runtime in the treewidth (upper bound) may result in a tower of exponents that is far from optimal. In contrast, the available upper bounds are more immediate for QSat: Chen [9] showed that one can decide validity for a given QBF in time exponential in the treewidth where the treewidth is on top of a *tower*[1] of iterated exponentials of height that equals the quantifier rank in the formula. Since the quantifier rank required to encode a problem directly matches the level on which the problem is located in the polynomial hierarchy, reductions to QSat seem natural. Lampis, Mitsou, and Mengel [41] employed this fact and proposed reductions from a collection of reasoning problems in AI to QSat that yield quite precise (up to a constant factor) upper bounds on the runtime. In consequence, these results highlight QBF encodings as a very handy and precise alternative to Courcelle's theorem. A natural question is whether one can significantly improve existing algorithms or establish limits that, unless very bad things happen in computational complexity theory, an algorithm with a certain runtime cannot exist. Lampis, Mitsou, and Mengel also consider this question using results [42] for QBF of quantifier rank two (2-QSat). While these results for the second level are applicable to numerous important problems, there is also a plethora of interesting problems that are even harder, which underlines the need for further research in this direction.

In this paper, we address lower bounds for the runtime of an algorithm that exploits treewidth in a more general setting. We establish results for QBFs of bounded treewidth and of *arbitrary* quantifier rank, thereby providing a novel method to generalize the result for 2-QSat in a non-incremental way.

A way to establish tight lower bounds in parameterized complexity theory is to assume the *exponential time hypothesis (ETH) [37]* and construct reductions. ETH is a widely accepted standard hypothesis in the fields of exact and parameterized algorithms. ETH states that there is some real $s > 0$ such that we cannot decide satisfiability of a given 3-CNF formula F in time $2^{s \cdot |F|} \cdot \|F\|^{\mathcal{O}(1)}$ [13, Ch.14], where $|F|$ refers to the number of variables and $\|F\|$ to the *size* of F, which is number of variables plus number of clauses in F. Recently, Lampis and Mitsou [42] established that 2-QSat ($\exists\forall$-Sat and $\forall\exists$-Sat) cannot be solved by an algorithm that runs in time single exponential in the treewidth of the primal graph (*primal treewidth*) when assuming ETH. The primal graph of a QBF Q has as vertices the variables of Q and there is an edge between two variables if they occur together in a clause or term, respectively. Pan and Vardi [47] mention in an earlier work that this extends to 3-QSat ($\forall\exists\forall$-Sat and $\exists\forall\exists$-Sat), and ℓ-QSat, if ℓ is an *odd* number. But it does not extend constructively to the case, where ℓ is even. Therefore, a new approach is needed to show the complete picture for QSat. While Marx and Mitsou [45] considered certain graph problems that are located on the third level of the polynomial hierarchy [57], they emphasize that the classical complexity results do not provide sufficient explanation why double- or triple-exponential dependence on treewidth is needed and one requires quite involved proofs for each problem separately. However, they state that intuitively the quantifier rank of the problem definitions are the common underlying reason for being on higher levels of the polynomial hierarchy and for requiring high dependence on treewidth. A natural generalization of the statement to arbitrary QBFs is formally stated in the following hypothesis.

Claim 1. *Under ETH, QSat for a closed formula Q in prenex normal form with n variables, primal treewidth k, and quantifier rank ℓ* cannot *be decided in time* $\text{tow}(\ell, o(k)) \cdot \text{poly}(n)$.

Contributions. In this paper, we prove Claim 1, which strengthens the importance of QBF encodings for problems parameterized by treewidth, and establish a general methodology to obtain treewidth lower bounds for problems of the polynomial hierarchy. Our *contributions* are as follows:

1. We consider ETH and QSat and establish the full picture of runtime lower bounds for algorithms parameterized by treewidth in connection to the quantifier rank of the formula. We present a reduction that significantly *compresses* treewidth and applies to any instance of QSat without restricting the quantifier rank while only assuming ETH. Note that this "compression" is constructive and independent of the original instance size, which is different from existing methods, e.g., [42, 45, 47]. In fact, compression only depends on the original parameter treewidth.

2. We provide a novel methodology for a more fine-grained analysis of algorithms parameterized by treewidth. This methodology relies only on the ETH and allows:
   a. for simply using reductions from QBF to exclude runtime results (height of the tower) for treewidth and
   b. for directly concluding lower bounds for projected model counting problems (PQSat), that is, $\#\Sigma_\ell$Sat and $\#\Pi_\ell$Sat [17],

[1] Function $\text{tow}(\ell, k)$ is a tower of iterated exponentials of 2 of height ℓ with k on top. More precisely, for integer k, we define $\text{tow} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ by $\text{tow}(1, k) = 2^k$ and $\text{tow}(\ell + 1, k) = 2^{\text{tow}(\ell, k)}$ for all $\ell \in \mathbb{N}$.

which serve as canonical problems in counting complexity, as well as for various other problems.
Instead of establishing problem-specific reductions from SAT for problems higher in the polynomial hierarchy, e.g., [42, 45], our reduction and methodology are very general. We illustrate its applicability in the showcases in Section 3.2.

Novel Techniques. We constructively encode core ideas of a dynamic programming algorithm on tree decompositions into a QBF that expresses solving an instance of QSat by means of a QSat oracle of *one level* higher in the hierarchy (self-reduction) while achieving a certain compression of treewidth. More precisely, we provide a reduction that reduces any instance Q of QSat of treewidth t and quantifier rank ℓ into an instance Q' of QSat of treewidth $O(\log t)$ and quantifier rank $\ell + 1$, while the size of Q' is linearly bounded in the size of Q. Notice that the treewidth of Q' only depends on the treewidth of Q, but is independent of, e.g., the number of variables and quantifier rank of Q. Hence, *treewidth* of Q' is *compressed* compared to the original treewidth of Q. Atserias and Oliva [1] cover a related setting: compressing pathwidth[2] for a fragment of path decompositions of QBFs thereby increasing the quantifier rank by two. However, we require a general, constructive method to compress the width of *arbitrary* tree decompositions of any QBF, thereby increasing quantifier rank by only one, and improve their result (Corollary 29).

Our reduction is novel in the following sense:

1. We use a given tree decomposition to guide the evaluation of the considered formula, which allows us to decouple the variables sufficiently to decrease treewidth and we thereby achieve exponential *compression* of the parameter treewidth. By construction of the reduction, the lower bound results carry over to the *larger* parameter *pathwidth*[2] and even 2-*local pathwidth*, where each variable occurs at most twice. Note that this direction is by construction and does not hold in general. However, particular novelty lies in encoding essentials of dynamic programming, which will be presented in the more general context of treewidth (tree decompositions).

2. In the proof, we use a reduction approach that balances redundancy and structural dependency (captured by treewidth or pathwidth), which allows us to apply this method to QBFs of *arbitrary* quantifier rank, thereby increasing quantifier rank by only one.

3. Our approach might help to improve solvers utilizing treewidth, as instances of huge treewidth might become solvable in practice (cf., [8, 28]) after applying our reduction. Indeed, our reduction encodes dynamic programming on tree decompositions into a Boolean formula, namely, *guessing* of finite states for table entries of decomposition nodes, *checking* whether certain entries sustain, and *propagating* entries among different nodes. As this technique, although presented for QBFs, does not explicitly encode quantifier dependencies into the Boolean formula, the technique is hopefully of general use.

[2]Pathwidth is similar to treewidth, but admits only *certain* tree decompositions, whose tree is just a path.

Connection to kernels. Note that our approach is orthogonal to kernelization as kernelizations tackle bounds of the *instance size* by the considered parameter, whereas here we target reducing the parameter itself and not the size of the input instance.

2 Preliminaries

Basics. For a set X, let 2^X be the *power set of* X. The function $\text{tow}(\ell, k)$ is defined as in Footnote 1. The domain $\mathcal{D}$ of a function $f : \mathcal{D} \to \mathcal{A}$ is given by $\text{dom}(f)$. By $f^{-1} : \mathcal{A} \to \mathcal{D}$ we denote the inverse function $f^{-1} := \{f(d) \mapsto d \mid d \in \text{dom}(f)\}$ of a given function f, if it exists. To permit operations such as $f \cup g$ for functions f and g, the functions may be viewed as relations. We use the symbol "$\cdot$" as placeholder for a value of an argument, which is clear from the context and the actual value is negligible. We let $\mathbb{N}$ contain all positive integers and $\mathbb{N}_0$ all non-negative integers. Throughout this paper, we refer by $\log(\cdot)$ to the binary logarithm.

Computational Complexity. We assume familiarity with standard notions in computational computational complexity [48], counting complexity classes [17], and parameterized complexity [13, 16, 30]. We recall some basic notions. Let Σ and Σ' be some finite alphabets. We call $I \in \Sigma^*$ an *instance* and $\|I\|$ denotes the size of I. Let $L \subseteq \Sigma^* \times \mathbb{N}$ and $L' \subseteq \Sigma'^* \times \mathbb{N}$ be two parameterized problems. An fpt-reduction r using g from L to L' is a many-to-one reduction from $\Sigma^* \times \mathbb{N}$ to $\Sigma'^* \times \mathbb{N}$ such that for all $I \in \Sigma^*$ we have $(I, k) \in L$ if and only if $r(I, k) = (I', k') \in L'$ with $k' \leq g(k)$, where $f, g : \mathbb{N} \to \mathbb{N}$ are fixed computable functions such that r is computable in time $f(k) \cdot \text{poly}(\|I\|)$. We call r also an f-*bounded* fpt-reduction using g for given f and g.

Quantified Boolean Formulas (QBFs). We define *Boolean formulas* and their evaluation in the usual way and *literals* are variables or their negations. For a Boolean formula F, we denote by $\text{var}(F)$ the set of variables of F. Logical operators $\land, \lor, \neg, \to, \leftrightarrow$ are used in the usual meaning. A *term* is a conjunction of literals and a *clause* is a disjunction of literals. F is in *conjunctive normal form (CNF)* if F is a conjunction of clauses and F is in *disjunctive normal form (DNF)* if F is a disjunction of terms. In both cases, we identify F by its set of clauses or terms, respectively. From now on assume that a Boolean formula is either in CNF or DNF. A formula is in c-*CNF* or c-*DNF* if each set in F consists of at most c many literals. Let $\ell \geq 0$ be integer. A *quantified Boolean formula Q (in prenex normal form)* is of the form $Q_1 V_1 . Q_2 V_2 . \cdots Q_\ell V_\ell . F$ where $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq \ell$ and $Q_j \neq Q_{j+1}$ for $1 \leq j \leq \ell - 1$; and where V_i are disjoint, non-empty sets of Boolean variables with $\bigcup_{i=1}^{\ell} V_i \subseteq \text{var}(F)$; and

F is a Boolean formula. We call ℓ the *quantifier rank* of Q and let $\mathsf{matrix}(Q) := F$. Further, we denote the set $\mathsf{fvar}(Q)$ of *free variables* of Q by $\mathsf{fvar}(Q) := \mathsf{var}(\mathsf{matrix}(Q)) \setminus (\bigcup_{i=1}^{\ell} V_i)$. If $\mathsf{fvar}(Q) = \emptyset$, then Q is referred to as *closed*, otherwise we say Q is *open*. Unless stated otherwise, we assume open QBFs. The truth (evaluation) of QBFs is defined in the standard way. An *assignment* is a mapping $\iota : X \rightarrow \{0,1\}$ defined for a set X of variables. An assignment ι' *extends* ι (by $\mathsf{dom}(\iota') \setminus \mathsf{dom}(\iota)$) if $\mathsf{dom}(\iota') \supseteq \mathsf{dom}(\iota)$ and $\iota'(y) = \iota(y)$ for any $y \in \mathsf{dom}(\iota)$. Given a Boolean formula F and an assignment ι for $\mathsf{var}(F)$. Then, for F in CNF, $F[\iota]$ is a Boolean formula obtained by removing every $c \in F$ with $x \in c$ and $\neg x \in c$ if $\iota(x) = 1$ and $\iota(x) = 0$, respectively, and by removing from every remaining clause $c \in F$ literals x and $\neg x$ with $\iota(x) = 0$ and $\iota(x) = 1$, respectively. Analogously, for F in DNF values 0 and 1 are swapped. For a given QBF Q and an assignment ι, $Q[\iota]$ is a QBF obtained from Q, where variables $x \in \mathsf{dom}(\iota)$ are removed from preceding quantifiers accordingly, and $\mathsf{matrix}(Q[\iota]) := (\mathsf{matrix}(Q))[\iota]$. A Boolean formula F *evaluates to true* if there exists an assignment ι for $\mathsf{var}(F)$ such that $F[\iota] = \emptyset$ if F is in CNF or $F[\iota] = \{\emptyset\}$ if F is in DNF. A closed QBF Q *evaluates to true (or is valid)* if $\ell = 0$ and the Boolean formula $\mathsf{matrix}(Q)$ evaluates to true. Otherwise, i.e., if $\ell \neq 0$, we distinguish according to Q_1. If $Q_1 = \exists$, then Q evaluates to true if and only if there exists an assignment $\iota : V_1 \rightarrow \{0,1\}$ such that $Q[\iota]$ evaluates to true. If $Q_1 = \forall$, then $Q[\iota]$ evaluates to true if for any assignment $\iota : V_1 \rightarrow \{0,1\}$, $Q[\iota]$ evaluates to true. An (open or closed) QBF Q is *satisfiable* if there is a truth assignment $\iota : \mathsf{fvar}(Q) \rightarrow \{0,1\}$ such that resulting closed QBF $Q[\iota]$ evaluates to true. Otherwise Q is *unsatisfiable*. Given a closed QBF Q, the *evaluation problem QSAT* of QBFs asks whether Q evaluates to true; ℓ-QSAT refers to the problem QSAT on QBFs of quantifier rank ℓ. The problem QSAT is PSPACE-complete and is therefore believed to be computationally harder than SAT [39, 48, 57]. For more details on QBFs we refer to [3, 39].

The *projected model counting problem* PQSAT takes an open QBF Q and asks to output the number of distinct assignments $\iota : \mathsf{fvar}(Q) \rightarrow \{0,1\}$ such that $Q[\iota]$ evaluates to true.

Example 2. *Consider the closed QBF $Q = \exists w, x.\forall y, z.D$, where $D := d_1 \vee d_2 \vee d_3 \vee d_4$, and $d_1 := w \wedge x \wedge \neg y$, $d_2 := \neg w \wedge \neg x \wedge y$, $d_3 := w \wedge y \wedge \neg z$, and $d_4 := w \wedge y \wedge z$. Observe that $Q[\iota]$ is valid under assignment $\iota = \{w \mapsto 1, x \mapsto 1\}$. In particular, $Q[\iota]$ can be simplified to $\forall y, z.(\neg y) \vee (y \wedge \neg z) \vee (y \wedge z)$, which is valid, since for any assignment $\kappa : \{y,z\} \rightarrow \{0,1\}$ the formula $Q[\iota][\kappa]$ (and therefore Q) evaluates to true.* ■

Tree Decompositions (TDs). For basic terminology on graphs and digraphs, we refer to standard texts [7, 15]. For an *arborescence* $T = (N, A, r)$, which is a directed, rooted tree with root r and a node $t \in N$, we let $\mathsf{cld}(t, T)$ be the set of all *child nodes* t', which have an outgoing edge $(t, t') \in A$ from t to t'. Let $G = (V, E)$ be a graph. A *tree decomposition (TD)* of graph G is a pair $\mathcal{T} = (T, \chi)$ where $T = (N, A, r)$ is

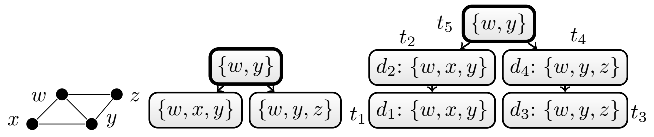

Figure 1. Primal graph P_Q of Q from Example 2 (left) with TDs $\mathcal{T}_1, \mathcal{T}_2$ of graph P_Q (right).

an arborescence with root $r \in N$, and χ is a mapping that assigns to each node $t \in N$ a set $\chi(t) \subseteq V$, called a *bag*, such that the following conditions hold: (i) $V = \bigcup_{t \in N} \chi(t)$ and $E \subseteq \bigcup_{t \in N} \{\{u, v\} \mid u, v \in \chi(t)\}$; and (ii) for each q, s, t, such that s lies on the path from q to t, we have $\chi(q) \cap \chi(t) \subseteq \chi(s)$. Then, $\mathsf{width}(\mathcal{T}) := \max_{t \in N} |\chi(t)| - 1$. The *treewidth* $\mathsf{tw}(G)$ of G is the minimum $\mathsf{width}(\mathcal{T})$ over all tree decompositions $\mathcal{T}$ of G. For arbitrary but fixed $w \geq 1$, it is feasible in linear time to decide if a graph has treewidth at most w and, if so, to compute a tree decomposition of width w [5]. Further, we call a tree decomposition $\mathcal{T} = (T, \chi)$ a *path decomposition (PD)* if $T = (N, \cdot, r)$ and $|\mathsf{cld}(t)| \leq 1$ for each node $t \in N$. Analogously, we define *pathwidth* $\mathsf{pw}(G)$ as the minimum $\mathsf{width}(\mathcal{T})$ over all path decompositions of G. Similarly, for $m \geq 2$, let *m-local pathwidth* of G refer to the pathwidth over all path decompositions of G, where each vertex in V occurs in at most m bags. For a given tree decomposition $\mathcal{T} = (T, \chi)$ with $T = (N, A, r)$, and an element $x \in \bigcup_{t \in N} \chi(t)$, we denote by $\mathcal{T}[x]$ the result $\mathcal{T}'$ of restricting $\mathcal{T}$ to nodes, whose bags contain x. Formally, $\mathcal{T}' := (T', \chi')$, where $T' := (N', A', r')$, $N' := \{t \mid t \in N, x \in \chi(t)\}$, $A' := A \cap (N' \times N')$, and for each $t \in N'$, $\chi'(t) = \chi(t)$. Finally, $r' \in N'$ is the first node reachable from r. The literature distinguishes so-called nice tree decompositions, which can be computed in linear time without increasing the width [40]. For our purposes, the following relaxed variant of almost nice tree decompositions suffices.

Definition 3. *Given an integer $c \in \mathbb{N}$. A tree decomposition $\mathcal{T} = (T, \chi)$, where $T = (N, \cdot, r)$, is called* almost c-nice, *if for each node $t \in N$ with $\mathsf{cld}(t) = \{t_1, \ldots, t_s\}$, the following conditions are true (i) $s \leq 2$ and (ii) $\left|\chi(t) \setminus \bigcup_{i=1}^{i=s} \chi(t_i)\right| \leq c$.*

In order to use tree decompositions for QBFs, we need a graph representation of Boolean formulas [55]. The *primal graph P_F* of a Boolean formula F in CNF or DNF has the variables $\mathsf{var}(F)$ of F as vertices and an edge $\{x, y\}$ if there exists a term or clause $f \in F$ such that $x, y \in \mathsf{var}(f)$, respectively. For a QBF Q, we identify its primal graph with the primal graph of its matrix, i.e., let $P_Q := P_{\mathsf{matrix}(Q)}$.

Example 4. *Figure 1 illustrates the primal graph P_Q of the QBF from Example 2 and two tree decompositions of P_Q of width 2. The graph P_Q has treewidth 2, since the vertices w,x,y are completely connected and hence width 2 is optimal [40].* ■

Definition 5. *Let $\mathcal{T} = (T, \chi)$ be a tree decomposition of a graph G. A* labeled tree decomposition (LTD) *$\mathcal{T}$ of a Boolean*

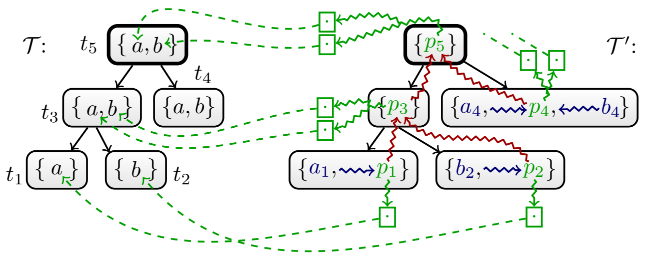

Figure 2. Simplified illustration of a certain tree decomposition $\mathcal{T}' = (T, \chi')$ of $P_{R(Q)}$ (yielded[3] by reduction R), and its relation to tree decomposition $\mathcal{T} = (T, \chi)$ of P_Q. Each bag $\chi'(t_i)$ of a node t_i of $\mathcal{T}'$ contains variable x_i for any variable x introduced in $\chi(t_i)$ and $\lceil \log(\text{width}(\mathcal{T})) \rceil$ many (green) pointer variables p_i selecting *one* variable in $\chi(t_i)$ of $\mathcal{T}$. Squiggly red arrows indicate the propagation between pointers p_i, p_j and ensure consistency. In particular, although truth values for variable a are "guessed" using a_1 and a_4 (and "propagated" via blue squiggly arrows to corresponding pointers p_1 and p_4, respectively), these red arrows ensure via pointers p_1, p_3, p_4, p_5 that truth values for a_1 and a_4 coincide.

formula F in CNF or DNF is a tuple $\mathcal{T} = (T, \chi, \delta)$ where (T, χ) is a tree decomposition of P_F with $T = (N, \cdot, r)$, and $\delta : N' \to F$ with $N' \subseteq N$ is a bijective mapping from TD nodes to clauses or terms in F such that for every $t \in N'$, $\text{var}(\delta(t)) \subseteq \chi(t)$.

Observation 6. *Given an almost c-nice tree decomposition $\mathcal{T}$ of a primal graph P_F for a given 3-CNF or 3-DNF formula F of width k. Then, one can easily create a labeled almost c-nice tree decomposition $\mathcal{T}'$ of P_F with a linear number of nodes in the number of nodes in $\mathcal{T}$ such that* $\text{width}(\mathcal{T}') = \text{width}(\mathcal{T})$.

Example 7. *Consider again Figure 1 (right). Observe that $\mathcal{T}_2$ is a labeled almost 3-nice tree decomposition of P_Q, where labeling function δ sets $\delta(t_i) = d_i$, for $1 \leq i \leq 4$.* ■

3 Decomposition-Guided Compression

Next, we present our approach to transform a given input instance of treewidth k into an instance of exponentially smaller treewidth ("compression") compared to the original treewidth k. Thereby, we trade the compression of the parameter for the cost of additional computation power required to solve the compressed instance. For the canonical QSat problem, we require an increased quantifier rank.

First, we introduce the reduction R that takes an instance Q of ℓ-QSat, and computes a corresponding tree decomposition $\mathcal{T}$ of the primal graph P_Q, where $\text{width}(\mathcal{T}) = \text{tw}(P_Q)$. Then, it returns a compressed instance $R(Q)$ of $(\ell + 1)$-QSat of treewidth $O(\log(\text{width}(\mathcal{T})))$. The reduction R, which is guided by TD $\mathcal{T}$, yields[3] a new *compressed tree decomposition $\mathcal{T}'$* of $P_{R(Q)}$ of width $O(\log(\text{width}(\mathcal{T})))$. For that it is crucial to balance introducing copies of variables (redundancy) and saving treewidth (structural dependency), such that, intuitively, we can still evaluate $R(Q)$ given the limitation of treewidth $O(\log(\text{width}(\mathcal{T})))$. To keep this balance, we can only analyze in a bag in $\mathcal{T}'$ a *constant* number of elements of the corresponding original bag of $\mathcal{T}$. Still, considering $\log(\text{width}(\mathcal{T}))$ many elements in a bag at once allows us to represent one "pointer" to address at most $\text{width}(\mathcal{T})$ many elements of each bag of $\mathcal{T}$ and, consequently, the restriction to $O(\log(\text{width}(\mathcal{T})))$ many elements in a bag at once enables *constantly* many such pointers. To give a first glance at the idea of the reduction R, Figure 2 provides an intuition and illustrates a tree decomposition $\mathcal{T}$ of P_Q together with a corresponding compressed tree decomposition $\mathcal{T}'$ of $P_{R(Q)}$, whose bags contain pointers to original bags of $\mathcal{T}$. Actually, we can encode the *propagation* of information from one bag of $\mathcal{T}'$ to its parent bag with the help of these pointers. Thereby, we ensure that information is consistent and this consistency can be preserved, even though we *guess* in R truth values for copies of the same variable in Q independently. Note that these "local" pointers for each bag are essential to achieve treewidth compression.

Below, we discuss the reduction R in more detail. Then, Section 3.2 provides a description of a general methodology for establishing lower bounds for problems parameterized by treewidth. In more detail, equipped with our lower bound results for QSat, we propose reductions from QSat as a general toolkit for proving lower bounds assuming ETH. Further, we discuss several showcases to illustrate this methodology.

3.1 The Reduction

The formula $R(Q)$ constructed by R mainly consists of three interacting parts. In the presentation, we refer to them as *guess*, *check*, and *propagate* part.

- "Guess" ($\mathcal{G}$): Contains clauses responsible for guessing truth values of variables occurring in the original QBF Q.
- "Check" ($\mathcal{CK}$): These clauses ensure that there is at least one 3-DNF term in Q that is satisfied, thereby maintaining 3 pointers for each node as discussed above.
- "Propagate" ($\mathcal{P}$): These clauses ensure consistency using a pointer for each node of the tree decomposition.

We commence with the formal description of R. Given a QBF Q of the form $Q := Q_1V_1.Q_2V_2.\cdots\forall V_\ell.D$, where D is in 3-DNF such that the quantifier blocks are alternating, i.e., quantifiers of quantifier blocks with even indices are equal, which are different from those of blocks with odd indices. Further, assume a labeled almost c-nice tree decomposition $\mathcal{T} = (T, \chi, \delta)$, where $T = (N, \cdot, \cdot)$ of the primal graph P_Q of D, which always exists by Observation 6. Notice that by Definition 5 for all terms $d \in D$, the inverse function $\delta^{-1}(d)$ is well-defined. Further, actually R can deal with open QBFs, i.e, QBF Q does not necessarily have to be closed.

[3]For the sake of readability, we defer the discussion of formal details on the construction of $\mathcal{T}'$ to the proof of Lemma 23.

Open formulas are needed later to simplify the correctness proof of Section 4.

We use the following sets of variables. Let $\mathit{NodeI}(x) := \{t \mid t \in N, x \in \chi(t) \setminus (\bigcup_{t_i \in \mathrm{cld}(t)} \chi(t_i))\}$ be the set of nodes, where a given element x is *introduced*. For a set $V \subseteq \mathrm{var}(D)$ of variables, we denote by $\mathit{VarI}(V) := \{x_t \mid x \in V, t \in \mathit{NodeI}(x)\}$ the set of fresh variables generated for each original variable x and node t, where x is introduced. Later, we need to distinguish whether the set V_i of variables is universally or existentially quantified. Universal quantification requires to shift for each $x \in V_i$ all but one representative of $\{x_t \mid t \in \mathit{NodeI}(x)\}$ to the next existential quantifier block Q_{i+1}. The representative variable that is not shifted is denoted by $\mathrm{rep}(x)$. In particular, given a quantifier block Q_2, its variables V_2 and the variables V_1 of the preceding quantifier block, we define: $\mathit{VarI}(Q_2, V_2, V_1) := \{x_t \mid x \in V_2, t \in \mathit{NodeI}(x), Q_2 = \exists\} \cup \{x_t \mid x \in V_2, t \in \mathit{NodeI}(x), x_t = \mathrm{rep}(x), Q_2 = \forall\} \cup \{x_t \mid x \in V_1, t \in \mathit{NodeI}(x), x_t \neq \mathrm{rep}(x), Q_2 = \exists\}$. We denote by $\mathit{VarSat} := \{sat_t, sat_{\leq t} \mid t \in N\}$ the set of fresh decision variables responsible for storing for each node $t \in N$ whether any term at t or at any node below t is satisfied, respectively. Finally, we denote by $\mathit{VarB} := \{b_t^0, \ldots, b_t^{\lceil \log(|\chi(t)|) \rceil - 1} \mid t \in N\}$, and $\mathit{VarBV} := \{v_t \mid t \in N\}$ the set of fresh variables for each node $t \in N$ that will be used to address particular elements of the corresponding bags (pointer as depicted in Figure 2 in binary representation), and to assign truth values for these elements, respectively. Overall, the variables in *VarB* allow us to guide the evaluation of formula D along the tree decomposition $\mathcal{T}$. For checking 3-DNF terms, we need the same functionality three more times, resulting in the sets $\mathit{VarB3} := \{b_{t,j}^0, \ldots, b_{t,j}^{\lceil \log(|\chi(t)|+1) \rceil - 1} \mid t \in N, 1 \leq j \leq 3\}$ that additionally may refer to a special fresh element *nil* (therefore the +1 in the exponent in definition of *VarB3*), and $\mathit{VarBV3} := \{v_{t,j} \mid t \in N, 1 \leq j \leq 3\}$ of fresh variables. Notice that the construction is designed in such a way that the focus lies only on certain elements of the bag (one at a time, and independent of other elements within the same bag). In the end, this ensures that the treewidth of our reduced instance is only logarithmic in the original treewidth of the primal graph of D. Reduction $R(Q)$ creates $Q' :=$

$$Q_1\ \mathit{VarI}(Q_1, V_1, \emptyset).\ Q_2\ \mathit{VarI}(Q_2, V_2, V_1). \cdots \forall\ \mathit{VarI}(\forall, V_\ell, V_{\ell-1}), \mathit{VarB}.$$
$$\exists\ \mathit{VarI}(\exists, \emptyset, V_\ell),\ \mathit{VarBV}, \mathit{VarBV3}, \mathit{VarB3}, \mathit{VarSat}.\ C,$$

where C is a CNF formula consisting of guess, check and propagate parts, i.e., sets $\mathcal{G}, \mathcal{CK}$, and $\mathcal{P}$ of clauses, respectively.

Example 8. *Consider again Q from Example 2. The resulting instance $R(Q)$ looks as follows assuming that $\mathrm{rep}(y) = y_{t_1}$, where C consists of a guess, check and, propagate part.*

$$\exists \underbrace{w_{t_1}, w_{t_3}, x_{t_1}.}_{\mathit{VarI}(\exists,\{w,x\},\emptyset)} \forall \underbrace{y_{t_1}, z_{t_3},}_{\mathit{VarI}(\forall,\{y,z\},\{w,x\})} \underbrace{b_{t_1}^0, b_{t_1}^1, b_{t_2}^0, \ldots, b_{t_4}^1, b_{t_5}^0.}_{\mathit{VarB}} \exists \underbrace{y_{t_3},}_{\mathit{VarI}(\exists,\emptyset,\{y,z\})}$$
$$\underbrace{v_{t_1}, \ldots, v_{t_5},}_{\mathit{VarBV}} \underbrace{v_{t_1,1}, v_{t_1,2}, v_{t_1,3}, v_{t_2,1}, \ldots, v_{t_5,3},}_{\mathit{VarBV3}}$$
$$\underbrace{b_{t_1,1}^0, b_{t_1,1}^1, b_{t_1,2}^0, \ldots, b_{t_5,3}^1,}_{\mathit{VarB3}} \underbrace{sat_{t_1}, \ldots, sat_{t_5}, sat_{\leq t_1}, \ldots, sat_{\leq t_5}.}_{\mathit{VarSat}} C$$

■

In the following, we define sets $\mathcal{G}, \mathcal{CK}$, and $\mathcal{P}$ of clauses. To this end, we require for the pointers a bit-vector (binary) representation of the elements in a bag of $\mathcal{T}$, and a mapping that assigns bag elements to its corresponding binary representation. In particular, we assume an arbitrary, but fixed total order $\prec$ of elements of a bag $\chi(t)$ of any given node $t \in N$. With $\prec$, we can then assign each element x in $\chi(t)$ its unique (within the bag) induced ordinal number $o(t, x)$. This ordinal number $o(t, x)$ is expressed in binary. For that we need precisely $\lceil \log(|\chi(t)|) \rceil$ many bit-variables $B := \{b_t^0, \ldots, b_t^{\lceil \log(|\chi(t)|) \rceil - 1}\}$. We denote by $[\![x]\!]_t$ the (consistent) set of literals over variables in B that encode (in binary) the ordinal number $o(t, x)$ of $x \in \chi(t)$ in t, such that whenever a literal b_t^i or $\neg b_t^i$ is contained in the set $[\![x]\!]_t$, the i-th bit in the unique binary representation of $o(t, x)$ is 1 or 0, respectively. Analogously, for $1 \leq j \leq 3$ we denote by $[\![x]\!]_{t,j}$ the (consistent) set of literals over variables in $B_j := \{b_{t,j}^0, \ldots, b_{t,j}^{\lceil \log(|\chi(t)|+1) \rceil - 1}\}$ that either binary-encode the ordinal number $o(t, x)$ of $x \in \chi(t)$ in t, or these literals binary-encode number $\max_{y \in \chi(t)} o(t, y) + 1$ for $x = \mathit{nil}$.

The guess part $\mathcal{G}$. The clauses in $\mathcal{G}$, which we denote as implications, are defined as follows.

$$x_t \wedge \bigwedge_{b \in [\![x]\!]_t} b \longrightarrow v_t \qquad \text{for each } x_t \in \mathit{VarI}(\mathrm{var}(D)) \qquad (1)$$

$$\neg x_t \wedge \bigwedge_{b \in [\![x]\!]_t} b \longrightarrow \neg v_t \qquad \text{for each } x_t \in \mathit{VarI}(\mathrm{var}(D)) \qquad (2)$$

Intuitively, this establishes that whenever a certain variable x_t for an introduced variable $x \in \chi(t)$ is assigned to true (false) and all the corresponding literals in $[\![x]\!]_t$ of the binary representation of $o(t, x)$ are satisfied (i.e, x is "selected"), then also $v_t \in \mathit{VarBV}$ of node t has to be set to true (false).

Analogously, set $\mathcal{G}$ further contains the following clauses:

$$x_t \wedge \bigwedge_{b \in [\![x]\!]_{t,j}} b \longrightarrow v_{t,j} \qquad \text{for each } x_t \in \mathit{VarI}(\mathrm{var}(D)), 1 \leq j \leq 3 \qquad (3)$$

$$\neg x_t \wedge \bigwedge_{b \in [\![x]\!]_{t,j}} b \longrightarrow \neg v_{t,j} \qquad \text{for each } x_t \in \mathit{VarI}(\mathrm{var}(D)), 1 \leq j \leq 3 \qquad (4)$$

Example 9. *Consider formula C from Example 8. Let $1 \leq j \leq 3$. Further, assume the following mapping of bag contents to bit-vector assignments. For any variable $a \in \mathrm{var}(D)$ with $t \in \mathit{NodeI}(a)$ and for $a = \mathit{nil}$ with $t \in N$, we arbitrarily fix the total ordering $\prec$ and have $[\![a]\!]_t$ and $[\![a]\!]_{t,j}$ as follows.*

a	$t \in \{t_1, t_2\}$		$t \in \{t_3, t_4\}$		$t = t_5$	
	$[\![a]\!]_t$	$[\![a]\!]_{t,j}$	$[\![a]\!]_t$	$[\![a]\!]_{t,j}$	$[\![a]\!]_t$	$[\![a]\!]_{t,j}$
w	$\{\neg b_t^0, \neg b_t^1\}$	$\{\neg b_{t,j}^0, \neg b_{t,j}^1\}$	$\{\neg b_t^0, \neg b_t^1\}$	$\{\neg b_{t,j}^0, \neg b_{t,j}^1\}$	$\{\neg b_t^0\}$	$\{\neg b_{t,j}^0, \neg b_{t,j}^1\}$
x	$\{\neg b_t^0, b_t^1\}$	$\{\neg b_{t,j}^0, b_{t,j}^1\}$	-		-	
y	$\{b_t^0, \neg b_t^1\}$	$\{b_{t,j}^0, \neg b_{t,j}^1\}$	$\{\neg b_t^0, b_t^1\}$	$\{\neg b_{t,j}^0, b_{t,j}^1\}$	$\{b_t^0\}$	$\{\neg b_{t,j}^0, b_{t,j}^1\}$
z	-		$\{b_t^0, \neg b_t^1\}$	$\{b_{t,j}^0, \neg b_{t,j}^1\}$	-	
nil	-	$\{b_{t,j}^0, b_{t,j}^1\}$	-	$\{b_{t,j}^0, b_{t,j}^1\}$	-	$\{b_{t,j}^0, \neg b_{t,j}^1\}$

The guess part of C contains for example for variable $w \in \mathrm{var}(D)$ the following clauses.

$$w_{t_1} \wedge \neg b_{t_1}^0 \wedge \neg b_{t_1}^1 \longrightarrow v_{t_1}, \quad \neg w_{t_1} \wedge \neg b_{t_1}^0 \wedge \neg b_{t_1}^1 \longrightarrow \neg v_{t_1},$$

$$w_{t_3} \wedge \neg b^0_{t_3} \wedge \neg b^1_{t_3} \longrightarrow v_{t_3}, \quad \neg w_{t_3} \wedge \neg b^0_{t_3} \wedge \neg b^1_{t_3} \longrightarrow \neg v_{t_3}.$$

Thereby, whenever we guess a certain truth value for w_{t_1} (w_{t_3}) it is ensured that there is a certain bit-vector, namely $[\![w]\!]_{t_1}$ ($[\![w]\!]_{t_3}$) such that v_{t_1} (v_{t_3}) has to be set to the same truth value. Analogously, clauses of the form (3) and (4) are in $\mathcal{G}$. ■

The check part $\mathcal{CK}$. In the following, we assume an arbitrary, but fixed total order of the (three) literals of each (3-DNF) term $d \in D$. We refer to the first, second, and third literal of d by $\mathrm{tlit}(d, 1)$, $\mathrm{tlit}(d, 2)$, and $\mathrm{tlit}(d, 3)$, respectively. Analogously, $\mathrm{tvar}(d, 1)$, $\mathrm{tvar}(d, 2)$, and $\mathrm{tvar}(d, 3)$ refers to the variable of the first, second, and third literal, respectively. Further, for a given term $d \in D$ and $1 \leq j \leq 3$, let $\mathrm{bv}(d, t, j)$ denote $v_{t,j}$ if $\mathrm{tlit}(d, j)$ is a variable, and $\neg v_{t,j}$ otherwise. Set $\mathcal{CK}$ contains the following clauses:

$$sat_{\leq t} \longrightarrow sat_{\leq t_1} \vee \cdots \vee sat_{\leq t_s} \vee sat_t \quad \text{for each } t \in N, \ \mathrm{cld}(t) = \{t_1, \ldots, t_s\} \tag{5}$$

Informally speaking, for any node t this ensures the propagation of whether we satisfied at least one term directly in node t, or in any descendant of t.

In order to check whether a particular term is satisfied, we add for each term $d \in D$ clauses encoding the implication $sat_{\delta^{-1}(d)} \longrightarrow \bigwedge_{1 \leq j \leq 3}\Big[\bigwedge_{b \in [\![\mathrm{tvar}(d,j)]\!]_{\delta^{-1}(d),j}} b \wedge \mathrm{bv}(d, \delta^{-1}(d), j)\Big]$ as follows:

$$sat_{\delta^{-1}(d)} \longrightarrow b \quad \text{for each } d \in D, 1 \leq j \leq 3, \ b \in [\![\mathrm{tvar}(d, j)]\!]_{\delta^{-1}(d),j} \tag{6}$$

$$sat_{\delta^{-1}(d)} \longrightarrow \mathrm{bv}(d, \delta^{-1}(d), j) \quad \text{for each } d \in D, 1 \leq j \leq 3 \tag{7}$$

Finally, we add $sat_{\leq r}$ for root r, and $\neg sat_t$ for each node t in $N \setminus \bigcup_{d \in D}\{\delta^{-1}(d)\}$ since these nodes are not used for checking satisfiability of any term.

$$sat_{\leq r} \tag{8}$$

$$\neg sat_t \quad \text{for each } t \in N \setminus \bigcup_{d \in D}\{\delta^{-1}(d)\} \tag{9}$$

Example 10. *Consider again formula C from Example 8. We discuss clauses of the check part for node $t_2 = \delta^{-1}(d_2)$ and root node t_5. Thereby, we encode satisfiability of term $d_2 = \neg w \wedge \neg x \wedge y$ assuming $\mathrm{tlit}(d_2, 1) = \neg w$, $\mathrm{tlit}(d_2, 2) = \neg x$, and $\mathrm{tlit}(d_2, 3) = y$.*

$$sat_{\leq t_2} \longrightarrow sat_{\leq t_1} \vee sat_{t_2},$$

$$sat_{t_2} \longrightarrow \neg b^0_{t_2,1}, \quad sat_{t_2} \longrightarrow \neg b^1_{t_2,1}, \quad sat_{t_2} \longrightarrow \neg v_{t_2,1},$$

$$sat_{t_2} \longrightarrow \neg b^0_{t_2,2}, \quad sat_{t_2} \longrightarrow b^1_{t_2,2}, \quad sat_{t_2} \longrightarrow \neg v_{t_2,2},$$

$$sat_{t_2} \longrightarrow b^0_{t_2,3}, \quad sat_{t_2} \longrightarrow \neg b^1_{t_2,3}, \quad sat_{t_2} \longrightarrow v_{t_2,3},$$

$$sat_{\leq t_5} \longrightarrow sat_{\leq t_2} \vee sat_{\leq t_4} \vee sat_{t_5}, \quad sat_{\leq t_5}, \quad \neg sat_{t_5}$$ ■

The propagate part $\mathcal{P}$. The sets $\mathcal{G}$ and $\mathcal{CK}$ contain clauses responsible for guessing truth values and checking that at least one term of the original formula D is satisfied accordingly. In particular, the guess of truth values for $\mathrm{var}(D)$ happens at different tree decomposition nodes "independently", whereas checking whether at least one term $d \in D$ is satisfied is achieved in exactly one tree decomposition node $\delta^{-1}(d)$. Intuitively, in order to ensure that these independent guesses of truth values for $\mathrm{var}(D)$, are consistent, clauses in $\mathcal{P}$ make use of the connectedness condition of TDs in order to guide the comparison of these independent guesses along the TD. More precisely, for each tree decomposition node $t \in N$, every node $t_i \in \mathrm{cld}(t)$, and every variable $x \in \chi(t) \cap \chi(t_i)$ that both nodes t and t_i have in common, the set $\mathcal{P}$ contains clauses:

$$v_t \wedge \bigwedge_{b \in [\![x]\!]_t} b \wedge \bigwedge_{b \in [\![x]\!]_{t_i}} b \longrightarrow v_{t_i} \quad \text{for each } t \in N, t_i \in \mathrm{cld}(t), \ x \in \chi(t) \cap \chi(t_i) \tag{10}$$

$$\neg v_t \wedge \bigwedge_{b \in [\![x]\!]_t} b \wedge \bigwedge_{b \in [\![x]\!]_{t_i}} b \longrightarrow \neg v_{t_i} \quad \text{for each } t \in N, t_i \in \mathrm{cld}(t), \ x \in \chi(t) \cap \chi(t_i) \tag{11}$$

Further, for each clause $d \in D$, every node t_i in $\mathrm{cld}(\delta^{-1}(d))$, and $1 \leq j \leq 3$ such that $\mathrm{tvar}(d, j) \in \chi(t_i)$, set $\mathcal{P}$ contains:

$$\bigwedge_{b' \in [\![\mathrm{tvar}(d,j)]\!]_{t,j}} b' \longrightarrow b \quad \text{for each } d \in D \text{ with } 1 \leq j \leq 3, \ t = \delta^{-1}(d), t_i \in \mathrm{cld}(t), \mathrm{tvar}(d, j) \in \chi(t_i), \ b \in [\![\mathrm{tvar}(d, j)]\!]_{t_i,j} \tag{12}$$

$$v_{t,j} \longleftrightarrow v_{t_i,j} \quad \text{for each } t \in N, t_i \in \mathrm{cld}(t), 1 \leq j \leq 3 \tag{13}$$

Vaguely speaking, this construction ensures that whenever a bag element (using *VarB3*) or a truth value (using *VarBV3*) is "selected" in node t, we also have to select the same (if exists) below in children of t.

Example 11. *Consider once more C from Example 8. We illustrate the propagate part for node $t_4 = \delta^{-1}(d_4)$ and variable w assuming that $w = \mathrm{tvar}(d_4, 1)$. Observe that $d_4 = w \wedge y \wedge z$, and $w \in \chi(t_4) \cap \chi(t_3)$.*

$$v_{t_4} \wedge \underbrace{\neg b^0_{t_4} \wedge \neg b^1_{t_4}}_{[\![w]\!]_{t_4}} \wedge \underbrace{\neg b^0_{t_3} \wedge \neg b^1_{t_3}}_{[\![w]\!]_{t_3}} \longrightarrow v_{t_3},$$

$$\neg v_{t_4} \wedge \underbrace{\neg b^0_{t_4} \wedge \neg b^1_{t_4}}_{[\![w]\!]_{t_4}} \wedge \underbrace{\neg b^0_{t_3} \wedge \neg b^1_{t_3}}_{[\![w]\!]_{t_3}} \longrightarrow \neg v_{t_3},$$

$$\neg b^0_{t_3,1} \wedge \neg b^1_{t_3,1} \longrightarrow \neg b^0_{t_4,1}, \quad \neg b^0_{t_3,1} \wedge \neg b^1_{t_3,1} \longrightarrow \neg b^1_{t_4,1},$$

$$v_{t_4,1} \longleftrightarrow v_{t_3,1}, \quad v_{t_4,2} \longleftrightarrow v_{t_3,2}, \quad v_{t_4,3} \longleftrightarrow v_{t_3,3}$$ ■

Remark 12. *Recalling Figure 2, we would like to highlight the relation between elements of the figure and variables or clauses of reduction R introduced above. Blue elements a_1, b_2, a_4, b_4 represent "introduce variables" VarI($\mathrm{var}(D)$) and the blue squiggly arrows visualize the guess part $\mathcal{G}$. Green elements p_1, p_2, p_3, p_4, p_5 represent "pointer variables" VarB and VarB3 and the green squiggly arrows point to elements of tree decomposition $\mathcal{T}$. Finally, red squiggly arrows visualize the propagate part $\mathcal{P}$. (The check part $\mathcal{CK}$ is not explicitly visualized.)*

Converting C to 3-CNF formula C'. Observe that by similar arguments (cf., [42]) one can transform using an additional reduction R' the CNF formula C of the QBF $R(Q)$ into 3-CNF, resulting in $Q'' = R'(R(Q))$ such that $\mathrm{tw}(P_{Q''}) \leq \mathrm{tw}(P_{R(Q)}) + 2$. To this end, one has to perform the following standard reduction (cf., [42]): As long as there exists a clause $c \in C$ consisting of more than 3 literals, we introduce a fresh existentially quantified variable v, remove c from C and replace it with two new clauses. The first new clause contains v and two literals of c, while the second clause contains $\neg v$ and the remaining literals of c. Note that this standard reduction R' does not affect satisfiability, and can be done such that it causes only constant increase of the treewidth (cf., Lemma 23 and [42]). Observe that by construction the same argument actually holds for pathwidth.

3.2 Methodology for Lower Bounds

The reduction discussed in the previous subsection allows us to establish our main result, which is the following theorem.

Theorem 13 (QBF lower bound)**.** *Given an arbitrary QBF of the form $Q = Q_1V_1.Q_2V_2.Q_3V_3 \cdots Q_\ell V_\ell.F$ where $\ell \geq 1$, and F is a 3-CNF formula (if $Q_\ell = \exists$), or F is a 3-DNF formula (if $Q_\ell = \forall$). Then, unless ETH fails, Q cannot be solved in time* $\mathsf{tow}(\ell, o(k)) \cdot \mathsf{poly}(|\mathsf{var}(F)|)$*, where k is the treewidth of the primal graph P_Q.*

In the following, we first use this theorem to establish a full methodology to obtain lower bound results for bounded treewidth and then provide a proof for the theorem in the next section. The result for $\ell = 2$ (cf., [42]) has already been applied as a strategy to show lower bound results for problems in artificial intelligence, as for example abstract argumentation, abduction, circumscription, and projected model counting, that are hard for the second level of the polynomial hierarchy when parameterized by treewidth [25, 27, 41]. With the generalization to an *arbitrary* quantifier rank in Theorem 13, one can obtain lower bounds for variants of these problems and even more general problems on the third level or higher levels of the polynomial hierarchy.

Methodology. This motivates our methodology to show lower bounds for problems parameterized by treewidth. To this end, we make use of a stricter notion of fpt-reductions, which *linearly preserves* the parameter. Given functions $f, g : \mathbb{N} \to \mathbb{N}$, where g is linear, and an f-bounded fpt-reduction r using g. Then, we call r an f-bounded *fptl-reduction* using g. Next, we discuss the methodology for proving lower bounds of a problem P for treewidth consisting of the following.

1. **Graph Representation:** Pick a graph representation $G(I)$ for a given instance I of problem P.
2. **Quantifier Rank:** Fix a quantifier rank ℓ such that there is a function $f : \mathbb{N} \to \mathbb{N}$ with $f(k) \in O(\mathsf{tow}(\ell, k))$ and aim for establishing lower bound $\mathsf{tow}(\ell, \Omega(k)) \cdot \mathsf{poly}(\|I\|)$.
3. **Establish Reduction:** Establish an f-bounded fptl-reduction from an arbitrary QBF Q of quantifier rank ℓ parameterized by treewidth of the primal graph of Q to an instance I of P parameterized by treewidth as well.
4. **Conclude lower bound:** Then, by applying Theorem 13 conclude that unless ETH fails, an arbitrary instance I of problem P cannot be solved in time $\mathsf{tow}(\ell, o(k)) \cdot \mathsf{poly}(\|I\|)$ where $k = \mathsf{tw}(G(I))$.

We can generalize this to "non-canonical" lower bounds. To this end, one aims in Step 2 for a lower bound of the form $\mathsf{tow}\left(\ell, \Omega(g^{-1}(k)\right) \cdot \mathsf{poly}(\|I\|)$ for some function $g : \mathbb{N} \to \mathbb{N}$ such that g^{-1} is well-defined, and $f(k) \in O\left(\mathsf{tow}(\ell, g^{-1}(k))\right)$. Then, in Step 3 one needs to establish an f-bounded fpt-reduction using g accordingly, in order to conclude in Step 4 that under ETH an arbitrary instance I of P cannot be solved in time $\mathsf{tow}(\ell, o(g^{-1}(k))) \cdot \mathsf{poly}(\|I\|)$, where $k = \mathsf{tw}(G(I))$.

With the help of this methodology one can show lower bounds $f(k)$ for certain problems P, parameterized by treewidth, by reducing from the canonical ℓ-QSat problem parameterized by treewidth k as well. Thus, one avoids directly using ETH via tedious reductions from Sat, which involves problem-tailored constructions of instances of P whose treewidth is ℓ-fold logarithmic in the number of variables or clauses of the given Sat formula.

Note that the methodology naturally extends to pathwidth, since the result of Theorem 13 easily extends to (2-local) pathwidth by construction of our reduction R, which works for *any* tree decomposition including the special case of path decompositions. Formal details will be provided in Section 4 on correctness in Corollary 27, followed by further consequences of Theorem 13.

Showcases. Table 1 gives a brief overview of selected problems and their respective runtime lower bounds under ETH. Then, the proof of Theorem 14 below serves as an example for applying the methodology, showing that Theorem 13 also allows for quite general results on projection. Note that this bounds are tight under ETH.

Theorem 14. *Given an open QBF of the form $Q = Q_1V_1.Q_2V_2.Q_3V_3 \cdots Q_\ell V_\ell.F$ where $\ell \geq 1$, and F is a 3-CNF formula (if $Q_\ell = \exists$), or a 3-DNF formula (if $Q_\ell = \forall$). Then, under ETH, PQSat is indeed harder than deciding validity of $Q[\iota]$ for any assignment $\iota :$* $\mathsf{fvar}(Q) \to \{0, 1\}$. *In particular, assuming ETH, PQSat cannot be solved in time* $\mathsf{tow}(\ell+1, o(k)) \cdot \mathsf{poly}(|\mathsf{var}(F)|)$*, where k is the pathwidth of the primal graph P_Q.*

Proof. Assume towards a contradiction that under ETH one can solve projected model counting of Q in time $\mathsf{tow}(\ell + 1, o(k)) \cdot \mathsf{poly}(|\mathsf{var}(F)|)$. In the following, we define an fptl-reduction r from QSat to the decision variant PQSat-at-least-u of PQSat, where a given open QBF Q is a yes instance if and only if the solution (count) to PQSat of Q is at least u.

In particular, we transform a closed QBF $Q' = Q_0V_0.Q_1V_1.Q_2V_2.Q_3V_3 \cdots Q_\ell V_\ell.F$, where k is the pathwidth of P_Q to an instance $Q = Q_1.V_1.Q_2V_2.Q_3V_3 \cdots Q_\ell V_\ell.F$ of PQSat-at-least-u, where $\mathsf{fvar}(Q) = V_0$, and we set $u := 1$ if $Q_0 = \exists$ and $u := 2^{|V_0|}$, otherwise. The reduction is indeed correct, since Q' is a yes-instance of QSat if and only if $Q = r(Q')$ is a yes-instance of PQSat-at-least-u. Then, one can solve Q' of quantifier rank $\ell + 1$ in time $\mathsf{tow}(\ell + 1, o(k)) \cdot \mathsf{poly}(|\mathsf{var}(F)|)$, which contradicts Theorem 13 and Corollary 27. □

Corollary 15. *Assuming ETH, an instance Q of the problem $\#\Sigma_\ell$Sat or $\#\Pi_\ell$Sat cannot be solved in time* $\mathsf{tow}(\ell+1, o(k)) \cdot \mathsf{poly}(|\mathsf{var}(\mathsf{matrix}(Q))|)$*, where k is the pathwidth of P_Q.*

Problem P	LB $\mathrm{tow}(i, \Omega(k)) \cdot \mathrm{poly}(\|I\|)$					
	i=1	i=2	i=3	i=4	i=5	i=ℓ
Min Vertex Cover / Dominating Set [32]	$\triangledown$[37]					
Max Indep. Set, Hamilt. Cycle [32]	$\triangledown$[37]					
3-Colorability, Sat, #Sat [32, 55]	$\triangledown$[37]					
Circumscription, PAP [41]		$\triangledown_t$[41] ▼				
MUS [41]		$\triangledown_t$[41] ▼				
Skep$_{\mathrm{pref}}$, Skep$_{\mathrm{semi\text{-}st}}$, Cred$_{\mathrm{semi\text{-}st}}$ [26]		$\triangledown_t$[26] ▼				
PMC [27]		$\triangledown_t$[27] ▼				
ASPCons, #ASP [25, 38]		$\triangledown_t$[25] ▼				
k-Choosability [45], $k \geq 3$		$\triangledown$[45] ▼				
k-Choosability Deletion [45], $k \geq 4$			$\triangledown$[45]			
#PPAP			▼			
PASP [2, 25]			▼[25]			
#PCred$_{\mathcal{S}}$ [26], $\mathcal{S} \in \{$pref, semi-st, stage$\}$			▼[26]			
Candidate World View Check [56]			▼			
World View Check [56]				▼		
#Projected Guesses to World Views					▼	
ℓ-QSat, #ℓ-QSat, $\ell \geq 1$ [9]						▼
PQSat [17]: #$\Sigma_{\ell-1}$Sat, #$\Pi_{\ell-1}$Sat, $\ell \geq 2$						▼

Table 1. Runtime lower bounds (under ETH) for selected problems, where I denotes an instance of problem P and k refers to the treewidth ("$\triangledown_t$"), pathwidth ("$\triangledown$"), or 2-local pathwidth ("▼") of the corresponding (primal) graph of I. Results known from the literature are marked by "$\triangledown_t$" and "$\triangledown$". By "▼", we indicate that the result holds due to lower bound advancements and the methodology described in this paper. We obtain results for "▼", with known lower bound ("$\triangledown_t$", "$\triangledown$"), by the existing lower bound proof together with our methodology for 2-local pathwidth. Bounds are asymptotically tight unter ETH; corresponding upper bounds (e.g.,[4, 9, 25, 26, 35, 38, 41, 45, 55]) are out of scope. For definitions, we refer to the problem compendium in an online self-archived version.

Next, we provide further examples (listed in Table 1) of the applicability of our methodology. We provide brief definitions of the problems discussed below in an online self-archived version.

Proposition 16 (cf., [25]). *Unless ETH fails, PASP for given ASP program Π and a set $P \subseteq \mathrm{var}(\Pi)$ of projection variables cannot be solved in time $\mathrm{tow}(3, o(k)) \cdot \mathrm{poly}(|\Pi|)$, where k is the pathwidth of the primal graph[4] of Π.*

Proof (Idea). Fptl-reduction from $\forall\exists\forall$-Sat to PASP, both parameterized by the pathwidth of its primal graph. □

Proposition 17 (cf., [26]). *Let $\mathcal{S} \in \{$pref, semi-st, stage$\}$ and F be an argumentation framework. Unless ETH fails, we cannot solve the problem #PCred$_{\mathcal{S}}$ in time $\mathrm{tow}(3, o(k)) \cdot \mathrm{poly}(\|F\|)$ where k is the pathwidth of F (underlying graph).*

Proof (Idea). Fptl-reduction from $\forall\exists\forall$-Sat parameterized by pathwidth of primal graph, to #PCred$_{SEM}$ (parameterized by pathwidth of the underlying graph). □

Theorem 18. *Unless ETH fails, #PPAP for given instance (T, H, M) and set P of projection variables cannot be solved in time $\mathrm{tow}(3, o(k)) \cdot \mathrm{poly}(|\mathrm{var}(T)|)$, where k is the pathwidth of the primal graph of T.*

[4]For a definition of the primal graph of a program, we refer to [38].

Proof (Idea). Fptl-reduction from $\forall\exists\forall$-Sat to the decision variant of #PPAP, either from scratch or by lifting existing reduction [41] from $\exists\forall$-Sat. □

Theorem 19. *Given an epistemic program Π and a variable $a \in \mathrm{var}(\Pi)$. Then, unless ETH fails, deciding the problem Candidate World View Check cannot be solved in time $\mathrm{tow}(3, o(k)) \cdot \mathrm{poly}(|\Pi|)$, and the problem World View Check for a cannot be solved in time $\mathrm{tow}(4, o(k)) \cdot \mathrm{poly}(|\Pi|)$, where k is the pathwidth of the primal graph of Π.*

Proof (Idea). Fptl-reduction from $\exists\forall\exists$-Sat, or $\exists\forall\exists\forall$-Sat (parameterized by pathwidth of primal graph), respectively. Actually the reductions from the literature for showing Σ_3^P-hardness and Σ_4^P-hardness [56] form fptl-reductions. □

Note that our work focuses on lower bounds. However, the corresponding upper bounds for treewidth can be established by reductions to ℓ-QSat to obtain asymptotically tight results under ETH, see Table 1.

4 Correctness, Compression and Runtime

In the following, we show correctness and properties of our reduction presented in Section 3.1. Therefore, we assume a given QBF $Q := Q_1V_1.Q_2V_2.\cdots\forall V_\ell.D$, where D is in 3-DNF. Further, let $\mathcal{T} = (T, \chi, \delta)$ such that $T = (N, \cdot, r)$ be a labeled almost c-nice tree decomposition of primal graph P_Q of width k. The reduced instance is addressed by $R(Q)$, where reduction R is defined as in Section 3.1. The resulting QBF of quantifier rank $\ell + 1$ is referred to by $R'(R(Q))$ and its matrix in 3-CNF is given by $C = \mathrm{matrix}(R'(R(Q)))$.

To simplify presentation, we introduce the following definitions. Let $d \in D$ be a term, $t \in N$ be a node of the tree decomposition, and $1 \leq j \leq 3$, then $\mathrm{bit\text{-}term}(d, t) := \bigcup_{1 \leq j \leq 3}[[\![\mathrm{tvar}(d, j)]\!]_{t,j} \cup \{\mathrm{bv}(d, t, j)\}]$. Further, given an assignment $\alpha : \mathrm{var}(D) \rightarrow \{0, 1\}$, we define a function $\mathrm{local}(\cdot)$ that produces new assignments to copies of the variables, therefore let $\mathrm{local}(\alpha) := \{x_t \mapsto \alpha(x) \mid x \in \mathrm{dom}(\alpha), t \in \mathit{NodeI}(x)\}$ denote the *matching* assignment of the corresponding guess variables. Further, for a set $\mathcal{S}$ of literals and an assignment ι, we say assignment ι *respects* $\mathcal{S}$, if $(\bigwedge_{l \in \mathcal{S}} l)[\iota]$ evaluates to true.

Correctness. Next, we establish correctness of reduction R.

Lemma 20. *Let κ be any assignment of at least two variables $x_t, x_{t'} \in \mathit{VarI}(\mathrm{var}(D))$ such that $\kappa(x_t) \neq \kappa(x_{t'})$ for nodes $t, t' \in \mathit{NodeI}(x)$ with $x \in \mathrm{var}(D)$. Then, $R(Q)[\kappa]$ is invalid.*

Proof. We construct an assignment κ' which extends κ and sets certain variables in VarB. Then, we show that $R(Q)[\kappa']$ is invalid, which suffices since VarB is universally quantified. The construction of κ' is as follows: for every $b \in [\![x]\!]_{t''}$ and every $t'' \in N$ where $x \in \chi(t'')$, we set $\kappa'(b) := 1$, if b is a variable; and $\kappa'(b) := 0$, otherwise. Assume towards contradiction that there is an assignment $\kappa'' : \mathit{VarI}(\mathrm{var}(D)) \cup \mathit{VarB} \cup \mathit{VarBV} \rightarrow \{0, 1\}$ that extends κ' such that $R(Q)[\kappa'']$ is valid. In particular, the assignment κ'' sets variables in *VarBV* such that every clause in the assigned variables of $R(Q)$, in particular, parts $\mathcal{G}$ and $\mathcal{P}$, is valid under the assignment κ''. By Condition (ii) of the definition of a tree decomposition (connectedness), $\mathcal{T}[x]$ induces a connected tree as well. In consequence, irrelevant of how κ'' assigns variable $v_{r'}$ for the root node r' of $\mathcal{T}[x]$, the clauses in Formulas (10) and (11) enforce that exactly the same truth value $v = \kappa''(v_n)$ has to be set for any node $n \in \mathit{NodeI}(x)$. Then, $\kappa''(v_t) = v$ and $\kappa''(v_{t'}) = v$ holds. By part $\mathcal{G}$ of our reduction, more precisely, Formulas (1) and (2), we conclude that both $\kappa''(x_t) = v$ and $\kappa''(x_{t'}) = v$, which contradicts that $\kappa(x_t) \neq \kappa(x_{t'})$. □

Lemma 21. *Given an assignment $\iota : \mathit{VarI}(\mathrm{var}(D)) \cup \mathit{VarB} \cup \mathit{VarBV} \rightarrow \{0,1\}$. Then, for any assignment $\kappa : \mathit{VarI}(\mathrm{var}(D)) \cup \mathit{VarB} \cup \mathit{VarBV} \cup \mathit{VarB3} \cup \mathit{VarBV3} \rightarrow \{0,1\}$ that extends ι, $R(Q)[\kappa]$ is invalid, if (a) there is no term $d_i \in D$ with $t = \delta^{-1}(d_i)$ such that κ respects* bit-term(d_i, t)*. Now assume that there is $d_i \in D$ with κ respecting* bit-term$(d_i, \delta^{-1}(d_i))$*, then $R(Q)[\kappa]$ is also invalid, if (b) $\kappa(v_{t,j}) \neq \kappa(x_{t'})$, where $x =$* tvar(d_i, j) *for some $1 \leq j \leq 3$ and $t' \in \mathit{NodeI}(x)$.*

Proof. Assume towards a contradiction that (a) is not the case, i.e., there is no $d_i \in D$ such that κ respects bit-term(d_i, t) for $t = \delta^{-1}(d_i)$ and still $R(Q)[\kappa]$ is valid. Observe that by $R(Q)$, in particular, by construction of the check part $\mathcal{CK}$ of R, $\kappa(sat_{\leq r}) = 1$ by (8) and therefore $\kappa(sat_t) = 1$ by (5) for at least one node $t \in N$ has to be set in κ. This, however, implies by (9) that $t = \delta^{-1}(d_i)$ for some $d_i \in D$. In consequence, by construction of (6) and (7), κ respects bit-term(d_i, t), where $t = \delta^{-1}(d_i)$, contradicting the assumption.

Towards contradicting (b), assume that there is $d_i \in D$ with $t = \delta^{-1}(d_i)$ and $x =$ tvar(d_i, j) as well as $t' \in \mathit{NodeI}(x)$ such that $\kappa(v_{t,j}) \neq \kappa(x_{t'})$ and still $R(Q)[\kappa]$ is valid. Observe that for any two nodes $t'', t''' \in \mathcal{T}[x]$, κ respects $[\![x]\!]_{t'',j}$ and $[\![x]\!]_{t''',j}$ by (12) and connectedness of $\mathcal{T}[x]$. Further, for any $t'', t''' \in \mathcal{T}[x]$, $\kappa(v_{t'',j}) = \kappa(v_{t''',j})$ by (13). Then, since (3) and (4) ensure that $\kappa(x_{t'}) = \kappa(v_{t',j})$, ultimately by connectedness of $\mathcal{T}[x]$, $\kappa(v_{t,j}) = \kappa(x_{t'})$ holds. □

Theorem 22 (Correctness)**.** *Let Q be a QBF of the form $Q = Q_1 V_1.Q_2 V_2. \cdots \forall V_\ell.D$ where D is in DNF. Then, for any assignment α :* fvar$(Q) \rightarrow \{0,1\}$*, we have $Q[\alpha]$ is valid if and only if $R(Q)[\alpha']$ is valid, where assignment:* fvar$(R(Q)) \rightarrow \{0,1\}$ *is such that $\alpha' =$* local(α).

Proof. Let $\mathcal{T} = (T, \chi, \delta)$ be the labeled tree decomposition that is computed when constructing R, where $T = (N, A, r)$. We proceed by induction on the quantifier rank ℓ.
Base case. Assume $\ell = 1$.

"$\Longrightarrow$": Let α be an assignment to the free variables of Q for which $Q[\alpha]$ is valid. Further, let $\alpha' :=$ local(α). We show that $R(Q)[\alpha']$ is valid as well. Let therefore ι be an arbitrarily chosen assignment to the variables in V_1. Since $\ell = 1$, we have $Q_1 = \forall$. We define an assignment $\kappa : \mathit{VarI}(\forall, V_1, \emptyset) \rightarrow \{0,1\}$ such that $\kappa(x_t) := \iota(x)$ for every $x_t \in \mathit{VarI}(\forall, V_1, \emptyset)$ with $t \in N$ and $x \in \mathrm{var}(D)$. Next, we define an assignment $\kappa' : \mathit{VarI}(\forall, V_1, \emptyset) \cup \mathit{VarI}(\exists, \emptyset, V_1) \rightarrow \{0,1\}$ that extends κ and sets $\kappa'(x_{t'}) := \iota(x)$ for every $x_{t'} \in \mathit{VarI}(\exists, \emptyset, V_1)$ with $t' \in N$. Assignment κ' has by construction the same truth value for each of the copies x_t of x, which is needed for $R(Q)[\alpha' \cup \kappa]$ to be valid in order to not contradict Lemma 20.

Then, we construct an assignment κ'', which extends κ' by the variables in *VarB3*, *VarBV3*, and *VarSat*. By construction of ι and since $Q[\alpha]$ is valid, $Q[\alpha \cup \iota]$ is valid, which is the same as $D[\alpha \cup \iota]$ is valid. In consequence, as D is in DNF, there is at least one term $d \in D$ such that $d[\alpha \cup \iota]$ is valid. Depending on the term d, we assign the variables in *VarB3*, *VarBV3*, and *VarSat* with assignment κ''. By Definition 5, there is a unique node $t = \delta^{-1}(d)$ in the labeled tree decomposition for the term d. Then, we set $\kappa''(sat_{\leq t}) := \kappa''(sat_t) := 1$. For every ancestor t' of $t \in N$, we assign $\kappa''(sat_{\leq t'}) := 1$. For every node $s \in N$ that is not an ancestor of t, we set $\kappa''(sat_{\leq s}) := 0$. Finally, for every node u, where $u \neq t$, we set $\kappa''(sat_u) := 0$. For every node $t \in N$ and $1 \leq j \leq 3$ with tvar$(d, j) \notin \chi(t)$, we set κ'' such that it respects $[\![nil]\!]_{t,j} \cup \{bv(d, t, j)\}$. Finally, for every node t and $1 \leq j \leq 3$ with tvar$(d, j) \in \chi(t)$, we set κ'' such that it respects bit-term(d, t).

It remains to prove that for every assignment $\beta : \mathit{VarB} \rightarrow \{0,1\}$, there is an assignment $\zeta : \mathit{VarBV} \rightarrow \{0,1\}$ for which $R(Q)[\alpha' \cup \kappa'' \cup \beta \cup \zeta]$ is valid. For every variable $x \in \chi(t)$, if for every node $t \in N$, assignment β respects $[\![x]\!]_t$, then we set $\zeta(v_t) := (\alpha \cup \iota)(x)$. Otherwise, $\zeta(v_t) := 0$, since we can assign any truth value here. By construction of α' and κ'', clauses in Formulas (3) and (4) are satisfied of $\mathcal{G}$. Every clause of $\mathcal{CK}$ of $R(Q)$ is satisfied by construction of $\kappa'' \setminus \kappa'$ (and also by κ'') and ζ. Clauses in Formulas (12) and (13) of $\mathcal{P}$ are satisfied by $\kappa'' \setminus \kappa'$. Further, clauses in Formulas (1) and (2) of $\mathcal{G}$ are satisfied because of β, κ'', α', and ζ. Finally, the clauses in Formulas (10) and (11) of $\mathcal{P}$ are satisfied by construction of β, ζ, and κ'.

"$\Longleftarrow$": Let α be an assignment to the free variables of Q for which $Q[\alpha]$ is invalid. We show that if QBF $Q[\alpha]$ is invalid, then $R(Q)[\alpha']$ is invalid as well. Since $Q[\alpha]$ is invalid, $Q[\alpha \cup \iota]$ is invalid for any assignment $\iota : V_1 \rightarrow \{0,1\}$. Assume towards a contradiction that $R(Q)[\alpha']$ is valid. We define an assignment $\kappa :=$ local(ι), which is $\kappa : \mathit{VarI}(\forall, V_1, \emptyset) \cup \mathit{VarI}(\exists, \emptyset, V_1) \rightarrow \{0,1\}$ such that $\kappa(x_t) := \iota(x)$ for every $x_t \in \mathit{VarI}(\forall, V_1, \emptyset) \cup \mathit{VarI}(\exists, \emptyset, V_1)$ with $t \in N$ and $x \in \mathrm{var}(D)$. Observe that by Lemma 20, κ is the only remaining option to obtain valid $R(Q)[\alpha]$. As a result, since $R(Q)[\alpha']$ is claimed valid, $R(Q)[\alpha' \cup \kappa]$ is valid as well. In consequence, by Lemma 21 Statement (a), there has to exist an extension κ' of $\alpha' \cup \kappa$ such that for some $d \in D$, κ' respects bit-term(d, t), where $t = \delta^{-1}(d)$. By Lemma 21 Statement (b), for $1 \leq j \leq 3$ and every node $t' \in \mathit{NodeI}(y)$, where $y :=$ tvar(d, j), we have $\kappa'(v_{t,j}) = \kappa'(y_{t'})$. However, by construction of κ' and connectedness of $\mathcal{T}[y]$, then $(\alpha \cup \iota)$ respects d. In consequence, this contradicts our assumption that $Q[\alpha \cup \iota]$ is invalid.

Induction step ($\ell > 1$): We assume that the theorem holds for a given $\ell - 1$ and it remains to prove that it then holds for ℓ.

"$\Longrightarrow$": We proceed by case distinction on the first quantifier, i.e., (Case 1) $Q_1 = \exists$ and (Case 2) $Q_1 = \forall$. Thereby, we show that if $Q[\alpha]$ is valid and has quantifier rank ℓ, then $R(Q)[\alpha']$ is valid as well.

(Case 1) $Q_1 = \exists$: Since $Q[\alpha]$ is valid, we can construct at least one assignment $\iota : V_1 \to \{0,1\}$ such that $Q[\alpha \cup \iota]$ is valid. By induction hypothesis, since the QBF $Q[\alpha \cup \iota]$ has quantifier rank $\ell - 1$, and is valid, there are α', ι' such that $R(Q)[\alpha' \cup \iota']$ is valid as well. In particular, by induction hypothesis, $\alpha' = \mathsf{local}(\alpha), \iota' = \mathsf{local}(\iota)$ and therefore $R(Q)[\alpha']$ is valid as well.

(Case 2) $Q_1 = \forall$: Since $Q[\alpha]$ is valid, for any assignment ι of V_1, we obtain that $Q[\alpha \cup \iota]$ is valid. In the following, we denote by $R''(Q)$ the QBF that is obtained from $R(Q)$, where variables in $\mathit{VarI}(\exists, \emptyset, V_1)$ do not appear in the scope of a quantifier, i.e., these variables, while existentially quantified in $R(Q)$, are free variables in $R''(Q)$. By induction hypothesis, since the QBF $Q[\alpha \cup \iota]$ has quantifier rank $\ell - 1$, and is valid, there are α', ι' with $\alpha' = \mathsf{local}(\alpha), \iota' = \mathsf{local}(\iota)$, such that $R''(Q)[\alpha' \cup \iota']$ is valid as well. Then, since ι was chosen arbitrarily, for every assignment κ of variables in $\mathsf{dom}(\iota') \cap \mathit{VarI}(Q_1, V_1, \emptyset)$, there is (by Lemma 20, since $R''(Q)[\alpha' \cup \iota']$ is valid) an assignment κ' of variables in $\mathsf{dom}(\iota') \cap \mathit{VarI}(\exists, \emptyset, V_1)$ such that $R(Q)[\alpha' \cup \kappa \cup \kappa']$ is valid. In consequence, $R(Q)[\alpha']$ is valid as well.

"$\Longleftarrow$": Again, we proceed by case distinction in order to show that if $Q[\alpha]$ is invalid and has quantifier rank ℓ, $R(Q)[\alpha']$ is invalid as well.

(Case 1) $Q_1 = \exists$: Since $Q[\alpha]$ is invalid, for every assignment ι of variables V_1 we have that $Q[\alpha \cup \iota]$ is also invalid. By induction hypothesis, since the QBF $Q[\alpha \cup \iota]$ has quantifier rank $\ell - 1$, and is invalid, there are assignments α' and ι' such that $R(Q)[\alpha' \cup \iota']$ is invalid as well, where $\alpha' = \mathsf{local}(\alpha), \iota' = \mathsf{local}(\iota)$. Therefore $R(Q)[\alpha']$ is invalid, since ι was chosen arbitrarily and by Lemma 20 ι' covers all relevant cases, where $R(Q)[\alpha']$ could be valid.

(Case 2) $Q_1 = \forall$: Since $Q[\alpha]$ is invalid, there is at least one assignment ι of variables V_1, such that $Q[\alpha \cup \iota]$ is invalid. By induction hypothesis, since QBF $Q[\alpha \cup \iota]$ has quantifier rank $\ell - 1$, and is invalid, there are assignments α' and ι' with $\alpha' = \mathsf{local}(\alpha), \iota' = \mathsf{local}(\iota)$, such that $R''(Q)[\alpha' \cup \iota']$ is invalid (R'' defined above), either. By Lemma 20, even for an assignment ι'' that restricts ι' to variables in $\mathsf{dom}(\iota') \cap \mathit{VarI}(Q_1, V_1, \emptyset)$, there cannot be an assignment κ to variables in $\mathsf{dom}(\iota') \cap \mathit{VarI}(\exists, \emptyset, V_1)$ such that $R(Q)[\alpha' \cup \iota'' \cup \kappa]$ is valid. In consequence, $R(Q)[\alpha']$ is invalid as well. □

Compression and Runtime. After having established the correctness of reduction R, we move on to showing that this reduction indeed compresses the treewidth of the resulting QBF $R(Q)$, as depicted in Figure 2. In particular, we prove this claim by constructing a tree decomposition $\mathcal{T}'$ of the primal graph of $R(Q)$ and show its relation to labeled almost c-nice tree decomposition $\mathcal{T}$ of Q, where $\mathsf{width}(\mathcal{T}) = \mathsf{tw}(P_Q)$. Then, we discuss runtime properties of the reduction.

Lemma 23 (Compression). *The reduction R exponentially decreases treewidth. In particular, $R'(R(Q))$ constructs a QBF such that the treewidth of the primal graph of $R'(R(Q))$ is $12 \cdot \lceil \log(k+1) \rceil + 7c + 6$, where k is the treewidth of P_Q and $c \leq k$.*

Proof. Assume a labeled almost c-nice tree decomposition $\mathcal{T} = (T, \chi, r)$ of P_Q of width k, where $T = (N, E)$. From this we will construct a tree decomposition $\mathcal{T}' = (T, \chi', r)$ of the primal graph of $R(Q)$. For each tree decomposition node $t \in N$ with $\mathsf{cld}(t) = \{t_1, \ldots, t_s\}$, we set its bag $\chi'(t) := \{b \mid x \in \chi(t), b \in [\![x]\!]_t \cup [\![x]\!]_{t_1} \cup \cdots \cup [\![x]\!]_{t_s}\} \cup \{b \mid x \in \chi(t), 1 \leq j \leq 3, b \in [\![x]\!]_{t,j} \cup [\![x]\!]_{t_1,j} \cup \cdots \cup [\![x]\!]_{t_s,j}\} \cup \{x_{t'} \mid x_{t'} \in \mathit{VarI}(V), t' = t\} \cup \bigcup_{t' \in \{t, t_1, \ldots, t_s\}} \{v_{t',1}, v_{t',2}, v_{t',3}, v_{t'}, \mathit{sat}_{t'}, \mathit{sat}_{\leq t'}\}$. Observe that all the properties of tree decompositions are satisfied. In particular, connectedness is not destroyed since the only elements that are shared among (at most two) different tree decompositions nodes are in *VarB*, *VarBV* and in *VarB3*, and *VarBV3*.

Each bag $\chi'(t)$ contains bit-vectors $[\![x]\!]_t, [\![x]\!]_{t_1}, \ldots, [\![x]\!]_{t_s}$ for each $x \in \chi(t)$, resulting in at most $3 \cdot \lceil \log(k) \rceil$ many elements, since each node can have at most $s = 2$ many children. Further, each bag additionally consists of bit-vectors $[\![x]\!]_{t,j}, [\![x]\!]_{t_1,j}, \ldots, [\![x]\!]_{t_s,j}$ for each $x \in \chi(t) \cup \{\mathit{nil}\}$, where $1 \leq j \leq 3$, which are at most $3 \cdot 3 \cdot \lceil \log(k+1) \rceil$ many elements. In total everything sums up to at most $12 \cdot \lceil \log(k+1) \rceil + 7c + 6$ many elements per node, since $|\{x_{t'} \mid x_{t'} \in \mathit{VarI}(V), t = t'\}| \leq c$, and moreover $|\bigcup_{t' \in \{t, t_1, \ldots, t_s\}} \{v_{t',1}, v_{t',2}, v_{t',3}, v_{t'}, \mathit{sat}_{t'}, \mathit{sat}_{\leq t'}\}| \leq 6 \cdot (c+1)$ due to $\mathcal{T}$ being labeled and almost c-nice. Note that the treewidth of $R'(R(Q))$ only marginally increases, since there are at most $O(k \cdot \lceil \log(k) \rceil)$ many clauses in each bag of $\chi'(t)$ for any node $t \in N$, each of size at most $O(\lceil \log(k) \rceil)$. However, the fresh variables, that were introduced during the 3-CNF reduction only turn up in at most two new clauses (that is, they have degree two in the primal graph). Further, the construction can be controlled in such a way, that each new clause consists of at most two fresh variables. In consequence, one can easily modify $\mathcal{T}'$, by adding at most $O(k \cdot \lceil \log(k) \rceil^2)$ many intermediate nodes for each node $t \in N$, such that the width of $\mathcal{T}'$ is at most $12 \cdot \lceil \log(k+1) \rceil + 7c + 6$. □

Theorem 24 (Runtime). *Given a QBF Q, where D= $\mathsf{matrix}(Q)$, k is the treewidth of the primal graph of Q. Then, constructing $R'(R(Q))$ takes time $O(2^{k^4} \cdot \|D\| \cdot c)$, where $c \leq k$.*

Proof. First, we construct [5] a tree decomposition of the primal graph of Q of width k in time $2^{O(k^3)} \cdot |\mathsf{var}(D)|$, consisting of at most $O(2^{k^3} \cdot |\mathsf{var}(D)|)$ many nodes. Then, we compute a labeled almost c-nice tree decomposition in time $O(k^2 \cdot 2^{k^3} \cdot (\|D\|)$ [40, Lemma 13.1.3] without increasing the width k, resulting in decomposition $\mathcal{T} = (T, \chi)$, where $T = (N, E, r)$ of the primal graph of Q. Note that thereby the number of nodes is at most $O(k \cdot 2^{k^3} \cdot \|D\|)$. The reduction $R(Q)$ then uses at most $O(k \cdot 2^{k^3} \cdot \|D\| \cdot c)$ many variables in $VarI(V)$ since in almost c-nice tree decompositions one node "introduces" at most c variables. The other sets of variables used in R are bounded by $O(\lceil\log(k+1)\rceil \cdot k^2 \cdot 2^{k^3} \cdot \|D\|)$. Overall, there are $O(\lceil\log(k+1)\rceil \cdot k^2 \cdot 2^{k^3} \cdot \|D\| \cdot c)$ many clauses constructed by $R(Q)$. Hence, the claim follows, since $R'(R(Q))$ runs in time $O(\lceil\log(k+1)\rceil^2 \cdot k^2 \cdot 2^{k^3} \cdot \|D\| \cdot c) \subseteq O(2^{k^4} \cdot \|D\| \cdot c)$. ◻

Proof of the main result. We are in position to prove the main result of this work. To this end, we show that the lower bounds are closed under negation and restate Theorem 13.

Lemma 25. *Assume a given closed QBF of the form $Q = Q_1V_1.Q_2V_2.Q_3V_3 \cdots Q_\ell V_\ell.F$, where $\ell \geq 1$ and F is in CNF if $Q_\ell = \exists$, and F is in DNF if $Q_\ell = \forall$. Under ETH, one cannot solve Q in time $\mathrm{tow}(\ell, o(k)) \cdot \mathrm{poly}(|\mathrm{var}(F)|)$ if and only if one cannot solve the negation $\neg Q$ in the same time.*

Proof. Assume towards a contradiction that $\neg Q$ can be solved in time $\mathrm{tow}(\ell, o(k)) \cdot \mathrm{poly}(|\mathrm{var}(F)|)$ under ETH. But then, since inverting the result can be achieved in constant time, under ETH we can solve Q in time $\mathrm{tow}(\ell, o(k)) \cdot \mathrm{poly}(|\mathrm{var}(F)|)$. Hence, we arrive at a contradiction. ◻

Theorem 13 (QBF lower bound). *Given an arbitrary QBF of the form $Q = Q_1V_1.Q_2V_2.Q_3V_3 \cdots Q_\ell V_\ell.F$ where $\ell \geq 1$, and F is a 3-CNF formula (if $Q_\ell = \exists$), or F is a 3-DNF formula (if $Q_\ell = \forall$). Then, unless ETH fails, Q cannot be solved in time $\mathrm{tow}(\ell, o(k)) \cdot \mathrm{poly}(|\mathrm{var}(F)|)$, where k is the treewidth of the primal graph P_Q.*

Proof. We assume that Q is closed, i.e., for Q we have $\mathrm{fvar}(Q) = \emptyset$. We show the theorem by induction on the quantifier rank ℓ. For the induction base, where $\ell = 1$, the result follows from the ETH in case of $Q_\ell = \exists$ since $k \leq |\mathrm{var}(F)|$. If $Q_\ell = \forall$, by Lemma 25, the result follows. Note that for the case of $\ell = 2$, the result has already been shown [42] as well.

For the induction step, we assume that the theorem holds for given Q of quantifier rank $\ell \geq 1$, where $Q_\ell = \forall$, the treewidth of primal graph P_Q is k, and F is in 3-DNF. We show that then the theorem also holds for quantifier rank $\ell+1$. Towards a contradiction, we assume that in general we can solve any QBF Q' of quantifier rank $\ell + 1$, in time $\mathrm{tow}(\ell + 1, o(\mathrm{tw}(P_{Q'}))) \cdot \mathrm{poly}(|\mathrm{var}(C')|)$, where $C' = \mathrm{matrix}(Q')$. We compute a labeled almost c-nice TD $\mathcal{T}$ of P_Q of width k, where c is in $O(\log(k))$. We proceed by case distinction on the last quantifier $Q'_{\ell+1}$ of Q'.

(Case 1) $Q'_{\ell+1} = \exists$: Let $Q' = R'(R(Q))$, $C' = \mathrm{matrix}(Q')$ be the matrix of Q', and k' be the treewidth of the primal graph of C'. Observe that Q' has quantifier rank $\ell + 1$ and is of the required form. By Lemma 23, $k' = 12 \cdot \lceil\log(k + 1)\rceil + 7c + 6$. As a result, since R is an fpt-reduction (including time for computing $\mathcal{T}$) according to Theorem 24, one can solve Q' in time $\mathrm{tow}(\ell+1, o(12 \cdot \lceil\log(k+1)\rceil + 7c + 6)) \cdot \mathrm{poly}(|\mathrm{var}(C')|) = \mathrm{tow}(\ell + 1, o(\log(k))) \cdot \mathrm{poly}(|\mathrm{var}(C')|)$. Therefore, by Theorem 22 we can solve Q in time $\mathrm{tow}(\ell, o(k)) \cdot \mathrm{poly}(|\mathrm{var}(F)|)$, which contradicts the induction hypothesis.

(Case 2) $Q'_{\ell+1} = \forall$: By Lemma 25 one can decide in time $\mathrm{tow}(\ell + 1, o(k)) \cdot \mathrm{poly}(|\mathrm{var}(C')|)$ whether Q' is valid if and only if we can decide in time $\mathrm{tow}(\ell+1, o(k)) \cdot \mathrm{poly}(|\mathrm{var}(C')|)$ whether $\neg Q'$ is valid. Note that after bringing $\neg Q'$ into prenex normal form, the last quantifier is $\exists$. Therefore, the remainder of this case is (Case 1). Hence, we have established the second case and this concludes the proof. ◻

Further Consequences. We generalize Theorem 13 to the incidence graph. The *incidence graph* of a formula F in CNF or DNF is the bipartite graph, which has as vertices the variables and clauses (terms) of F and an edge vc between every variable v and clause (term) c whenever v occurs in c in F [55]. We obtain the following.

Corollary 26. *Given an arbitrary QBF Q of quantifier rank $\ell \geq 1$. Then, under ETH one cannot solve Q in time $\mathrm{tow}(\ell, o(k)) \cdot \mathrm{poly}(|\mathrm{var}(\mathrm{matrix}(Q))|)$, where k is the treewidth of the incidence graph of $\mathrm{matrix}(Q)$.*

Proof. The claim follows from Theorem 13, since, in general, the treewidth k of the incidence graph of Q is bounded [29, 55] by treewidth k' of P_Q, i.e., $k \leq k' + 1$. As a result, if the weaker lower bound of this corollary did not hold, Theorem 13 would be violated. ◻

Corollary 27 (2-Local Pathwidth bound). *Given an arbitrary QBF of the form $Q = Q_1V_1.Q_2V_2.Q_3V_3 \cdots Q_\ell V_\ell.F$, where $\ell \geq 1$ and F in 3-CNF (if $Q_\ell = \exists$) or 3-DNF (if $Q_\ell = \forall$). Then, unless ETH fails, Q cannot be solved in time $\mathrm{tow}(\ell, o(k)) \cdot \mathrm{poly}(|\mathrm{var}(F)|)$, where k is the 2-local pathwidth of graph P_Q.*

Proof. First, we show the claim for *pathwidth* by induction, which can be easily established for base case $\ell = 1$. For the case of $\ell = 2$, related work [42] holds only for treewidth. However, the cases for $\ell \geq 2$ follow from the proof of Theorem 13, since every path decomposition is also a tree decomposition, and the proofs of lemmas and theorems used intermediately only rely on an *arbitrary* tree decomposition. To be more concrete, the proof of Theorem 13 relies on Lemma 23, whose proof shows compression for any tree decomposition, which hence also works for any PD as well. Similarly, Theorem 24 also holds for pathwidth, since a PD of fixed pathwidth can be computed [5] even in time $O(2^{k^2} \cdot |\mathrm{var}(F)|)$, and since computation of labeled almost c-nice decompositions works anologously for PDs. Further, the remainder of the proof holds for the thereby obtained PD, since the construction works for any TD. Finally, Lemma 25 holds independently of the parameter. As a result, reductions R' and R used by Theorem 13 indeed are sufficient for PDs.

The proof above can be lifted to the case of 2-local pathwidth by observing that the 2-local pathwidth of $P_{R'(R(Q))}$ is also bounded by $12 \cdot \lceil\log(k + 1)\rceil + 7c + 6$ (cf. Lemma 23), since each variable of $P_{R'(R(Q))}$ occurs at most twice in the constructed (path) decomposition $\mathcal{T}'$ of $R'(R(Q))$. □

Remark 28. *We remark that reduction R can be generalized to finite,* non-Boolean *domains (QCSP, e.g., [24]). For given variables V of a QCSP formula Q, the variables in $VarI(V)$, $VarBV$, and $VarBV3$ have to be made non-Boolean, whereas the other variables used in R stay Boolean. Consequently, one obtains similar results as in related work [42], but for quantifier rank $\ell \geq 3$. We conject that under ETH, validity of QCSPs Q over domain $\mathcal{D}$ of quantifier rank ℓ, where $k = \mathrm{pw}(P_Q)$, cannot be decided in time* $\mathrm{tow}(\ell - 1, |\mathcal{D}|^{o(k)}) \cdot \mathrm{poly}(|\mathrm{var}(Q)|)$.

Finally, we establish a corollary that improves a result from the literature. To this end, we denote for given positive number n by $\log^*(n)$ the smallest value i such that $\mathrm{tow}(i, 1) \geq n$. A known result [1, Corollary 1] states Σ_ℓ^P-hardness for instances Q of $(4 \cdot \log^*(|\mathrm{var}(\mathrm{matrix}(Q))|))$-QSat, wheras here we establish para-Σ_ℓ^P-hardness for instances Q' of $(\log^*(|\mathrm{var}(\mathrm{matrix}(Q'))|))$-QSat. This is possible by applying our established reduction R, which is rather fine-grained since it only increases quantifier rank by one, and it works indeed for any QBF, and not just for a certain classes of QBFs in contrast to the known result. As a consequence, whenever a new class of ℓ-QBFs with a certain treewidth or pathwidth guarantee was discovered, which is still Σ_ℓ^P-hard, one *immediately* obtains para-Σ_ℓ^P-hardness by using reduction R. Then, one could potentially further improve quantifier alternations by applying reduction R, which is (asymptotically) tight under ETH.

Corollary 29. *Given any integer $\ell \geq 1$. Then, deciding QSat is para-Σ_ℓ^P-hard when parameterized by (2-local) pathwidth of the primal graph P_Q for input QBFs of the form $Q = Q_1 V_1.Q_2 V_2.Q_3 V_3 \cdots Q_{\ell+\log^*(|\mathrm{var}(F)|)} V_{\ell+\log^*(|\mathrm{var}(F)|)}.F$, where is F in 3-CNF (if $Q_\ell = \exists$) or 3-DNF (if $Q_\ell = \forall$).*

Proof. Given a closed QBF of the form $Q' = Q_1 V_1'.Q_2 V_2'.Q_3 V_3' \cdots Q_\ell V_\ell'.F'$, where $\ell \geq 1$ and F' is in 3-CNF if $Q_\ell = \exists$, and F' is in 3-DNF if $Q_\ell = \forall$, and k' is the 2-local pathwidth of $P_{Q'}$. Then, we apply our reduction R followed by R' on Q' and iteratively apply R and R'. We repeat this step exactly $\log^*(k')$ many times and refer to the final result by Q''. Note that the solutions to problem QSat on Q' and Q'' are equivalent by Theorem 22. Then, the resulting 2-local pathwidth k'' of $P_{Q''}$ is in $\mathcal{O}(1)$ by Lemma 23, i.e., parameter k'' is constant. Hence, since Q' is hard for Σ_ℓ^P, also Q'' is hard for Σ_ℓ^P, and Q'' is para-Σ_ℓ^P-hard since k'' is a constant. Observe that $k' \leq |\mathrm{var}(F')| \leq |\mathrm{var}(\mathrm{matrix}(Q''))|$. As a result, QSat for QBFs of the form Q above is hard for para-Σ_ℓ^P. □

5 Conclusion

In this work, we presented a lower bound for deciding the validity (QSat) of quantified Boolean formulas (QBFs). Thereby, we have significantly extended the current state-of-the-art of this line of research: So far, lower bound results under ETH for QSat parameterized by treewidth were not available for all levels of the polynomial hierarchy. The generalization of this result in Theorem 13 does not only cover QBFs, parameterized by treewidth and an arbitrary quantifier rank, but solves a natural question for a well-known problem in complexity theory. Interestingly, the result confirms the (asymptotic) optimality of the algorithm by Chen [9] for solving QSat and thereby answers a longstanding open question. Indeed, one cannot expect to solve QSat of quantifier rank ℓ significantly better than in time $\Omega^*(\mathrm{tow}(\ell, k))$ in the treewidth k. The proof of this result relies on a novel reduction approach that makes use of a fragile balance between redundancy and structural dependency (captured by treewidth) and uses tree decompositions as a "guide" in order to achieve exponential *compression* of the parameter treewidth. We encode dynamic programming on tree decompositions and obtain a technique for compressing treewidth. Note that both our technique and the results naturally carry over to path decompositions and (2-local) pathwidth.

Given the nature of our reduction, we observe that the reduction might also serve in reducing treewidth in practice. In particular, solvers based on tree decompositions such as the QBF solver dynQBF [8] could benefit from significantly reduced treewidth; at the cost of increased quantifier rank by one. Since dynQBF is capable [44] of solving instances up to treewidth 80 with quantifier rank more than two, slightly increasing the quantifier rank might be in practice a good trade-off for decreasing the treewidth significantly.

Another advantage of our reduction is that it gives rise to a versatile *methodology* for showing lower bounds for arbitrary problems (depending on the ETH, parameterized by treewidth) by reduction from ℓ-QSat, parameterized by treewidth as well. Thereby we avoid tedious reductions from Sat (directly using ETH), which involves problem-tailored gadgets to construct instances whose treewidth is ℓ-fold logarithmic in the number of variables or clauses of the given Sat formula. Further, we have listed a number of showcases to illustrate the applicability of this approach to natural problems that are beyond the second level of the polynomial hierarchy. As a by-product we have established that the canonical problems #Σ_ℓSat and #Π_ℓSat of projected model counting applied to QBFs when parameterized by treewidth always come at the price of an additional level of exponentiality in the treewidth (compared to ℓ-QSat).

One direction for future work is to explore further problems parameterized by treewidth and to establish tightness of the so far existing upper bounds. Another important direction is to work out techniques and showcases for "non-canonical" lower bounds, where fptl-reductions are not sufficient and using a customized function g is necessary. Hence, our goal is to continue this line of research in order to use this toolkit for problems that do not exhibit (e.g., [43]) canonical runtimes, where fptl-reductions suffice. We hope this work will foster research and new insights on lower bounds.

References

[1] Albert Atserias and Sergi Oliva. 2014. Bounded-width QBF is PSPACE-complete. *J. Comput. Syst. Sci.* 80, 7 (2014), 1415–1429. https://doi.org/10.1016/j.jcss.2014.04.014

[2] Rehan A. Aziz. 2015. *Answer Set Programming: Founded Bounds and Model Counting*. Ph.D. Dissertation. Department of Computing and Information Systems , The University of Melbourne.

[3] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh (Eds.). 2009. *Handbook of Satisfiability*. FAIA, Vol. 185. IOS Press.

[4] Hans L. Bodlaender. 1988. Dynamic Programming on Graphs with Bounded Treewidth. In *Proceedings of the 15th International Colloquium on Automata, Languages and Programming (ICALP'88) (LNCS)*, Vol. 317. Springer, 105–118. https://doi.org/10.1007/3-540-19488-6

[5] Hans L. Bodlaender. 1996. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25, 6 (1996), 1305–1317. https://doi.org/10.1137/S0097539793251219

[6] Hans L. Bodlaender and Arie M. Koster. 2008. Combinatorial Optimization on Graphs of Bounded Treewidth. *Comput. J.* 51, 3 (2008), 255–269. https://doi.org/10.1093/comjnl/bxm037

[7] John A. Bondy and Uppaluri S. R. Murty. 2008. *Graph theory*. Graduate Texts in Mathematics, Vol. 244. Springer. 655 pages. https://doi.org/10.1007/978-1-84628-970-5

[8] Günther Charwat and Stefan Woltran. 2019. Expansion-based QBF Solving on Tree Decompositions. *Fundam. Inform.* 167, 1-2 (2019), 59–92. https://doi.org/10.3233/FI-2019-1810

[9] Hubie Chen. 2004. Quantified Constraint Satisfaction and Bounded Treewidth. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04)*, Vol. IOS Press. 161–170.

[10] Markus Chimani, Petra Mutzel, and Bernd Zey. 2012. Improved Steiner tree algorithms for bounded treewidth. *J. Discrete Algorithms* 16 (2012), 67–78. https://doi.org/10.1016/j.jda.2012.04.016

[11] Bruno Courcelle. 1990. Graph Rewriting: An Algebraic and Logic Approach. In *Handbook of Theoretical Computer Science, Vol. B*. Elsevier, 193–242. https://doi.org/10.1016/b978-0-444-88074-1.50010-x

[12] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. 2001. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discr. Appl. Math.* 108, 1-2 (2001), 23–52. https://doi.org/10.1016/S0166-218X(00)00221-3

[13] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer. XVII, 613 pages. https://doi.org/10.1007/978-3-319-21275-3

[14] Rina Dechter. 2006. Tractable Structures for Constraint Satisfaction Problems. In *Handbook of Constraint Programming*. Vol. I. Elsevier, Chapter 7, 209–244. https://doi.org/10.1016/S1574-6526(06)80011-8

[15] Reinhard Diestel. 2012. *Graph Theory, 4th Edition*. Graduate Texts in Mathematics, Vol. 173. Springer. 410 pages.

[16] Rodney G. Downey and Michael R. Fellows. 2013. *Fundamentals of Parameterized Complexity*. Springer. https://doi.org/10.1007/978-1-4471-5559-1

[17] Arnaud Durand, Miki Hermann, and Phokion G. Kolaitis. 2005. Subtractive reductions and complete problems for counting complexity classes. *Theor. Comput. Sci.* 340, 3 (2005), 496–513. https://doi.org/10.1016/j.tcs.2005.03.012

[18] Wolfgang Dvořák, Reinhard Pichler, and Stefan Woltran. 2012. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artif. Intell.* 186 (2012), 1–37. https://doi.org/10.1016/j.artint.2012.03.005

[19] Uwe Egly, Thomas Eiter, Hans Tompits, and Stefan Woltran. 2000. Solving Advanced Reasoning Tasks Using Quantified Boolean Formulas. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence (AAAI/IAAI 2000)*. AAAI Press / The MIT Press, 417–422.

[20] Thomas Eiter and Georg Gottlob. 1995. The Complexity of Logic-Based Abduction. *J. ACM* 42, 1 (1995), 3–42. https://doi.org/10.1145/200836.200838

[21] Thomas Eiter and Georg Gottlob. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Ann. Math. Artif. Intell.* 15, 3–4 (1995), 289–323. https://doi.org/10.1007/BF01536399

[22] Michael Elberfeld, Andreas Jakoby, and Till Tantau. 2010. Logspace versions of the theorems of Bodlaender and Courcelle. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*. IEEE, 143–152. https://doi.org/10.1109/FOCS.2010.21

[23] Ronald Fagin. 1974. Generalized First-Order Spectra and Polynomial-Time Recognizable Sets. In *Proceedings of the 7th Symposia in Applied Mathematics*. AMS.

[24] Alex Ferguson and Barry O'Sullivan. 2007. Quantified Constraint Satisfaction Problems: From Relaxations to Explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI 2007)*. The AAAI Press, 74–79.

[25] Johannes K. Fichte and Markus Hecher. 2019. Treewidth and Counting Projected Answer Sets. In *Proceedings of the 15th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2019) (LNCS)*, Vol. 11481. Springer, 105–119. https://doi.org/10.1007/978-3-030-20528-7_9

[26] Johannes K. Fichte, Markus Hecher, and Arne Meier. 2019. Counting Complexity for Reasoning in Abstract Argumentation. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, (AAAI 2019)*. The AAAI Press, 2827–2834. https://doi.org/10.1609/aaai.v33i01.33012827

[27] Johannes K. Fichte, Markus Hecher, Michael Morak, and Stefan Woltran. 2018. Exploiting Treewidth for Projected Model Counting and its Limits. In *Proceedings of the 21th International Conference on Theory and Applications of Satisfiability Testing (SAT'18) (LNCS)*, Vol. 10929. Springer, 165–184. https://doi.org/10.1007/978-3-319-94144-8_11

[28] Johannes K. Fichte, Markus Hecher, Patrick Thier, and Stefan Woltran. 2020. Exploiting Database Management Systems and Treewidth for Counting. In *Proceedings of the 22nd International Symposium on Practical Aspects of Declarative Languages (PADL 2020) (LNCS)*, Vol. 12007. Springer, 151–167. https://doi.org/10.1007/978-3-030-39197-3_10

[29] Johannes K. Fichte and Stefan Szeider. 2015. Backdoors to Tractable Answer-Set Programming. *Artif. Intell.* 220, C (2015), 64–103. https://doi.org/10.1016/j.artint.2014.12.001

[30] Jörg Flum and Martin Grohe. 2006. *Parameterized Complexity Theory*. Springer. 495 pages. https://doi.org/10.1007/3-540-29953-X

[31] Eugene C. Freuder. 1985. A sufficient condition for backtrack-bounded search. *J. ACM* 32, 4 (1985), 755–761. https://doi.org/10.1145/4221.4225

[32] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[33] Georg Gottlob, Reinhard Pichler, and Fang Wei. 2010. Bounded treewidth as a key to tractability of knowledge representation and reasoning. *Artif. Intell.* 174, 1 (2010), 105–132. https://doi.org/10.1016/j.artint.2009.10.003

[34] Martin Grohe. 2017. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Vol. 47. Cambridge University Press.

[35] Markus Hecher, Michael Morak, and Stefan Woltran. 2020. Structural Decompositions of Epistemic Logic Programs. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, (AAAI 2020)*. The AAAI Press. In press.

[36] Neil Immerman. 1999. *Descriptive complexity*. Springer.

[37] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. 2001. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.* 63, 4 (2001), 512–530. https://doi.org/10.1006/jcss.2001.1774

[38] Michael Jakl, Reinhard Pichler, and Stefan Woltran. 2009. Answer-Set Programming with Bounded Treewidth. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, Vol. 2. 816–822.

[39] Hans Kleine Büning and Theodor Lettman. 1999. *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, New York, NY, USA. 420 pages.

[40] Ton Kloks. 1994. *Treewidth, Computations and Approximations*. LNCS, Vol. 842. Springer. https://doi.org/10.1007/BFb0045375

[41] Michael Lampis, Stefan Mengel, and Valia Mitsou. 2018. QBF as an Alternative to Courcelle's Theorem. In *Proceedings of the 21th International Conference on Theory and Applications of Satisfiability Testing (SAT'18)*. Springer, 235–252. https://doi.org/10.1007/978-3-319-94144-8_15

[42] Michael Lampis and Valia Mitsou. 2017. Treewidth with a Quantifier Alternation Revisited. In *Proceedings of the 12th International Symposium on Parameterized and Exact Computation (IPEC'17) (LIPIcs)*, Vol. 89. Dagstuhl Publishing, 26:1–26:12. https://doi.org/10.4230/LIPIcs.IPEC.2017.26

[43] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. 2018. Slightly Superexponential Parameterized Problems. *SIAM J. Comput.* 47, 3 (2018), 675–702. https://doi.org/10.1137/16M1104834

[44] Florian Lonsing and Uwe Egly. 2018. Evaluating QBF Solvers: Quantifier Alternations Matter. In *Proceedings of the 24th International Conference on Principles and Practice of Constraint Programming (CP 2018) (LNCS)*, Vol. 11008. Springer, 276–294. https://doi.org/10.1007/978-3-319-98334-9_19

[45] Dániel Marx and Valia Mitsou. 2016. Double-Exponential and Triple-Exponential Bounds for Choosability Problems Parameterized by Treewidth. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016) (LIPIcs)*, Vol. 55. Dagstuhl Publishing, 28:1–28:15. https://doi.org/10.4230/LIPIcs.ICALP.2016.28

[46] Sebastian Ordyniak and Stefan Szeider. 2013. Parameterized Complexity Results for Exact Bayesian Network Structure Learning. *J. Artif. Intell. Res.* 46 (2013), 263–302. https://doi.org/10.1613/jair.3744

[47] Guoqiang Pan and Moshe Y. Vardi. 2006. Fixed-Parameter Hierarchies inside PSPACE. In *LICS*. IEEE Computer Society, 27–36. https://doi.org/10.1109/LICS.2006.25

[48] Christos H. Papadimitriou. 1994. *Computational Complexity*. Addison-Wesley. 523 pages.

[49] Reinhard Pichler, Stefan Rümmele, and Stefan Woltran. 2010. Counting and Enumeration Problems with Bounded Treewidth. In *Proceedings of the 16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'10) (LNCS)*, Vol. 6355. Springer, 387–404. https://doi.org/10.1007/978-3-642-17511-4_22

[50] Neil Robertson and Paul D. Seymour. 1983. Graph Minors. I. Excluding a Forest. *J. Comb. Theory, Ser. B* 35, 1 (1983), 39–61. https://doi.org/10.1016/0095-8956(83)90079-5

[51] Neil Robertson and Paul D. Seymour. 1984. Graph Minors. III. Planar Tree-Width. *J. Comb. Theory, Ser. B* 36, 1 (1984), 49–64. https://doi.org/10.1016/0095-8956(84)90013-3

[52] Neil Robertson and Paul D. Seymour. 1985. Graph Minors – a Survey. In *Surveys in Combinatorics 1985: Invited Papers for the 10th British Combinatorial Conference (London Mathematical Society Lecture Note Series)*. Cambridge University Press, 153–171. https://doi.org/10.1017/CBO9781107325678.009

[53] Neil Robertson and Paul D. Seymour. 1986. Graph Minors. II. Algorithmic Aspects of Tree-Width. *J. Algorithms* 7, 3 (1986), 309–322. https://doi.org/10.1016/0196-6774(86)90023-4

[54] Neil Robertson and Paul D. Seymour. 1991. Graph minors. X. Obstructions to tree-decomposition. *J. Comb. Theory, Ser. B* 52, 2 (1991), 153–190.

[55] Marko Samer and Stefan Szeider. 2010. Algorithms for propositional model counting. *J. Discrete Algorithms* 8, 1 (2010), 50–64. https://doi.org/10.1016/j.jda.2009.06.002

[56] Yi-Dong Shen and Thomas Eiter. 2017. Evaluating Epistemic Negation in Answer Set Programming (Extended Abstract). In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI 2017)*. ijcai.org, 5060–5064. https://doi.org/10.24963/ijcai.2017/722

[57] Larry J. Stockmeyer and Albert R. Meyer. 1973. Word problems requiring exponential time. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC'73)*. ACM, 1–9. https://doi.org/10.1145/800125.804029