QBF as an Alternative to Courcelle's Theorem

Michael Lampis, Stefan Mengel, and Valia Mitsou

[1] Université Paris-Dauphine, PSL Research University, CNRS, UMR 7243
[2] CNRS, CRIL UMR 8188
[3] Université Paris-Diderot, IRIF, CNRS, UMR 8243

Abstract. We propose reductions to quantified Boolean formulas (QBF) as a new approach to showing fixed-parameter linear algorithms for problems parameterized by treewidth. We demonstrate the feasibility of this approach by giving new algorithms for several well-known problems from artificial intelligence that are in general complete in the second level of the polynomial hierarchy. By reduction from QBF we show that all resulting algorithms are essentially optimal in their dependence on the treewidth. Most of the problems that we consider were already known to be fixed-parameter linear by using Courcelle's theorem or dynamic programming, but we argue that our approach has clear advantages over these techniques: on the one hand, in contrast to Courcelle's theorem, we get concrete and tight guarantees for the runtime dependence on the treewidth. On the other hand, we avoid tedious dynamic programming and, after showing some normalization results for CNF-formulas, our upper bounds often boil down to a few lines.

1 Introduction

Courcelle's seminal theorem [7] states that every graph property definable in monadic second order logic can be decided in linear time on graphs of constant treewidth. Here treewidth is the famous width measure used to measure intuitively how similar a graph is to a tree. While the statement of Courcelle's theorem might sound abstract to the unsuspecting reader, the consequences are tremendous. Since a huge number of computational problems can be encoded in monadic second order logic, this gives automatic linear time algorithms for a wealth of problems in such diverse fields as combinatorial algorithms, artificial intelligence and databases; out of the plethora of such papers let us only cite [19, 10] that treat problems that will reappear in this paper. This makes Courcelle's theorem one of the cornerstones of the field of parameterized algorithms.

Unfortunately, its strength comes with a price: while the runtime dependence on the size of the problem instance is linear, the dependence on the treewidth is unclear when using this approach. Moreover, despite recent progress, see e.g. the survey [26], Courcelle's theorem is largely considered impractical due to the gigantic constants involved in the construction. Since generally these constants are unavoidable [18], showing linear time algorithms with Courcelle's theorem can hardly be considered as a satisfying solution.

As a consequence, linear time algorithms conceived with the help of Courcelle's theorem are sometimes followed up with more concrete algorithms with more explicit runtime guarantees often by dynamic programming or applications of a datalog approach [12, 19, 21]. Unfortunately, these hand-written algorithms tend to be very technical, in particular for decision problems outside of NP. Furthermore, even this meticulous analysis usually gives algorithms with a dependance on treewidth that is a tower of exponentials.

The purpose of this paper is two-fold. On the one hand we propose reductions to QBF combined with the use of a known QBF-algorithm by Chen [6] as a simple approach to constructing linear-time algorithms for problems beyond NP parameterized by treewidth. The advantage of this approach over Courcelle's metatheorem or tedious dynamic programming is that the algorithms we provide, while being almost straightforward to produce, give bounds on the treewidth that match asymptotically those of careful dynamic programming. On the other hand, we show that our algorithms are best possible, giving matching complexity lower bounds. Our problem pool will be the area of artificial intelligence that contains a variety of problems complete for classes in the second level of the polynomial hierarchy: abduction, circumscription, abstract argumentation and the computation of minimal unsatisfiable sets in unsatisfiable formulas.

Our algorithmic approach might at first sight seem surprising: since the QBF with a fixed number of alternations is complete for the different levels of the polynomial hierarchy, there are trivially reductions from all problems in that hierarchy to the corresponding QBF problem. So what is new about this approach? The crucial observation here is that in general reductions to QBF guaranteed by completeness do not maintain the treewidth of the problem. So we might start with an instance of small treewidth and then reduce to a QBF-instance with high treewidth such that Chen's algorithm does not give us any meaningful runtime guarantees. Moreover, while Chen's algorithm runs in linear time, there is no reason for the reduction to QBF to run in linear time which would result in an algorithm with overall non-linear runtime.

The runtime bounds that we give are mostly of the form $2^{2^{O(k)}}n$ where k is the treewidth and n the size of the input. Furthermore, starting from recent lower bounds for QBF [25], we also show that these runtime bounds are essentially tight as there are no algorithms with runtime $2^{2^{o(k)}}2^{o(n)}$ for the considered problems. Our lower bounds are based on the *exponential time hypothesis (ETH)* which posits that there is no algorithm for 3SAT with runtime $2^{o(n)}$ where n is the number of variables in the input. ETH is by now widely accepted as a standard assumption in the fields of exact and parameterized algorithms for showing tight lower bounds, see e.g. the survey [29]. We remark that our bounds confirm the observation already made in [30] that problems complete in the second level of the polynomial hierarchy parameterized by treewidth tend to have runtime double-exponential in the treewidth.

As a consequence, the main contribution of this paper is to show that reductions to QBF can be used as a simple technique to show algorithms with essentially optimal runtime for a wide range of problems.

2 Preliminaries

In this section, we only introduce notation that we will use in all parts of the paper. The background for the problems on which we demonstrate our approach will be given in the individual sections in which these problems are treated.

2.1 Treewidth

Throughout this paper, all graphs will be undirected and simple unless explicitly stated otherwise. A *tree decomposition* $(T, (B_t)_{t \in T})$ of a graph $G = (V, E)$ consists of a tree T and a subset $B_t \subseteq V$ for every node t of T with the following properties:

- every vertex $v \in V$ is contained in at least one set B_t,
- for every edge $uv \in E$, there is a set B_t that contains both u and v, and
- for every $v \in V$, the set $\{t \mid v \in B_t\}$ induces a subtree of T.

The last condition is often called the connectivity condition. The sets B_t are called *bags*. The *width* of a tree decomposition is $\max_{t \in T}(|B_t|) - 1$. The *treewidth* of G is the minimum width of a tree decomposition of G. We will sometimes tacitly use the fact that any tree decomposition can always be assumed to be of size linear in $|V|$ by standard simplifications. Computing the treewidth of a graph is NP-hard [1], but for every fixed k there is a linear time algorithm that decides if a given graph has treewidth at most k and if so computes a tree decomposition witnessing this [3].

A tree decomposition is called *nice* if every node t of T is of one of the following types:

- **leaf node:** t is a leaf of T.
- **introduce node:** t has a single child node t' and $B_t = B_{t'} \cup \{v\}$ for a vertex $v \in V \setminus B_{t'}$.
- **forget node:** t has a single child node t' and $B_t = B_{t'} \setminus \{v\}$ for a vertex $v \in B_{t'}$.
- **join node:** t has exactly two children t_1 and t_2 with $B_t = B_{t_1} = B_{t_2}$.

Nice tree decompositions were introduced in [22] where it was also shown that given a tree decomposition of a graph G, one can in linear time compute a nice tree decomposition of G with the same width.

2.2 CNF formulas

A *literal* is a propositional variable or the negation of a propositional variable. A *clause* is a disjunction of literals and a CNF-formula is a conjunction of clauses. For technical reasons we assume that there is an injective mapping from the variables in a CNF formula ϕ to $\{0, \ldots, cn\}$ for an arbitrary but fixed constant c where n is the number of variables in ϕ and that we can evaluate this mapping in constant time. This assumption allows us to easily create in linear time in n lists

which store data assigned to the variables that we can then look up in constant time. Note that formulas in the DIMACS format [8], the standard encoding for CNF formulas, generally have this assumed property. Alternatively, we could use perfect hashing to assign the variables to integers, but this would make some of the algorithms randomized.

Let ϕ and ϕ' be two CNF formulas. We say that ϕ is a projection of ϕ' if and only if $\mathsf{var}(\phi) \subseteq \mathsf{var}(\phi')$ and $a : \mathsf{var}(\phi) \to \{0, 1\}$ is a model of ϕ if and only if a can be extended to a model of ϕ'.

(a) Primal graph (b) Incidence graph

Fig. 1: Primal and incidence graphs for $\phi = (\neg x \vee z) \wedge (x \vee y \vee \neg w) \wedge (\neg z \vee w)$.

To every CNF formula ϕ we assign a graph called *primal graph* whose vertex set is the set of variables of ϕ. Two vertices are connected by an edge if and only if they appear together in a clause of ϕ (See Figure 1a). The *primal treewidth* of a CNF formula is the treewidth of its primal graph. We will also be concerned with the following generalization of primal treewidth: the *incidence graph* of a CNF formula has as vertices the variables *and* the clauses of the formula. Two vertices are connected by an edge if and only if one vertex is a variable and the other is a clause such that the variable appears in the clause (See Figure 1b). The *incidence treewidth* of a formula is then the treewidth of its incidence graph.

It is well-know that the incidence treewidth of a CNF-formula can be arbitrarily higher than the primal treewidth (for example consider a single clause of size n). The other way round, formulas of primal treewidth k can easily be seen to be of incidence treewidth at most $k + 1$ [17].

3 Preprocessing CNF

In this section, we will show several linear time transformations of CNF-formulas that will be useful in the remainder of the paper. Some of these transformations are somewhat tedious but none of them are particularly hard.

Later on, it will be convenient to assume that the CNF formulas we encounter have bounded arity (clause size). There are of course well-known standard reductions for this. Proposition 1 suggests that we can apply them in linear time while roughly maintaining the treewidth.

Proposition 1. *There is an algorithm that, given a CNF formula ϕ of incidence treewidth k, computes in time $2^{O(k)}|\phi|$ a 3CNF formula ϕ' of incidence treewidth $O(k)$ with $\mathsf{var}(\phi) \subseteq \mathsf{var}(\phi')$ such that ϕ is a projection of ϕ'.*

Proof.(Sketch) We use the classic reduction from SAT to 3SAT that cuts big clauses into smaller clauses by introducing new variables. During this reduction we have to take care that the runtime is in fact linear and that we can bound the treewidth appropriately. For the complete proof see Appendix A. □

It is well-known that if the clauses in a formula ϕ of incidence treewidth k have at most size d, then the primal treewidth of ϕ is at most $(k+1)d$, see e.g. [17] so the following result follows directly.

Corollary 1. *There is an algorithm that, given a CNF-formula ϕ of incidence treewidth k, computes in time $O(2^k|\phi|)$ a 3CNF-formula ϕ' of primal treewidth $O(k)$ such that ϕ is a projection of ϕ'.*

We will in several places in this paper consider Boolean combinations of functions expressed by CNF formulas of bounded treewidth. The following technical lemma states that we can under certain conditions construct CNF formulas of bounded treewidth for the these Boolean combinations.

Lemma 1. *a) There is an algorithm that, given a 3CNF-formula ϕ and a tree decomposition $(T, (B_t)_{t\in T})$ of its incidence graph of width $O(k)$, computes in time $\mathrm{poly}(k)n$ a CNF-formula ϕ' and a tree decomposition $(T', (B'_t)_{t\in T})$ of the incidence graph of ϕ' such that $\neg\phi$ is a projection of ϕ', for all $t \in T$ we have $B'_t \cap \mathsf{var}(\phi) = B_t$ and the width of $(T', (B'_t)_{t\in T})$ is $O(k)$.*

b) There is an algorithm that, given two 3CNF-formulas ϕ_1, ϕ_2 and two tree decompositions $(T, (B^i_t)_{t\in T})$ for $i = 1, 2$ of the incidence graphs of ϕ_i of width $O(k)$ such that for every bag either $B^1_t \cap B^2_t = \emptyset$ or $B^1_t \cap \mathsf{var}(\phi_1) = B^2_t \cap \mathsf{var}(\phi_2)$, computes in time $\mathrm{poly}(k)n$ a tree decomposition $(T', (B'_t)_{t\in T})$ of the incidence graph of $\phi_1 \wedge \phi_2$ such that $\phi' \equiv \phi_1 \wedge \phi_2$, for all $t \in T$ we have $B^1_t \cup B^2_t = B'_t$ and the width of $(T, (B'_t)_{t\in T})$ is $O(k)$.

Proof. a) Because every clause has at most 3 literals, we assume w.l.o.g. that every bag B that contains a clause C contains also all variables of C.

In a first step, we add for every clause $C = \ell_1 \vee \ell_2 \vee \ell_3$ a variable x_C and substitute C by clauses with at most 3-variables encoding the constraint $\mathcal{C} = x_C \leftrightarrow l_1 \vee l_2 \vee l_3$ introducing some new variables. The result is a CNF-formula ϕ_1 in which every assignment a to $\mathsf{var}(\phi)$ can be extended uniquely to a satisfying assignment a_1 and in a_1 the variable x_C is true if and only if C is satisfied by a. Note that, since every clause has at most 3 variables, the clauses for $\mathcal{C}$ can be constructed in constant time. Moreover, we can construct a tree decomposition of width $O(k)$ for ϕ_1 from that of ϕ by adding all new clauses for $\mathcal{C}$ and x_C to every bag containing C.

In a next step, we introduce a variable x_t for every $t \in T$ and a constraint $\mathcal{T}$ defining $x_t \leftrightarrow (x_{t_1} \wedge x_{t_2} \wedge \bigwedge_{C\in B_t} x_C)$ where t_1, t_2 are the children of t and the variables are omitted in case they do not appear. The resulting CNF formula ϕ_2

is such that every assignment a to $\mathsf{var}(\phi)$ can be uniquely extended to a satisfying assignment a_2 of ϕ_2 and x_t is true in a_2 if and only if all clauses that appear in the subtree of T rooted in t are satisfied by a. Since every constraint $\mathcal{T}$ has at most k variables, we can construct the 3CNF-formula simulating it in time $O(k)$, e.g. by Tseitin transformation. We again bound the treewidth as before.

The only thing that remains is to add a clause $\neg x_r$ where r is the root of T. This completes the proof of a).

b) We simply set $B'_t = B^1_t \cup B^2_t$. It is readily checked that this satisfies all conditions. □

Lemma 2. *There is an algorithm that, given a 3CNF formula ϕ with a tree decomposition $(T, (B_t)_{t\in T})$ of width k of the incidence graph of ϕ and sequences of variables $X := (x_1, \ldots, x_\ell)$, $Y = (y_1, \ldots, y_\ell) \subseteq \mathsf{var}(\phi)^\ell$ such that for every $i \in [\ell]$ there is a bag B_t with $\{x_i, y_i\} \in B_t$, computes in time $\mathrm{poly}(k)|\phi|$ a formula ψ that is a projection of $X \subseteq Y = \bigwedge_{i=1}^{\ell}(x_i \leq y_i)$ and a tree decomposition $(T, (B_t)_{t\in T})$ of ψ of width $O(1)$. The same is true for $\subset$ instead of $\subseteq$.*

Proof. For the case $\subseteq$, ψ is simply $\bigwedge_{i=1}^{\ell}(x_i \leq y_i) = \bigwedge_{i=1}^{\ell} \neg x_i \vee y_i$. ψ satisfies all properties even without projection and with the same tree decomposition.

The case $\subset$ is slightly more complex. We first construct $\bigwedge_{i=1}^{\ell}(x_i = y_i) = \bigwedge_{i=1}^{\ell}(\neg x_i \vee y_i) \wedge (x_i \vee \neg y_i)$. Then we apply Lemma 1 a) to get a CNF formula that has $X \neq Y$ as a projection. Finally, we use Lemma 1 to get a formula for $X \subset Y = (X \subseteq Y) \wedge (X \neq Y)$. It is easy to check that this formula has the right properties for the tree decomposition. □

4 2-QBF

Our main tool in this paper will be QBF, the quantified version of CNF. In particular, we will be concerned with the version of QBF which only has two quantifier blocks which is often called 2-QBF. Let us recall some standard definitions. A $\forall\exists$-QBF is a formula of the form $\forall X \exists Y \phi(X, Y)$ where X and Y are disjoint vectors of variables and $\phi(X, Y)$ is a CNF-formula called the *matrix*. We assume the usual semantics for $\forall\exists$-QBF.

It is well-known that deciding if a given $\forall\exists$-QBF is true is complete in the second level of the polynomial hierarchy, and thus generally considered intractable. Treewidth has been used as an approach for finding tractable fragments of $\forall\exists$-QBF and more general bounded alternation QBF. Let us define the primal (resp. insidence) treewidth of a $\forall\exists$-QBF to be the primal (resp. incidence) treewidth of the underlying CNF formula. Chen [6] showed the following result.

Theorem 1. *[6] There is an algorithm that given a $\forall\exists$-QBF of primal treewidth k decides in time $2^{2^{O(k)}}|\phi|$ if ϕ is true.*

We note that the result of [6] is in fact more general than what we state here. In particular, the paper gives a more general algorithm for QBF-i with running time $2^{2^{\cdot^{\cdot^{\cdot^{O(k)}}}}}|\phi|$, where the height of the tower of exponentials is i. For readability we

will restrict ourselves to the case $i = 2$ that is general enough for our needs. We also remark that Chen does not state that his algorithm in fact works in linear time. We sketch in Appendix B why there is indeed a linear runtime bound.

In the later parts of this paper, we require a version of Theorem 1 for incidence treewidth which fortunately follows directly from Theorem 1 and Corollary 1.

Corollary 2. *There is an algorithm that given a $\forall\exists$-QBF of incidence treewidth k decides in time $2^{2^{O(k)}}|\phi|$ if ϕ is true.*

We remark that general QBF of bounded treewidth without any restriction on the quantifier prefix is PSPACE-complete [2], and finding tractable fragments by taking into account the structure of the prefix and notions similar to treewidth is quite an active area of research, see e.g. [14, 13].

To show tightness of our upper bounds, we use the following theorem from [25].

Theorem 2. *There is no algorithm that, given a $\forall\exists$-QBF ϕ with n variables and treewidth k, decides if ϕ is true in time $2^{2^{o(k)}}2^{o(n)}$, unless ETH is false.*

5 Abstract Argumentation

Abstract argumentation is an area of artificial intelligence which tries to assess the acceptability of arguments within a set of possible arguments based only the relation between them, i.e., which arguments defeat which. Since its creation in [9], abstract argumentation has developed into a major and very active subfield. In this section, we consider the most studied setting introduced in [9].

An argumentation framework is a pair $F = (A, R)$ where A is a finite set and $R \subseteq A \times A$. The elements of A are called *arguments*. The elements of R are called the *attacks* between the arguments and we say for $a, b \in A$ that a attacks b if and only if $ab \in R$. A set $S \subseteq A$ is called *conflict-free* if and only if there are no $a, b \in S$ such that $ab \in R$. We say that a vertex a is *defended* by S if for every b that attacks a, i.e. $ba \in R$, there is an argument $c \in S$ that attacks b. The set S is called *admissible* if and only if it is conflict-free and all elements of S are defended by S. An admissible set S is called *preferred* if and only if it is subset-maximal in the set of all admissible sets.

There are two main notions of acceptance: A set S of arguments is accepted *credulously* if and only if there is a preferred admissible set such that $S \subseteq S'$. The set S is accepted *skeptically* if and only if for all preferred admissible sets S' we have $S \subseteq S'$. Both notions of acceptance have been studied extensively in particular with the following complexity results: it is NP hard to decide, given an argumentation framework $F = (A, R)$ and a set $S \subseteq A$, if S is credulously accepted. For skeptical acceptance, the analogous decision problem is Π_p^2-complete [11]. Credulous acceptance is easier to decide, because when S is contained in any admissible set S' then it is also contained in a preferred admissible set S'': a simple greedy algorithm that adds arguments to S' that are not in any conflicts constructs such an S''.

Concerning treewidth, after some results using Courcelle's theorem [10], it was shown in [12] by dynamic programming that credulous acceptance can be

decided in time $2^{O(k)}n$ while skeptical acceptance can be decided in time $2^{2^{O(k)}}n$ for argument frameworks of size n and treewidth k. Here an argument framework is seen as a directed graph and the treewidth is that of the underlying undirected graph. We reprove these results in our setting. To this end, we first encode conflict free sets in CNF. Given an argumentation framework $F = (A, R)$, construct a CNF formula ϕ_{cf} that has an indicator variable x_a for every $a \in A$ as

$$\phi_{cf} := \bigwedge_{ab \in R} \neg x_a \vee \neg x_b.$$

It is easy to see that the satisfying assignments of ϕ_{cf} encode the conflict-free sets for F. To encode the admissible sets, we add an additional variable P_a for every $a \in A$ and define:

$$\phi_d := \phi_{cf} \wedge \bigwedge_{a \in A} ((\neg P_a \vee \bigvee_{b:ba \in R} x_b) \wedge \bigwedge_{b:ba \in R} (P_a \vee \neg x_b))$$

The clauses for each P_a are equivalent to $P_a \leftrightarrow \bigvee_{b:ba \in R} x_b$, i.e., P_a is true in a model if and only if a is attacked by the encoded set. Thus by setting

$$\phi_{adm} := \phi_d \wedge \bigwedge_{ba \in R} \neg P_b \vee \neg x_a$$

we get a CNF formula whose models restricted to the x_a variables are exactly the admissible sets. We remark that in [24] the authors give a similar SAT-encoding for argumentation problems with slightly different semantics.

Claim. If F has treewidth k, then ϕ_{adm} has incidence treewidth k.

Proof. We start from a tree decomposition $(T, (B_t)_{t \in T})$ of width k of F and construct a tree decomposition of ϕ_{adm}. First note that $(T, (B_t)_{t \in T})$ is also a tree decomposition of the primal graph of ϕ_{cf} up to renaming each a to x_a. For every $ba \in R$ there is thus a bag B that contains both b and a. We connect a new leaf to B containing where $\{C_{a,b}, a, b\}$ is a clause node for the clause $\neg x_a \vee \neg x_b$ to construct a tree decomposition of the primal graph of ϕ_d.

Now we add P_a to all bags containing x_a, so that for every clause $P_a \vee \neg x_b$ we have a bag containing both variables, and we add new leaves for the corresponding clause nodes as before. Then we add for every clause $C_a := \neg P_a \vee \bigvee_{b:ba \in R} x_b$ the node C_a to every bag containing a. This covers all edges incident to C_a in the incidence graph of ϕ_d and since for every a we only have one such edge, this only increases the width of the decomposition by a constant factor. We obtain a tree decomposition of width $O(k)$ for the incidence graph of ϕ_d.

The additional edges for ϕ_{adm} are treated similarly to above and we get a tree decomposition of width $O(k)$ of ϕ_{adm} of ϕ as desired. □

Combining Claim 5 with the fact that satisfiability of CNF-formulas of incidence treewidth k can solved in time $2^{O(k)}$, see e.g. [33], we directly get the first result of [12].

Theorem 3. *There is an algorithm that, given an argumentation framework $F = (A, R)$ of treewidth k and a set $S \subseteq A$, decides in time $2^{O(k)}|A|$ if S is credulously accepted.*

We also give a short reproof of the second result of [12].

Theorem 4. *There is an algorithm that, given an argumentation framework $F = (A, R)$ of treewidth k and a set $S \subseteq A$, decides in time $2^{2^{O(k)}}|A|$ if S is skeptically accepted.*

Proof. Note that the preferred admissible sets of $F = (A, R)$ are exactly the subset maximal assignments to the x_a that can be extended to a satisfying assignment of ϕ_{adm}. Let $X := \{x_a \mid a \in A\}$, then we can express the fact that an assignment is a preferred admissible set by

$$\phi'(X) = \exists P \forall X' \forall P' \left(\phi_{adm}(X, P) \wedge (\neg\phi(X', P') \vee \neg(X \subset X'))\right)$$

where the sets P,X' and P' are defined analogously to X. Then S does not appear in all preferred admissible sets if and only if

$$\exists X(\phi'(X) \wedge \bigvee_{a \in S} \neg x_a).$$

Negating and using Lemma 1 yields a $\forall\exists$-QBF of incidence treewidth $O(k)$ that is true if and only if S appears in all preferred admissible sets. This gives the result with Corollary 2. □

We now show that Theorem 4 is essentially tight.

Theorem 5. *There is no algorithm that, given an argumentation framework $F = (A, R)$ of size n and treewidth k and a set $S \subseteq A$, decides if S is in every preferred admissible set of F in time $2^{2^{o(k)}}2^{o(n)}$, unless ETH is false.*

Proof. We use a construction from [11, 12]: for a given $\forall\exists$-QBF $\forall Y \exists Z \phi$ in variables $Y \cup Z = \{x_1, \ldots, x_n\}$ and clauses $C_1, \ldots, C_m$, define $F_\phi = (A, R)$ with

$$\begin{aligned}
A =& \{\phi, C_1, \ldots, C_m\} \vee \{x_i, \bar{x}_i \mid i = 1 \leq i \leq n\} \cup \{b_1, b_2, b_3\} \\
R =& \{(C_j, \phi) \mid 1 \leq j \leq m\} \cup \{(x_i, \bar{x}_i), (\bar{x}_i, x_i) \mid 1 \leq i \leq n\} \\
& \cup \{(x_i, C_j) \mid x_i \text{ in } C_j \ , 1 \leq j \leq m\} \cup \{(\bar{x}_i, C_j) \mid \neg x_i \text{ in } C_j \ , 1 \leq j \leq m\} \\
& \cup \{(\phi, b_1), (\phi, b_2), (\phi, b_3), (b_1, b_2), (b_2, b_3), (b_3, b_1)\} \cup \{(b_1, z), (b_1, \bar{z}) \mid z \in Z\}
\end{aligned}$$

One can show that ϕ is in every preferred admissible set of F_ϕ if and only if ϕ is true. Moreover, from a tree decomposition of the primal graph of ϕ we get a tree decomposition of F as follows: we add every $\bar{x}_i$ to every bag that contains x_i and we add b_1, b_2, b_3 to all bags. This increases the treewidth from k to $2k+3$ and thus we get the claim with Theorem 2. □

6 Abduction

In this section, we consider *(propositional) abduction*, a form of non-monotone reasoning that aims to find explanations for observations that are consistent with an underlying theory. A *propositional abduction problem* (short PAP) consists of a tuple $P = (V, H, M, T)$ where T is a propositional formula called the *theory* in variables V, the set $M \subseteq V$ is called the set of *manifestations* and $H \subseteq V$ the set of hypotheses. We assume that T is always in CNF. In abduction, one identifies a set $S \subseteq V$ with the formula $\bigwedge_{x \in S} x$. Similarly, given a set $S \subseteq H$, we define $T \cup S := T \wedge \bigwedge_{x \in S} x$. A set $S \subseteq H$ is a *solution* of the PAP, if $T \cup S \models M$, i.e., all models of $T \cup S$ are models of M.

There are three main problems on PAPs that have been previously studied:

- Solvability: Given a PAP P, does it have a solution?
- Relevance: Given a PAP P and $h \in H$, is h contained in *at least one* solution?
- Necessity: Given a PAP P and $h \in H$, is h contained in *all* solutions?

The first two problems are Σ_p^2-complete while necessity is Π_p^2-complete [16]. In [19], it is shown with Courcelle's theorem that if the theory T of an instance P is of bounded treewidth, then all three above problems can be solved in linear time. Moreover, [19] gives an algorithm based on a Datalog-encoding that solves the solvability and relevance problems in time $2^{2^{O(k)}}|T|$ on instances of treewidth k. Our first result gives a simple reproof of the latter results and gives a similar runtime for necessity.

Theorem 6. *There is a linear time algorithm that, given a PAP $P = (V, H, M, T)$ such that the incidence treewidth of T is k and $h \in H$, decides in time $2^{2^{O(k)}}|T|$ the solvability, relevance and necessity problems.*

Proof. We first consider solvability. We identify the subsets $S \subseteq H$ with assignments to H in the obvious way. Then, for a given choice S, we have that $T \cup S$ is consistent if and only if

$$\psi_1(S) := \exists X T(X) \wedge \bigwedge_{s_i \in H} (s_i \to x_i),$$

is true where X has a variable x_i for every variable $v_i \in V$. Moreover, $T \cup S \models M$ if and only if

$$\psi_2 := \forall X' \left(\bigwedge_{s_i \in H} (s_i \to x_i') \to \left(T(X') \to \bigwedge_{v_i \in M} x_i' \right) \right),$$

where X' similarly to X has a variables x_i for every variable $v_i \in V$. To get a $\forall\exists$-formula, we observe that the PAP has no solution if and only if

$$\forall S \neg(\psi_1(S) \wedge \psi_2(S)) = \forall S \forall X' \exists X' \neg(T(X) \wedge S \subseteq X|_H) \vee (S \subseteq X'|_H \wedge T(X') \wedge \neg \bigwedge_{v_i \in M} x_i')$$

is true, where $X|_H$ denotes the restriction of X to the variables of H. Now applying Lemmata 1 and 2 in combination with de Morgan laws to express $\vee$ yields a $\forall\exists$-QBF of incidence treewidth $O(k)$ and the result follows with Corollary 2.

For relevance, we simply add the hypothesis h to T and test for solvability.

For necessity, observe that h is in all solutions if and only if

$$\forall S(\psi_1(S) \wedge \psi_2(S)) \rightarrow h,$$

which can easily be brought into $\forall\exists$-QBF slightly extending the construction for the solvability case. □

Using the Σ_p^2-hardness reduction from [16], it is not hard to show that the above runtime bounds are tight.

Theorem 7. *There is no algorithm that, given a PAP P whose theory has primal treewidth k, decides decides solvability of P in time $2^{2^{o(k)}}2^{o(n)}$, unless ETH is false. The same is true for relevance and necessity of a variable h.*

Proof. Let $\phi' = \forall X \exists Y \phi$ be a $\forall\exists$-QBF with $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_\ell\}$. Define a PAP $P = (V, H, M, T)$ as follows

$$\begin{aligned} V &= X \cup Y \cup X' \cup \{x\} \\ H &= X \cup X' \\ M &= Y \cup \{s\} \\ T &= \bigwedge_{i=1}^{m}(x_i \leftrightarrow \neg x_i') \wedge \underbrace{(\phi \rightarrow s \wedge \bigwedge_{j=1}^{\ell} y_j)}_{\psi} \wedge \bigwedge_{j=1}^{\ell}(s \rightarrow y_j) \end{aligned}$$

where $X' = \{x_1', \ldots, x_m'\}$ and s are fresh variables. It is shown in [16] that ϕ' is true if and only if P has a solution. We show that T can be rewritten into CNF-formula T' with the help of Lemma 1. The only non-obvious part is the rewriting of ψ. We solve this part by first negating into $(\phi \wedge (\neg s \vee \bigvee_{j=1}^{\ell} \neg y_j)$ and observing that the second conjunct is just a clause, adding it to ϕ only increases the treewidth by 2. Finally, we negate the resulting formula to get a CNF-formula for ψ with the desired properties. The rest of the construction of T' is straightforward. The claim then follows with Theorem 2.

The result is a PAP with theory T' of treewidth $O(k)$ and $O(n)$ variables and the result for solvability follows with Theorem 2. The result for relevance and necessity we point the reader to the proof of Theorem 4.3 in [16]. There for a PAP P a new PAP P' with three additional variables and 5 additional clauses is constructed such that solvability of P reduces to the necessity (resp. relevance) of a variable in P'. Since adding a fixed number of variables and clauses only increases the primal treewidth at most by a constant, the claim follows. □

6.1 Adding $\subseteq$-Preferences

In abduction there are often preferences for the solution that one wants to consider for a given PAP. One particular interesting case is $\subseteq$-preference where

one tries to find (subset-)minimal solutions, i.e. solutions S such that no strict subset $S' \subseteq S$ is a solution. This is a very natural concept as it corresponds to finding minimal explanations for the observed manifestations. We consider two variations of the problems considered above, $\subseteq$-relevance and $\subseteq$-necessity. Surprisingly, complexity-wise, both remain in the second level of the polynomial hierarchy [15]. Below we give a linear-time algorithm for these problems.

Theorem 8. *There is a linear time algorithm that, given a PAP $P = (V, H, M, T)$ such that the incidence treewidth of T is k and $h \in H$, decides in time $2^{2^{2^{O(k)}}}|T|$ the $\subseteq$-relevance and $\subseteq$-necessity problems.*

Proof.(Sketch) We have seen how to express the property of a set S being a solution as a formula $\psi(S)$ in the proof of Theorem 6. Then expressing that S is a minimal model can written by

$$\psi'(S) := \psi(S) \wedge (\forall S'(S' \subseteq S \rightarrow \neg\psi(S'))).$$

This directly yields QBFs for encoding the $\subseteq$-necessity and $\subseteq$-relevance problems as before which can again be turned into treewidth $O(k)$. The only difference is that we now have three quantifier alternations leading to a triple-exponential dependence on k when applying the algorithm from [6] (see Appendix B). □

We remark that [19] already gives a linear time algorithm for $\subseteq$-relevance and $\subseteq$-necessity based on Courcelle's algorithm and thus without any guarantees for the dependence on the runtime. Note that somewhat disturbingly the dependence on the treewidth in Theorem 8 is triple-exponential. We remark that the lower bounds we could get with the techniques from the other sections are only double-exponential. Certainly, having a double-exponential dependency as in our other upper bounds would be preferable and thus we leave this as an open question.

7 Circumscription

In this section, we consider the problem of circumscription. To this end, consider a CNF-formula T encoding a propositional function called the *theory*. Let the variable set X of T be partitioned into three variable sets P, Q, Z. Then a model a of T is called (P, Q, Z)-*minimal* if and only if there is no model a' such that $a'|_P \subset a|_P$ and $a'|_Q = a|_Q$. In words, a is minimal on P for the models that coincide with it on Q. Note that a and a' can take arbitrary values on Z. We denote the (P, Q, Z)-minimal models of T by $\mathrm{MM}(T, P, Q, Z)$. Given a CNF-formula F, we say that $\mathrm{MM}(T, P, Q, Z)$ entails F, in symbols $\mathrm{MM}(T, P, Q, Z) \models F$, if all assignments in $\mathrm{MM}(T, P, Q, Z)$ are models of F. The problem of *circumscription* is, given T, P, Q, Z and F as before, to decide if $\mathrm{MM}(T, P, Q, Z) \models F$.

Circumscription has been studied extensively and is used in many fields, see e.g. [28, 31]. We remark that circumscription can also be seen as a form of closed world reasoning which is equivalent to reasoning under the so-called extended closed world assumption, see e.g. [5] for more context. On general instances circumscription is Π_p^2-complete [15] and for bounded treewidth instances, i.e. if the

treewidth of $T \wedge F$ is bounded, there is a linear time algorithm shown by Courcelle's theorem [19]. There is also a linear time algorithm for the corresponding counting problem based on datalog [21]. We here give a version of the result from [19] more concrete runtime bounds.

Theorem 9. *There is an algorithm that, given an instance T, P, Q, Z and F of incidence treewidth k, decides if* $\mathrm{MM}(T, P, Q, Z) \models F$ *in time* $2^{2^{O(k)}}(|T| + |F|)$.

Proof. Note that $\mathrm{MM}(T, P, Q, Z) \models F$ if and only if for every assignment (a_P, a_Q, a_Z) to P, Y, Z, we have that (a_P, a_Q, a_Z) is not a model of T, or (a_P, a_Q, a_Z) is a model of F or there is a model (a'_P, a'_Q, a'_Z) of T such that $a'_P \subset a_P$ and $a'_Q = a_Q$. This can be written as a $\forall\exists$-formula as follows:

$$\psi := \forall P \forall Q \forall Z \exists P' \exists Z' (\neg T(P, Q, Z) \vee F(P, Q, Z) \vee (T(P', Q, Z') \wedge P' \subset P)).$$

We first compute a tree decomposition of $T \wedge F$ of width $O(k)$ in time $2^{O(k)}(|T| + |F|)$. We can use Lemma 1, Lemma 2 and Proposition 1 to compute in time $\mathrm{poly}(k)(|T| + |F|)$ a CNF-formula ϕ such that the matrix of ψ is a projection of ϕ and ϕ has incidence treewidth $O(k)$. Applying Corollary 2, yields the result. □

We now show that Theorem 9 is essentially optimal by analyzing the proof in [15].

Theorem 10. *There is no algorithm that, given an instance T, P, Q, Z and F of size n and treewidth k, decides if* $\mathrm{MM}(T, P, Q, Z) \models F$ *in time* $2^{2^{o(k)}} 2^{o(n)}$, *unless ETH is false.*

Proof. Let $\psi = \forall X \exists Y \phi$ be a QBF with $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_\ell\}$. We define the theory T as follows:

$$T = \left(\bigwedge_{i=1}^{m} (x_i \neq z_i) \right) \wedge ((u \wedge y_1 \wedge \ldots y_\ell) \vee \phi),$$

where $z_1, \ldots, z_m$ and u are fresh variables. Set $P = \mathsf{var}(T)$ and $Q = \emptyset$ and Z the rest of the variables. In [15], it is shown that all $\mathrm{MM}(T, P, Q, Z) \models \neg u$ if and only if ϕ is true. Now using Lemma 1 we turn ψ into a 2-QBF ψ' with the same properties. Note that ψ' has treewidth $O(k)$ and $O(m + \ell)$ variables and thus the claim follows directly with Theorem 2. □

8 Minimal Unsatisfiable Subsets

Faced with unsatisfiable CNF-formula, it is in many practical setting highly interesting to find the sources of unsatisfiability. One standard way of describing them if by so-called minimal unsatisfiable sets. A *minimal unsatisfiable set (short MUS)* is an unsatisfiable set $\mathcal{C}$ of clauses of a CNF-formula such that every proper subset of $\mathcal{C}$ is satisfiable. The computation of MUS has attracted a lot of attention, see e.g. [23, 20, 35] and the references therein.

In this section, we study the following question: given a CNF-formula ϕ and a clause C, is C contained in a MUS of ϕ? Clauses for which this is the case can in a certain sense be considered as not problematic for the satisfiability of ϕ. As for the other problems studied in this paper, it turns out that the above problem is complete for the second level of the polynomial hierarchy, more specifically for Σ_p^2 [27]. Treewidth reductions seem to not have been considered before, but we show that our approach gives a linear time algorithm in a simple way.

Theorem 11. *There is an algorithm that, given a CNF-formula ϕ incidence treewidth k and a clause C of ϕ, decides C is in a MUS of ϕ in time $2^{2^{O(k)}}|\phi|$.*

Proof. Note that C is in a MUS of ϕ if and only if there is an unsatisfiable clause set $\mathcal{C}$ such that $C \in \mathcal{C}$ and $\mathcal{C} \setminus \{C\}$ is satisfiable. We will encode this in $\forall\exists$-QBF. In a first step, similarly to the proof of Lemma 1, we add a new variable x_C for every clause C of ϕ and substitute ϕ by clauses expressing $C \leftrightarrow x_c$. Call the resulting formula ψ. It is easy to see that the incidence treewidth of ψ is at most double that of ϕ. Moreover, for every assignment a to $\mathsf{var}(\phi)$, there is exactly one extension to a satisfying assignment a' of ψ. Moreover, in a' a clause variable x_C is true if and only if a satisfies the clause C. Let $\mathcal{C}$ be a set of clauses, then $\mathcal{C}$ is unsatisfiable if and only if for every assignment a to $\mathsf{var}(\phi)$, $\mathcal{C}$ is not contained in the set of satisfied clauses. Interpreting sets by assignments as before, we can write this as a formula by

$$\psi'(\mathcal{C}) := \forall X \forall \mathcal{C}' : \psi_C(X, \mathcal{C}') \rightarrow \neg(\mathcal{C} \subseteq \mathcal{C}').$$

Let now $\mathcal{C}$ range over the sets of clauses not containing C. Then we have by the considerations above that C appears in a MUS if and only if

$$\begin{aligned}\psi^* &= \exists \mathcal{C} \psi'(\mathcal{C} \cup \{C\}) \wedge \neg\psi'(\mathcal{C}) \\ &= \exists \mathcal{C} \exists X' \exists \mathcal{C}' \forall X'' \forall \mathcal{C}''(\phi_C(X', \mathcal{C}') \rightarrow \neg(\mathcal{C} \cup \{C\} \subseteq \mathcal{C}')) \wedge \phi_C(X', \mathcal{C}'') \wedge \mathcal{C} \subseteq \mathcal{C}''\end{aligned}$$

Negating and rewriting the matrix of the resulting QBF with Lemma 1, we get in linear time a $\forall\exists$-QBF of treewidth $O(k)$ that is true if and only if C does not appear in a MUS of ϕ. Using Theorem 2 completes the proof. □

We now show that Theorem 11 is essentially tight.

Theorem 12. *There is no algorithm that, given a CNF-formula ϕ with n variables and primal treewidth k and a clause C of ϕ, decides if C is in a MUS of ϕ in time $2^{2^{o(k)}}2^{o(n)}$, unless ETH is false.*

Proof. Given a $\forall\exists$-QBF $\psi = \forall X \exists Y \phi$ of incidence treewidth k where $C_1, \ldots, C_m$ are the clauses of ϕ, we construct the CNF-formula

$$\phi' = \bigwedge_{x \in X} (x \wedge \neg x) \wedge w \wedge \bigwedge_{i=1}^{m} \neg w \vee C_i.$$

In [27] it is shown that ψ is true if and only if the clause w appears in a MUS of ϕ'. Note that ϕ' has primal treewidth $k+1$: in a tree decomposition of the

primal graph of ϕ, we can simply add the variable w into all bags to get a tree decomposition of the primal graph of ϕ'. Since clearly $|\phi'| = O(|\phi|)$, any algorithm to check if w is in a MUS of ϕ' in time $2^{2^{o(k)}}2^{o(n)}$ contradicts ETH with Theorem 2. □

9 Conclusion

In this paper, we took an alternate approach in the design of optimal algorithms mainly for the second level of the polynomial hierarchy parameterized by treewidth: we used reductions to QBF-2.We stress that, apart from some technical transformations on CNF-formulas which we reused throughout the paper, our algorithms are straightforward and all complexity proofs very simple. We consider this as a strength of what we propose and not as a lack of depth, since our initial goal was to provide a black-box technique for designing optimal linear-time algorithms with an asymptotically optimal guarantee on the treewidth. We further supplement the vast majority of our algorithms by tight lower-bounds, using ETH reductions again from QBF-2.

We concentrated on areas of artificial intelligence, investigating a collection of well-studied and diverse problems that are complete for Σ_p^2 and Π_p^2. However we conjecture that we could apply our approach to several problems with similar complexity status. Natural candidates are problems complete for classes in the polynomial hierarchy, starting from the second level, see e.g. [34] for an overview (mere NP-complete problems can often be tackled by other successful techiques).

Of course, our approach is no silver bullet that magically makes all other techniques obsolete. On the one hand, for problems whose formulation is more complex than what we consider here, Courcelle's theorem might offer a richer language to model problems than QBF. This is similar in spirit to some problems being easier to model in declarative languages like ASP than in CNF. On the other hand, handwritten algorithms probably offer better constants than what we get by our approach. For example, the constants in [12] are more concrete and smaller than what we give in Section 5. However, one could argue that for double-exponential dependencies, the exact constants probably do not matter too much simply because already for small parameter values the algorithms become infeasible[4]. Despite these issues, in our opinion, QBF encodings offer a great trade-off between expressivity and tightness for the runtime bounds and consider it as a valuable alternative.

Acknowledgments

Most of the research in this paper was performed during a stay of the first and third authors at CRIL that was financed by the project PEPS INS2I 2017 CODA. The second author is thankful for many valuable discussions with members of CRIL, in particular Jean-Marie Lagniez, Emmanuel Lonca and Pierre Marquis, on the topic of this article.

[4] To give the reader an impression: $2^{2^5} \approx 4.2 \times 10^9$ and already $2^{2^6} \approx 1.8 \times 10^{19}$.

References

1. Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for np-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, 1989.
2. Albert Atserias and Sergi Oliva. Bounded-width QBF is PSPACE-complete. *J. Comput. Syst. Sci.*, 80(7):1415–1429, 2014.
3. Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.
4. Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. An o(cˆk n) 5-approximation algorithm for treewidth. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 499–508, 2013.
5. Marco Cadoli and Maurizio Lenzerini. The complexity of propositional closed world reasoning and circumscription. *J. Comput. Syst. Sci.*, 48(2):255–310, 1994.
6. Hubie Chen. Quantified constraint satisfaction and bounded treewidth. In Ramon López de Mántaras and Lorenza Saitta, editors, *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004*, pages 161–165, 2004.
7. Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.
8. DIMACS. Satisfiability: Suggested Format. *DIMACS Challenge. DIMACS*, 1993.
9. Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.
10. Paul E. Dunne. Computational properties of argument systems satisfying graph-theoretic constraints. *Artif. Intell.*, 171(10-15):701–729, 2007.
11. Paul E. Dunne and Trevor J. M. Bench-Capon. Coherence in finite argument systems. *Artif. Intell.*, 141(1/2):187–203, 2002.
12. Wolfgang Dvorák, Reinhard Pichler, and Stefan Woltran. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artif. Intell.*, 186:1–37, 2012.
13. Eduard Eiben, Robert Ganian, and Sebastian Ordyniak. Using decomposition-parameters for QBF: mind the prefix! In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 964–970, 2016.
14. Eduard Eiben, Robert Ganian, and Sebastian Ordyniak. Small resolution proofs for QBF using dependency treewidth. *CoRR*, abs/1711.02120, 2017.
15. Thomas Eiter and Georg Gottlob. Propositional circumscription and extended closed-world reasoning are π_2^p-complete. *Theor. Comput. Sci.*, 114(2):231–245, 1993.
16. Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. *J. ACM*, 42(1):3–42, 1995.
17. Eldar Fischer, Johann A. Makowsky, and Elena V. Ravve. Counting truth assignments of formulas of bounded tree-width or clique-width. *Discrete Applied Mathematics*, 156(4):511–529, 2008.
18. Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004.
19. Georg Gottlob, Reinhard Pichler, and Fang Wei. Bounded treewidth as a key to tractability of knowledge representation and reasoning. *Artif. Intell.*, 174(1):105–132, 2010.

20. Alexey Ignatiev, Alessandro Previti, Mark H. Liffiton, and Joao Marques-Silva. Smallest MUS extraction with minimal hitting set dualization. In *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015*, volume 9255, pages 173–182, 2015.
21. Michael Jakl, Reinhard Pichler, Stefan Rümmele, and Stefan Woltran. Fast counting with bounded treewidth. In *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2008*, volume 5330, pages 436–450, 2008.
22. Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.
23. Jean-Marie Lagniez and Armin Biere. Factoring out assumptions to speed up MUS extraction. In Matti Järvisalo and Allen Van Gelder, editors, *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, volume 7962 of *Lecture Notes in Computer Science*, pages 276–292. Springer, 2013.
24. Jean-Marie Lagniez, Emmanuel Lonca, and Jean-Guy Mailly. Coquiaas: A constraint-based quick abstract argumentation solver. In *27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2015*, pages 928–935, 2015.
25. Michael Lampis and Valia Mitsou. Treewidth with a quantifier alternation revisited. In *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, 2017.
26. Alexander Langer, Felix Reidl, Peter Rossmanith, and Somnath Sikdar. Practical algorithms for MSO model-checking on tree-decomposable graphs. *Computer Science Review*, 13-14:39–74, 2014.
27. Paolo Liberatore. Redundancy in logic I: CNF propositional formulae. *Artif. Intell.*, 163(2):203–232, 2005.
28. Vladimir Lifschitz. Circumscription. In Dov M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming (Vol. 3)*, pages 297–352. Oxford University Press, Inc., New York, NY, USA, 1994.
29. Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.
30. Dániel Marx and Valia Mitsou. Double-exponential and triple-exponential bounds for choosability problems parameterized by treewidth. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, pages 28:1–28:15, 2016.
31. John McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artif. Intell.*, 28(1):89–116, 1986.
32. Guoqiang Pan and Moshe Y. Vardi. Fixed-parameter hierarchies inside PSPACE. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 27–36, 2006.
33. Marko Samer and Stefan Szeider. Algorithms for propositional model counting. *J. Discrete Algorithms*, 8(1):50–64, 2010.
34. Marcus Schaefer and Christopher Umans. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT news*, 33(3):32–49, 2002.
35. João P. Marques Silva and Inês Lynce. On improving MUS extraction algorithms. In *Theory and Applications of Satisfiability Testing - SAT 2011*, pages 159–173, 2011.

A Proof of Proposition 1

In this section we prove that given a CNF formula with bounded incidence treewidth and unbounded arity we can compute in linear time a 3CNF formula which has essentially the same treewidth (give or take a constant).

Proposition 2. *There is an algorithm that, given a CNF formula ϕ of incidence treewidth k, computes in time $2^{O(k)}|\phi|$ a 3CNF formula ϕ' of incidence treewidth $O(k)$ with $\mathsf{var}(\phi) \subseteq \mathsf{var}(\phi')$ such that ϕ is a projection of ϕ'.*

Proof. We use the classic reduction from SAT to 3SAT that cuts big clauses into smaller clauses by introducing new variables. During this reduction we have to take care that the runtime is in fact linear and that we can bound the treewidth appropriately.

In a first step, we compute a tree decomposition $(T, (B_t)_{t \in T})$ of width k for the incidence graph of ϕ in time $2^{O(k)}|\phi|$ with the algorithm from [4].

We store ϕ in the following format: for every clause C we have a doubly linked list L_C storing pointers to all variables in C and their polarity, i.e., if they appear positively or negatively. Moreover, for every variable x we have a doubly linked list L_x storing pointers to the clauses that x appears in. So far, this is essentially an adjacency list representation of the incidence graph of ϕ. As additional information, we add for every entry in L_C that points to a variable x also a pointer to the cell in L_x that points to C. Symmetrically, we add in the cell pointing to C in L_x a pointer to the cell in L_c pointing towards x. The purpose of this data structure is that it allows us efficient deletions: if we are in a cell in L_x that points towards C, we can delete the edge xC from the adjacency list in constant time without having to search for the right entry in L_C. Similarly, if we have the entry in L_C representing the edge xC, we can delete this edge in constant time. Note that this data structure can be computed easily in linear time in a single pass over ϕ.

We now construct the formula ϕ' along a postfix DFS order on T (that can easily be computed in linear time). For each clause node C appearing in a bag, we store a variable $F_{B,C}$ that will be put in the next clause we print out for C. $F_{B,C}$ will always be a variable that does not appear outside the subtree below B in T and $F_{B,C}$ might be empty.

We now describe the construction in the different types of nodes B in T:

- if B introduces a new variable, we copy the values $F_{B,C}$ from its child and do nothing else.
- if B introduces a new clause C, we initialize $F_{B,C}$ as empty and copy all other values $F_{B,C'}$ from the child node.
- if B is forget node for a variable x, we first copy all $F_{B,C}$ as before. Then, for every clause C in B such that C contains x, we do the following: if $F_{B,C}$ is empty, we set $F_{B,C}$ to x with the polarity as in C. If $F_{B,C}$ contains a literal ℓ, we write out a clause $\ell \vee \ell_x \vee z$ where ℓ_x is the variable x with the same polarity as in C and z is a fresh variable we have not used before. Then we set $F_{B,C}$ to $\neg z$. Finally, in any case, we delete the edge xC in our data structure.

– if B is a forget node for a clause C, we again first copy all $F_{B,C}$. Then define the set S_C that consists of $F_{B,C}$ and all literals whose variables are in B that appear in C . We arbitrarily split S into a 3CNF by adding some more fresh variables and print it out. Afterwards, we delete all edges containing x in our data structure.
– if B is a join node with two children B_1 and B_2, we compute the $F_{B,C}$ as follows: if $F_{B_1,C}$ and $F_{B_2,C}$ are empty, we set $F_{B,C}$ empty as well. If exactly one of the $F_{B_1,C}$ and $F_{B_2,C}$ contains a literal, we set $F_{B,C}$ to that literal. If $F_{B_1,C}$ contains ℓ_1 and $F_{B_2,C}$ contains ℓ_2, we print out a clause $\ell_1 \vee \ell_2 \vee z$ for a fresh variable z and set $F_{B,C}$ to $\neg z$.

This completes the algorithm. The clauses we have printed out in the various steps form the formula ϕ'. We have to check that the algorithm indeed runs in linear time in $|\phi|$ and that ϕ' has the desired properties.

Obviously, ϕ' is in 3CNF, because all clauses we print out only contain at most 3 variables.

We next argue that the treewidth of ϕ' is $O(k)$. To this end, we construct a tree decomposition $(T, (B'_t)_{t\in T}$. For every bag B_t the corresponding bag B'_t contains all variables in B and all variables in the $F_{B_t,C}$. Moreover, for every B_t for which the algorithm prints out a clause C', we put C' and the variables of C' into B'_t. Since in every bag we print out at most k clauses and all of them have size at most 3, the resulting B'_t has size $O(k)$. By construction, the bags B'_t cover all edges in the incidence graph of ϕ'. Finally, the connectivity condition is easy to verify. It follows that the treewidth of ϕ' is $O(k)$.

We next claim that the construction of ϕ' from ϕ and $(T, (B_t)_{t\in T})$ can be done in time $O(\text{poly}(k)|\phi|)$ with the help of our data structure. To see this, first observe when a variable is forgotten in a bag B, it contains only at most k neighbors in our data structure and those all lie in B. This is because for all neighbors that have been forgotten before, the corresponding edges have been deleted in the adjacency lists. The same is true when forgetting clauses. Thus, we can find the clauses that a variable is in in time $O(k)$. Since we can delete edges in constant time, it is easy to see that every bag can be treated in time polynomial in k. Since T can be assumed to be of size linear in $|\phi|$, the desired runtime bound follows.

Finally, it is easy to see that ϕ is a projection of ϕ'. This follows exactly as in the usual reduction from SAT to 3SAT. The only slight difference is that instead of cutting of pieces of the formula from the left to the right side, we decompose clauses potentially in a treelike fashion which results in clauses that contain only fresh variables. However, this does changes neither the correctness of the reduction not the argument. □

B Bounded Treewidth *k*-QBF in Linear Time

We sketch a proof for linear runtime in Theorem 1 and refer the reader to [6] for the technical details. In fact, since this will not be much more work, we treat the case of r-QBF, the generalization of 2-QBF to r quantifier blocks.

To give the runtime bound, define $g(r,k)$ recursively by $g(0,k) = k$ and $g(r+1,k) = 2^{g(k,r)}$. We now state the linear time version of the main result in [6].

Theorem 13. *There is an algorithm that given a r-QBF of primal treewidth k decides in time $g(r, O(k))|\phi|$ if ϕ is true.*

We remark that Theorem 13 was already observed in [32] but that paper gives no justification of that claim, we decided to give some more details.

The crucial data structure in [6] are *choice constraints* which consist of a variable scope and a rooted tree with unbounded fanout in which all leaves are at depth r and for every leaf a relation $R \subseteq \{0,1\}^s$.

Chen then defines *choice quantified formulas* which are consist of a variable prefix and a conjunction of choice constraints in the variables of the prefix. We omit the semantics of choice quantified formulas here since they are not important for our sketch and refer the reader to [6] for details. We remark however that any given QBF ϕ where all clauses have at most k variables can be turned in time $O(2^k n)$ into a choice quantified formula ψ such that ϕ is true if and only if ψ is.

We define a notion of equivalence for nodes in choice constraints: leaves are equivalent if and only if they have the same relation. Equivalence of two nodes t, t' of depth $\ell < r$ is defined recursively: let $t_1 \ldots, t_s$ be the children of t and $t'_1, \ldots, t'_s$ be children of t'. Then t and t' are equivalent if and only if for every t_i there is an equivalent t'_j and for every t'_j there is an equivalent t_i[5]. A choice constraint is in normal form if and only if no node has any equivalent children.

Chen shows that one can *normalize* a choice constraint by deleting iteratively for all equivalent pairs of nodes one of them and its subtree. Crucially, applying this operation for a constraint in a choice quantified formula yields an equivalent formula.

Observation 14 *There are $g(r+1,k)$ non-equivalent normal choice constraints with scope of size k whose leaves are at depth r and all of these constrains have size $g(r,2k)2^k$.*

Proof. Choice constraints of depth 0 are just relations of arity k over $\{0,1\}$, so there are $g(1,k) = 2^k$ of those. Moreover, each of those relations can be decribed in size $2^{O(k)}$, e.g. by a value table.

For $r > 0$, by definition of equivalence, the root can have as children any subset of normal choice constraints of depth $r-1$ and the same scope. Since there are by induction $g(r,k)$ of those, the first claim follows by definition of g. The description size is at most $g(r,k) \cdot g(r-1,2k)2^k \leq g(r,2k)2^k$ □

[5] We remark that the notion of equivalence in [6] is slightly different. We chose to modify the definition since our notion gives slightly smaller size bounds for normalized choice constraints and have the same properties concerining truth of the resulting formulas.

Note that we can check the equivalence of two children in time polynomial in the size of a given choice constraint and thus normalization can also be done in time $\text{poly}(g(r, 2k)2^k) = g(r, O(k))$.

Chen also introduces a polynomial time computable join operation on choice constraints with the property that substituting two choice constraints by their join in a choice quantified formula one gets a new formula that is equivalent to the old. A naive solution to solve choice quantified formulas would thus be to simply join all its choice constraints and then check the single resulting constraint. The problem with this is that it would grow the variables scope of the resulting constraint (the variables of a join are the unions those of the joined constraints) such that the size bound in Observation 14 would become meaningless and the runtime bound would explode.

The solution to this is working along a tree decomposition: in every forget node, one joins all choice constraints having the forgotten variable in their scope. Note that these choice constraints and thus also the resulting join only have at most the $k + 1$ variables of the current bag in the scope, so by normalizing after every join, the resulting choice constraint will have size at most $g(r, O(k))$. Now the variable to forget only appears in a single choice constraint and Chen shows how to forget it in that case in an operation that may grow the choice constraint but by applying normalization during this forget operation one maintains the size bound of Observation 14. Applying this on all forget nodes iteratively, one gets a trivial choice constraint formula that can be decided in constant time.

Let us explain why this algorithm runs within the claimed time bounds: by similar preprocessing than that in Proposition 1, we can make sure that for every node that is forgotten, we can look up all constraints it appears in time linear in the number of those constraints, so at most $O(2^k)$. Now we compute at most 2^k pairwise joins followed by normalization which run each in time $g(r, O(k))$. Thus the overall time for joining and normalizing is $2^k g(r, O(k)) = g(r, O(k))$. Observing that the forgetting can also be done in polynomial time and thus in $g(r, O(k))$ leads to an overall cost per forgotten variables of $g(r, O(k))$. Now noting that the computation of the tree decomposition and a traversal to find the order in which the variables are forgotten can be done in time $2^{O(k)}n$ where n is the number of variables, completes the proof.