Logical Methods in Computer Science
Vol. 8 (1:32) 2012, pp. 1–32
www.lmcs-online.org

Submitted Mar. 2, 2010
Published Mar. 29, 2012

TREE-WIDTH FOR FIRST ORDER FORMULAE

ISOLDE ADLER [a] AND MARK WEYER [b]

[a] Institut für Informatik, Goethe-Universität Frankfurt am Main
e-mail address: iadler@informatik.uni-frankfurt.de

[b] not affiliated
e-mail address: mark@weyer-zuhause.de

Abstract. We introduce tree-width for first order formulae φ, fotw(φ). We show that computing fotw is fixed-parameter tractable with parameter fotw. Moreover, we show that on classes of formulae of bounded fotw, model checking is fixed parameter tractable, with parameter the length of the formula. This is done by translating a formula φ with fotw(φ) $< k$ into a formula of the k-variable fragment $\mathcal{L}^k$ of first order logic. For fixed k, the question whether a given first order formula is equivalent to an $\mathcal{L}^k$ formula is undecidable. In contrast, the classes of first order formulae with bounded fotw are fragments of first order logic for which the equivalence is decidable.

Our notion of tree-width generalises tree-width of conjunctive queries to arbitrary formulae of first order logic by taking into account the quantifier interaction in a formula. Moreover, it is more powerful than the notion of elimination-width of quantified constraint formulae, defined by Chen and Dalmau (CSL 2005): for quantified constraint formulae, both bounded elimination-width and bounded fotw allow for model checking in polynomial time. We prove that fotw of a quantified constraint formula φ is bounded by the elimination-width of φ, and we exhibit a class of quantified constraint formulae with bounded fotw, that has unbounded elimination-width. A similar comparison holds for strict tree-width of non-recursive stratified datalog as defined by Flum, Frick, and Grohe (JACM 49, 2002).

Finally, we show that fotw has a characterization in terms of a cops and robbers game without monotonicity cost.

1. Introduction

Model checking is an important problem in complexity theory. It asks for a given formula φ of some class $\mathcal{C}$ of formulae and a structure $\mathcal{A}$, whether $\mathcal{A}$ satisfies φ.

MC($\mathcal{C}$)
Input: A structure $\mathcal{A}$ and a formula $\varphi \in \mathcal{C}$.
Question: $\mathcal{A} \models \varphi$?

1998 ACM Subject Classification: F.2, F.4.1, H.2.3.

Key words and phrases: treewidth, model checking, conjunctive queries, quantified constraint formulae, first-order logic, elimination-width, cops and robbers game.

DOI:10.2168/LMCS-8 (1:32) 2012

Let $\mathcal{L}$ denote first order logic. It is well-known, that $\mathrm{MC}(\mathcal{L})$ is PSPACE-complete. Motivated by this, much research has been done on finding fragments of $\mathcal{L}$ having a tractable model checking problem. For instance, for fixed k, the problem $\mathrm{MC}(\mathcal{L}^k)$ can be solved in polynomial time, where $\mathcal{L}^k$ denotes the fragment of first order formulae with at most k variables (see e.g. [17]).

The class of *conjunctive queries*, CQ, is an important fragment of first order logic. Many queries that occur in practice are conjunctive queries, and model checking of conjunctive queries on relational databases (i.e. relational structures) is an important and well-studied problem in database theory [29, 8, 18, 10, 20, 23]. It is equivalent to conjunctive query containment, to the constraint satisfaction problem studied in artificial intelligence and to the homomorphism problem for structures [7, 15]. A *conjunctive query* is a first order formula starting with a quantifier prefix using only existential quantifiers, followed by a conjunction of relational atoms. While MC(CQ) is NP-hard in general, several researchers proved independently that conjunctive queries of bounded *tree-width* can be evaluated in polynomial time [8, 18]. One way to prove this is the following. Suppose φ is a conjunctive query having tree-width k. Then we can compute a tree decomposition of width k in linear time using Bodlaender's algorithm [6]. From the decomposition we can actually read off the syntax of an equivalent formula $\varphi' \in \mathcal{L}^{k+1}$. Finally, we use the fact that $\mathrm{MC}(\mathcal{L}^{k+1})$ is solvable in polynomial time. Essentially, bounded tree-width is even necessary for polynomial time solvability of MC(CQ) [24, 22].

In this paper, we introduce a notion of *tree-width for first order formulae* φ, $\mathrm{fotw}(\varphi)$. Our notion generalises the notion of tree-width of conjunctive queries, and we show that the class $\mathcal{C}_k$ of all first order formulae φ with $\mathrm{fotw}(\varphi) \leq k$ satisfies the following properties.

(1) $\mathcal{C}_k$ has a polynomial time membership test (Corollary 4.8).
(2) $\mathcal{C}_k$ has the same expressive power as $\mathcal{L}^{k+1}$, the fragment of first order formulae with at most $k+1$ variables (Theorem 5.5).
(3) There is an algorithm that computes for given $\varphi \in \mathcal{C}_k$ an equivalent formula $\varphi' \equiv \varphi$ with $\varphi' \in \mathcal{L}^{k+1}$ (Theorem 5.5).
(4) $\mathrm{MC}(\mathcal{C}_k)$ is fixed parameter tractable with parameter the length of φ, i.e. for input $\varphi \in \mathcal{C}_k$ and $\mathcal{A}$, the running time is $p(\|\mathcal{A}\|)f(|\varphi|)$ for a polynomial p and a computable function f (Corollary 5.6).

Obviously, properties 1 and 3 imply property 4. While $\mathrm{MC}(\mathcal{L}^k)$ is solvable in polynomial time, we do not obtain a polynomial algorithm for $\mathrm{MC}(\mathcal{C}_k)$. Nevertheless, in typical applications one can expect the length of the formula to be small compared to the size of the structure (database). For a fixed formula the running time is polynomial, and moreover, the problem is *fixed-parameter tractable* (in FPT), meaning that changing φ does not alter the exponent of the polynomial (see [13, 17]).

Note that for fixed $k > 0$ it is undecidable, whether a first order formula φ is equivalent to an $\mathcal{L}^k$ formula. Hence it is not surprising that our notion of k-bounded first order tree-width does not capture semantic equivalence to $\mathcal{L}^k$ (we will give more details in Section 5).

Quantified constraint formulae generalise conjunctive queries by allowing arbitrary quantifiers in the quantifier prefix. In [9], Chen and Dalmau introduce *elimination orderings* for quantified constraint formulae. These elimination orderings must respect the quantifier prefix. In this way, Chen and Dalmau obtain a notion of *elimination-width*[1], which

[1]Actually, the notion is called *tree-width* for quantified constraint formulae in [9], But since the notion is defined via elimination orderings, we prefer the term *elimination-width*.

allows for model checking of quantified constraint formulae of bounded elimination-width in polynomial time, using a consistency algorithm. Hereby, they answer a question posed in [19] positively, whether bounded tree-width methods work for formulae more general than conjunctive queries. Introducing a notion of tree-width for arbitrary first order formulae, we even go further. We show that for quantified constraint formulae φ, elimination-width of φ is at least as large as $\mathrm{fotw}(\varphi)$, and we exhibit a class of quantified constraint formulae with bounded first order tree-width and unbounded elimination-width. We show that quantified constraint formulae of bounded fotw allow for model checking in polynomial time. Hence fotw is more powerful than elimination-width.

In [16], Flum, Frick and Grohe introduce *strict tree-width*[2] for *non-recursive stratified datalog (*NRSD*) programs*. They show that model checking for NRSD programs of bounded strict tree-width can be done in polynomial time. Since NRSD programs have a canonical translation into first order formulae, our notion of tree-width can be transfered from first order formulae to NRSD programs. We show that if an NRSD program Π has strict tree-width at most k, then the formula φ_Π obtained from Π has elimination-width at most k and hence it satisfies $\mathrm{fotw}(\varphi_\Pi) \leq k$. Again there are classes of NRSD programs with unbounded strict tree-width, whose corresponding first order formulae have bounded first order tree-width. Hence our notion of first order tree-width yields larger subclasses of $\mathcal{L}$, that still allow for tractable model checking.

Actually, we introduce first order tree-width as a special case of a more abstract notion which we term stratified tree-width. We expect that stratified tree-width will find further, quite different, applications.

The rest of this paper is organised as follows. Section 2 fixes some terminology. Section 3 introduces the notion of stratified tree-width, the special case of first order tree-width, and the notion of *xenerp normal form* of a formula φ – a kind of opposite of prenex normal form. We show that fotw is invariant under transformation into xenerp normal form. In Section 3.4 we relate fotw to the natural notion of tree-width stratified by the alternation depth of a formula. In Section 4 we show how to compute stratified tree decompositions and, in particular, how to compute first order tree-width. In Section 5 we prove that bounded first order tree-width is expressively equivalent to bounded variable fragments of first order logic and that model checking for formulae of bounded first order tree-width is fixed-parameter tractable. In Section 6 we relate our notion to existing notions and give a game characterisation of stratified tree-width. We conclude with some open problems in Section 7.

We wish to thank the anonymous referees for many useful suggestions.

2. Well-known definitions

A *vocabulary* $\sigma = \{R_1, \ldots, R_n, c_1, \ldots, c_m\}$ is a finite set of *relation symbols* R_i, $1 \leq i \leq n$, and *constant symbols* c_j, $1 \leq j \leq m$. Every R_i has an associated *arity*, an integer $\mathrm{ar}(R_i) > 0$. A *σ-structure* is a tuple $\mathcal{A} = (A, R_1^{\mathcal{A}}, \ldots, R_n^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_m^{\mathcal{A}})$ where A is a finite set, the *universe* of $\mathcal{A}$, $R_i^{\mathcal{A}} \subseteq A^{\mathrm{ar}(R_i)}$ for $1 \leq i \leq n$, and $c_j^{\mathcal{A}} \in A$ for $1 \leq j \leq m$.

Given a σ-structure $\mathcal{A}$ we distinguish between the cardinality $|A|$ of the universe A of $\mathcal{A}$ and the *size* $\|\mathcal{A}\|$ of $\mathcal{A}$, given by $\|\mathcal{A}\| = |\sigma| + |A| + \sum_{i=1}^{n} \left|R_i^{\mathcal{A}}\right| \cdot \mathrm{ar}(R_i)$.

[2] In [16], the authors also introduce a notion of *tree-width for first order formulae*. But their notion disregards the quantifier interaction, and they only use it for conjunctive queries with negation.

We use $\mathcal{L}$ to denote relational first order logic with constants, and for simplicity, we refer to $\mathcal{L}$ as *first order logic*. We assume that the reader is familiar with the basic notions of first order logic (see for instance [14]). For a formula φ we let $\text{free}(\varphi)$ denote the set of free variables of φ. A formula φ is a *sentence*, if $\text{free}(\varphi) = \emptyset$. We sometimes write $\varphi(x_1, \ldots, x_n)$ to indicate that $\text{free}(\varphi) \subseteq \{x_1, \ldots, x_n\}$.

For a structure $\mathcal{A}$, a formula $\varphi(x_1, \ldots, x_n)$, and elements $a_1, \ldots a_n \in A$ we write $\mathcal{A} \models \varphi(a_1, \ldots a_n)$ to denote that $\mathcal{A}$ satisfies φ if the variables $x_1, \ldots, x_n$ are interpreted by $a_1, \ldots, a_n$, respectively. We let

$$\varphi(\mathcal{A}) := \{(a_1, \ldots, a_n) \mid \mathcal{A} \models \varphi(a_1, \ldots, a_n)\}.$$

For sentences we have $\varphi(\mathcal{A}) = \text{TRUE}$, if $\mathcal{A}$ satisfies φ, and FALSE otherwise. If the vocabularies of φ and $\mathcal{A}$ are different, we let $\varphi(\mathcal{A}) = \emptyset$.

The *Query Evaluation Problem* for a class $\mathcal{C}$ of formulae is the following problem:

EVAL($\mathcal{C}$)
Input: A structure $\mathcal{A}$ and a formula $\varphi \in \mathcal{C}$.
Problem: Compute $\varphi(\mathcal{A})$.

Note that if φ is a sentence, then EVAL($\mathcal{C}$) and MC($\mathcal{C}$) coincide. We say that a formula $\varphi \in \mathcal{L}$ is *straight*, if no variable in φ is quantified over twice, if no free variable is also a quantified variable, and if each quantified variable actually occurs in some atom. All formulae are straight, unless stated otherwise. Moreover, we assume that all formulae are in negation normal form, i.e. the negation symbols only appear in front of atoms.

We denote a graph G as a pair $G = (V(G), E(G))$, where the set $V(G)$ of *vertices* is finite, and every *edge* $e \in E(G)$ is a two-element subset of $V(G)$. A *tree decomposition* of a graph $G = (V, E)$ is a pair (T, B), consisting of a rooted tree T and a family $B = (B_t)_{t \in T}$ of subsets of V, the *pieces* of T, satisfying:

- **(TD1):** For each $v \in V$ there exists $t \in T$, such that $v \in B_t$. We say the node t *covers* v.
- **(TD2):** For each edge $e \in E$ there exists $t \in T$, such that $e \subseteq B_t$. We say the node t *covers* e.
- **(TD3):** For each $v \in V$ the set $\{t \in T \mid v \in B_t\}$ is connected in T.

The *width* of (T, B) is defined as $\text{w}(T, B) := \max\big\{\, |B_t| \;\big|\; t \in T \big\} - 1$.

The *tree-width of* G is defined as

$$\text{tw}(G) := \min\big\{\text{w}(T, B) \;\big|\; (T, B) \textit{ is a tree decomposition of } G\big\}.$$

Fact 2.1. Every graph G of tree-width at most k has at most $k \cdot |V(G)|$ edges. □

Fact 2.1 can be shown by induction on the number of vertices (see e.g. [17]). We will make frequent use of the following well-known fact about tree decompositions (see [11]):

Fact 2.2. Let (T, B) be a tree decomposition of some graph G, and let $C \subseteq V(G)$. If for all $v, w \in C$, some piece of (T, B) covers both v and w, then there is some piece B_t covering C entirely, i.e. $C \subseteq B_t$.

In particular, every clique in G is covered by some piece. □

3. First order tree-width

3.1. **Stratified tree-width.** We start with defining stratified tree-width. Then, first order tree-width is defined as a special case. Although it is our only application of stratified tree-width, stating results in greater generality allows us to focus on their essence. It is also quite possible, that further applications will arise in the future.

Any rooted tree T induces a natural partial order $<_T$ on its nodes, where the smallest element is the root. For a tree decomposition (T, B) of a graph G and a vertex $v \in V(G)$, let $t_v \in T$ denote the $<_T$-minimal tree node that covers v. By (TD3), the node t_v is well-defined. Now, let $d\colon V(G) \to \mathbb{N}$ be a function. We say that a tree decomposition (T, B) of G is *d-stratified*, if all $u, v \in V(G)$ with $t_u <_T t_v$ satisfy $d(u) \leq d(v)$. The *tree-width of* (G, d) is defined as

$$\mathrm{tw}(G, d) := \min\big\{\mathrm{w}(T, B) \;\big|\; (T, B) \text{ is a } d\text{-stratified tree decomposition of } G\big\}.$$

It will sometimes be convenient to work with an alternative characterization of stratified tree width: let $G = (V, E)$ be a graph and $d : V \to \mathbb{N}$. An *elimination ordering* of (G, d) is a linear ordering $(v_1, \ldots, v_n)$ of V which *respects* d, i.e. $i < j$ implies $d(v_i) \leq d(v_j)$. With an elimination ordering we associate a sequence of graphs as follows:

- $G_n := G$
- $V(G_{i-1}) := V(G_i) \setminus \{v_i\}$, and
- $E(G_{i-1}) := \big\{e \in E(G_i) \;\big|\; v_i \notin e\big\} \cup \big\{\{u, w\} \;\big|\; u \neq w,\ \{u, v_i\}, \{v_i, w\} \in E(G_i)\big\}$ for $1 < i \leq n$.

The *width* of the elimination ordering is $\max_{i\in[n]}\{\deg(v_i) \text{ in } G_i\}$. The *elimination-width* of (G, d), $\mathrm{ew}(G, d)$, is the minimum width of an elimination ordering of (G, d). It is well-known that the tree-width of a graph G equals the elimination-width of G (see [5]), and this fact can be generalised to our setting.

Theorem 3.1. *Let G be a graph and $d : V(G) \to \mathbb{N}$. Then* $\mathrm{tw}(G, d) = \mathrm{ew}(G, d)$.

Proof. Towards a proof of $\mathrm{tw}(G, d) \geq \mathrm{ew}(G, d)$, let (T, B) be a d-stratified tree decomposition for G of width k. We may assume that (T, B) is *small*, i.e. all nodes $s, t \in V(T)$ with $s \neq t$ satisfy $B_s \not\subseteq B_t$. Recall that for a vertex $v \in V(G)$, t_v denotes the $<_T$-minimal node of T with $v \in B_t$. We now define an ordering $v_1, \ldots, v_n$ of $V(G)$ such that for all $1 \leq i, j \leq n$ we have

- $i < j$ implies $d(v_i) \leq d(v_j)$,
- there is a piece B_i of (T, B) containing v_i and all the neighbours of v_i in G_i.

In particular, $v_1, \ldots, v_n$ is an elimination ordering of (G, d) of width at most k.

Claim 1. There exists a vertex $v \in V(G)$ with $d(v)$ maximum, such that v appears in exactly one piece B_ℓ of (T, B), and ℓ is a leaf.

Proof. Choose any vertex w with $d(w)$ maximum. If w is contained in a piece B_t of (B, T) where t is not a leaf (otherwise we are done), then choose a leaf $\ell \geq_T t$ of T. Let s be the parent of ℓ. Choose $v \in B_\ell \setminus B_s$ (such a v exists since the decomposition is small). Since (T, B) is d-stratified and $t_w <_T \ell = t_v$, we have $d(w) \leq d(v)$, and hence by maximality $d(w) = d(v)$, proving the claim. ∎

Let $v_n := v$. Then we replace G by G_{n-1}, we restrict d and (T, B) to G_{n-1} and we proceed by induction.

Towards $\mathrm{tw}(G,d) \leq \mathrm{ew}(G,d)$, let $v_1,\dots,v_n$ be an ordering of $V(G)$ of width at most k and let $G_1,\dots,G_n = G$ be the associated sequence of graphs. For $i = 1,\dots,n$ we define tree decompositions for the G_i that respect d and have width at most k. For $i = 1$ we take the trivial decomposition. Given a tree decomposition of G_{i-1}, we choose a piece containing all the neighbours of v_i in G_i (such a piece exists, because the neighbours induce a clique in G_{i-1}), and we attach to it a new piece containing v_i and all neighbours of v_i in G_i. Let (T,B) be the tree decomposition obtained for $G = G_n$. Obviously, (T,B) has width at most k. Moreover, (T,B) is d-stratified: let $v_i, v_j \in V(G)$. If $t_{v_i} <_T t_{v_j}$, then, by construction, we have $i < j$. Since $v_1,\dots,v_n$ is an elimination ordering of (G,d), this implies $d(v_i) \leq d(v_j)$. □

3.2. **First order tree-width.** For a formula φ, the *formula graph* is the undirected graph G_φ, with vertices $\mathrm{var}(\varphi)$, and edges $\{x,y\}$ whenever x and y are free variables, or when x and y occur together in some atom of φ. (If φ is not straight, then we obtain the formula graph of φ by first making it straight.) Note that the formula graph depends on the syntax of the formula. Logically equivalent formulae may have different formula graphs.

We now introduce a partial order $\preceq_\varphi$ on the variables of a formula φ, from which we then obtain the *essential alternation depth*, $\mathrm{ead}_\varphi(x)$, of a variable $x \in \mathrm{var}(\varphi)$. Given a tree decomposition of G_φ of width $k-1$ that respects ead_φ, we show in Section 5, how to transform the formula φ bottom up along the decomposition into an equivalent $\mathcal{L}^k$-formula. In this transformation, we want to 'reuse' as many variables as possible, so, intuitively, the 'worst case' is that φ is in prenex normal form. Hence we want to 'undo' prenex normal form, pushing quantifiers as far as possible away from the root in the syntax tree. Of course, we have to make sure that we obtain an equivalent formula. Intuitively, $\preceq_\varphi$ gives us a partial order of quantifications that we have to respect while undoing prenex normal form.

For a bound variable x, let $Q_x \in \{\exists,\forall\}$ be the type of quantifier used to quantify x in φ. Then the *scope* of x is the unique subformula ψ of φ such that $Q_x x\psi$ is a subformula of φ. For bound variables x,y of φ, we write $x \leq_\varphi y$ to denote that $x = y$ or y is quantified in the scope of x. For a set X of variables, we use $\varphi_{[X]}$ to denote the minimal (with respect to subformulaship) subformula of φ which contains all atoms using variables from X.

Definition 3.2. Let $\trianglelefteq$ be a binary relation on the variables of some formula φ. Then two variables x and y are *entangled* with respect to $\trianglelefteq$ and φ, if x occurs in $\varphi_{[y\trianglelefteq]}$ and y occurs in $\varphi_{[x\trianglelefteq]}$ (as usual, we use $x \trianglelefteq$ to denote $\{x' \mid x \trianglelefteq x'\}$).

Definition 3.3. Let φ be a straight formula. Then $\preceq_\varphi$ is the minimal (with respect to $\subseteq$) binary relation on $\mathrm{var}(\varphi)$, such that the following hold.

(1) $\preceq_\varphi$ is reflexive.
(2) $\preceq_\varphi$ is transitive.
(3) If $x \leq_\varphi y$, $Q_x \neq Q_y$ and there is a sequence $x = z_0,\dots,z_n = y$ of bound variables such that for all $0 \leq i < n$ we have that z_i, z_{i+1} are entangled with respect to $\preceq_\varphi$ and φ and that $x \preceq_\varphi z_i$ or $y \preceq_\varphi z_i$, then $x \preceq_\varphi y$ (*Alternation*).

In order to see that $\preceq_\varphi$ is well-defined, observe that Definition 3.3 is in fact an inductive definition: all three conditions can be restated as closure of $\preceq_\varphi$ under some operator on binary relations, and all three operators are monotone with respect to $\subseteq$. The least obvious case is the one of the operator underlying Alternation. To establish monotonicity in this case, assume that $\trianglelefteq$ and $\trianglelefteq'$ are binary relations on $\mathrm{var}(\varphi)$ and that $(\trianglelefteq) \subseteq (\trianglelefteq')$. We have to show that whenever two variables x,y satisfy Alternation with respect to $\trianglelefteq$, then they also

do with respect to $\trianglelefteq'$. For any variable z we have $(z \trianglelefteq) \subseteq (z \trianglelefteq')$, so $\varphi_{[z\trianglelefteq]}$ is a subformula of $\varphi_{[z\trianglelefteq']}$. Thus, entanglement of some variables with respect to $\trianglelefteq$ and φ implies entanglement with respect to $\trianglelefteq'$ and φ. Hence any witness for Alternation with respect to $\trianglelefteq$ is also one with respect to $\trianglelefteq'$.

At many places, we will use proof by induction on the inductive definition of $\preceq_\varphi$. Therefore, we explicate how the inductive principle works in this case.

Lemma 3.4. *Let φ be a formula and P a property of pairs of variables from φ. If*

(1) *$P(x,x)$ holds for all $x \in \mathrm{var}(\varphi)$,*
(2) *$x \preceq_\varphi y$, $y \preceq_\varphi z$, $P(x,y)$ and $P(y,z)$ imply $P(x,z)$, and*
(3) *if $x \leq_\varphi y$, $Q_x \neq Q_y$, $(\trianglelefteq) \subseteq (\preceq_\varphi)$ such that $P(x',y')$ holds for all $x' \trianglelefteq y'$, and for some sequence $x = z_0, \ldots, z_n = y$ and all $0 \leq i < n$ we have $z_i \in (x \trianglelefteq) \cup (y \trianglelefteq)$ and entanglement of z_i and z_{i+1} with respect to $\trianglelefteq$ and φ, then $P(x,y)$,*

then $P(x,y)$ holds for all x, y such that $x \preceq_\varphi y$. □

Remark 3.5.

(1) The relation $\preceq_\varphi$ is a subrelation of $\leq_\varphi$: $(\preceq_\varphi) \subseteq (\leq_\varphi)$,
(2) the relation $\preceq_\varphi$ is a partial order, and
(3) $x \preceq_\varphi y$ holds whenever x and y are entangled, $Q_x \neq Q_y$, and $x \leq_\varphi y$.

Proof. 1 follows since $\leq_\varphi$ satisfies all closure conditions.
2: $x \preceq_\varphi y$ is reflexive and transitive by definition, and it inherits anti-symmetry from $(\leq_\varphi)$ by 1.
3: this follows by letting $n = 1$ in Alternation. □

Example 3.6. Let $\varphi := \exists x \forall y \exists z \big(Pxy \wedge \forall u (Ryu \vee Pzu) \big)$. Then $x \preceq_\varphi y$ and $z \preceq_\varphi u$ by Remark 3.5, 3, $y \preceq_\varphi z$ by Alternation (witnessed by the sequence y, u, z), and $x \preceq_\varphi z$, $y \preceq_\varphi u$, and $x \preceq_\varphi u$ by Transitivity. In this example, all entanglements are due to the two variables in question occuring in the same atom.

We use φ_x as a shorthand for $\varphi_{[x\preceq_\varphi]}$, and we say that x and y are entangled in φ, if they are entangled with respect to $\preceq_\varphi$ and φ. Note that this is the case if and only if x occurs in φ_y and y occurs in φ_x. Observe further that φ_x is a subformula of the scope of x. The idea behind entanglement is to capture interaction between variables.

Example 3.7. Let $\varphi := \forall x \forall x' \exists y (((Py \wedge Px) \vee Px) \wedge ((Py \wedge Px') \vee Px'))$. Then $\varphi_{[\{y\}]}$ already is the whole quantifier free part of φ, hence so is φ_y. Further, φ_x contains $\varphi_{[\{x\}]} = (Py \wedge Px) \vee Px$. Thus x occurs in φ_y and y occurs in φ_x, so x and y are entangled. It follows that $x \preceq_\varphi y$, so φ_x contains φ_y. As this is the whole quantifier free part, we have $\varphi_x = \varphi_y$. In a similar way we obtain $\varphi_{x'} = \varphi_y$. Consequently, x and x' are entangled as well. Intuitively, x and x' interact through y.

Example 3.8. Let $\varphi := \forall x \exists y \forall z (Rzy \vee (Px \wedge Py))$. Then y and z are entangled, because they occur in the same atom. x, however, is not entangled with any other variable, because $\varphi_{[\{x\}]} = Px$ does not contain any variable besides x. The same holds for $\psi := \exists y (\forall z Rzy \vee (\forall x Px \wedge Py))$, which illustrates that x does not interact at all. Thus, $(\preceq_\varphi) = \{(y,z), (x,x), (y,y), (z,z)\}$.

Example 3.9. For $n > 0$ let

$$\varphi_n := \exists x_1 \dots \exists x_n \forall y \exists z \left(\bigwedge_{1 \leq i \leq n} Rx_i z \wedge Py \right)$$

and

$$\psi_n := \exists x_1 \dots \exists x_n \forall y \exists z \left(\bigwedge_{1 \leq i \leq n} (Rx_i z \wedge Py) \right) .$$

In φ_n, the only entanglements are between the x_i and z. Consequently, $\preceq_{\varphi_n}$ is the equality relation on $\mathrm{var}(\varphi_n)$. On the other hand, both $(\psi_n)_y$ and $(\psi_n)_z$ coincide with the quantifier free part of ψ_n, so y and z are entangled in ψ_n. It follows that $y \preceq_{\psi_n} z$. Nevertheless, as y does not occur in $(\psi_n)_{x_i} = Rx_i z$, y is not entangled with x_i in ψ_n.

Definition 3.10. Let φ be a first order formula and $x \in \mathrm{var}(\varphi)$. The *essential alternation depth* of x in φ, denoted by $\mathrm{ead}_\varphi(x)$, is the maximum over all $\preceq_\varphi$-paths P ending in x of the number of quantifier changes in P, adding $+1$ in case the first variable on P is existentially quantified and $+2$ if it is universally quantified. If x is a free variable, we let $\mathrm{ead}_\varphi(x) = 0$.

The $+1$ respectively $+2$ in the definition makes sure that $\mathrm{ead}_\varphi(x)$ is odd if and only if $Q_x = \exists$.

Example 3.11. The formula from Example 3.6 satisfies $\mathrm{ead}_\varphi(x) = 1$, $\mathrm{ead}_\varphi(y) = 2$, $\mathrm{ead}_\varphi(z) = 3$, and $\mathrm{ead}_\varphi(u) = 4$.

The formula from Example 3.7 satisfies $\mathrm{ead}_\varphi(x) = \mathrm{ead}_\varphi(x') = 2$ and $\mathrm{ead}_\varphi(y) = 3$.

The formulae from Example 3.8 satisfy $\mathrm{ead}_\varphi(x) = \mathrm{ead}_\varphi(z) = 2$, $\mathrm{ead}_\varphi(y) = 1$, and $\mathrm{ead}_\psi = \mathrm{ead}_\varphi$.

For the formulae from Example 3.9 we have $\mathrm{ead}_{\varphi_n}(x_i) = \mathrm{ead}_{\psi_n}(x_i) = \mathrm{ead}_{\varphi_n}(z) = 1$, $\mathrm{ead}_{\varphi_n}(y) = \mathrm{ead}_{\psi_n}(y) = 2$, and $\mathrm{ead}_{\psi_n}(z) = 3$.

Definition 3.12. If we replace $\preceq_\varphi$ by $\leq_\varphi$ in Definition 3.10, we obtain the (usual) *alternation depth* of x in φ, which we denote by ad_φ.

Remark 3.13. Every formula φ satisfies $\mathrm{ead}_\varphi \leq \mathrm{ad}_\varphi$.

Example 3.14. For the formuale from Examples 3.6 and 3.7 we have $\mathrm{ad}_\varphi = \mathrm{ead}_\varphi$.

The formulae from Example 3.8 satisfy $\mathrm{ad}_\varphi(x) = 2$, $\mathrm{ad}_\varphi(y) = 3$, $\mathrm{ad}_\varphi(z) = 4$, and $\mathrm{ad}_\psi = \mathrm{ead}_\psi$.

For the formulae from Example 3.9 we have $\mathrm{ad}_{\varphi_n} = \mathrm{ad}_{\psi_n} = \mathrm{ead}_{\psi_n}$.

Definition 3.15 (First order tree-width)**.** For a formula φ we define the *first order tree-width* of φ by $\mathrm{fotw}(\varphi) := \mathrm{tw}(G_\varphi, \mathrm{ead}_\varphi)$.
Accordingly, we say that (T, B) is a *tree decomposition for* φ, if (T, B) is an ead_φ-stratified tree decomposition for G_φ.

Note that for a formula φ, the variables $\mathrm{free}(\varphi)$, as well as the variables of any atom or literal in φ induce cliques in G_φ. In particular, since $\mathrm{ead}_\varphi(x) = 0$ for any free variable x, by Fact 2.2 it is no restriction to require that the free variables be covered in the root of a tree decomposition.

In general, the difference between $\mathrm{fotw}(\varphi)$ and $\mathrm{tw}(G_\varphi)$ can be unbounded:

Proposition 3.16. *For every $n > 0$ there is a formula φ_n with $\mathrm{fotw}(\varphi_n) = n$ and $\mathrm{tw}(G_{\varphi_n}) = 1$.*

Proof. Let

$$\varphi_n = \exists x_1 \dots \exists x_n \forall y \bigwedge_{1 \leq i \leq n} E x_i y \,.$$

Then G_{φ_n} is the n-star with center y, and we have $\mathrm{ead}_{\varphi_n}(x_i) = 1$ for all $1 \leq i \leq n$, and $\mathrm{ead}_{\varphi_n}(y) = 2$. It is easy to see that any ead_{φ_n}-stratified tree decomposition (T, B) of G_{φ_n} has a piece $\{x_1, x_2, \dots, x_n, y\}$, namely B_{t_y}. On the other hand, such a tree decomposition needs no other pieces. Hence $\mathrm{fotw}(\varphi_n) = n$. Since G_{φ_n} is a tree we have $\mathrm{tw}(G_{\varphi_n}) = 1$. □

Lemma 3.17. *Given a formula φ, we can compute $\preceq_\varphi$ and ead_φ in polynomial time.*

Proof. In order to compute $\preceq_\varphi$, consider the three closure operators implicit in its definition. As $\preceq_\varphi$ is a binary relation on $\mathrm{var}(\varphi)$, a quadratic number of applications of the closure operators suffices to produce $\preceq_\varphi$. Hence it remains to show that each closure operator is computable in polynomial time. This is immediate for Reflexivity and Transitivity. For Alternation, let $(\trianglelefteq) \subseteq \mathrm{var}(\varphi)^2$ be the current approximation of $\preceq_\varphi$. First, we compute the formulae $\varphi_{[z \trianglelefteq]}$ for all variables z, and from these the entanglement relation. Then, checking whether some pair (x, y) needs to be added to R because of Alternation basically amounts to reachability in the entanglement graph restricted to $(x \trianglelefteq) \cup (y \trianglelefteq)$.

It is clear that ead_φ can be computed from φ and $\preceq_\varphi$ in polynomial time. □

3.3. **Xenerp normal form.** Prenex normal form aims to make the scopes of quantifiers as large as possible. Working in the opposite direction, we obtain what we term xenerp normal form.

Definition 3.18. A subformula χ of a formula φ is in *xenerp normal form with respect to* φ, if for all variables x quantified in χ the following holds: φ_x is immediately preceeded by a quantifier sequence which contains $Q_x x$.

A formula φ is in *xenerp normal form*, if it is in xenerp normal form with respect to itself.

Example 3.19. Recall the formulae φ and ψ from Example 3.8. We have already seen that $\mathrm{ead}_\varphi = \mathrm{ead}_\psi$. Furthermore, φ and ψ are equivalent and ψ is in xenerp normal form, whereas φ is not in xenerp normal form.

The following lemma presents equivalent transformations of formulae, such that neither the formula graph, nor the essential alternation depth, nor the corresponding tree decompositions change.

Lemma 3.20. *Let φ and ψ be formulae satisfying either 1, 2 or 3.*

1. *There are formulae $\chi_1, \dots, \chi_n$, a positive Boolean combination θ of n arguments, and a variable x which does not occur in $\chi_2, \dots, \chi_n$, such that ψ is obtained from φ by replacing a subformula $\theta(Q_x x \chi_1, \chi_2, \dots, \chi_n)$ by $Q_x x \theta(\chi_1, \chi_2, \dots, \chi_n)$.*
2. *There are a formula χ, and variables x, y with $Q_x = Q_y$ such that ψ is obtained from φ by replacing a subformula $Q_x x Q_y y \chi$ by $Q_y y Q_x x \chi$.*
3. *There are a formula χ, and variables x, y with φ_x a proper subformula of φ_y, such that ψ is obtained from φ by replacing a subformula $Q_x x Q_y y \chi$ by $Q_y y Q_x x \chi$, where $Q_y y \chi$ is xenerp with respect to φ.*

Then $\varphi \equiv \psi$, $\mathrm{ead}_\varphi = \mathrm{ead}_\psi$, and $G_\varphi = G_\psi$. Consequently, tree decompositions for φ coincide with tree decompositions for ψ and in particular $\mathrm{fotw}(\varphi) = \mathrm{fotw}(\psi)$.

Implicitly, we assume in these cases that Q_x and Q_y are the same with respect to the formula φ and with respect to the formula ψ.

Proof. In all three cases the formulae differ only in the position of quantifiers, so $G_\varphi = G_\psi$ is immediate. Also, for all $X \subseteq \mathrm{var}(\varphi) = \mathrm{var}(\psi)$, the formulae $\varphi_{[X]}$ and $\psi_{[X]}$ are essentially equal: the only potential difference between them is the same shift of quantifiers which led from φ to ψ. In particular, the same variables occur in $\varphi_{[X]}$ as in $\psi_{[X]}$. Hence, all differences between $\preceq_\varphi$ and $\preceq_\psi$ (and thus between ead_φ and ead_ψ) must ultimately stem from differences between $\leq_\varphi$ and $\leq_\psi$.

For the first replacement, the equivalence $\varphi \equiv \psi$ is well-known. For the other parts of the statement, the only change between $\leq_\varphi$ and $\leq_\psi$ is, that for all variables y quantified in χ_i for some $2 \leq i \leq n$, we have $x \not\leq_\varphi y$ but $x \leq_\psi y$. We show that this change has no impact on $\preceq$. Clearly $(\preceq_\varphi) \subseteq (\preceq_\psi)$ because of $(\leq_\varphi) \subseteq (\leq_\psi)$. For the converse we use induction on derivations. More precisely, we show by induction on the inductive definition of $\preceq_\psi$ that for all $x' \preceq_\psi y'$ we have $x' \preceq_\varphi y'$. That is the inductive property $P(x', y')$ as in Lemma 3.4 is $x' \preceq_\varphi y'$. The first two inductive rules are trivial, because we know that $\preceq_\varphi$ is reflexive and transitive. Hence we can concentrate on Alternation. So let $x' \leq_\psi y'$, $(\trianglelefteq) \subseteq (\preceq_\psi) \cap (\preceq_\varphi)$, and $x' = z_0, \ldots, z_n = y'$ be given such that $Q_x \neq Q_y$, and for all $0 \leq i < n$ we have $z_i \in (x' \trianglelefteq) \cap (y' \trianglelefteq)$ and entanglement of z_i, z_{i+1} with respect to $\trianglelefteq$ and ψ. By our above observation on the similarity of $\varphi_{[X]}$ and $\psi_{[X]}$ for any set X of bound variables, it follows that z_i and z_{i+1} are also entangled with respect to $\trianglelefteq$ and φ, and thus with respect to $\preceq_\varphi$ and φ. Consequently, either $x' \preceq_\varphi y'$ holds (and we are done), or $x' \not\leq_\varphi y'$. Then, together with $x' \leq_\psi y'$ it follows that $x' = x$ and y' is quantified in χ_j for some $2 \leq j \leq n$. As $x \leq_\varphi z_i$ or $y' \leq_\varphi z_i$ for all $0 \leq i \leq n$, all z_i are quantified either in $Q_x x \chi_1$ or in χ_j. As the former is true for $z_0 = x$ and the latter for $z_n = y'$, there is some $0 \leq i < n$ such that z_i is quantified in $Q_x x \chi_1$ and z_{i+1} in χ_j. Then $\varphi_{[z_i \trianglelefteq]}$ is a subformula of φ_{z_i}, which in turn is a subformula of the scope of z_i and thus of $Q_x x \chi_1$. Similarly, the scope of z_{i+1} is a subformula of χ_j. As all occurences of z_{i+1} are in its scope and χ_j is disjoint from $Q_x x \chi_1$, this contradicts the fact that z_{i+1} occurs in $\varphi_{[z_i \trianglelefteq]}$.

In the second replacement, the equivalence $\varphi \equiv \psi$ also is well-known. So let us show $(\preceq_\varphi) = (\preceq_\psi)$. As the replacement is symmetric, it suffices to show $(\preceq_\psi) \subseteq (\preceq_\varphi)$, which we do by induction. Again, the cases of Reflexivity and Transitivity are clear. For Alternation, let $\trianglelefteq$ and $x' = z_0, \ldots, z_n = y'$ be given as above. Again, we obtain that Alternation also yields $x' \preceq_\varphi y'$ unless $x' \not\leq_\varphi y'$. But then $x' = x$ and $y' = y$ contradicting $Q_x = Q_y$.

In the third replacement, if $Q_x = Q_y$ then this case is subsumed by the second replacement so we may assume $Q_x \neq Q_y$. Let us start by showing $\varphi \equiv \psi$. We will even show $Q_x x Q_y y \chi \equiv Q_y y Q_x x \chi$. As $Q_y y \chi$ is in xenerp normal form we have, for variables v, w quantified in $Q_y y \chi$, that φ_w is a subformula of φ_v whenever $v \leq_\varphi w$. In this case w occurs in φ_v (as the formula is straight, w does occur somewhere, and it can only occur in φ_w which is a subformula of φ_v). So for entanglement of v and w it suffices to show that v occurs in φ_w. Let $V := \{v \in \mathrm{var}(\varphi) \mid y \leq_\varphi v \text{ and } \varphi_x \text{ is a subformula of } \varphi_v\}$. Observe, that V is linearly ordered by $\leq_\varphi$.

Claim 1. There is a subformula θ of χ which is a superformula of φ_x such that no variable from V occurs free in θ.

Proof. For contradiction, assume the opposite. Then we inductively define a sequence $v_0, v_1, \ldots$ of variables. Take v_0 to be x. For all $i \geq 1$ we will have $v_i \in V$. For defining v_{i+1} from v_i, let θ_i be φ_{v_i}, together with the sequence of quantifications of variables from V which immediately preceeds φ_{v_i}. Hence θ_i is a subformula of $Q_y y\chi$. If $\theta_i = Q_y y\chi$, then we terminate the sequence. Otherwise, by our assumption, some variable from V occurs free in θ_i. Then let v_{i+1} be such a variable. As $Q_y y\chi$ is xenerp, θ_i is a proper subformula of $\varphi_{v_{i+1}}$. This implies that v_i occurs in $\varphi_{v_{i+1}}$ and the converse holds by choice of v_{i+1}, because θ_i essentially coincides with φ_{v_i}. Hence v_i and v_{i+1} are entangled whenever both are defined. If furthermore $i \neq 0$, then we also have $v_{i+1} \leq_\varphi v_i$ and $v_{i+1} \neq v_i$, because $Q_y y\chi$ is xenerp and φ_{v_i} is a proper subformula of $\varphi_{v_{i+1}}$. Hence, as V is finite, the sequence $v_0, \ldots$ must terminate, say with v_n. As φ_x is a proper subformula of φ_y, we have $\theta_0 \neq Q_y y\chi$, so $n > 0$. Now let $0 \leq m \leq n$ be minimal such that $Q_{v_m} = Q_y$. The case that there are no such m will be handled later. As $Q_{v_0} = Q_x \neq Q_y$, we have $m > 0$ so φ_x is a proper subformula of φ_{v_m}. We obtain $v_m \preceq_\varphi v_i$ for all $1 \leq i \leq m$ using a backwards induction as follows. The base case is Reflexivity. For the inductive step we have a chain $v_m, \ldots, v_i$ of entanglements and by the inductive hypothesis we have $v_m \preceq_\varphi v_j$ for all intermediate $j > i$. Further $v_m \leq v_i$ and $Q_{v_m} = Q_y \neq Q_{v_i}$, so we obtain $v_m \preceq_\varphi v_i$ using Alternation. Then, in total we have a chain $x = v_0, v_1, \ldots, v_m$ of entanglements such that $x \preceq_\varphi v_0$ by Reflexivity, $v_m \preceq_\varphi v_i$ for all $1 \leq i < m$ and $Q_x \neq Q_y = Q_{v_m}$, so Alternation implies $x \preceq_\varphi v_m$. But then $(v_m \preceq_\varphi) \subseteq (x \preceq_\varphi)$, contradicting that φ_x is a proper subformula of φ_{v_m}. Now for the case that $Q_{v_i} = Q_x$ for all i such that v_i is defined. Then $\theta_n = Q_y y\chi$, so $\varphi_{v_n} = \varphi_y$ and thus v_n and y are entangled. This time we have a chain $y, v_n, \ldots, v_1, x$ of entanglements with the same properties as the sequence $v_m, \ldots, v_1, x$ above: it is descending with respect to $\leq_\varphi$ except for the last step, the first variable has the same quantifier as y and all other variables have the same quantifier as x. Thus similar to the above we obtain $x \preceq_\varphi y$, this time in contradiction to φ_x being a proper subformula of φ_y. This proves the claim. ∎

So let θ be a subformula of χ and a superformula of φ_x such that no variable from V occurs free in θ. Hence all free variables of θ except x are also free variables of $Q_x x Q_y y\chi$. Without loss of generality we may assume that $Q_x = \exists$ and $Q_y = \forall$. It is well-known that $\exists x \forall y \chi$ implies $\forall y \exists x \chi$. For the converse, let $\mathcal{I}$ be an interpretation for $\exists x \forall y \chi$ such that $\mathcal{I} \models \forall y \exists x \chi$. We need to show that $\mathcal{I} \models \exists x \forall y \chi$. For a value a from the universe of the interpretation $\mathcal{I}$, we have that $\mathcal{I}\frac{a}{x}$ is an interpretation for $\forall y \chi$. As all free variables of θ are also free in $\forall y \chi$, we have that $\mathcal{I}\frac{a}{x}$ also is an interpretation for θ. Now let a_0 be some value such that $\mathcal{I}\frac{a_0}{x} \models \theta$. If no such a_0 exists, let instead a_0 be arbitrary. Now for all a we have that $\mathcal{I}\frac{a}{x} \models \theta$ implies $\mathcal{I}\frac{a_0}{x} \models \theta$. Now let b be an arbitrary value. As $\mathcal{I} \models \forall y \exists x \chi$, we have that $\mathcal{I}_1 := \mathcal{I}\frac{b}{y}\frac{a'}{x} \models \chi$ for some value a'. Let us examine the impact that replacing $\mathcal{I}_1$ by $\mathcal{I}_2 := \mathcal{I}\frac{b}{y}\frac{a_0}{x}$ has on the formula χ. Recall that φ_x is a subformula of θ, so in particular x does not occur free in χ except in θ. Hence the only possible change comes from θ. If $\mathcal{I}_1 \models \theta$, then also $\mathcal{I}_2 \models \theta$ and nothing changes, that is $\mathcal{I}_2 \models \chi$. The same argument holds if $\mathcal{I}_1, \mathcal{I}_2 \not\models \theta$. Otherwise $\mathcal{I}_1 \not\models \theta$ and $\mathcal{I}_2 \models \theta$. In this case recall, that all formulae are in negation normal form, so χ is positive in θ. Thus $\mathcal{I}_1 \models \chi$ again implies $\mathcal{I}_2 \models \chi$. So $\mathcal{I}\frac{a_0}{x}\frac{b}{y} = \mathcal{I}\frac{b}{y}\frac{a_0}{x} \models \chi$ for all values b, which implies $\mathcal{I}\frac{a_0}{x} \models \forall y \chi$ and then $\mathcal{I} \models \exists x \forall y \chi$.

Next, let us compare $\preceq_\varphi$ with $\preceq_\psi$. The only difference between $\leq_\varphi$ and $\leq_\psi$ is, that $x \leq_\varphi y \not\leq_\varphi x$ while $x \not\leq_\psi y \leq_\psi x$. To show $(\preceq_\varphi) = (\preceq_\psi)$, we show inclusion in both directions by induction. Let us start with $(\preceq_\varphi) \subseteq (\preceq_\psi)$. As in the proofs for the other replacements, the only interesting case is where $x' \preceq_\varphi y'$ but $x' \not\preceq_\psi y'$. This implies $x' = x$

and $y' = y$. Then $x \preceq_\varphi y$ in contradiction to φ_x being a proper subformula of φ_y. For the other inclusion we need to recall a little more from the above cases: we obtain a sequence of entanglements (which are such both in ψ and in φ) $x' = z_0, \dots, z_n = y'$ such that for all $0 \leq i < n$ we have $x' \preceq_\psi z_i$ and $x' \preceq_\varphi z_i$ or $y' \preceq_\psi z_i$ and $y' \preceq_\varphi z_i$. Also we have $x' \leq_\psi y'$ and we are done unless $x' \not\leq_\varphi y'$, hence $x' = y$ and $y' = x$. But then we have $x \leq_\varphi y$ and a chain $x = v_n, \dots, v_0 = y$ of entanglements. As further $x \preceq_\varphi v_n$ by Reflexivity, we obtain $x \preceq_\varphi y$ which again gives a contradiction. This concludes the proof of Lemma 3.20 □

Observe, that the first class of replacements in the previous lemma are exactly what is used in turning a formula into prenex normal form. For xenerp normal form, we need the first and third class.

Corollary 3.21. *Let φ be a formula and let ψ be a prenex normal form (obtained in the usual way) of φ. Then* $\mathrm{fotw}(\varphi) = \mathrm{fotw}(\psi)$. □

Lemma 3.22. *From a formula φ we can compute in polynomial time a formula ψ in xenerp normal form, such that $\varphi \equiv \psi$, $G_\varphi = G_\psi$,* $\mathrm{ead}_\varphi = \mathrm{ead}_\psi$, *and consequently, tree decompositions for φ coincide with those for ψ.*

Proof. We work by applying replacements from Lemma 3.20 in one direction or the other. More precisely, we apply replacements of the first kind backwards whenever possible. Whenever no such replacement is applicable, then each quantifier sequence ends with some quantifier $Q_x x$ such that the scope of x is an atom, a negated atom, or a conjunction or disjunction of two subformulae, both of which contain x. In all three cases, φ_x contains the scope of x. As the reversed inclusion always holds, we have that $Q_x x$ immediately precedes φ_x.

If the formula φ' at hand is xenerp, we are done. Otherwise let x be $\leq_{\varphi'}$-maximal such that the quantifier sequence containing $Q_x x$ does not precede φ_x. By the above, this sequence does not end with $Q_x x$, so the scope of x has the form $Q_y y \chi$. By maximality of x, $Q_y y \chi$ is xenerp with respect to φ'. In particular, the quantifier sequence containing $Q_x x$ is followed by φ_y. By the choice of x we conclude that $\varphi_x \neq \varphi_y$, so φ_x is a proper subformula of φ_y. Hence a replacement of the third kind is applicable (in the forward direction).

It remains to show, that repeatedly applying these replacements terminates. As a first semi-invariant, consider the sum, taken over all variables x, of the distance that $Q_x x$ has from the root in the syntax tree of the formula. (Backwards) replacements of the first kind increase this semi-invariant while replacements of the third kind do not change it at all. On the other hand, replacements of the third kind decrease the number of variable pairs (x, y), such that $x \leq_{\varphi'} y$ and φ_x is a proper subformula of φ_y. As both semi-invariants are polynomially bounded in $|\varphi|$, and each replacement (including the test for applicability) requires only polynomial time (recall Lemma 3.17), the procedure runs in polynomial time. □

3.4. **Comparing ead with ad.** Let φ be a first order formula. As $\preceq_\varphi$ is coarser than $\leq_\varphi$, it is immediate that $\mathrm{ead}_\varphi \leq \mathrm{ad}_\varphi$. However, this does not directly imply that $\mathrm{fotw}(\varphi) \leq \mathrm{tw}(G_\varphi, \mathrm{ad}_\varphi)$ also holds. Variables which are incomparable by $\leq_\varphi$ (and thus by $\preceq_\varphi$) are given an order by ead_φ and by ad_φ, but not neccessarily the same one. Consequently, not every ad_φ-stratified tree decomposition is also ead_φ-stratified. Nevertheless $\mathrm{fotw}(\varphi) \leq \mathrm{tw}(G_\varphi, \mathrm{ad}_\varphi)$ does hold, and it is the purpose of this subsection to show this fact.

At the core of our proof there is a double induction which we cast into two auxiliary lemmata. An *entanglement chain* (in some formula φ) is, of course, a sequence $v_0, \dots, v_n$

of variables such that for all $0 \leq i < n$ the variables v_i and v_{i+1} are entangled in φ. The entanglement chain is *hanging*, if $v_0, v_n \leq_\varphi v_i$ for all $0 < i < n$. It is *crossing*, if $Q_{v_0} \neq Q_{v_n}$. It is *nice*, if $\min_{\leq_\varphi}(v_0, v_n) \preceq_\varphi v_i$ for all $0 \leq i \leq n$. (We will only talk about nice chains in contexts where the existence of $\min_{\leq_\varphi}(v_0, v_n)$ is guaranteed a priori. As a side note it is not hard to see that this minimum exists for all hanging chains.)

Lemma 3.23. *Let φ be a formula such that $\leq_\varphi$ is a total order (for example φ is in prenex normal form). Let $x = v_{-m}, \ldots, v_{-1}, y = v_0, v_1, \ldots, v_n = z$ be an entanglement chain without repetitions such that the following hold:*

(1) *$Q_x \neq Q_y = Q_z$.*
(2) *$x \leq_\varphi v_i$ for all $-m \leq i < n$.*
(3) *The subchain $y = v_0, v_1, \ldots, v_n = z$ is hanging.*
(4) *$x \preceq_\varphi v_i$ for all $-m \leq i \leq 0$, i.e. the subchain $x = v_{-m}, \ldots, v_{-1}, v_0 = y$ is nice.*
(5) *For all $0 \leq k < \ell \leq n$, if the subchain $v_k, \ldots, v_\ell$ is hanging and crossing, then it is also nice.*

Then the chain is nice.

Proof. Otherwise assume a counterexample with n minimal. Obviously, $n > 0$. If $n = 1$, then we can use the chain to derive either $x \preceq_\varphi z$ or $z \preceq_\varphi x$ by Alternation. In the first case we are done, in the second case we use Transitivity with x to derive $z \preceq_\varphi v_i$ for all $-m \leq i \leq 0$.

Hence in the following we may assume $n > 1$. Let $0 < k < n$ be such, that $v_k \leq_\varphi v_i$ for all $0 < i < n$, that is the subchains $v_0, \ldots, v_k$ and $v_k, \ldots, v_n$ are hanging. If $Q_{v_k} = Q_x$, then the last premise gives $y \preceq_\varphi v_i$ for all $0 \leq i \leq k$ and $z \preceq_\varphi v_i$ for all $k \leq i \leq n$. The former, together with $x \preceq_\varphi v_0 = y$, gives $x \preceq_\varphi v_i$ for all $0 \leq i \leq k$. Hence, for all $-m \leq i \leq n$ we have $x \preceq_\varphi v_i$ or $z \preceq_\varphi v_i$. Thus the chain again witnesses $x \preceq_\varphi z$ or $z \preceq_\varphi x$, so transitivity also gives $x \preceq_\varphi v_i$ for all $k \leq i \leq n$ or $z \preceq_\varphi v_i$ for all $-m \leq i \leq k$.

So we are left with the case $Q_{v_k} = Q_y = Q_z$. Then the subchain $x = v_{-m}, \ldots, v_k$ satisfies all conditions of this lemma. Minimality of n implies that $x \preceq_\varphi v_i$ for all $-m \leq i \leq k$. Now we set $m' = m + k$, $n' = n - k$, and $v'_i = v_{i+k}$ for all $-m' \leq i \leq n'$. The thus shifted entanglement chain again satisfies all conditions, so minimality of the counterexample implies that the shifted chain is nice. Then so is the original chain. □

Lemma 3.24. *Let φ be a formula such that $\leq_\varphi$ is a total order. Then every hanging and crossing entanglement chain is nice.*

Proof. Otherwise assume a counterexample $v_0, \ldots, v_n$ with minimal n. The fact that the chain is crossing prohibits $n = 0$. If $n = 1$, the claim follows from Alternation. Hence we assume $n > 1$. Let $0 < k < n$ be such, that $v_k \leq_\varphi v_i$ for all $0 < i < n$. We have $Q_{v_k} = Q_x$ or $Q_{v_k} = Q_y$, without loss of generality the latter. Then by minimality of the counterexample the shifted entanglement chain with v_k in the middle satisfies all conditions of Lemma 3.23. Hence the claim follows from that lemma. □

Theorem 3.25. *For all $\varphi \in \mathcal{L}$ we have* $\mathrm{fotw}(\varphi) \leq \mathrm{tw}(G_\varphi, \mathrm{ad}_\varphi)$.

Proof. Without loss of generality, we may assume that φ is in prenex normal form: otherwise let φ' be a prenex normal form of φ such that $\mathrm{ad}_\varphi = \mathrm{ad}_{\varphi'}$. Such a φ' can be obtained by moving quantifiers to the left in φ (which is the normal procedure for making a formula prenex) while always chosing a variable with minimal ad_φ. Among the prenex normal forms of φ, this φ' is a sensible prenex normal form anyway, in that it does not introduce

unneccessary alternation. Observe that $G_\varphi = G_{\varphi'}$. Now by Lemma 3.20 we have $\mathrm{fotw}(\varphi) = \mathrm{fotw}(\varphi')$ while the choice of φ' implies $\mathrm{tw}(G_\varphi, \mathrm{ad}_\varphi) = \mathrm{tw}(G_{\varphi'}, \mathrm{ad}_{\varphi'})$.

By virtue of Theorem 3.1 it suffices to prove $\mathrm{ew}(G_\varphi, \mathrm{ead}_\varphi) \leq \mathrm{ew}(G_\varphi, \mathrm{ad}_\varphi)$. We show that for all formulae φ in prenex normal form, all k, and all elimination orderings of $(G_\varphi, \mathrm{ad}_\varphi)$ of width k, there is an elimination ordering of $(G_\varphi, \mathrm{ead}_\varphi)$ of width k. Let $\varphi = Q_{x_1}x_1 \dots Q_{x_n}x_n\theta$ where θ is quantifier free.

For any ordering $y_1, \dots, y_n$ of $\{x_1, \dots, x_n\}$, a *φ-fault* is a pair $1 \leq i < i' \leq n$ such that $\mathrm{ead}_\varphi(y_i) > \mathrm{ead}_\varphi(y_{i'})$. Let us assume that φ and $y_1, \dots, y_n$ form a counterexample with a minimal number of φ-faults, that is $y_1, \dots, y_n$ is an elimination ordering of $(G_\varphi, \mathrm{ad}_\varphi)$, $(G_\varphi, \mathrm{ead}_\varphi)$ has no elimination ordering of equal width, and the number of φ-faults is minimal for all choices of φ and $y_1, \dots, y_n$. If this number of φ-faults is 0, then $y_1, \dots, y_n$ also is an elimination ordering of $(G_\varphi, \mathrm{ead}_\varphi)$, contradicting the counterexample property. Hence there is some φ-fault.

Let k be the width of $y_1, \dots, y_n$ with respect to G_φ. Let $\psi := Q_{y_1}y_1 \dots Q_{y_n}y_n\theta$. As $y_1, \dots, y_n$ is obtained from $x_1, \dots, x_n$ only by rearranging variables within quantifier blocks, we can obtain ψ from φ by a sequence of replacements as in Part 2 of Lemma 3.20. Hence that Lemma implies $G_\varphi = G_\psi$ and $\mathrm{ead}_\varphi = \mathrm{ead}_\psi$. Also, $\mathrm{ad}_\varphi = \mathrm{ad}_\psi$, so in particular $y_1, \dots, y_n$ is an elimination ordering of (ψ, ad_ψ) of width k. By Definition of ψ, we have $y_i \leq_\psi y_{i'}$ if and only if $i \leq i'$. Furthermore, φ-faults and ψ-faults of $y_1, \dots, y_n$ coincide.

As there is a ψ-fault, there also is one which concerns two subsequent variables, that is for some i we have $\mathrm{ead}_\psi(y_i) > \mathrm{ead}_\psi(y_{i+1})$. Let $z_1, \dots, z_n$ be the sequence $y_1, \dots, y_{i-1}$, y_{i+1}, y_i, $y_{i+2}, \dots, y_n$, i.e. y_i and y_{i+1} change places. Further, let $\chi := Q_{z_1}z_1 \dots Q_{z_n}z_n\theta$. Obviously, $z_1, \dots, z_n$ is an elimination ordering of $(G_\chi, \mathrm{ad}_\chi)$. We claim that $y_i \not\preceq_\psi y_{i+1}$ and that $\{y_i, y_{i+1}\}$ is no edge of $G_{\psi,i+1}$, where this graph is as in the definition of elimination width with respect to the sequence $y_1, \dots, y_n$. These claims are proved later, let us first show how to make use of them. The fact $G_\chi = G_\psi$ and the non-edge between y_i and y_{i+1} in $G_{\psi,i+1}$ imply that the width of $z_1, \dots, z_n$ with respect to χ and ψ is also k. From $y_i \not\preceq_\psi y_{i+1}$ it is easy to see (and somewhat implicit in the proof of Lemma 3.20) that $\mathrm{ead}_\chi = \mathrm{ead}_\psi$. In particular, ψ-faults and χ-faults coincide. By construction, $z_1, \dots, z_n$ has one such fault less than $y_1, \dots, y_n$, so by minimality of the counterexample there is some elimination ordering of $(G_\chi, \mathrm{ead}_\chi)$ of width k. As $G_\chi = G_\psi = G_\varphi$ and $\mathrm{ead}_\chi = \mathrm{ead}_\psi = \mathrm{ead}_\varphi$, it is also one of φ of equal width, in contradiction to φ, k being a counterexample.

Now for the claims. First, assume for contradiction that $y_i \preceq_\psi y_{i+1}$. As $y_i \neq y_{i+1}$, this is not due to Reflexivity. As y_{i+1} is the $\leq_\psi$-successor of y_i and $(\preceq_\psi) \subseteq (\leq_\psi)$, there can be no intermediate variable, hence $y_i \preceq_\psi y_{i+1}$ is due to Alternation. But then $Q_{y_i} \neq Q_{y_{i+1}}$, so $\mathrm{ead}_\psi(y_{i+1}) \geq \mathrm{ead}_\psi(y) + 1$ in contradiction to $i, i+1$ being a ψ-fault.

For the second claim assume, again for contradiction, that y_i forms an edge with y_{i+1} in $G_{\psi,i+1}$, i.e. in G_ψ there is a path from y_i to y_{i+1} such that all internal vertices of that path are of the form y_j with $j > i+1$. We view the path as an entanglement chain $y_i = v_0, \dots, v_n = y_{i+1}$. As the elimination ordering is $\leq_\psi$, we have $y_i, y_{i+1} \leq_\psi v_j$ for all $0 < j < n$, that is the chain is hanging. If $Q_{y_i} \neq Q_{y_{i+1}}$, then the chain is nice by Lemma 3.24. In particular, $y_i \preceq_\psi y_{i+1}$, contradicting the previous claim. Now for the case $Q_{y_i} = Q_{y_{i+1}}$. Then $\mathrm{ead}_\psi(y_{i+1}) < \mathrm{ead}_\psi(y_i)$ implies that there is some $w \in \mathrm{var}(\psi)$ such that $\mathrm{ead}_\psi(w) = \mathrm{ead}_\psi(y_i) - 1$ and $w \preceq_\psi y_i$. It follows that $Q_w \neq Q_{y_i}$. The fact $w \preceq_\psi y_i$ is not due to Reflexivity because $Q_w \neq Q_{y_i}$. It is also not due to Transitivity: otherwise let u be the intermediate variable. Then $Q_u = Q_{y_i}$ or $Q_u = Q_w$, without loss of generality the latter. Then $w \preceq_\psi u$ is due to Transitivity and further unfolding Transitivity until

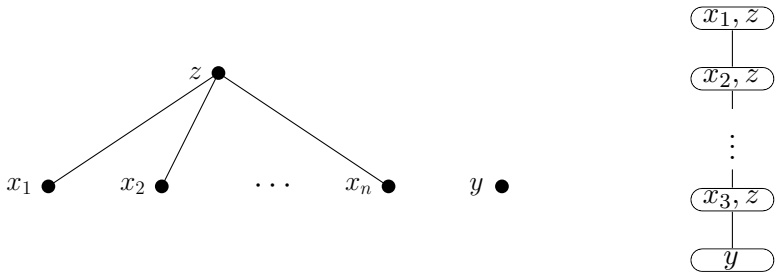

Figure 1: Example 3.26: the formula graph of φ_n and an ead_{φ_n}-stratified tree decomposition of width 1 of φ_n.

Alternation is applicable eventually yields some u' such that $w \preceq_\psi u' \preceq_\psi u \preceq_\psi y_i$ and $Q_w \neq Q_{u'} \neq Q_u \neq Q_{y_i}$. Then $\mathrm{ead}_\psi(w) + 1 = \mathrm{ead}_\psi(y_i) \geq \mathrm{ead}_\psi(w) + 3$, a contradiction. Hence $w \preceq_\psi y_i$ is due to Alternation, so there is some corresponding entanglement chain. Prepending it to the one we have yields a chain $w = v_{-m}, \ldots, v_{-1}, v_0 = y_i, v_1, \ldots, v_n = y_{i+1}$. Furthermore this chain satisfies the first three conditions of Lemma 3.23 (with respect to ψ). By virtue of Lemma 3.24, the other two conditions are implied. Hence Lemma 3.23 yields $w \preceq_\psi y_{i+1}$, so $\mathrm{ead}_\psi(y_{i+1}) \geq \mathrm{ead}_\psi(w)+1 = \mathrm{ead}_\psi(y_i)$, contradicting $\mathrm{ead}_\psi(y_{i+1}) < \mathrm{ead}_\psi(y_i)$. □

The following example shows that the difference in the opposite direction can be unbounded.

Example 3.26. For $n > 0$, recall the formula φ_n from Example 3.9. We have seen that $\mathrm{ead}_{\varphi_n}(x_i) = \mathrm{ad}_{\varphi_n}(x_i) = 1 = \mathrm{ead}_{\varphi_n}(z)$ for $i \leq n$, and $\mathrm{ead}_{\varphi_n}(y) = \mathrm{ad}_{\varphi_n}(y) = 2$, whereas $\mathrm{ad}_{\varphi_n}(z) = 3$. Figure 1 shows the formula graph of φ_n together with an ead_{φ_n}-stratified tree decomposition of width 1 of φ_n. We actually have $\mathrm{fotw}(\varphi_n) = 1$. On the other hand, it is easy to see that every ad_{φ_n}-stratified tree decomposition of φ_n has a piece $\mathrm{var}(\varphi_n)$ or $\mathrm{var}(\varphi_n) \setminus \{y\}$, and hence $\mathrm{tw}(G_{\varphi_n}, \mathrm{ad}_{\varphi_n}) = n$.

4. Computing stratified tree decompositions

In this section, we show that computing stratified tree decompositions of optimal width is fixed-parameter tractable, where the parameter is the stratified tree-width. In fact, the running time is essentially linear for bounded stratified tree-width. In the next section, we will use the algorithm developed here as a first step for formula evaluation.

Definition 4.1. Let $G = (V, E)$ be a graph and $d : V \to \mathbb{N}$. We say that (G, d) is *normalized*, if for $n := |V|$ we have $V = \{1, \ldots, n\}$ and $d(v) \leq n$ for all $v \in V$.

We obtain a linear time algorithm only for normalized inputs. In the following, let (G, d) with $G = (V, E)$ be normalized. We set $d_{\max} := \max_{v \in V} d(v)$ and for any integer i we let $X_i := \{v \in V \mid d(v) = i\}$. Let $N_G(C)$ denote the set of neighbours of C in $V \setminus C$.

Definition 4.2. For $0 \leq i \leq d_{\max}$, let $G^{(i)}$ be the graph with vertex set V, such that two vertices x, y form an edge in $G^{(i)}$, if in G there is a path from x to y with all internal vertices in $\bigcup_{j > \max(i, d(x), d(y))} X_j$.

In particular, every edge of G is present in all $G^{(i)}$ by virtue of a path without any internal vertices. For $G^{(d_{\max})}$, we can use only such paths, so $G^{(d_{\max})} = G$. Observe the similarity of the graphs $G^{(i)}$ to the graphs G_j in the definition of elimination orderings: in both cases the connectivity through some vertices is redirected to hold immediately between those vertices thus connected. The main differences are granularity (the number of graphs is $d_{\max}+1$, respectively $|V|+1$) and the fact that in the definition of $G^{(i)}$, no vertices are deleted. Indeed, we have $G^{(i)}[\bigcup_{j\leq i} X_j] = G_{|\bigcup_{j\leq i} X_j|}$. This also explains our interest in the $G^{(i)}$. We are (implicitly) looking for elimination orderings for G which respect d. Such elimination orderings keep the X_i intact, so the $G^{(i)}$ will (up to vertex deletion) occur as the G_j at the boundaries between the different X_i.

Definition 4.3. The *component tree* of (G,d) is a rooted tree with nodes t labelled by two subsets of V, denoted by C_t and D_t. For the root r, we let $C_r := V$. For a node t at level i (where the root has level 0), we let $D_t := C_t \cap \bigcup_{j\leq i} X_j$. For each (nonempty) connected component C of $G[C_t \setminus D_t]$, node t has a child t_C and we let $C_{t_C} := C \cup N_G(C)$. Further, we let $D_t^1 := D_t \cap X_i$ and $D_t^2 := D_t \setminus D_t^1$.

Lemma 4.4. *Let G be a graph and $d\colon V(G)\to\mathbb{N}$. Let t,u be nodes of the component tree of (G,d), where u is a child of t. Then*

(1) $D_t \cap D_u = D_t \cap C_u$.
(2) *Let (T,B) be a d-stratified tree decomposition of G. Then some piece of (T,B) covers $D_t \cap D_u$.*

Proof. Let i be the depth of t in the component tree and let C be the connected component of $G[C_t \setminus D_t]$ such that $C_u = C \cup N_G(C)$.

1) It suffices to prove $D_t \cap D_u \supseteq D_t \cap C_u$. If $x \in D_t \cap C_u$, then $x \in D_t \subseteq \bigcup_{0\leq j\leq i} X_j$ and hence $x \in C_u \cap \bigcup_{0\leq j\leq i+1} X_j = D_u$.

2) By Fact 2.2 it suffices to show that any pair of distinct vertices $x,y \in D_t \cap D_u$ occurs together in some piece of (T,B). As $x,y \in D_t$, we have $x,y \notin C$, so $x,y \in N_G(C)$. Since C is a connected component, there is a path P from x to y with all internal vertices in C. Hence $d(z) > d(x)$ and $d(z) > d(y)$ for all internal vertices z of P.

Let $x = x_0, x_1, \ldots, x_n = y$ be the path P. If $n = 1$, then G already contains the edge $\{x,y\}$, which hence is covered by (T,B). Otherwise, the set of tree nodes covering vertices from $P \setminus \{x,y\}$ induces a nonempty connected subtree T' in T. Let s be the $\leq_T$-minimal node of T' and let $1 < i < n$ be such that $s = t_{x_i}$. As $\{x,x_1\}$ is an edge of G, the vertex x is covered by some node t' of T'. By definition of s and t_x, we have $s \leq_T t'$ and $t_x \leq_T t'$. Hence, both t_x and s lie on the unique path from the root of T to t' and thus they are comparable by $\leq_T$. As $t_{x_i} = s <_T t_x$ would contradict d-stratifiedness, $t_x \leq_T s \leq_T t'$ follows. Using (TD3), we conclude that s covers x. Analogously, s covers y, so x and y occur together in B_s. □

While the component tree is useful for mentally addressing the task of computing a stratified tree decomposition, we cannot afford to actually compute it: its size is superlinear and we want to achieve a linear running time. The algorithm instead works with a modified variant. First, we can do without the C_ts, so the algorithm only computes the tree itself and the D_ts. This projection is necessary to obtain a linear size, as is the following second modification: in case for some node t at level $i > 0$ we have $D_t \cap X_i = \emptyset$, then we omit the node t, making its only child instead a child of the parent of t. We term t a *dropped node*.

Whenever we talk about the level of a node in the modified component tree, we always refer to the level the node had prior to dropping any nodes. As a third modification, for each node t, say at level i, we store $G^{(i-1)}[D_t]$ alongside D_t. For technical reasons which will become clear in the proof of Lemma 4.5, we allow the encodings of these $G^{(i-1)}[D_t]$ to have multiedges. To keep the notation simple, let us fix the convention that $|E|$ for some edge set E with multiedges denotes the sum of multiplicities of edges from E.

Lemma 4.5. *We can compute a modified component tree of a normalized (G, d) in time $O(|V| \cdot \mathrm{tw}(G, d)^2)$.*

Proof. Let $k := \mathrm{tw}(G, d)$. The modified component tree is computed in a bottom-up fashion. For this observe, that the various C used in the definition of the component tree at level i are the connected components of $G[\bigcup_{j \geq i} X_j]$. Hence each C_t for t at level i is $C \cup N_G(C)$ for such a C. In order to determine the corresponding D_t, we do not need the full graph: it is irrelevant which edges between vertices of $\bigcup_{j>i} X_j$ are present; it suffices to know what connectivity these induce on $\bigcup_{j \leq i} X_j$. This information is present in $G^{(i)}$. More precisely, we can transform the defining equality $D_t = (C \cup N_G(C)) \cap \bigcup_{j \leq i} X_j$, where C is some connected component of $G[\bigcup_{j \geq i} X_j]$, into the equality $D_t = D \cup (N_{G^{(i)}}(D) \cap \bigcup_{j<i} X_j)$, where D is the corresponding connected component of $G^{(i)}[X_i]$.

Recall that, as (G, d) is normalized, the vertices of G are $1, \ldots, n$, and $d_{\max} \leq n$, where $n = |V|$. More precisely, we assume that G is given as an array of adjacency lists and that d is given as an array of d's values, where both arrays are indexed with vertices. Using bucket sort, we compute the sets $X_0, \ldots, X_{d_{\max}}$ in time $O(n + d_{\max})$, which is $O(n)$ thanks to $d_{\max} \leq n$.

For the bottom-up run, the algorithm uses a loop $i = d_{\max}, \ldots, 1$. As an invariant, at the beginning of run i it has the following data:

- The graph $G^{(i)}$ (possibly with multiedges).
- An array of lists of subtrees of the modified component tree. Overall, the lists contain all subtrees rooted at level $i + 1$. For all $0 \leq j \leq d_{\max}$, the list at entry j contains the subtrees which, after dropping nodes, end up at level j.

  For $j > 0$, we also store an element from $D_t \cap X_{j-1}$ alongside each subtree, where t is the root of the subtree.

The graph is initialized to $G^{(d_{\max})} = G$. This can be done in zero time, because G is not needed any more. The array is initialized with empty lists, using time $O(d_{\max})$ and thus $O(n)$. For a single run of the loop, we first do a depth-first search through $G^{(i)}[X_i]$. More precisely, the search is in $G^{(i)}$, it uses all elements from X_i as entry points, and it terminates recursion in elements from other X_j. In this way, the connected components D of $G^{(i)}[X_i]$ are found, and we also directly obtain the corresponding $D \cup (N_{G^{(i)}}(D) \cap \bigcup_{j<i} X_j)$, that is some new D_t. During the search, we do some more things, which only increase the running time by a constant factor: for each such D_t, we generate the tree node t and we label each vertex of D_t with t. The label persists only until the next i. Also, for each such t, while constructing D_t, we also compute the sets D_t^1 and D_t^2, and we start constructing $G^{(i-1)}[D_t]$ by building a graph with vertex set D_t and all edges from $G^{(i)}[D_t]$ incident to (at least one vertex of)

D_t^1. Furthermore, we determine the maximal j, such that $D_t^2 \cap X_j \neq \emptyset$ and we remember some $x \in D_t^2 \cap X_j$. The running time of the search is $O(|X_i| + |\{e \in E(G^{(i)}) | e \cap X_i \neq \emptyset\}|)$. The second part, the number (respecting multiplicities) of edges of $G^{(i)}$ incident with X_i, is bounded by the number of edges of $G^{(0)}$ incident with X_i. Summing over all i, the total running time of all searches is then $O(n + |E(G^{(0)})|)$.

Next, we update $G^{(i)}$ to $G^{(i-1)}$ by introducing, for each new t, edges between each two vertices from D_t^2. We also do this in the graph stored at node t, which thus becomes $G^{(i-1)}[D_t]$ as needed. We do not check whether the edges were already present (because we do not have enough time to do so), hence multiedges may be introduced. The running time is $O(|D_t| + \ell)$, where ℓ is the number of edges introduced. Summing over all iterations, the first part $|D_t|$ is bounded in the same way as the running time of the depth-first searches. The second part ℓ is bounded by $\binom{k}{2}$, where $k := \mathrm{tw}(G, d) + 1$: as $D_t \cap \bigcup_{j<i} X_j = D_t \cap D_u$ for the parent u of t, we can conclude $|D_t^2| = |D_t \cap \bigcup_{j<i} X_j| \leq k$ from Lemma 4.4, 2. Hence in total, the second part induces a running time of $O(m \cdot k^2)$, where m is the size of the tree. As this is the only place, where we add edges to $G^{(i)}$, we can also bound $|E(G^{(0)})|$ by $|E(G)| + m \cdot \binom{k}{2}$.

We already have all the nodes for level i. We only need to connect them to the nodes of level $i+1$ (in case $i < d_{\max}$). For this, for each tree in the list at the array entry $i+1$, say with root u, we look at its element from $D_u \cap X_i$, say y, then look at the label of y generated above (as $y \in X_i$, we did consider y in the depth-first search), say t, and make u a child of t. As this is done at most once for each tree node, the total running time of doing this throughout the loop is linear in the size of the tree. Last, for all new t, we recall j and x determined above, and we add the tree rooted at t together with x to the list at array entry $j+1$.

This concludes the description of the loop. After it has finished, we generate the root node r, set $D_r := X_0$ and make all trees in the list at array entry 0 children of r. The total running time of computing the modified component tree is thus $O(n + |E(G)| + m \cdot k^2)$, where, again, m is the number of nodes of the modified component tree. For the second term we have $|E(G)| \leq n \cdot \mathrm{tw}(G) \leq n \cdot k$ by Fact 2.1. Now, let us estimate m. Each non-root node t of the modified component tree is a non-root node of the original component tree, say at level $i > 0$, satisfying $D_t \cap X_i \neq \emptyset$. Hence, there is an element $x_t \in D_t \cap X_i$. Let u be the parent of t in the original component tree. Then x_t belongs to the connected component C of $C_u \setminus D_u$, such that $C_t = C \cup N_G(C)$. For different t at level i, the x_t come from different connected components, so there can be at most $|X_i|$ such t. In total, $m \leq n$, so the running time is $O(n \cdot k^2)$. □

Remark 4.6. As a corollary to the previous proof, let us observe that $G^{(0)}$ is obtained from G by turning all D_t^2 into cliques.

Theorem 4.7. *There is an algorithm that, given a normalized (G, d), computes a d-stratified tree decomposition of G of minimum width in time $O(|V(G)| \cdot 2^{\mathrm{poly}(\mathrm{tw}(G,d))})$.*

Proof. Observe that, if u is a child of t in the component tree, then $D_t \cap D_u = D_u^2$. Then Lemma 4.4, 2 and Remark 4.6 imply that d-stratified tree decompositions of G coincide with those of $G^{(0)}$. Hence it suffices to compute a d-stratified tree decomposition for $G^{(0)}$ of minimum width. This is achieved in four phases. First, a modified component tree is computed as in Lemma 4.5.

In the second phase, for each node t of the component tree, an optimal tree decomposition of $G^{(0)}[D_t]$ is computed. Observe, that, if i is the level of t, then $G^{(0)}[D_t]$ coincides with $G^{(i-1)}[D_t]$, which we have stored at node t. Hence, for a single t, using Bodlaender's algorithm [6] (which works well even in the presence of multiedges), the tree decomposition can be obtained in time $\|G^{(i-1)}[D_t]\| \cdot 2^{\mathrm{poly}(k)}$. Observe, that the sum of all $\|G^{(i-1)}[D_t]\|$ is bounded by the encoding size of the modified component tree, which is bounded by the time it took to compute it. Hence, overall this phase takes time $n \cdot 2^{\mathrm{poly}(k)}$.

In the third phase, for each node t of the component tree and each parent or child u of t, we determine a node $x_{t,u}$ of the tree decomposition of $G^{(0)}[D_t]$, whose piece fully contains $D_t \cap D_u$. Observe, that $D_t \cap D_u$ coincides with D_t^2 or with D_u^2, depending on whether u is a parent or a child of t. Hence, $D_t \cap D_u$ is a clique in $G^{(0)}$, and then also in $G^{(0)}[D_t]$, so there must be some $x_{t,u}$ as above. We find $x_{t,u}$ by first reading D_t^2 respectively D_u^2 from the modified component tree, and then checking each element from each piece of the tree decomposition against this set $D_t \cap D_u$, until some adequate $x_{t,u}$ is found. Even without sophisticated search structures, $O(\ell \cdot |D_t \cap D_u| \leq \ell \cdot k)$ time suffices, where ℓ is the size of the tree decomposition. Overall, this phase takes time $O(n \cdot k^3)$.

In the fourth phase, we use a bottom-up recursion to construct, for all nodes t, an optimal tree decompositions of $G^{(0)}[C_t]$. It will happen to extend the above tree decomposition of $G^{(0)}[D_t]$. The recursion works as follows: we cycle through the children of t. By virtue of bottom-up-ness, for each such child u, we already know an optimal tree decomposition of $G^{(0)}[C_u]$. As it contains the known decomposition of $G^{(0)}[D_u]$, it also contains the node $x_{u,t}$. Hence we connect it to the known decomposition of $G^{(0)}[D_t]$ by adding an edge between $x_{u,t}$ and $x_{t,u}$. The result still is a tree decomposition, because the intersection $D_t \cap D_u$ separates D_t from C_u (Lemma 4.4, 1). It is an optimal tree decomposition, because its width is the maximum of the widths of the participating tree decompositions, which were optimal for their respective subgraphs. In the end, the recursion yields an optimal tree decomposition (T, B) of $G^{(0)}[C_r] = G^{(0)}$. It takes time $O(m)$.

Last, pick some node s from the tree decomposition of $G^{(0)}[D_r]$. As $B_s \subseteq D_r \subseteq X_0$, choosing s as the root of (T, B) makes it d-stratified. This concludes the proof of Theorem 4.7. □

As each (G, d) can be normalized in time $O(n \log n)$, a similar statement holds for arbitrary (G, d), albeit not with linear running time. Theorem 4.7 implies that we can efficiently decide whether a formula φ satisfies $\mathrm{fotw}(\varphi) \leq k$:

Corollary 4.8. *There is an algorithm that, given a formula $\varphi \in \mathcal{L}$, computes a tree decomposition of φ of minimum width in time* $\mathrm{poly}(\|\varphi\|) + |\mathrm{var}(\varphi)| \cdot 2^{\mathrm{poly}(\mathrm{fotw}(\varphi))}$.

Proof. Given φ, it is first turned into a straight formula. Then, G_φ and ead_φ are computed and normalized in polynomial time. The last step is a call to the algorithm from Theorem 4.7. □

5. Query evaluation on bounded first order tree-width

This section contains the second main result: evaluating formulae of bounded tree-width is fixed parameter tractable with parameter the length of the formula. Moreover, we show that evaluating quantified constraint formulae of bounded first order tree-width can be done in polynomial time. This is stronger than Chen and Dalmau's result [9] for quantified

constraint formulae of bounded elimination-width: as we will see in Section 6, bounded elimination-width (i.e. bounded *tree-width* in [9]) implies bounded first order tree-width, but there are classes of quantified constraint formulae with unbounded elimination-width, that have bounded first order tree-width.

Definition 5.1. Let $k \geq 0$ be an integer. The fragment $\mathcal{L}^k$ of $\mathcal{L}$ consists of those (not necessarily straight) formulae φ such that $|\mathrm{var}(\varphi)| \leq k$. In contrast, let $\tilde{\mathcal{L}}^k$ be the fragment of formulae φ of $\mathcal{L}$ such that all subformulae of φ have at most k free variables.

Remark 5.2. Obviously, $\mathcal{L}^k \subseteq \tilde{\mathcal{L}}^k$. On the other hand, formulae from $\tilde{\mathcal{L}}^k$ can be turned into equivalent formulae from $\mathcal{L}^k$ by renaming bound variables. The algorithm runs in polynomial time, and in linear time for fixed k. □

By Remark 5.2, we can use $\mathcal{L}^k$ and $\tilde{\mathcal{L}}^k$ interchangeably. It will be more convenient to work with the latter.

Remark 5.3. For all k, the question whether a given formula is equivalent to an $\mathcal{L}^k$ formula is undecidable.

The statement is folklore, but we provide a proof for completeness' sake.

Proof. We reduce from satisfiability of $\mathcal{L}$. By introducing a new unary predicate U and relativizing all quantifiers to U, satisfiability reduces to the question, whether a given formula from $\mathcal{L}$ is satisfiable by an infinite structure. By the theorem of Löwenheim and Skolem, this in turn is equivalent to satisfiability by a countable infinite structure. Now let φ be some fixed formula such that φ does not have finite models, and such that φ is not equivalent to any $\mathcal{L}^k$-formula. Then a given formula ψ is unsatisfiable by infinite structures, if and only if $\varphi \wedge \psi'$ is equivalent to an $\mathcal{L}^k$ formula, where ψ' is obtained from ψ by renaming all symbols (constant symbols, relation symbols, and free variables) to be disjoint from all symbols of φ. For the 'only if' part, unsatisfiability by infinite structures of ψ implies the same for ψ'. Then, $\varphi \wedge \psi'$ is unsatisfiable and hence equivalent to $Pc \wedge \neg Pc$ from $\mathcal{L}^0$. For the 'if' part, assume that ψ is satisfiable and that $\varphi \wedge \psi'$ is equivalent to $\chi \in \mathcal{L}^k$. We obtain χ' from χ by renaming all symbols which do not occur in φ to be disjoint from all symbols of ψ'. By choice of χ we have $\chi \models \varphi$ and then $\chi' \models \varphi$. The converse would contradict that φ is not equivalent to any $\mathcal{L}^k$-formula, so there is some interpretation $\mathcal{I}_1$ such that $\mathcal{I}_1 \models \varphi \wedge \neg\chi'$. As φ does not have any finite models, the domain of $\mathcal{I}_1$ is infinite, and by Löwenheim-Skolem we can assume without loss of generality that it is countable. On the other hand, satisfiability of ψ implies that of ψ', say by an interpretation $\mathcal{I}_2$ with countable infinite domain. By virtue of all the renaming done above, $\mathcal{I}_1$ and $\mathcal{I}_2$ do not have any symbols in common. Furthermore, their domains are of the same cardinality. Hence there is an interpretation $\mathcal{I}$ which extends both (up to isomorphisms), that is we have $\mathcal{I} \models \varphi \wedge \psi'$ but $\mathcal{I} \not\models \chi$, a contradiction to the assumed equivalence. □

As a consequence, there is no computable width parameter such that width-k captures all formulae logically equivalent to a formula of $\mathcal{L}^k$. For any width parameter, only formulae which are 'syntactically close' to a formula of $\mathcal{L}^k$ are captured, for varying values of 'syntactically close'.

The following example shows that first order tree-width indeed does not capture equivalence to $\mathcal{L}^2$.

Example 5.4. Let $n > 2$. $\varphi_n = \psi_n \vee \chi$, where $\chi \in \mathcal{L}^2$ and $\mathrm{fotw}(\psi_n) = n$, but ψ_n is unsatisfiable. Then $\varphi_n \equiv \chi \in \mathcal{L}^2$, but $\mathrm{fotw}(\varphi_n) = n$.

First order sentences of tree-width at most $k-1$ have the same expressive power as $\tilde{\mathcal{L}}^k$ (and by Remark 5.2 hence as $\mathcal{L}^k$). More generally we have the following.

Theorem 5.5. *Let $k \geq 0$.*

(1) *For any formula φ with $\mathrm{fotw}(\varphi) \leq k-1$ there is a formula $\psi \in \tilde{\mathcal{L}}^k$ with $\varphi \equiv \psi$ which is computable from φ.*
(2) *Any formula $\psi \in \tilde{\mathcal{L}}^k$ satisfies $\mathrm{fotw}(\psi) \leq k-1$.*

Proof. 2. By Theorem 3.25 it suffices to show that $\mathrm{tw}(G_\psi, \mathrm{ad}_\psi) \leq k-1$. Without loss of generality, ψ is already straight. Let T be the syntax tree of ψ. For a node t, say t corresponds to the subformula χ of ψ, let B_t consist of the free variables of χ. As $\psi \in \tilde{\mathcal{L}}^k$, the width of (T, B) is at most $k-1$. (TD2) holds for (T, B), because edges arise from atoms and atoms are among the subformulae χ considered in the definition of (T, B). Then (TD1) follows, because ψ is straight. (TD3) holds, because for all variables x, the set $\{t \in T \mid x \in B_t\}$ union of all paths from (nodes corresponding to) atoms in which x occurs to (the node corresponding to) the scope of x. It is easy to see that (T, B) is ad_ψ-stratified.

1. Given φ, we use the algorithm of Corollary 4.8 to compute an ead_φ-stratified tree decomposition (T, B) of width $\mathrm{fotw}(\varphi) \leq k-1$ for G_φ.

Intuitively, we will iteratively replace subformulae of φ by equivalent $\tilde{\mathcal{L}}^k$-formulae, until we obtain an $\tilde{\mathcal{L}}^k$ formula ψ equivalent to φ. The replacement is done along a tree decomposition (T, B) of φ of width at most $k-1$. For our purposes it is more convenient only to work in leaves of the decomposition, so in every iteration we restrict the decomposition to the part of the formula that still has to be transformed into an $\tilde{\mathcal{L}}^k$ formula. In doing so, we treat the subformulae φ' of φ already in $\tilde{\mathcal{L}}^k$ as new atoms, defined on the variables $\mathrm{free}(\varphi')$. For this we make sure that these new atoms are covered in some piece of the remaining part of the tree decomposition.

More precisely, we describe an iterative algorithm. In every iteration, we are given a formula φ' in xenerp normal form, a tree decomposition (T', B') of φ' of width at most $k-1$, and a second order substitution S, substituting relation symbols of φ' by $\tilde{\mathcal{L}}^k$ formulae with the appropriate number of free variables, such that we have $\varphi' S \equiv \varphi$. We will need to extend the syntax of formulae: we allow any monotone Boolean function as a single Boolean connective. Of course, this extended syntax still allows for negation normal form, and we still assume that all formulae are in this normal form. This syntax extension carries over to the definition of $\varphi_{[X]}$ and hence of ead_φ.

We start by letting φ' be some xenerp normal form of φ, $(T', B') = (T, B)$, and $S = \emptyset$. Now in every step we do the following.

1. If φ' is quantifier free, then $\mathrm{var}(\varphi') = \mathrm{free}(\varphi)$. As $|\mathrm{free}(\varphi)| \leq \mathrm{fotw}(\varphi) + 1 \leq k$ (recall, that $\mathrm{free}(\varphi)$ forms a clique in G_φ), we have $\varphi' \in \tilde{\mathcal{L}}^k$, so the algorithm stops with output $\varphi' S$. More precisely, $\varphi' S$ may still use our additional Boolean connectives, but it is trivial to eliminate these without leaving $\tilde{\mathcal{L}}^k$ (but in general making the formula non-straight).

2. Otherwise, if (T', B') has a leaf ℓ with parent t satisfying $B_\ell \subseteq B_t$, then we remove ℓ and B_ℓ from (T', B'), keeping φ' and S.

3. If neither 1 nor 2 apply, then we choose a variable $x \in \mathrm{var}(\varphi')$ as in the following claim.

Claim 1. There exists a bound variable $x \in \mathrm{var}(\varphi')$ with $\mathrm{ead}_{\varphi'}(x)$ maximum, such that x appears in exactly one piece B'_ℓ of (T', B'), and ℓ is a leaf.

Proof. This claim is a variant of Claim 1 in the proof of Theorem 3.1, and in can be shown in the same way. Instead of smallness of the decomposition, we use the fact that Case 2 does not apply. Thus we obtain a variable x with $\mathrm{ead}_{\varphi'}(x)$ maximum and a leaf ℓ such that x appears only in B_ℓ. As Case 1 does not apply, there are bound variables. As $\mathrm{ead}_{\varphi'}(x)$ is maximum, x is a bound variable. ∎

Without loss of generality, suppose that $Q_x = \exists$. Let ψ be the scope of x in φ' and let V be the set of variables quantified in ψ. We partition V into the subsets V_1 and V_2, where V_1 contains the variables with which x is entangled in φ'. As φ' is xenerp, ψ is φ'_x preceded by some quantifications of variables from V_1. By maximality of $\mathrm{ead}_{\varphi'}(x)$, we have $Q_y = \exists$ for all $y \in V_1$.

Claim 2. No variable $y \in V_1$ is quantified in the scope of some variable $z \in V_2$.

Proof. Otherwise, x occurs in φ'_y (because x and y are entangled), φ'_y is a subformula of φ'_z (because φ' is xenerp and $z \preceq_{\varphi'} y$), and z occurs in φ'_z which is a subformula of φ'_x. Hence x and z are entangled, contradicting $z \in V_2$. Thus the claim holds. ∎

Claim 3. x is $\preceq_{\varphi'}$-maximal.

Proof. Otherwise there is some $y \neq x$ such that $x \preceq_{\varphi'} y$. In the inductive definition of $\preceq_{\varphi'}$, the pair (x, y) is not introduced by Reflexivity. If it is introduced by Transitivity, say with intermediate variable z, we can replace y by z, so we can eventually assume that $x \preceq_{\varphi'} y$ is due to Alternation. In particular $Q_x \neq Q_y$. Thus $\mathrm{ead}_{\varphi'}(y) > \mathrm{ead}_{\varphi'}(x)$, contradicting $\mathrm{ead}_{\varphi'}$-maximality of x. ∎

In the same way we can show that there are no $\preceq_{\varphi'}$-relationships among V_1.

Let ψ' be obtained from ψ by removing all quantifications for variables from V_1. Letting $\exists V_1$ denote the sequence of all these quantifications in arbitrary order, $\exists V_1 \psi'$ is obtained from ψ by a sequence of replacements as in Lemma 3.20 parts 1 and 2. In particular, $\exists x \psi$ is equivalent to $\exists x \exists V_1 \psi'$.

As all variables quantified in ψ' are from V_2 and their scopes in ψ' are the same as in ψ, we have that x does not occur in any quantified subformula of ψ'. Hence ψ' is a positive Boolean combination of atoms, of negated atoms, and of subformulae in which x does not occur. By choosing the subformulae in which x does not occur maximal, ψ' is a positive Boolean combination of atoms using x, of negated atoms using x, and of maximal subformulae of ψ' in which x does not occur. We transform this Boolean combination into disjunctive normal form $\bigvee_{i \in I} \bigwedge_{j \in J_i} L_j$ with literals L_j. Here each L_j is an atom using x, a negated atom using x, or a maximal subformula of ψ' in which x does not occur. Then

$$\exists x \psi \equiv \exists x \exists V_1 \psi' \equiv \exists x \exists V_1 \bigvee_{i \in I} \bigwedge_{j \in J_i} L_j \equiv \exists V_1 \bigvee_{i \in I} \exists x \bigwedge_{j \in J_i} L_j.$$

For $i \in I$ let $J_i^- \subseteq J_i$ be the subset of indices j such that x does not occur in L_j and let $J_i^+ = J_i \setminus J_i^-$. For convenience we let $J^+ := \bigcup_{i \in I} J_i^+$ and $J^- := \bigcup_{i \in I} J_i^-$. Now

$$\exists x \psi \equiv \exists V_1 \bigvee_{i \in I} \exists x \bigwedge_{j \in J_i} L_j \equiv \exists V_1 \bigvee_{i \in I} \Big(\bigwedge_{j \in J_i^-} L_j \wedge \exists x \bigwedge_{j \in J_i^+} L_j \Big).$$

Let $j \in J^+$. As (T', B') is a tree decomposition of φ', (the atom underlying) L_j is covered by some piece of (T', B'). Since x is a variable of L_j and x occurs in B'_ℓ only, L_j is covered in B'_ℓ. Hence for $i \in I$ and $\psi_i^x := \exists x \bigwedge_{j \in J_i^+} L_j$ we have $\mathrm{var}(\psi_i^x) \subseteq B'_\ell$. Let $x\bar{y}$ be an enumeration

of B'_ℓ, and for all $i \in I$ let A^x_i be a new relation symbol of arity $|B'_\ell| - 1 \leq k-1$. Replace $\exists x\psi$ in φ' by $\exists V_1\psi''$, where

$$\psi'' := \bigvee_{i\in I} \big(\bigwedge_{j\in J_i^-} L_j \wedge A^x_i \bar{y}\big)$$

and let φ'' be the formula thus obtained from φ'. Let S_x be the substitution which replaces every $A^x_i\bar{y}$ by ψ^x_i, respectively. Clearly, $\varphi' \equiv \varphi'' S_x$, so by setting $S' := S_x S$ we have $\varphi \equiv \varphi' S \equiv \varphi'' S'$. More precisely, in the definition of ψ'', we use a single positive Boolean connective (as allowed by the above syntax extension) for the entire disjunctive normal form. This Boolean connective has only one input for each L_j used. In particular, we retain a single quantifier per variable quantified in ψ, that is we retain straightness.

We obtain a tree decomposition (T'', B'') for φ'' by letting $T'' := T'$ and removing x from B'_ℓ. This tree decomposition still covers all (edges of $G_{\varphi''}$ created by) atoms of φ'' which also occur in φ', because these do not use x. The new atoms $A^x_i\bar{y}$ are covered by $B''_\ell = B'_\ell \setminus \{x\}$. This shows (TD2). (TD1) and (TD3) are inherited from (T', B').

In the rest of the proof we show that (T'', B'') is $\mathrm{ead}_{\varphi''}$-stratified.

Claim 4. Let y be a bound variable and Z a set of bound variables such that $x \neq y$ and $x \notin Z$. Then y occurs in $\varphi''_{[Z]}$ if and only if y occurs in $\varphi'_{[Z]}$ or $\varphi'_{[Z]}$ is a subformula of φ'_x but not of any L_j with $j \in J^-$ and y occurs in φ'_x.

Proof. Observe that $\varphi'_{[Z]}$ is generated already by at most two occurrences of variables from Z. If one of these occurences is outside of ψ, then $\varphi'_{[Z]}$ is disjoint from ψ and φ'_x, or ψ and φ'_x are proper subformulae of $\varphi'_{[Z]}$. In either case we have to show that the same variables $y \neq x$ occur in $\varphi'_{[Z]}$ as in $\varphi''_{[Z]}$. This follows, because in the first case $\varphi'_{[Z]} = \varphi''_{[Z]}$ whereas in the second case $\varphi''_{[Z]}$ is obtained from $\varphi'_{[Z]}$ by replacing the subformula $\exists x\psi$ with $\exists V_1\psi''$. It remains to consider the case where all (generating) occurrences of variables from Z in φ' are in ψ. Let j_1, j_2 be such that the generating occurrences are in L_{j_1} and L_{j_2}. If $j_1 = j_2 \in J^-$, then $\varphi'_{[Z]}$ is a subformula of L_{j_1}. Furthermore $\varphi''_{[Z]} = \varphi'_{[Z]}$, so the same variables y occur in these formulae. If any generating variable is from B'_ℓ, then it occurs in all atoms $A^x_i\bar{y}$, so $\varphi''_{[Y]} = \psi''$. In the last case, $j_1, j_2 \in J^-$ and $j_1 \neq j_2$. Again, $\varphi''_{[Y]} = \psi''$. In both cases there is no single $j \in J^-$ such that $\varphi'_{[Z]}$ is a subformula of L_j, so we need to show that the same variables y occur in $\varphi''_{[Z]}$ as in φ'_x (including the special case of occurrence in the subformula $\varphi'_{[Z]}$ of φ'_x). This follows because $\varphi''_{[Z]} = \psi''$ and ψ'' uses the same variables as φ'_x. This shows the claim. ∎

For arbitrary variables y, z different from x, the two following claims show that $y \preceq_{\varphi'} z$ if and only if $y \preceq_{\varphi''} z$.

Claim 5. Let y, z be variables different from x. Then $y \preceq_{\varphi'} z$ implies $y \preceq_{\varphi''} z$.

Proof. First, consider the following modification of $\preceq_{\varphi'}$. Relax, in the definition of entanglement with respect to $\trianglelefteq$ and φ', the conditions of the form 'v occurs in $\varphi'_{[w\trianglelefteq]}$' by '$v$ occurs in $\varphi'_{[w\trianglelefteq]}$ or in $\varphi''_{[(w\trianglelefteq)\setminus\{x\}]}$', with the convention that no variable occurs in $\varphi''_{[\emptyset]}$. Let $\preceq$ denote the modified relation. It is clear that $y \preceq_{\varphi'} z$ implies $y \preceq z$. By induction on the definition of $\preceq$ it is easy to see that x also is $\preceq$-maximal.

In order to transform derivations for $\preceq$ into derivations for $\preceq_{\varphi''}$, we start by normalizing them. First, we may assume that no entanglement chain used in any application of Alternation repeats elements. Also, we may assume that Reflexivity and Transitivity are

always applied as early as possible. In particular, at each application of Alternation, the current approximation for $\preceq$ is reflexive and transitive. The third normalization concerns applications of Alternation, say with chain $v_0, \dots, v_n$ and with $\trianglelefteq$ as the current approximation of $\preceq$. Such an application is normalized, if for entanglement, instead of the sets $v_i \trianglelefteq$, already the sets $(v_i \trianglelefteq) \setminus \{x\}$ suffice (unless $v_i = x$, in which case $(v_i \trianglelefteq) = \{x\}$ due to maximality of x). Now we claim that this normalization is always possible. We prove the claim by induction on derivations for $\preceq$ which are already in the first two normal forms. Only the Alternation step is nontrivial, so consider an application of Alternation with chain $v_0, \dots, v_n$ and approximation $\trianglelefteq$ for $\preceq$. By the inductive hypothesis, all pairs from $\trianglelefteq$ can be derived in a normalized way. If our application of Alternation is not already normalized, then there is some $0 \leq i \leq n$ such that $x \neq v_i \trianglelefteq x$. After unfolding Transitivity, we obtain some w such that $v_i \trianglelefteq w$ and such that $w \trianglelefteq x$ holds due to Alternation (possibly $w = v_i$). Let $(\trianglelefteq') \subseteq (\trianglelefteq)$ be the approximation of $\preceq$ pertaining to this application of Alternation. Then there is another chain of entanglements ending with u, x, such that $w \trianglelefteq' u$ or $x \trianglelefteq' u$. Due to the normalizations from the inductive hypothesis, x already occurs in $\varphi'_{[(u\trianglelefteq')\setminus\{x\}]}$ or in $\varphi''_{[(u\trianglelefteq')\setminus\{x\}]}$. As x does not occur in φ'' at all, the former must be the case. As $x \trianglelefteq' u$ would contradict $\preceq$-maximality of x, we have $w \trianglelefteq' u$. Transitivity gives $v_i \trianglelefteq u$. Now consider one of the up to two neighbours of v_i in the entanglement chain, without loss of generality we pick v_{i+1}. We distinguish the cases $v_{i+1} = x$ and $v_{i+1} \neq x$. In the first case, we already know that x occurs in $\varphi'_{[(u\trianglelefteq')\setminus\{x\}]}$. From $v_i \trianglelefteq u$ and transitivity we conclude $(u \trianglelefteq') \setminus \{x\} \subseteq (v_i \trianglelefteq) \setminus \{x\}$, so x occurs in $\varphi'_{[(v_i\trianglelefteq)\setminus\{x\}]}$ as needed. Now for the case that $v_{i+1} \neq x$. As v_{i+1} occurs in $\varphi'_{[v_i\trianglelefteq]}$ or in $\varphi''_{[(v_i\trianglelefteq)\setminus\{x\}]}$, it is easy to see that v_{i+1} also occurs in $\varphi''_{[(v_i\trianglelefteq)\setminus\{x\}\cup\{u\}]}$. But as $v_i \trianglelefteq u$, this is $\varphi''_{[(v_i\trianglelefteq)\setminus\{x\}]}$. In a last normalization step, we also eliminate x from occurrences in entanglement chains in derivations. Hence assume that v, x, w is part of such a chain, again with $\trianglelefteq$ as approximation of $\preceq$. As x is $\preceq$-maximal, we have $(x \trianglelefteq) = \{x\}$, hence v occurs in $\varphi'_{[\{x\}]}$ and then also in ψ''. As x occurs in $\varphi'_{[(w\trianglelefteq)\setminus\{x\}]}$ (recall the previous normalization), we conclude that ψ'' is a subformula of $\varphi''_{[(w\trianglelefteq)\setminus\{x\}]}$. Thus v occurs in $\varphi''_{[(w\trianglelefteq)\setminus\{x\}]}$ and similarly w occurs in $\varphi''_{[(v\trianglelefteq)\setminus\{x\}]}$, so x can be omitted from the chain. After this normalization, we can assume that x occurs in applications of Alternation only as an endpoint of the chain.

Now assume some counterexample y, z to the claim. Then $y \preceq z$. Let y, z be derivation-minimal with respect to normalized derivations for $\preceq$. Then $y \neq z$ because $\preceq_{\varphi''}$ is reflexive. Next assume that $y \preceq z$ is due to transitivity, say with intermediate variable v. If $v = x$ then $x \preceq z \neq x$ contradicting $\preceq$-maximality of x. Thus $v \neq x$, so $y \preceq_{\varphi''} v \preceq_{\varphi''} z$ by minimality of the counterexample. $y \preceq_{\varphi''} z$ follows. Finally for Alternation, let us assume some chain $y = v_0, \dots, v_n = z$ of entanglements in the sense of $\preceq$, such that $y \preceq v_i$ or $z \preceq v_i$ for all $0 \leq i < n$. As always, denote the approximation of $\preceq$ by $\trianglelefteq$. By normalization of derivations, no v_i equals x and x is not needed for entanglement. From derivation-minimality we conclude $y \preceq_{\varphi''} v_i$ or $z \preceq_{\varphi''} v_i$, and that $(v_i \trianglelefteq) \setminus \{x\} \subseteq (v_i \preceq_{\varphi''})$ for all $0 \leq i \leq n$. By Claim 4, the occurrence of v_{i+1} in $\varphi'_{[(v_i\trianglelefteq)\setminus\{x\}]}$ or $\varphi''_{[(v_i\trianglelefteq)\setminus\{x\}]}$ implies occurrence of v_{i+1} in $\varphi''_{[v_i\preceq_{\varphi''}]}$. Similarly, v_i occurs in $\varphi''_{[v_{i+1}\preceq_{\varphi''}]}$. Thus, the entanglement chain is also such a one in φ''. Also $y \leq_{\varphi'} z$ implies $y \leq_{\varphi''} z$ so we conclude $y \preceq_{\varphi''} z$. ∎

Now for the converse.

Claim 6. Let y, z be variables different from x. Then $y \preceq_{\varphi''} z$ implies $y \preceq_{\varphi'} z$.

Proof. We work by induction on $\preceq_{\varphi''}$. The cases Reflexivity and Transitivity are immediate. For Alternation let $\trianglelefteq$ be some approximation of $\preceq_{\varphi''}$, assume $y \leq_{\varphi''} z$ and $Q_y \neq Q_z$ and let $y = w_0, \ldots, w_n = z$ be given such that for all $0 \leq i < n$ we have $y \preceq_{\varphi''} w_i$ or $z \preceq_{\varphi''} w_i$, occurrence of w_i in $\varphi''_{[w_{i+1} \trianglelefteq]}$, and occurrence of w_{i+1} in $\varphi''_{[w_i \trianglelefteq]}$. The inductive hypotheses are $y \preceq_{\varphi'} w_i$ or $z \preceq_{\varphi'} w_i$, and $(\trianglelefteq) \subseteq (\preceq_{\varphi'})$. As $\leq_{\varphi'}$ and $\leq_{\varphi''}$ differ only among V_1 and we have $Q_y \neq Q_z$, we conclude $y \leq_{\varphi'} z$. If furthermore all occurrences also hold in φ', then we are done. So assume that there is some i such that w_i is not entangled with w_{i+1} in φ', that is w_i does not occur in $\varphi'_{w_{i+1}}$ or w_{i+1} does not occur in φ'_{w_i}. Without loss of generality let us assume the former. Claim 4 then implies that w_i occurs in φ'_x and that $\varphi'_{w_{i+1}}$ is a subformula of φ'_x but not of any L_j with $j \in J^-$. By maximality of these L_j among the subformulae not containing x, this implies that x occurs in $\varphi'_{w_{i+1}}$. Of course w_{i+1} occurs in $\varphi'_{w_{i+1}}$ and thus in φ'_x. Hence x and w_{i+1} are entangled in φ'. Next let us show that also w_i and x are entangled. If w_{i+1} does not occur in φ'_{w_i}, then we can proceed as above, so we may assume that w_{i+1} does occur in φ'_{w_i}. Then w_i and w_{i+1} are comparable by $\leq_{\varphi'}$. If $w_{i+1} \leq_{\varphi'} w_i$, then xenerp normal form of φ' implies that w_i occurs in $\varphi'_{w_{i+1}}$ in contradiction to our assumption. Hence $w_i \leq_{\varphi'} w_{i+1}$ and by xenerp we have that $\varphi'_{w_{i+1}}$ is a subformula of φ'_{w_i}. In particular x occurs in φ'_{w_i} and we already know the converse. In this way, whenever a link of the φ''-chain does not hold in φ', we can insert x to obtain a longer valid chain. So far, validity only means that each link is an entanglement in φ'.

It remains to show that $y \preceq_{\varphi'} x$ or $z \preceq_{\varphi'} x$, if x needed to be inserted. Let v and v' be the neighbours of (one occurence of) x in the chain. If v and v' both are in $\{y, z\}$, then, because x is only inserted between non-entangled variables, one of them is y and the other one is z. As $Q_y \neq Q_z$, we have $Q_v \neq Q_x$ or $Q_{v'} \neq Q_x$, without loss of generality the former. As x and v are entangled in φ' by the above, they are comparable by $\leq_{\varphi'}$. Thus by Alternation we either have $x \preceq_{\varphi'} v$, contradicting $\preceq_{\varphi'}$-maximality of x, or $v \preceq_{\varphi'} x$, in which case we are done because $v \in \{y, z\}$. Now for the case that one of v and v' is not from $\{y, z\}$, without loss of generality $y \neq v \neq z$. Still, v is entangled with x. Furthermore, v is an original element of the chain, so $y \preceq_{\varphi'} v$ or $z \preceq_{\varphi'} v$. Without loss of generality let us assume the latter and recall that $z \neq v$. After unfolding, in the derivation of $z \preceq_{\varphi'} v$, some applications of Transitivity, we obtain some u such that $z \preceq_{\varphi'} u \preceq_{\varphi'} v$ and $u \preceq_{\varphi'} v$ is due to Alternation. Accordingly, let $u = t_0, \ldots, t_k = v$ be a witnessing entanglement chain. If $v \leq_{\varphi'} x$, then we can extend this chain by x to obtain (together with the fact $u \preceq_{\varphi'} v$) a witnessing chain for $u \preceq_{\varphi'} x$ which implies $z \preceq_{\varphi'} x$ using Transitivity. It remains to consider the case $v \not\leq_{\varphi'} x$. But as v, x are entangled, they are comparable by $\leq_{\varphi'}$, so $x \leq_{\varphi'} v$ and thus $v \in V_1$ implying $Q_v = Q_x$ and then $Q_u \neq Q_x$. As $u \preceq_{\varphi'} v$ we have $u \leq_{\varphi'} v$, so x and u are comparable by $\leq_{\varphi'}$. $x \leq_{\varphi'} u$ would imply $u \in V_2$ (since $Q_u \neq Q_x$), contradicting Claim 2. Hence $u \leq_{\varphi'} x$ so the chain $t_0, \ldots, t_k$ can be extended by x to show $u \preceq_{\varphi'} x$ and then $z \preceq_{\varphi'} x$. ∎

Hence $\preceq_{\varphi'}$ and $\preceq_{\varphi''}$ coincide outside of x. As x is $\preceq_{\varphi'}$-maximal, this implies that $\mathrm{ead}_{\varphi'}$ and $\mathrm{ead}_{\varphi''}$ coincide outside of x. Hence $\mathrm{ead}_{\varphi'}$-stratification of (T', B') implies $\mathrm{ead}_{\varphi''}$-stratification of (T'', B''). Finally, we turn φ'' into xenerp normal form.

It is easy to see that the algorithm terminates: the second case decreases the size of the tree decomposition while it leaves the set of variables intact. The third case leaves the tree intact and eliminates one variable. As both the tree and the set of variables are finite, eventually the first case must trigger and the algorithm stops. This concludes the proof of Theorem 5.5 □

It is well-known that first order query evaluation for $\mathcal{L}^k$ can be done in time $n^{k+O(1)}$, see [28].

Corollary 5.6. *Evaluating queries of bounded first order tree-width is fixed parameter tractable with parameter the length of the formula.*

More precisely, given a finite structure $\mathcal{A}$ and a formula $\varphi \in \mathcal{L}$, there is an algorithm that computes $\varphi(\mathcal{A})$ in time $\|\mathcal{A}\|^{\mathrm{fotw}(\varphi)+O(1)} f(|\varphi|)$ for some computable function f. □

Here the function f is basically the running time we need for translating the formula φ with $\mathrm{fotw}(\varphi) \leq k-1$ into an $\mathcal{L}^k$ formula. It is q-times exponential, where q is the alternation depth of φ. The exponentiations arise from converting some subformulae into disjunctive or conjunctive normal form. In special cases where this step is not needed, the running time is much lower.

Corollary 5.7. *Evaluating formulae without disjunctions and of bounded first order tree-width can be done in polynomial time.*

Proof. Let φ be a formula without disjunctions such that $\mathrm{fotw}(\varphi) < k$. Let us recall the algorithm underlying the proof of Theorem 5.5, 1. The argument for termination (at the very end of the proof) actually shows that the main loop is iterated only a quadratic number of times. Case 1 of the main loop is only executed once, and it is polynomial in the data then present. Case 2 basically requires a search through the current tree, which is a subtree of the original one. It remains to show that Case 3 runs in polynomial time and to bound the way it increases the size of the data. In fact, all individual steps of Case 3 run in polynomial time even for general φ, with the exception of turning ψ' into disjunctive normal form. But as our φ does not contain any disjunctions, the same holds for φ' and ψ', so ψ' already is in disjunctive normal form and there is nothing to do. Formally, the index set I only contains one element and (up to reordering) we have $\psi' = \psi_- \wedge \psi_+$, where $\psi_- := \bigwedge_{j \in J^-} L_j$ and $\psi_+ := \bigwedge_{j \in J^+} L_j$.

Let us now bound the data increase. The data consist of the tree decomposition (which only becomes smaller), the formula φ', and the substitution S. It is easier to consider, instead of φ' and S, the formula $\varphi' S$. The size difference between the two stems only from a collection of pairs (x, i), where x is a variable and i is an index from the respective I. As each such I is a singleton, there are at most $|\mathrm{var}(\varphi)|$ such pairs. With respect to $\varphi' S$, all that happens in Case 3 is shifting quantifiers and then replacing $\exists x \exists V_1(\psi_- \wedge \psi_+)$ by $\exists V_1(\psi_- \wedge \exists x \psi_+)$. The size does not change.

Hence the algorithm from Theorem 5.5, 1 runs in polynomial time. So does turning the formula from $\tilde{\mathcal{L}}^k$ into one from $\mathcal{L}^k$. We conclude with the time $n^{k+O(1)}$ for evaluating the latter. □

Of course, the same applies to formulae without conjunctions. One example of formulae without disjunctions are quantified constraint formulae, which we define and discuss in detail in Section 6.

The fixed parameter tractability from Corollary 5.6 does not remain when we make the first order tree-width part of the parameter. This is even true for model checking instead of evaluation. Instead, the problem becomes AW[∗]-hard: fotw is bounded by the length of the formula, and the model checking problem for first order logic, parameterized by the formula length, is AW[∗]-complete [12].

6. Relation to similar notions

In this section we show that fotw and tree-width coincide on conjunctive queries, while fotw is more powerful than both elimination-width of quantified constraint formulae and strict tree-width of non-recursive stratified datalog programs. Finally, we extend the cops and robber game characterizing tree-width to stratified tree-width and we prove that requiring monotonicity does not limit the cops.

A *(Boolean) conjunctive query* is a sentence $\varphi = \exists x_1 \dots \exists x_n \psi$, where ψ is a conjunction of relational atoms such that $\mathrm{var}(\psi) = \{x_1, \dots, x_n\}$. The *tree-width* of a conjunctive query φ, $\mathrm{tw}(\varphi)$, is defined as the tree-width of G_φ (see [25]). Any conjunctive query φ satisfies $\mathrm{ead}_\varphi = 1$. Hence the notion of fotw generalises the notion of tree width of conjunctive queries.

Remark 6.1. Any conjunctive query φ satisfies $\mathrm{fotw}(\varphi) = \mathrm{tw}(\varphi)$. □

6.1. **Quantified constraint formulae.** A *quantified constraint formula* [9] is a sentence

$$\varphi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \psi,$$

where $Q_i \in \{\forall, \exists\}$ for $i = 1 \dots n$ and ψ is a conjunction of relational atoms. In [9], Chen and Dalmau introduce the notion of *tree-width* of a quantified constraint formula and they show that model checking for quantified constraint formulae of bounded tree-width can be done in polynomial time using the k-consistency algorithm. Since their notion of tree-width is defined via an elimination ordering rather than via a decomposition, we call it *elimination-width* instead of *tree-width*. We show that fotw is less than or equal to Chen and Dalmau's elimination-width, and we give an example of a class of quantified constraint formulae having bounded fotw and unbounded elimination-width. By Corollary 5.7, model checking of quantified constraint formulae of bounded fotw can also be done in polynomial time.

Recall that by Theorem 3.1, any graph G and $d : V(G) \to \mathbb{N}$ satisfies $\mathrm{tw}(G, d) = \mathrm{ew}(G, d)$. Let φ be a quantified constraint formula with formula graph G_φ. Let ad'_φ be the mapping that assigns to a variable $v \in \mathrm{var}(\varphi)$ the number of quantifier changes occurring before v in the quantifier prefix of φ, adding $+1$ (note that here we always add $+1$, regardless of the first quantifier of φ). Then Chen and Dalmau's notion of elimination-width can be equivalently reformulated in our setting as $\mathrm{ew}(G_\varphi, \mathrm{ad}'_\varphi)$.

Lemma 6.2. *Let φ be a first order formula. Then* $\mathrm{fotw}(\varphi) \leq \mathrm{ew}(G_\varphi, \mathrm{ad}'_\varphi)$.

Proof. By Theorem 3.1, we may use tw and ew interchangeably. Then, by Theorem 3.25, it suffices to show $\mathrm{tw}(G_\varphi, \mathrm{ad}_\varphi) = \mathrm{tw}(G_\varphi, \mathrm{ad}'_\varphi)$. First, assume that φ starts with a quantifier. If this quantifier is existential, then $\mathrm{ad}_\varphi = \mathrm{ad}'_\varphi$. If it is universal, then $\mathrm{ad}_\varphi(x) = \mathrm{ad}'_\varphi(x) = 0$ for all free variables x, and $\mathrm{ad}_\varphi(x) - 1 = \mathrm{ad}'_\varphi(x) > 0$ for all bound variables x. In both cases, an elimination ordering respects ad_φ if, and only if, it respects ad'_φ. Hence $\mathrm{ew}(G_\varphi, \mathrm{ad}_\varphi) = \mathrm{ew}(G_\varphi, \mathrm{ad}'_\varphi)$ and thus $\mathrm{tw}(G_\varphi, \mathrm{ad}_\varphi) = \mathrm{tw}(G_\varphi, \mathrm{ad}'_\varphi)$.

The same holds trivially, when φ is quantifier free.

Now for the general case: φ is a positive boolean combination of formulae $\varphi_1, \dots, \varphi_n$ which are quantifier free or start with quantifiers. Let F be the set of free variables of φ. We already know that $\mathrm{tw}(G_{\varphi_i}, \mathrm{ad}_{\varphi_i}) = \mathrm{tw}(G_{\varphi_i}, \mathrm{ad}'_{\varphi_i})$ for all $1 \leq i \leq n$. As the various φ_i

have pairwise distinct bound variables, $G_\varphi - F$ is the disjoint union of the $G_{\varphi_i} - F$. Then it is easy to see that

$$\mathrm{tw}(G_\varphi, \mathrm{ad}_\varphi) = \max(|F| - 1, \max_{1 \le i \le n} \mathrm{tw}(G_{\varphi_i}, \mathrm{ad}_{\varphi_i}))$$

and

$$\mathrm{tw}(G_\varphi, \mathrm{ad}'_\varphi) = \max(|F| - 1, \max_{1 \le i \le n} \mathrm{tw}(G_{\varphi_i}, \mathrm{ad}'_{\varphi_i})),$$

from which it follows that $\mathrm{tw}(G_\varphi, \mathrm{ad}_\varphi) = \mathrm{tw}(G_\varphi, \mathrm{ad}'_\varphi)$. □

Remark 6.3. There exists a class of quantified constraint formulae having first order tree-width 1, where Chen and Dalmau's elimination-width is unbounded: let the class consist of the φ_n as in Example 3.26. Then $\mathrm{ad}'_{\varphi_n} = \mathrm{ad}_{\varphi_n}$, and hence $\mathrm{tw}(G_{\varphi_n}, \mathrm{ad}'_{\varphi_n}) = \mathrm{ew}(G_{\varphi_n}, \mathrm{ad}'_{\varphi_n}) = n$, while $\mathrm{fotw}(\varphi_n) = 1$. □

6.2. **Non-recursive stratified datalog.** In [16], Flum, Frick and Grohe define strict tree-width of non-recursive stratified datalog (NRSD) programs and they show that the evaluation problem for NRSD programs can be solved in polynomial time on programs of bounded strict tree-width [16, Corollary 5.26]. NRSD programs have the same expressive power as $\mathcal{L}$ and there are simple translations in both directions. This allows us to compare their notion with fotw. We show that if the NRSD program has tree-width at most k, then the corresponding first order formula has fotw at most k, and we exhibit a class of formulae with bounded fotw, whose corresponding NRSD programs have unbounded tree-width.

We assume that the reader is familiar with datalog and we only fix our notation, and we refer the reader to [16] otherwise. A *datalog rule ρ with negation* is an expression $Qx_1 \dots x_l \leftarrow \bigwedge_{i=1}^n \lambda_i$, where Q is a relation symbol and $x_1, \dots, x_n \in \mathrm{var}(\bigwedge_{i=1}^n \lambda_l)$ are pairwise distinct variables, and the λ_i are literals ($i = 1, \dots, n$). $Qx_1 \dots x_l$ is called the *head* of ρ, and $\bigwedge_{i=1}^n \lambda_i$ is called the *body* of ρ. To define the semantics, let $\mathcal{A}$ be a structure whose vocabulary contains all the relation symbols occurring in the body of ρ. Let $\bar{y}$ be a tuple that consists of all variables of $\mathrm{var}(\bigwedge_{i=1}^n \lambda_i) \setminus \{x_1, \dots, x_l\}$, and let $\varphi_\rho(x_1 \dots x_l) = \exists \bar{y} \bigwedge_{1 \le i \le n} \lambda_i$. We let $\rho(\mathcal{A}) := \varphi_\rho(\mathcal{A})$. A non-recursive stratified datalog (NRSD) program is a sequence $\Pi = (\Pi^1, \dots, \Pi^n)$ of non-recursive datalog programs Π^i (called the strata of Π) as defined in [16]. We denote the intentional vocabulary of Π by $\mathrm{int}(\Pi)$ and the extensional vocabulary of Π by $\mathrm{ext}(\Pi)$.

The *strict tree-width* of a datalog rule ρ is defined as $\mathrm{stw}(\rho) := \mathrm{tw}(G_{\varphi_\rho})$, and for an NRSD program $\Pi = (\Pi^1, \dots, \Pi^n)$ the *strict tree-width* of Π is defined as $\mathrm{stw}(\Pi) := \max\{\mathrm{stw}(\rho) \mid \rho \in \bigcup_{i=1}^n \Pi^i\}$.[3] The following is proved in [16], Corollary 5.26 (2).

Theorem 6.4 (Flum, Frick, Grohe)**.** *For fixed integer $k > 0$, the evaluation problem for* NRSD *programs of strict tree-width at most k can be solved in polynomial time.* □

[3] In [16], the term *strict* tree-width refers to the fact that tree decompositions are required to cover all variables in the head of a datalog rule together in some piece.

It is well known that a query is definable in first order logic if and only if it is NRSD definable. Actually, an NRSD program Π_Q (i.e. an NRSD program with goal predicate Q) defines an equivalent first order formula φ_{Π_Q} in a natural way, and vice versa. For proving our theorem, we make one direction explicit, associating a first order formula to an NRSD program as follows.

Let $\Pi = (\Pi^1, \ldots, \Pi^n)$ be an NRSD program and let $Q \in \mathrm{int}(\Pi^i)$ for some $i \leq n$. Suppose for all $Q' \in \mathrm{int}(\Pi)$ occuring in $(\Pi^1, \ldots, \Pi^{i-1})$, the formula $\varphi_{\Pi_{Q'}}$ is already defined. Let

$$\bar{\varphi}_{\Pi_Q} := \bigvee_{\rho \in \Pi^i,\ Q \text{ occurs in the head of } \rho} \varphi_\rho,$$

and let

$$\varphi_{\Pi_Q} := \bigvee_{\rho \in \Pi^i,\ Q \text{ occurs in the head of } \rho} \varphi^*_\rho,$$

where φ^*_ρ is obtained from φ_ρ by recursively replacing relation symbols $Q' \in \mathrm{int}(\Pi)$ occuring in φ_ρ by the corresponding formula $\bar{\varphi}_{\Pi_{Q'}}$ (i.e. φ_{Π_Q} is an $\mathrm{ext}(\Pi)$-formula). Let $\mathcal{A}$ be an $\mathrm{ext}(\Pi)$-structure. It is easy to see that we have $\Pi_Q(\mathcal{A}) = \varphi_{\Pi_Q}(\mathcal{A})$.

Recall that ad'_φ is the mapping that assigns to a variable $v \in \mathrm{var}(\varphi)$ the alternation depth of v in φ.

Theorem 6.5. *Any* NRSD *program* Π *with* $Q \in \mathrm{int}(\Pi)$ *satisfies* $\mathrm{fotw}(\varphi_{\Pi_Q}) \leq \mathrm{tw}(G_{\varphi_\Pi}, \mathrm{ad}'_{\varphi_{\Pi_Q}}) \leq \mathrm{stw}(\Pi)$.

Proof. The first inequality follows from Theorem 3.1 and Lemma 6.2.

Towards the second inequality, let $\Pi = (\Pi^1, \ldots, \Pi^n)$ and let $k := \mathrm{stw}(\Pi) = \max\{\mathrm{stw}(\varphi_\rho) \mid \rho \in \bigcup_{i=1}^n \Pi_i\}$. We prove by induction on the number n of strata of Π that all $Q \in \mathrm{int}(\Pi)$ satisfy $\mathrm{tw}(G_{\varphi_{\Pi_Q}}, \mathrm{ad}'_{\varphi_{\Pi_Q}}) \leq k$. Let $Q\bar{x}$ be the head corresponding to Q and let i be such that $Q \in \mathrm{int}(\Pi^i)$. Then φ_{Π_Q} has exactly the free variables $\bar{x}$ and we have

$$\varphi_{\Pi_Q} = \bigvee_{\rho \in \Pi^i,\ Q \text{ occurs in the head of } \rho} \varphi^*_\rho.$$

Suppose all $Q' \in \mathrm{int}(\Pi)$ occuring in $(\Pi^1, \ldots, \Pi^{i-1})$ satisfy $\mathrm{tw}(\mathrm{ad}'_{\varphi_{\Pi_{Q'}}}, G_{\varphi_{\Pi_{Q'}}}) \leq k$.

We may assume that $\bar{\varphi}_{\Pi_Q}$ and φ_{Π_Q} are straight.

First we construct a tree decomposition for $\bar{\varphi}_{\Pi_Q}$ as follows. For every $\rho \in \Pi^i$ with head $Q\bar{x}$, we take a tree decomposition of width at most k of G_{φ_ρ}. Each of these decompositions has a piece containing the variables $\bar{x}$. We glue them together at one new root covering the variables $\bar{x}$. Then we orient the edges of the decomposition tree away from the root, and we obtain a tree decomposition $(\bar{T}, \bar{B})$ for $\bar{\varphi}_{\Pi_Q}$ of width at most k. By the inductive hypothesis, for every $Q' \in \mathrm{int}(\Pi)$ such that $Q'\bar{y}$ occurs in $\bar{\varphi}_{\Pi_Q}$, we have an $\mathrm{ad}'_{\varphi_{\Pi_{Q'}}}$-stratified tree decomposition $(T^{Q'}, B^{Q'})$ of $\varphi_{\Pi_{Q'}}(\bar{y})$. By definition, the variables in $\bar{y}$ are contained in the piece at the root r of $T^{Q'}$. Moreover, $\bar{y}$ is covered in some piece of $(\bar{T}, \bar{B})$. We choose such a piece $\bar{B}_t$ and we attach $(T^{Q'}, B^{Q'})$ to this piece such that r becomes a new successor of t. Having done this for all atoms $Q'\bar{y}$ (with $Q \neq Q'$), that occur in $\bar{\varphi}_{\Pi_Q}$, we obtain a tree decomposition (T, B) for $G_{\varphi_{\Pi_Q}}$ of width at most k. We may assume that φ_{Π_Q} is in negation normal form. (If not, we transform φ_{Π_Q} into negation normal form. Note that this does not change the formula graph.) Then, by construction, (T, B) is $\mathrm{ad}'_{\varphi_{\Pi_Q}}$-stratified. □

The following remark shows that the difference can be unbounded in the opposite direction. Moreover, it shows that the difference between Chen and Dalmau's elimination-width and tree-width of NRSD programs can be unbounded.

Remark 6.6. There is a class $\mathcal{C}$ of NRSD programs with unbounded strict tree-width, such that $\mathrm{fotw}(\varphi_\Pi) = \mathrm{tw}(G_{\varphi_\Pi}, \mathrm{ad}'_{\varphi_\Pi}) = 0$ for all $\Pi \in \mathcal{C}$.

Proof. For an integer $n > 0$ let $\psi_n := \exists x_1 \dots \exists x_{n-1} \forall x_n \big(\bigwedge_{i=1}^{n} P x_i \big)$. Take $\mathcal{C}$ to consist of the natural NRSD programs which are equivalent to the formulae ψ_n, for $n > 0$. □

6.3. **Cops, Robbers and stratified tree-width.** We now introduce the cops and robbers game as defined in [27]. Let G be a graph and let $k \geq 0$ be an integer. The cops and robbers game on G (with game parameter k) is played by two players, the *cop player* and the *robber player*, on the graph G. The cop player controls k cops and the robber player controls the robber. Both the cops and the robber move on the vertices of G. Some of the cops move to at most k vertices and the robber stands on a vertex r not occupied by the cops. In each move, some of the cops fly in helicopters to at most k new vertices. During the flight, the robber sees which position the cops are approaching and before they land she quickly tries to escape by running arbitrarily fast along paths of G to a vertex r', not being allowed to run through a standing cop. Hence, if $X \subseteq V(G)$ is the cops' first position, the robber stands on $r \in V(G) \setminus X$, and after the flight, the cops occupy the set $Y \subseteq V(G)$, then the robber can run to any vertex r' within the connected component of $G \setminus (X \cap Y)$ containing r. The cops win if they land a cop via helicopter on the vertex occupied by the robber. The robber wins if she can always elude capture. *Winning strategies* are defined in the usual way. The *cop-width* of G, $\mathrm{cw}(G)$, is the minimum number of cops having a winning strategy on G.

A winning strategy for the cops is *monotone*, if for all plays played according to the strategy, if $X_1, X_2, \dots$ is the sequence of cop positions, then the connected components R_i of $G \setminus X_i$ containing the robber form a decreasing (with respect to $\subseteq$) sequence. The R_i are called the *robber spaces*. The *monotone cop-width* of G, $\text{mon-cw}(G)$, is the minimum number of cops having a monotone winning strategy on G.

Theorem 6.7 (Seymour, Thomas [27])**.** *Any graph G satisfies* $\mathrm{tw}(G) + 1 = \mathrm{cw}(G) = \text{mon-cw}(G)$. □

Now let G be a graph and let d be a function $d\colon V(G) \to \mathbb{N}$. The *$d$-stratified* cops and robbers game on G is played as the cops and robbers game on G, but in every move the cops have to satisfy the following additional condition. Intuitively, they can only clear vertices v with $d(v) = i$ after they have cleared all vertices w with $d(w) < i$. More precisely: for every move (X, R), where $X \subseteq V(G)$ is the cop position and R is the robber space, the cops have to make sure that $\max\{d(x) \mid x \in X\} \leq \min\{d(r) \mid r \in R\}$. Then $\mathrm{cw}(G, d)$ and $\text{mon-cw}(G, d)$ are defined analogously, and for a formula φ we let $\mathrm{cw}(\varphi) := \mathrm{cw}(G_\varphi, \mathrm{ead}_\varphi)$ and $\text{mon-cw}(\varphi) := \text{mon-cw}(G_\varphi, \mathrm{ead}_\varphi)$.

Although proving the following theorem is not very hard, it seems interesting to know that $\mathrm{cw}(G, d)$ and $\text{mon-cw}(G, d)$ coincide. In many generalisations of the cops and robbers game to other settings, the analogous statements become false [1, 2, 26], and it might be helpful to explore the borderline.

Theorem 6.8. *Let G be a graph and $d : V(G) \to \mathbb{N}$. The following statements are equivalent:*

(1) $\mathrm{tw}(G, d) \leq k - 1$,
(2) $\text{mon-cw}(G, d) \leq k$, *and*
(3) $\mathrm{cw}(G, d) \leq k$.

In particular, any first order formula φ satisfies $\mathrm{fotw}(\varphi) + 1 = \text{mon-cw}(\varphi) = \mathrm{cw}(\varphi)$.

Proof. $1 \Rightarrow 2$: suppose $\mathrm{tw}(G, d) \leq k - 1$. Let (T, B) be a d-stratified tree decomposition of width at most $k - 1$ for G. From (T, B), the k cops can read off a monotone winning strategy in the usual way, first moving to B_r and then following the robber down the tree decomposition into the unique direction where the robber space is covered (see e.g. [3]). Since (T, B) is d-stratified, the winning strategy is also d-stratified.

$2 \Rightarrow 3$: any monotone winning strategy is a winning strategy.

$3 \Rightarrow 1$: suppose k cops have a winning strategy for the d-stratified game on G.

Claim 1. Let t, u be nodes of (G, d)'s component tree, where u is a child of t. Let $x, y \in D_t \cap D_u$. If, while playing against the k cops, the robber can move to x and no cop will land on y, then the robber can also move to y.

Let i be the depth of t in (G, d)'s component tree and let C be the connected component of $G[C_t \setminus D_t]$ with $C_u = C \cup N(C)$. As $x, y \in D_t$ we have $x, y \notin C$, and hence $x, y \in N_G(C)$ and there is a path from x to y with all internal vertices in C. Hence $d(z) > i \geq d(x)$ and $d(z) > i \geq d(y)$ for all internal vertices of the path. Therefore, as long as the cops play on vertices with d at most i, the path is free and the robber can use it. But the cops can never move to a vertex with $d > i$ before they have cleared $D_t \cap D_u$ completely, so the path from x to y is free whenever the robber can move to x. ∎

Recall, from Section 4, that the graph $G^{(0)}$ is obtained from G by adding all edges between any pair of distinct vertices $x, y \in D_t \cap D_u$ for all directed edges (t, u) of the component tree. By the claim, any d-stratified winning strategy for k cops on G is also a d-stratified winning strategy on $G^{(0)}$. Forgetting d, obviously, k cops have a winning strategy on $G^{(0)}$ and hence in particular on $G^{(0)}[D_t]$ for all nodes t of (G, d)'s component tree. Thus $\mathrm{cw}(G^{(0)}[D_t]) \leq k$ and by Theorem 6.7 this implies $\mathrm{tw}(G^{(0)}[D_t]) \leq k - 1$. From the tree decompositions of the $G^{(0)}[D_t]$ of width $\leq k$ we can now construct tree decompositions for the $G^{(0)}[C_t]$ of width $\leq k - 1$ in a bottom-up manner as in the proof of Theorem 4.7. For the root r of (G, d)'s component tree, we have $G^{(0)} = G^{(0)}[C_r]$ and it is easy to see that the tree decomposition for $G^{(0)}[C_r]$ of width $\leq k$ obtained in this way is d-stratified. □

7. Conclusion

We introduced a notion of tree-width for first order formulae φ, $\mathrm{fotw}(\varphi)$, generalising tree-width of conjunctive queries and elimination-width of quantified constraint formulae [9]. Our notion can also be seen as an adjustment of the notion of tree-width of first order formulae as defined in [16] (which only works for conjunctive queries with negation).

We proved that computing fotw is fixed-parameter tractable with parameter fotw (Theorem 4.7). Moreover, we showed that evaluating formulae of k-bounded first order tree-width is fixed-parameter tractable, with parameter the length of the formula (Theorem 5.5). This is done by first computing a tree decomposition of width at most k for the

formula, and then translating the formula equivalently into a formula of the k-variable fragment $\mathcal{L}^k$ of first order logic. It is well-known that evaluating $\mathcal{L}^k$ formulae can be done in polynomial time. When translating the formula φ into an equivalent $\mathcal{L}^k$ formula, we get a non-elementary explosion in the running time.

Conjecture 7.1. When translating a formula φ satisfying $\text{fotw}(\varphi) \leq k$ into an equivalent $\mathcal{L}^k$ formula, a non-elementary explosion cannot be avoided.

Moreover, it is still unknown whether the explosion can be avoided in parameterized algorithms for evaluating queries of bounded first order tree-width.

We show that first order tree-width can be characterised by other notions such as elimination-width (Theorem 3.1), and the minimum number of cops necessary to catch the robber in the stratified cops and robbers game, as well as the minimum number of cops necessary in the monotone version of the game (Theorem 6.8). Hence our notion is very natural and robust.

Moreover, we showed that fotw is more powerful than the notion of elimination-width of quantified constraint formulae as defined in [9]: for quantified constraint formulae, both bounded elimination-width and bounded fotw allow for model checking in polynomial time. We proved that if φ is a quantified constraint formula, then $\text{fotw}(\varphi)$ is bounded by the elimination-width of φ, and there are classes of quantified constraint formulae with bounded fotw and unbounded elimination-width.

Finally, we showed that fotw is more powerful than tree-width of non-recursive stratified datalog (NRSD) programs [16]. NRSD programs have the same expressive power as first order logic, in the sense that NRSD programs correspond to first order formulae and vice versa. We showed that first-order tree-width of (formula versions of) NRSD programs is bounded by the strict tree-width of the programs and that there are classes of first order formulae with bounded fotw, whose corresponding NRSD programs have unbounded strict tree-width.

For conjunctive query evaluation, methods more powerful than bounded tree-width are known. Conjunctive queries of bounded hypertree-width [20], bounded fractional hypertree-width [23] and bounded (hyper)closure tree-width [4] yield even larger tractable classes of instances. For example, conjunctive queries of bounded hypertree-width correspond to the k-guarded fragment of first order logic [21], and similar correspondences can be found for the other invariants. Why not generalise these notions to first order formulae? By generalising these notions to first order formulae φ in the obvious way, a decomposition of bounded width would not give us an instruction how to translate φ into the corresponding guarded fragment of first order logic (transforming subformulae of φ into conjunctive normal form as in the proof of Theorem 5.5, 1 does not necessarily yield guarded subformulae).

Nevertheless, generalising these notions to quantified constraint formulae should indeed yield classes with an efficient query evaluation, that are strictly larger than classes of quantified constraint formulae of bounded first order tree-width. It would be interesting to find the largest fragment of first order formulae for which such a generalization is possible.

References

[1] Isolde Adler. Marshals, monotone marshals, and hypertree-width. *Journal of Graph Theory*, 47(4):275–296, 2004.

[2] Isolde Adler. Directed tree-width examples. *J. Comb. Theory, Ser. B*, 97(5):718–725, 2007.

[3] Isolde Adler. Tree-related widths of graphs and hypergraphs. *SIAM J. Discrete Math.*, 22(1):102–123, 2008.

[4] Isolde Adler. Tree-width and functional dependencies in databases. In M. Lenzerini and D. Lembo, editors, *PODS*, pages 311–320. ACM, 2008.
[5] Stefan Arnborg. Efficient algorithms for combinatorial problems with bounded decomposability - a survey. *BIT*, 25(1):2–23, 1985.
[6] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.
[7] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90. ACM, 1977.
[8] Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. *Theor. Comput. Sci.*, 239(2):211–229, 2000.
[9] Hubie Chen and Víctor Dalmau. From pebble games to tractability: An ambidextrous consistency algorithm for quantified constraint satisfaction. In C.-H. Luke Ong, editor, *CSL*, volume 3634 of *Lecture Notes in Computer Science*, pages 232–247. Springer, 2005.
[10] Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In Pascal Van Hentenryck, editor, *CP*, volume 2470 of *Lecture Notes in Computer Science*, pages 310–326. Springer, 2002.
[11] Reinhard Diestel. *Graph theory*. Springer, Berlin, 2006.
[12] Rod G. Downey, Michael R. Fellows, and Udayan Taylor. The parameterized complexity of relational database queries and an improved characterization of W[1]. In D. S. Bridges, C. Calude, P. Gibbons, S. Reeves, and I. H. Witten, editors, *Combinatorics, Complexity, and Logic Proceedings of DMTCS 96*, pages 194–213. Springer-Verlag, 1996.
[13] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.
[14] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer, 1990.
[15] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.
[16] Jörg Flum, Markus Frick, and Martin Grohe. Query evaluation via tree-decompositions. *J. ACM*, 49(6):716–752, 2002.
[17] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Secaucus, NJ, USA, 2006.
[18] Eugene C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *AAAI*, pages 4–9, 1990.
[19] Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. The complexity of quantified constraint satisfaction problems under structural restrictions. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 150–155. Professional Book Center, 2005.
[20] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64:579–627, 2002.
[21] Georg Gottlob, Nicola Leone, and Francesco Scarcello. Robbers, marshals, and guards: Game theoretic and logical characterizations of hypertree width. *Journal of Computer and System Sciences*, 66:775–808, 2003.
[22] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), 2007.
[23] Martin Grohe and Dániel Marx. Constraint solving via fractional edge covers. In *SODA*, pages 289–298. ACM Press, 2006.
[24] Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In *STOC*, pages 657–666, 2001.
[25] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. Comput. Syst. Sci.*, 61(2):302–332, 2000.
[26] Stephan Kreutzer and Sebastian Ordyniak. Digraph decompositions and monotonicity in digraph searching. In Hajo Broersma, Thomas Erlebach, Tom Friedetzky, and Daniël Paulusma, editors, *WG*, volume 5344 of *Lecture Notes in Computer Science*, pages 336–347, 2008.
[27] Paul D. Seymour and Robin Thomas. Graph searching and a min-max theorem for tree-width. *J. Comb. Theory, Ser. B*, 58(1):22–33, 1993.
[28] Moshe Y. Vardi. On the complexity of bounded-variable queries (extended abstract). In *PODS '95: Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 266–276, New York, NY, USA, 1995. ACM.

[29] Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, pages 82–94. IEEE Computer Society, 1981.