Structural Tractability of Counting of Solutions to Conjunctive Queries

Arnaud Durand[*]
IMJ UMR 7586 - Logique
Université Paris Diderot
F-75205 Paris, France
durand@logique.jussieu.fr

Stefan Mengel[†]
Institute of Mathematics
University of Paderborn
D-33098 Paderborn, Germany
smengel@mail.uni-paderborn.de

ABSTRACT

In this paper we explore the problem of counting solutions to conjunctive queries. We consider a parameter called the *quantified star size* of a formula φ which measures how the free variables are spread in φ. We show that for conjunctive queries that admit nice decomposition properties (such as being of bounded treewidth or generalized hypertree width) bounded quantified star size exactly characterizes the classes of queries for which counting the number of solutions is tractable. This also allows us to fully characterize the conjunctive queries for which counting the solutions is tractable in the case of bounded arity. To illustrate the applicability of our results, we also show that computing the quantified star size of a formula is possible in time $n^{O(k)}$ for queries of generalized hypertree width k. Furthermore, quantified star size is even fixed parameter tractable parameterized by some other width measures, while it is **W[1]**-hard for generalized hypertree width and thus unlikely to be fixed parameter tractable. We finally show how to compute an approximation of quantified star size in polynomial time where the approximation ratio depends on the width of the input.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Languages—*Query languages*; F.1.3 [**Computation by Abstract Devices**]: Complexity measures and classes; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Hypergraphs*

General Terms

Theory

Keywords

conjunctive queries, counting complexity, hypergraph decomposition techniques

1. INTRODUCTION

Conjunctive queries (CQs) are a fundamental class of logical queries that consist of evaluating an existential conjunctive first-order formula over a finite structure. They admit a number of equivalent formulations for example as select-project-join queries in database theory or as homomorphism problems in constraint satisfaction and thus have been extensively studied in various contexts. Deciding if a Boolean CQ is true or not on a structure is well known to be **NP**-complete, so the main interest of study has been to identify tractable subclasses, so-called "islands of tractability", where the decision question is tractable, i.e. can be solved in polynomial time.

One main direction in finding tractable classes of CQs has been imposing structural restrictions on the formula of the query – more exactly on the hypergraph associated to it – while the database is assumed to be arbitrary. In a seminal paper Yannakakis [25] proved that if the formula is acyclic, then the Boolean CQ question becomes tractable. The main idea behind structural restrictions is to extend this result by generalizing it to "nearly acyclic" queries. This has lead to many different decompositions for graphs and hypergraphs and associated width measures (see e.g. [13, 8, 23]). The common approach for these decompositions is to group together vertices or edges (of the graphs or hypergraphs) into clusters of some fixed constant size and to arrange these clusters into a tree. The resulting width measures are often sought to have two desirable properties:

- For every k the class of queries of width k should be tractable, i.e. Boolean CQ should be solvable in polynomial time.

- Given an instance it should be possible to decide if there is a decomposition of width k and construct one if it exists.

While decomposition techniques without the first property do not make any sense in the context of CQs, the second property is sometimes relaxed. For some decomposition

techniques one does not actually need the decomposition to solve the Boolean query problem [6], a promise of the existence is enough. For other decompositions one only knows approximation algorithms that construct decompositions of width that is near the optimal width, which is enough to guarantee tractability of Boolean CQ [22, 1].

More recently there has also been interest in enumerating all solutions to CQs and in the corresponding counting question. For enumeration of the query answers it turns out that the picture is less clear than for decision [2, 4, 16]. Also the situation for counting is more subtle: For quantifier free queries – which correspond to queries without projections in the database perspective – most commonly considered structural restrictions yield tractable counting problems (see, e.g. [24]). While this is nice it is not fully satisfying, because quantifiers/projections are very natural and essential in database queries. While introducing projections does not make any difference for the complexity of Boolean CQ, the situation for the associated counting problem, denoted #CQ, is dramatically different. In [24] it is shown that even one single existentially quantified variable is enough to make counting answers to CQs #**P**-hard even when the structure of the query is a tree (which implies width 1 for all commonly considered decomposition techniques). This underlines the gain of expressive power obtained by existential quantification in the context of counting. It also follows that the decomposition techniques used for Boolean CQ are not enough to guarantee tractability for counting.

In a previous paper [10] the authors of this paper have proposed a way out of this dilemma for counting by introducing a parameter called *quantified star size* for acyclic conjunctive queries (ACQs). This parameter measures how the free variables are spread in the formula. We represented a query formula $\varphi(\mathbf{x})$ with a list $\mathbf{x}$ of free variables, by extending the hypergraph $\mathcal{H} = (V, E)$ associated to $\varphi(\mathbf{x})$ with a set $S \subseteq V$. Then the quantified star size is the size of a maximum independent set consisting of vertices from the set S in some specified subhypergraphs of $\mathcal{H}$. It turns out that this measure precisely characterizes the tractable subclasses of ACQs. The main result is that (under the widely believed assumption $\mathbf{FPT} \neq \#\mathbf{W}[\mathbf{1}]$ from parameterized complexity) solutions to a class of ACQs can be counted in polynomial time if and only if the queries in the class are of bounded quantified star size.

Overview of the results

Counting solutions to queries. In this paper we extend the results of [10] from acyclic queries to commonly considered decomposition techniques. To do so we generalize the notion of quantified star size from acyclic queries to general conjunctive queries. We show that every class of CQs that allows efficient counting must be of bounded quantified star size – again under the same assumption from parameterized complexity. We then go on showing that for all decomposition techniques for CQs commonly considered in the literature combining them with bounded quantified star size leads to tractable counting problems. The key feature that makes this result work is the organization of atoms into a tree of clusters that is prominent in all decomposition methods for CQs known so far. Combining the results above we get an exact characterization of the classes of tractable CQ counting problems for commonly considered decomposition techniques.

Let us illustrate these results for the example of generalized hypertree decomposition [13], which is one of the most general decomposition methods and one of the most studied too [13, 15, 23]. We have that, under the assumption that $\mathbf{FPT} \neq \#\mathbf{W}[\mathbf{1}]$, for any (recursively enumerable) class $\mathcal{C}$ of hypergraphs of bounded generalized hypertreewidth the following statements are equivalent:

- #CQ for instances in $\mathcal{C}$ can be solved in polynomial time

- $\mathcal{C}$ is of bounded quantified star size.

In our considerations, the arity of atoms of queries is not a priori bounded. In this setting, there is no known *ultimate* measure resulting from a decomposition method that fully characterizes tractability even for Boolean CQ. This explains why our characterizations are stated for *each* decomposition method. For bounded arity however, the situation is different. It is well known that being of bounded treewidth completely characterizes tractability for decision [19, 17] and counting [9] for CSP (corresponding to quantifier free conjunctive queries in this setting). Combining [19, 17] and our results from above we derive a complete characterization of tractability for #CQ in terms of tree width and quantified star size for the bounded arity case.

Note that our results are for counting with set semantics, i.e. we count each solution only once. Counting for bag semantics in which multiple occurences of identical tuples are counted has already been essentially solved in [24].

Discovering quantified star size. To exploit tractability results of the above kind it is helpful if the membership in a tractable class can be decided efficiently, i.e. in our case if computing the quantified star size is also tractable. In the second part of the paper, we turn to these "discovery problems" of determining the quantified star size of queries.

In [10] it is shown that quantified star size of acyclic CQs can be determined in polynomial time. Since star size is equivalent to independent sets, we cannot expect this to be true on more general queries anymore. Fortunately, it turns out that for queries of generalized hypertree width k, there is a n^k algorithm that computes the quantified star size. We show that this is in a sense optimal, because under the assumption $\mathbf{FPT} \neq \mathbf{W}[\mathbf{1}]$ there is no efficient (fixed paramater tractable in k) algorithm computing the quantified star size for queries parameterized by generalized hypertree width.

Still some natural decomposition methods admit fixed parameter discovery algorithms. We prove that this is the case for the class of CQ having bounded hingetree width (see [8]). This result is interesting on his own from a hypergraph algorithms perspective. Because of the connection between star size and maximum independent set, it provides a new class of hypergraphs for which computing the maximum independent set is $\mathbf{FPT}$. Note that the preceding hardness result shows that fixed parameterized tractability of this problem is unlikely for other hypergraph decomposition techniques.

We then turn our attention to star size approximation. We show that there is a polynomial time approximation algorithm with ratio k that given a decomposition of width k runs in time independent of k.

Summing these results up, quantified star size does not only imply tractable counting if combined with well known

decomposition techniques, but in case the decomposition is given or can be efficiently computed (hypertreewidth, hinge-tree width) or approximated (generalized hypertreewidth), then computing quantified star size is itself tractable.

Finally, we investigate the problem of counting solution and computing quantified star size for queries of bounded fractional hypertree width [18, 22]. This decomposition method is of a somewhat different nature than the ones studied before so we treat it individually. We again prove that counting is tractable in this setting and that the discovery problem can be decided in $O(n^{k^{O(1)}})$ i.e. with a slightly bigger dependency in k than before.

2. PRELIMINARIES

Conjunctive queries. We assume the reader to be familiar with the basics of (first order) logic (see [21]). We assume all formulas to be in prenex form. If ϕ is a first order formula, $\mathsf{var}(\phi)$ denotes the set of its variables, $\mathsf{free}(\phi) \subseteq \mathsf{var}(\phi)$ the set of its free variables and $\mathsf{atom}(\phi)$ the set of its atomic formulas. Let $\mathbf{x} = x_1, ..., x_k$, we denote $\phi(\mathbf{x})$ the formula with free variables $\mathbf{x}$. If ϕ is such that $\mathsf{free}(\phi) = \mathsf{var}(\phi)$ then ϕ is said to be *quantifier-free*. The *Boolean query problem* $\Phi = (\mathcal{A}, \phi)$ associated to a formula $\phi(\mathbf{x})$ and a structure $\mathcal{A}$, asks whether the set

$$\phi(\mathcal{A}) = \{\mathbf{a} : (\mathcal{A}, \mathbf{a}) \models \phi(\mathbf{x})\}$$

called the *query result* is empty or not. The (general) query problem consists of computing the set $\phi(\mathcal{A})$, while the corresponding counting problem is computing the size of $\phi(\mathcal{A})$, denoted by $|\phi(\mathcal{A})|$. We call two instances $\Phi = (\mathcal{A}, \phi), \Phi' = (\mathcal{A}', \phi')$ solution equivalent, if $\mathsf{free}(\phi) = \mathsf{free}(\phi')$ and $\phi(\mathcal{A}) = \phi'(\mathcal{A}')$. When ϕ is a $\{\exists, \wedge\}$-first order formula the boolean query problem is known as the *Conjunctive Query Problem*, CQ for short. It is well known that the the Boolean CQ problem is **NP**-complete. We denote by #CQ the associated counting problem: given a query instance $\Phi = (\mathcal{A}, \phi)$, return $|\phi(\mathcal{A})|$.

Any $\mathbf{a} \in \phi(\mathcal{A})$ will be alternatively seen as an assignment $\mathbf{a} : \mathsf{free}(\phi) \to D$ or as a tuple of dimension $|\mathsf{free}(\phi)|$. Two assignments $\mathbf{a}$ and $\mathbf{a}'$ are *compatible* (symbol: $\mathbf{a} \sim \mathbf{a}'$) if they agree on their common variables.

DEFINITION 2.1. *Let $\phi(\mathbf{x}, \mathbf{y})$, $\psi(\mathbf{y}, \mathbf{z})$ be two conjunctive queries with $\mathbf{x} \cap \mathbf{z} = \emptyset$ and let $\mathcal{A}, \mathcal{A}'$ be two finite structures. The the natural join of ϕ and ψ is $\phi(\mathcal{A}) \bowtie \psi(\mathcal{A}') :=$ $\{(\mathbf{a}, \mathbf{b}, \mathbf{c}) : (\mathbf{a}, \mathbf{b}) \in \phi(\mathcal{A})$ and $(\mathbf{b}, \mathbf{c}) \in \psi(\mathcal{A}')\}$*

When $\mathcal{A} = \mathcal{A}'$, $\phi(\mathcal{R}) \bowtie \psi(\mathcal{A})$ is simply $[\phi \wedge \psi](\mathcal{A})$.

Query size and Model of computation. The underlying model of computation for our algorithms will be the RAM model with unit costs. We assume the relations of a conjunctive query to be encoded by listing their tuples. For a relation $\mathcal{R}$ let $\mathrm{arity}(\mathcal{R})$ denote the arity of $\mathcal{R}$ and $|\mathcal{R}|$ the number of tuples in $\mathcal{R}$. Then the size of an encoding of $\mathcal{R}$ is $\|\mathcal{R}\| := \Theta(\mathrm{arity}(\mathcal{R}) \cdot |\mathcal{R}|)$. For a vocabulary τ let $|\tau|$ be the number of predicate symbols. Finally, let $|D|$ be the size of a domain D. Then encoding a structure $\mathcal{A}$ over the vocabulary τ with domain D takes space $\|\mathcal{A}\| := |\tau| + |D| + \sum_{\mathcal{R} \in \tau} \|\mathcal{R}^{\mathcal{A}}\|$.

Furthermore, it takes space $\|\phi\| := \Theta(\sum_{P \in \mathsf{atom}(\phi)} \mathrm{arity}(P))$ to encode a formula ϕ. The size of an encoding of a CQ instance $\Phi = (\phi, \mathcal{A})$ is then $\|\Phi\| := \|\phi\| + \|\mathcal{A}\|$.

For a detailed discussion and justification of these conventions see [11, Section 2.3]

Parameterized complexity. This section is a very short introduction to some notions from parameterized complexity used in the remainder of this paper (for more details see [12]).

A parameterized decision problem over an alphabet Σ is a language $L \subseteq \Sigma^*$ together with a computable parameterization $\kappa : \Sigma^* \to \mathbb{N}$. The problem (L, κ) is said to be fixed parameter tractable, or $(L, \kappa) \in \mathbf{FPT}$, if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that there is an algorithm that decides for $x \in \Sigma^*$ in time $f(\kappa(x))|x|^{O(1)}$ if x is in L.

Let (L, κ) and (L', κ') be two parameterized decision problems over the alphabets Σ resp. Π. A parameterized many-one reduction from (L, κ) to (L', κ') is a function $r : \Sigma^* \to \Pi^*$ such that for all $x \in \Sigma^*$:

- $x \in L \Leftrightarrow r(x) \in L'$,

- $r(x)$ can be computed in time $f(\kappa(x))|x|^c$ for a computable function f and a constant c, and

- $\kappa'(r(x)) \leq g(\kappa(x))$ for a computable function g.

It is easy to see that $\mathbf{FPT}$ is closed under parameterized many-one reductions.

Let p-Clique be the problem of deciding on an input (G, k) where G is a graph and k and integer, if G has a k-clique. Here the parameterization κ is simply defined by $\kappa(G, k) := k$. The class $\mathbf{W}[\mathbf{1}]$ consists of all parameterized problems that are parameterized many-one reducible to p-Clique. A problem (L, κ) is called $\mathbf{W}[\mathbf{1}]$-hard, if there is a parameterized many-one reduction from p-Clique to (L, κ).

It is widely believed that $\mathbf{FPT} \neq \mathbf{W}[\mathbf{1}]$ and thus in particular p-Clique and all $\mathbf{W}[\mathbf{1}]$-hard problems are not fixed parameter tractable.

Parameterized counting complexity theory is developed similarly to decision complexity. A parameterized counting problem is a function $F : \Sigma^* \times \mathbb{N} \to \mathbb{N}$, for an alphabet Σ. Let $(x, k) \in \Sigma^* \times \mathbb{N}$, then we call x the input of F and k the parameter. A parameterized counting problem F is fixed parameter tractable, or $F \in \mathbf{FPT}$, if there is an algorithm computing $F(x, k)$ in time $f(k) \cdot |x|^c$ for a computable function $f : \mathbb{N} \to \mathbb{N}$ and a constant $c \in \mathbb{N}$.

Let $F : \Sigma^* \times \mathbb{N} \to \mathbb{N}$ and $G : \Pi^* \times \mathbb{N} \to \mathbb{N}$ be two parameterized counting problems. A parameterized parsimonious reduction from F to G is an algorithm that computes for every instance (x, k) of F an instance (y, l) of G in time $f(k) \cdot |x|^c$ such that $l \leq g(k)$ and $F(x, k) = G(y, l)$ for computable functions $f, g : \mathbb{N} \to \mathbb{N}$ and a constant $c \in \mathbb{N}$. A parameterized T-reduction from F to G is an algorithm with an oracle for G that solves any instance (x, k) of F in time $f(k) \cdot |x|^c$ in such a way that for all oracle queries the instances (y, l) satisfy $l \leq g(k)$ for computable functions f, g and a constant $c \in \mathbb{N}$.

Let p-#Clique be the problem of counting k-cliques in a graph where k is the parameter and the graph is the input. A parameterized problem F is in $\#\mathbf{W}[\mathbf{1}]$ if there is a parameterized parsimonious reduction from F to p-#Clique. F is $\#\mathbf{W}[\mathbf{1}]$-hard, if there is a parameterized T-reduction from p-#Clique to F. As usual, F is $\#\mathbf{W}[\mathbf{1}]$-complete if it is in $\#\mathbf{W}[\mathbf{1}]$ and hard for it, too.

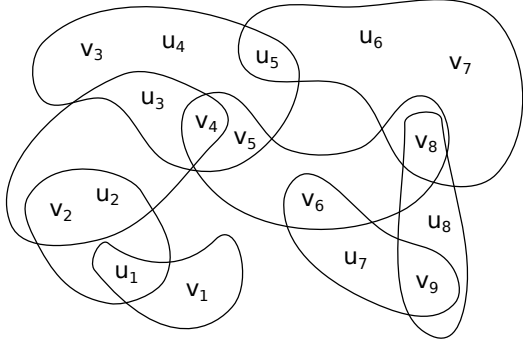Again, it is widely believed that there are problems in $\#\mathbf{W}[\mathbf{1}]$ (in particular the complete problems) that are not

Figure 1: The hypergraph associated to the formula ϕ of Example 2.2.
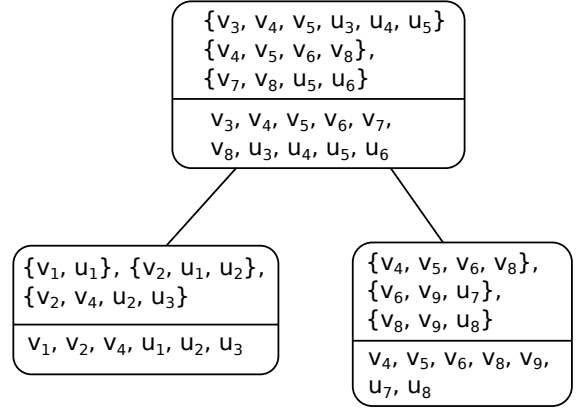

Figure 2: A generalized hypertree decomposition of width 3 for the hypergraph from Figure 1. The boxes are the guarded blocks. In the upper parts the guards are given while the lower parts show the blocks.

fixed parameter tractable. Thus, from showing that a problem F is $\#\mathbf{W}[\mathbf{1}]$-hard it follows that F can be assumed to be not fixed parameter tractable.

Hypergraph decompositions. In this section we present some well known hypergraph decompositions methods. For more details and more decomposition techniques see e.g. [8, 13, 23].

A (finite) hypergraph $\mathcal{H}$ is a pair (V, E) where V is a finite set and $E \subseteq \mathcal{P}(V)$. We associate a hypergraph $\mathcal{H} = (V, E)$ to a formula ϕ (the *canonical* structure describing ϕ) by setting $V := \mathsf{var}(\phi)$ and $E := \{\mathsf{var}(a) \mid a \in \mathsf{atom}(\phi)\}$.

EXAMPLE 2.2. *Consider the formula*

$$\phi \quad := \quad \exists u_1 \exists u_2 \exists u_3 \exists u_4 \exists u_5 \exists u_6 \exists u_7 \exists u_8$$

$$P_1(v_1, u_1) \wedge P_2(v_2, u_1, u_2) \wedge P_3(v_2, v_4, u_2, u_3)$$

$$\wedge P_4(v_3, v_4, v_5, u_3, u_4, u_5) \wedge P_5(v_4, v_5, v_6, v_8)$$

$$\wedge P_6(v_7, v_8, u_5, u_6) \wedge P_2(v_6, v_9, u_7) \wedge P_2(v_8, v_9, u_8)$$

The associated hypergraph is illustrated in Figure 1.

An independent set I in $\mathcal{H}$ is a set of vertices $I \subseteq V$ such that no two of them lie in one edge together. An edge cover C of $\mathcal{H}$ is an edge set $E' \subseteq E$ such that $\bigcup_{e \in E'} e = V$.

DEFINITION 2.3. *A generalized hypertree decomposition of a hypergraph $\mathcal{H} = (V, E)$ is a triple $(\mathcal{T}, (\lambda_t)_{t \in T}, (\chi_t)_{t \in T})$ where $\mathcal{T} = (T, F)$ is a rooted tree and $\lambda_t \subseteq E$ and $\chi_t \subseteq V$ for every $t \in T$ satisfying the following properties:*

1. *For every $v \in V$ the set $\{t \in T \mid v \in \chi_t\}$ induces a subtree of $\mathcal{T}$.*

2. *For every $e \in E$ there is a $t \in T$ such that $e \subseteq \chi_t$.*

3. *For every $t \in T$ we have $\chi_t \subseteq \bigcup_{e \in \lambda_t} e$.*

The first property is called the connectedness *condition. The sets χ_t are called* blocks *or* bags *of the decomposition, while the sets λ_t are called the* guards *of the decomposition. A pair (λ_t, χ_t) is called* guarded block.

The width *of a decomposition $(\mathcal{T}, (\lambda_t)_{t \in T}, (\chi_t)_{t \in T})$ is defined as $\max_{t \in T}(|\lambda_t|)$. The generalized hypertree width of $\mathcal{H}$ is the minimum width over all generalized hypertree decompositions of $\mathcal{H}$.*

We sometimes identify a guarded block (λ_t, χ_t) with the vertex t.

EXAMPLE 2.4. *Figure 2 shows a generalized hypertree decomposition of width 3 for the hypergraph from Figure 1.*

DEFINITION 2.5. *A hypergraph is acyclic if it has generalized hypertree width 1. In this case, the decomposition restricted to its blocks is called a join tree.*

Let us fix some notation: For an edge set $\lambda \subseteq E$ we use the shorthand $\bigcup \lambda := \bigcup_{e \in \lambda} e$. For a decomposition $(\mathcal{T}, (\lambda_t)_{t \in T}, (\chi_t)_{t \in T})$ we write $\mathcal{T}_t$ for the subtree of $\mathcal{T}$ that has t as its root. We also write $\chi(\mathcal{T}_t) := \bigcup_{t' \in V(\mathcal{T}_t)} \chi_{t'}$.

DEFINITION 2.6. *A generalized hypertree decomposition is called* hingetree decomposition[1] *if it satisfies the following conditions:*

4. *For each pair $t_1, t_2 \in T$ with $t_1 \neq t_2$ there are edges $e_1 \in \lambda_{t_1}$ and $e_2 \in \lambda_{t_2}$ such that $\chi_{t_1} \cap \chi_{t_2} \subseteq e_1 \cap e_2$.*

5. *For each $t \in T$ we have $\bigcup \lambda_t = \chi_t$.*

6. *For each $e \in E$ there is a $t \in T$ such that $e \in \lambda_t$.*

Hingetree width (also called degree of cyclicity) is defined analogously to generalized hypertree width.

EXAMPLE 2.7. *The decomposition from Figure 2 is also a hingetree decomposition.*

DEFINITION 2.8. *The* primal graph *of a hypergraph $\mathcal{H} = (V, E)$ is the graph $\mathcal{H}_P = (V, E_p)$ with $E_p := \{uv \in \binom{V}{2} \mid \exists e \in E : u, v \in E\}$.*

DEFINITION 2.9. *A tree decomposition of a hypergraph $\mathcal{H}$ is a generalized hypertree decomposition of its primal graph $\mathcal{H}_P$. The width of a tree decomposition is the size of its biggest bag minus 1. The treewidth of $\mathcal{H}$ is the minimum width over all tree decompositions of $\mathcal{H}$.*

---

[1]Note that this is not the original definition from [20] but an alternative, equivalent definition from [8].

For all decompositions defined above we define the width of a CQ-instance to be the width of the associated hypergraph.

We now recall some known results on the various decomposition methods.

LEMMA 2.10.   *a) (see e.g. [8]) For all of the width measures defined above Boolean CQ-instances of width k can be solved in time $\|\Phi\|^{p(k)}$ for a polynomial p.*

b) ([20]) There is an algorithm that given a hypergraph $\mathcal{H} = (V, E)$ computes a minimum width hingetree decomposition in time $|V|^{O(1)}$.

c) ([3])Computing minimum width tree decompositions is fixed parameter tractable parameterized by the treewidth.

d) ([1, 14]) There is an algorithm that given a hypergraph $\mathcal{H} = (V, E)$ of generalized hypertree width k constructs a generalized hypertree decomposition of width $O(k)$ of $\mathcal{H}$ in time $|V|^{O(k)}$.

DEFINITION 2.11. *Let $\mathcal{H} = (V, E)$ be a hypergraph and $V' \subseteq V$. The* induced subhypergraph $\mathcal{H}[V']$ *of $\mathcal{H}$ is the hypergraph $\mathcal{H}[V'] = (V', \{e \cap V' \mid e \in E, e \cap V' \neq \emptyset\})$.*

Let $x, y \in V$, a path *between x and y is a sequence of vertices $x = v_1, ..., v_k = y$ such that for each $i \in [k-1]$ there is an edge $e_i \in E$ with $v_i, v_{i+1} \in e_i$.*

A (connected) component *of $\mathcal{H}$ is the induced subhypergraph $\mathcal{H}[V']$ for a maximal vertex set V' such that for each pair $x, y \in V'$ there is a path between x and y in $\mathcal{H}$.*

OBSERVATION 2.12. *Let β be any decomposition technique defined in this section. Let $\mathcal{H} = (V, E)$ be a hypergraph of β-width k. Then for every $V' \subseteq V$ the induced subhypergraph $\mathcal{H}[V']$ has β-width at most k.*

PROOF. Let $(\mathcal{T}, (\lambda_t)_{t \in T}, (\chi_t)_{t_i n T})$ be a β-decomposition of $\mathcal{H}$ of width k. For each guarded block (λ_t, χ_t) compute a guarded block (λ'_t, χ'_t) with $\chi_t := \chi_t \cap V'$ and $\lambda_t := \{e \cap V' \mid e \in \lambda\}$. It is easy to check that $(\mathcal{T}, (\lambda'_t)_{t \in T}, (\chi'_t)_{t_i n T})$ is a β-decomposition of width at most k. □

3. QUANTIFIED-STAR SIZE

In this section we generalize quantified star size which was introduced in [10] for acyclic conjunctive queries to general conjunctive queries.

DEFINITION 3.1. *Let $\mathcal{H} = (V, E)$ be a hypergraph and $S \subseteq V$. Let C be the vertex set of a connected component of $\mathcal{H}[V-S]$. Let E_C be the set of hyperedges $\{e \in E \mid e \cap C \neq \emptyset\}$ and $V' := \bigcup_{e \in E_C} e$. Then $\mathcal{H}[V']$ is called an S-component of $\mathcal{H}$.*

DEFINITION 3.2. *Let $\mathcal{H} = (V, E)$ be a hypergraph. For a set $S \subseteq V$ the S-star size of $\mathcal{H}$ is the maximum size of an independent set consisting only of vertices in S in an S-component of $\mathcal{H}$. We say that this independent set forms the S-star.*

EXAMPLE 3.3. *Take $S = \{v_1, ..., v_9\}$ in the hypergraph of Figure 1. It has three S-components with respective edge lists:*

1. $\{v_1, u_1\}$, $\{v_2, u_1, u_2\}$, $\{v_2, v_4, u_2, u_3\}$, $\{v_7, v_8, u_5, u_6\}$, $\{v_4, v_5, v_6, v_8\}$, $\{v_3, v_4, v_5, u_3, u_4, u_5\}$, $\{v_8\}$

2. $\{v_8, v_9, u_8\}$, $\{v_8\}$

3. $\{v_6, v_9, u_7\}$, $\{v_6\}$

The S-star size i.e. the size of a maximum independent of S-vertices in a S-component is 4. The set $\{v_1, v_2, v_3, v_7\}$ forms an S-star (there are several other possibilities).

It is easy to see that for acyclic hypergraphs this definition of S-star size coincides with the definition in [10] which was only defined for acyclic hypergraphs.

DEFINITION 3.4. *An S-hypergraph is a pair $(\mathcal{H}, S)$ where $\mathcal{H} = (V, E)$ is a hypergraph and $S \subseteq V$. To each formula ϕ we associate an S-hypergraph $(\mathcal{H}, S)$ where $\mathcal{H}$ is the hypergraph associated to ϕ and $S := \mathsf{free}(\phi)$. The quantified star size of a CQ instance $\Phi = (\mathcal{A}, \phi)$ is the S-star size of $(\mathcal{H}, S)$.*

Let $\mathscr{G}_{star}$ be the class of S-hypergraphs $(\mathcal{H}_n, S_n)$, $n \in \mathbb{N}$, where $\mathcal{H}_n$ is a star graph and S_n consists of its leaves. More precisely, $H_n = (V_n, E_n), S_n$ are defined as

- $V_n = \{z, y_1, ..., y_n\}$,

- $E_n = \{\{z, y_i\} \mid i = 1, ..., n\}$,

- $S_n = \{y_1, ..., y_n\}$.

We will use the following lemma from [10].

LEMMA 3.5   ([10]). *#CQ is #$\mathbf{W}[\mathbf{1}]$-hard restricted to instances that have S-hypergraphs in $\mathscr{G}_{star}$ parameterized by the size of the stars.*

4. THE COMPLEXITY OF COUNTING

In this section we show that the decomposition techniques introduced in Section 2 lead to efficient counting when combined with bounded quantified star size. Furthermore, we show that bounded quantified star size is necessary for efficient counting under standard assumptions.

THEOREM 4.1. *There is an algorithm that given a #CQ-instance $\Phi = (\mathcal{A}, \phi)$ of quantified starsize ℓ and a generalized hypertree decomposition $\Xi = (\mathcal{T}, (\lambda_t)_{t \in T}, (\chi_t)_{t \in T})$ of Φ of width k counts the solutions of ϕ in time $\|\Phi\|^{p(k, \ell)}$ for a fixed polynomial p.*

In the proof we will use the following lemma from [10].

LEMMA 4.2. *For acyclic hypergraphs the size of a maximum independent set and a minimum edge cover coincide. Moreover, there is a polynomial time algorithm that given an acyclic hypergraph $\mathcal{H}$ computes a maximum independent set I and a minimum edge cover E^* of $\mathcal{H}$.*

PROOF OF THEOREM 4.1. Given $\Phi = (\mathcal{A}, \phi)$, we construct a solution equivalent instance Φ'' in two steps which is of generalized hypertree width k, too, and has a quantifier free formula.

Let $\mathcal{H} = (V, E)$ be the hypergraph of ϕ. Let $V_1, \ldots, V_m$ be the vertex sets of the components of $\mathcal{H}[V - S]$ and let $V'_1, \ldots, V'_m$ be the vertex sets of the S-components of $\mathcal{H}$. Clearly, $V_i \subseteq V'_i$ and $V'_i - V_i = V'_i \cap S =: S_i$. Let Φ_i be the #CQ-instance whose formula ϕ_i is obtained by restricting all atoms of ϕ to the variables in V'_i and whose structure $\mathcal{A}_i$ is obtained by projecting all relations of $\mathcal{A}$ accordingly. The associated hypergraph of ϕ_i is $\mathcal{H}[V'_i]$ and $\mathcal{H}[V'_i]$ has a generalized hypertree decomposition Ξ_i of width at most k with tree a $\mathcal{T}_i$ that is a subtree of $\mathcal{T}$ (see Observation 2.12).

For each Φ_i we construct a new #CQ-instance $\Phi_i' = (\mathcal{A}_i', \phi_i')$ as follows. For each guarded block $b = (\lambda, \chi) \in \Xi_i$ we construct a new atomic formula φ_b in the variables χ. The associated relation is given by $\pi_\chi\big(\bowtie_{\phi \in \text{atom}(\Phi_i): \text{var}(\phi) \subseteq \bigcup \lambda} \phi\big)$ i.e. by taking the natural join of all relations whose variables are guarded in the guarded block and projecting on χ. The formula ϕ_i' for Φ_i' is obtained as the conjunction of all φ_b. The decomposition Ξ_i has width at most k so this can be done in time $\|\Phi\|^{O(k)}$. Obviously, Φ_i and Φ_i' are solution equivalent. Furthermore ϕ_i' is acyclic, because it has a decomposition with tree $\mathcal{T}_i$, the same blocks as Ξ_i and width 1. Let $\mathcal{H}_i$ be the associated hypergraph of ϕ_i', then $\mathcal{H}_i$ has only one single S_i-component, because all the vertices in V_i are connected in $\mathcal{H}$ and thus also in $\mathcal{H}_i$. Also the S_i-star size of $\mathcal{H}_i$ is at most ℓ. To see this consider two independent vertices u, v in $\mathcal{H}_i$. The edges of $\mathcal{H}_i$ are equal to the blocks of Ξ_i, so u and v do not appear in a common block in Ξ_i. But then u and v cannot appear in one common block in Ξ, because of $\mathcal{T}$ being a tree and the connectedness condition. So u and v are independent in $\mathcal{H}$, too, and thus every independent set in $\mathcal{H}_i$ is also independent in $\mathcal{H}$. So $\mathcal{H}_i$ indeed has S_i-star size at most ℓ. Thus the vertices in S_i can be covered by at most ℓ edges $e_1, \ldots, e_\ell$ in $\mathcal{H}_i$ which we can compute in polynomial time by Lemma 4.2. Let $\alpha_1, \ldots, \alpha_\ell$ be the corresponding atoms. We again construct a new atomic formula ϕ_i'' in the variables S_i only and an associated relation $\mathcal{A}_i''$ as follows: For each combination $t_1, \ldots, t_\ell$ of tuples in $\alpha_1(\mathcal{A}_i'), \ldots, \alpha_\ell(\mathcal{A}_i')$ fix the free variables in ϕ_i' to the constants prescribed by the tuples $t_1, \ldots, t_\ell$ if these do not contradict. If the resulting CQ instance has a solution, add the projection of $t_1 \bowtie \ldots \bowtie t_\ell$ on S_i to the relation $\mathcal{A}_i''$ of ϕ_i''. By construction Φ_i' and $(\mathcal{A}_i'', \varphi_i'')$ are solution equivalent. Observe that the instances to be solved in this construction are tractable [25], so all of this can be done in time $\|\Phi_i\|^{p'(k,\ell)}$ for a polynomial p'.

We now eliminate all quantified variables in the original formula ϕ. To do so we add the atom ϕ_i'' for $i \in [m]$ and delete all atoms that contain any quantified variable, i.e. we delete each ϕ_i'. Add the $\mathcal{A}_i''$ to the structure $\mathcal{A}$ and call the resulting #CQ instance $\Phi'' = (\mathcal{A}'', \phi'')$. Because $(\mathcal{A}_i'', \phi_i'')$ is solution equivalent to Φ_i', we have that Φ and Φ'' are solution equivalent, too. We construct a guarded decomposition of ϕ'' by doing the following: For each guarded block (λ, χ) of Ξ with $\chi \cap V_i \neq \emptyset$ we construct a guarded block (λ', χ') by deleting all edges e with $e \cap V_i \neq \emptyset$ from λ and adding the edge S_i for ϕ_i''. Furthermore we set $\chi' = (\chi - V_i) \cup S_i$. It is easy to see that the result is indeed a generalized hypertree decomposition of ϕ'' of width at most k.

With standard techniques (see e.g. [8]) we construct in polynomial time a quantifier free acyclic #CQ-instance that is solution equivalent to Φ''. Its solutions and thus those of Φ can then be counted with the algorithm in [24] or [10]. $\square$

We now show that bounded quantified star size is necessary for efficient counting no matter which other structural restrictions we put on S-hypergraphs.

LEMMA 4.3. *Let $\mathcal{G}$ be a recursively enumerable class of S-hypergraphs such that #CQ for all instances whose S-hypergraph is in $\mathcal{G}$ is fixed parameter tractable parameterized by the size of the formulas. Then $\mathcal{G}$ has bounded S-star size or #$\mathbf{W}[\mathbf{1}] = \mathbf{FPT}$.*

PROOF SKETCH. The proof is a generalization of the respective proof in [10]: We show that if the S-star size of $\mathcal{G}$

is not bounded, then there is an FPT algorithm for #CQ on $\mathcal{G}_{star}$, the class of stars with a single quantified variable in the center. As this problem is #$\mathbf{W}[\mathbf{1}]$-hard by Lemma 3.5, it follows that #$\mathbf{W}[\mathbf{1}] = \mathbf{FPT}$.

So assume that #CQ is tractable on $\mathcal{G}$ and $\mathcal{G}$ has unbounded S-star size. We will construct a fixed parameter algorithm for #CQ on $\mathcal{G}_{star}$. So let $\Phi = (\mathcal{A}, \varphi)$ be an instance of this latter problem, i.e. Φ has the formula $\varphi := \exists z \bigwedge_{i=1}^k E_i(y_i, z)$. Let the domain of $\mathcal{A}$ be D. Because $\mathcal{G}$ is recursively enumerable and of unbounded S-star size, there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that for $k \in \mathbb{N}$ one can compute $(\mathcal{H}, S) \in \mathcal{G}$ with S-star size at least k in time $g(k)$. We will embed Φ into $\mathcal{H}$ to construct an #CQ-instance $\Phi' = (\mathcal{A}', \psi)$ of size $g(k)n^{O(1)}$ where n is the size of Φ. Furthermore, ψ will have the S-hypergraph $\mathcal{H}$ and $\mathcal{A}'$ the same domain D as $\mathcal{A}$. For convenience, Φ' will be built on a language containing one distinct relation symbol for each hyperedges in $\mathcal{H}$.

Let $\mathcal{H}'$ be the S-component of $\mathcal{H}$ that contains k independent vertices in the respective primal component. Call these vertices $s_1, \ldots s_k$. We may assume that the s_i are also computed in time $g(k)$ during the construction of $\mathcal{H}$. Observe that there must be a vertex v that is connected to each of the s_i by a path P_i such that the only vertex in P_i that is in S is s_i, because all the s_i lie in the same S-component. We now construct a #CQ instance Φ' that has the associated S-hypergraph $\mathcal{H}$.

All vertices that do not lie on any P_i are forced to a dummy value d in a straightforward way by all their constraints. All vertices on the P_i that are no s_j may take arbitrary but equal values in D. This is possible, because they are all connected to the common vertex v by paths. Let v_i be the predecessor of s_i on P_i. For all constraints that contain v_i and s_i we allow for them exactly the combinations allowed by the relation of $E_i^{\mathcal{A}}$. Observe that there is no edge that contains more than one of the s_i by definition, so each constraint has at most $|D|^2$ tuples.

Clearly, Φ and Φ' have the same number of solutions. Furthermore, we have $|\psi| \leq g(k)$ and Φ' can be constructed in time at most $g(k)\|\Phi\|^2$, because $\mathcal{H}$ has size at most $g(k)$ and the size of the relations for the constraints is bounded by $|D|^2$. But by assumption the solutions of Φ' can be counted in time $h(|\psi|)\|\Phi'\|^c$ for a constant c and a computable function h. Thus the solutions of Φ can be counted in time $h(g(k))\|\Phi\|^{c'}$ for a constant c' which completes the proof. $\square$

With Theorem 4.1 and Lemma 4.3 we have a solid understanding of the complexity of #CQ for structural classes that can be characterized by restrictions of generalized hypertree width. For each decomposition method with what Cohen et al. [8] call the "tractable construction" property, i.e. there must be a way to construct a decomposition efficiently, quantified star size is essentially the only parameterization that makes counting tractable. For the definitions of decomposition techniques not defined in this paper see [13].

COROLLARY 4.4. *Let β be one of the following decomposition techniques: biconnected component, cycle-cutset, cyclehypercutset, hinge-tree, hypertree, or generalized hypertree decomposition. Let furthermore $\mathcal{G}$ be a recursively enumerable class of S-hypergraphs of bounded β-width. Then counting solutions to all #CQ-instances whose associated hypergraph is in $\mathcal{G}$ is tractable if and only if $\mathcal{C}$ is of bounded S-star size (assuming $\mathbf{FPT} \neq \text{#}\mathbf{W}[\mathbf{1}]$).*

5. AN OPTIMAL RESULT FOR BOUNDED ARITY

In this section we show that for bounded arity #CQ we can exactly characterize which classes of instances allow polynomial time counting. This result is derived by combining the results of the preceding sections and the following theorem from [19] that we rephrase in our slighlty different wording.

THEOREM 5.1 ([19]). *Let $\mathscr{G}$ be a recursively enumerable class of hypergraphs of bounded arity. Assume* $\mathbf{FPT} \neq \mathbf{W}[\mathbf{1}]$. *Then the following three statements are equivalent:*

- *Boolean CQ for all instances with hypergraphs in $\mathscr{G}$ can be decided in polynomial time.*

- *Boolean CQ for all instances with hypergraphs in $\mathscr{G}$ is fixed parameter tractable parameterized by the size of the formulas.*

- *The hypergraphs in $\mathscr{G}$ are of bounded treewidth.*

Theorem 5.1 is originally stated to be true even for every fixed vocabulary. It has been generalized to any recursively enumerable class of conjunctive formulas [17]. In this context, a characterization of tractability for counting solutions of *quantifier-free* conjunctive queries is given in [9] in almost the same terms as Theorem 5.1 but with the weaker assumption that $\mathbf{FPT} \neq \#\mathbf{W}[\mathbf{1}]$. We show here a complete characterization of tractability for counting for general conjunctive queries. Not too surprisingly, tractability depends on both treewidth and star size of the underlying hypergraph.

THEOREM 5.2. *Let $\mathscr{G}$ be a recursively enumerable class of S-hypergraphs of bounded arity. Assume that $\mathbf{W}[\mathbf{1}] \neq \mathbf{FPT}$. Then the following statements are equivalent:*

1. *#CQ for all instances whose S-hypergraph is in $\mathscr{G}$ is solvable in polynomial time.*

2. *#CQ for all instances whose S-hypergraph is in $\mathscr{G}$ is fixed parameter tractable parameterized by the size of the formulas.*

3. *There is a constant c such that for each S-hypergraph $(\mathcal{H}, S)$ in $\mathscr{G}$ the treewidth of $\mathcal{H}$ and the S-star size are at most c.*

PROOF. The direction $1 \to 2$ is trivial. Furthermore, $3 \to 1$ follows directly from Theorem 4.1. So it remains only to show $2 \to 3$.

So assume that there is a recursively enumerable class $\mathscr{G}$ of S-hypergraphs such that counting solutions to #CQ-instances whose S-hypergraph are in $\mathscr{G}$ is fixed parameter tractable but 3 is not satisfied by $\mathscr{G}$. From Lemma 4.3 we know that the S-starsize of $\mathscr{G}$ must be bounded, so it follows that the treewidth of $\mathscr{G}$ must be unbounded.

We construct a class $\mathscr{G}'$ of hypergraphs by doing the following: For each S-hypergraph $(\mathcal{H}, S)$ in $\mathscr{G}$ we add $\mathcal{H}$ to $\mathscr{G}'$. Clearly $\mathscr{G}'$ is recursively enumerable and of unbounded treewidth. We will show that Boolean CQ for all instances whose hypergraphs are in $\mathscr{G}'$ is fixed parameter tractable parameterized by the size of the formula. This leads to a contradiction with Theorem 5.1.

Because $\mathscr{G}$ is recursively enumerable, there is an algorithm that that for each $\mathcal{H}$ in $\mathscr{G}'$ constructs an S-hypergraph $(\mathcal{H}, S)$ in $\mathscr{G}$ that has lead to the addition of $\mathcal{H}$ to $\mathscr{G}'$. For example one

can simply enumerate all S-hypergraphs in $\mathscr{G}$ until finding such a $(\mathcal{H}, S)$. Let $f(\mathcal{H})$ be the number of steps the algorithm needs on input $\mathcal{H}$. The function $f(\mathcal{H})$ is well defined and computable. We then define $g : \mathbb{N} \to \mathbb{N}$ by setting $g(k) := \max_{\mathcal{H}}(f(\mathcal{H}))$, where the maximum is over all hypergraphs $\mathcal{H}$ of size k in $\mathscr{G}'$. Clearly, g is again well defined and computable. Thus for each $\mathcal{H}$ in $\mathscr{G}'$ we can compute in time $g(|\mathcal{H}|)$ an S-hypergraph $(\mathcal{H}, S)$ in $\mathscr{G}$.

Now let $\Phi = (\mathcal{A}, \phi)$ be a CQ-instance with hypergraph $\mathcal{H}$ in $\mathscr{G}'$. To solve it we first compute $(\mathcal{H}, S)$ as above and construct a #CQ-instance $\Psi = (\mathcal{A}, \psi)$ with $(\mathcal{H}, S)$ as associated S-hypergraph for ψ by adding existential quantifiers for all variables not in S. Obviously Φ has solutions if and only if Ψ has any. But by assumption the solutions of Ψ can be counted in time $h(|\psi|)\|\Psi\|^{O(1)}$ for some computable function h, so Φ can be decided in time $(g(|\phi|) + h(|\phi|))\|\Phi\|^{O(1)}$ and thus is fixed parameter tractable. This is the desired contradiction to Theorem 5.1. $\square$

REMARK 5.3. *Note that our characterization relies on the underlying hypergraph structures of the query. In [17, 9], the corresponding characterizations are stronger in the sense that they are true for any recursively enumerable class of conjunctive formulas. Also these results and the one from [19] can be proved for every fixed vocabulary, while our proofs of the Lemmas 3.5 and 4.3 and thus also Theorem 5.2 rely on the fact that we can choose our vocabulary in the construction. It remains an open question whether our result can be improved similarly to the others.*

Also, the result in [9] (for quantifier free #CQ) is proved under the weaker assumption $\#\mathbf{W}[\mathbf{1}] \neq \mathbf{FPT}$. Showing the same equivalent result for general #CQ seems to be hard since our case also contains decision problems (e.g. #CQ with no free variables).

6. COMPUTING STAR SIZE

In this section we consider the problem of computing the quantified star size of bounded width instances. Observe that the computation of quantified star size is not strictly necessary. The algorithm of Theorem 4.1 does not need to find S-stars for graphs of width k but only for acyclic hypergraphs, which is easy by Lemma 4.2. Still it is of course desirable to know the quantified star size of an instance before applying the counting algorithm, because quantified star size has an exponential influence on the runtime. We show that for all decomposition techniques considered in this paper the quantified star size can be computed rather efficiently, roughly in $|V|^{O(k)}$ where k is the width of the input. For small values of k, this bound is reasonable. We then proceed by showing that, on the one hand, for some decomposition measures such as treewidth or hingetree, the computation of quantified star size is even fixed parameter tractable parameterized by the width. On the other hand, we show that for decomposition measures above hypertree width it is unlikely that fixed parameter tractability can be obtained (under standard assumptions).

Instead of tackling quantified star size directly, we consider the combinatorially less complicated notion of independent sets, which is justified by the following observation:

OBSERVATION 6.1. *Let β be any decomposition technique considered in this paper. Then for every $k \in \mathbb{N}$ computing the S-starsize of S-hypergraphs of β-width at most k polynomial*

time Turing-reduces to computing the size of a maximum independent set for hypergraphs of β-width at most k. Furthermore, there is a polynomial time many one reduction from computing the size of a maximum independent set in hypergraphs of β-width at most k to computing the S-star size of hypergraphs of β-width at most $k+1$.

PROOF. By definition computing S-starsize reduces to the computation of independent sets of S-components. S-components are induced subhypergraphs, so we get the first direction form Observation 2.12.

For the other direction let $\mathcal{H} = (V, E)$ be a hypergraph for which we want to compute the size of a maximum independent set. Let $x \notin V$. We construct the hypergraph $\mathcal{H}'$ of vertex set $V' = V \cup \{x\}$ and edge set $E' = \{e \cup \{x\} \mid e \in E\}$ and set $S := V$. The hypergraph is one single S-component, because x is in every edge. Furthermore, the S-starsize of $\mathcal{H}'$ is obviously the size of a maximum independent set in $\mathcal{H}$. It is easy to see that the construction increases the treewidth of the hypergraph by at most 1 and does not increase the β-width for all other decomposition considered here at all. $\square$

Because of Observation 6.1 we will not talk about S-star size in this section anymore but instead formulate everything with independent sets.

6.1 Exact computation

PROPOSITION 6.2. *There is an algorithm that given a hypergraph $\mathcal{H} = (V, E)$ and a generalized hypertree decomposition $\Xi = (\mathcal{T}, (\lambda_t)_{t \in T}, (\chi_t)_{t \in T})$ of $\mathcal{H}$ of width k computes a maximum independent set of $\mathcal{H}$ in time $k|V|^{O(k)}$.*

PROOF. We apply dynamic programming along the decomposition. Let $b = (\lambda, \chi)$ be a guarded block of $\mathcal{T}$. Let $\mathcal{T}_b$ be the subtree of $\mathcal{T}$ with b as its root. We set $V_b := \chi(\mathcal{T}_b)$. Observe that $I \subseteq V_b$ is independent in $\mathcal{H}$ if and only if it is independent in $\mathcal{H}[V_b]$ so we do not differentiate between the two notions. For each independent set $\sigma \subseteq \chi$ we will compute an independent set $I_{b,\sigma} \subseteq V_b$ that is maximum under the independent sets containing exactly the vertices σ from χ. Observe that because λ contains at most k edges that cover χ we have to compute at most kn^k independent sets $I_{b,\sigma}$ for each b.

If b is a leaf of $\mathcal{T}$, the construction of the $I_{b,\sigma}$ is straightforward and can certainly be done in time $k|V|^{O(k)}$.

Let now $b = (\lambda, \chi)$ be an inner vertex of $\mathcal{T}$ with children $b_1, \ldots, b_r$. For each independent set $\sigma \subseteq \chi$ we do the following: Let $b_i = (\lambda_i, \chi_i)$, then let σ_i be an independent set of χ_i such that $\sigma \cap \chi \cap \chi_i = \sigma_i \cap \chi \cap \chi_i$ and $|I_{b_i, \sigma_i}|$ is maximal. We claim that we can set $I_{b,\sigma} := \sigma \cup I_{b_1, \sigma_1} \cup \ldots \cup I_{b_r, \sigma_r}$.

We first show that $I_{b,\sigma}$ defined this way is independent. Assume this is not true, then $I_{b,\sigma}$ contains x, y that are in one common edge e in $\mathcal{H}[V_b]$. But then x, y do not lie both in χ, because $I_{b,\sigma} \cap \chi = \sigma$ and σ is independent. By induction x, y do not lie in one V_{b_i} either. Assume that $x \in \chi$ and $y \in V_{b_i}$ for some i. Then certainly $x \notin V_{b_i}$ and $y \notin \chi$. But the edge e must lie in one guard λ' such that the corresponding block χ' contains e. Because of the connectivity condition for y the guarded block (λ', χ') must lie in the subtree with root b_i, which contradicts $x \in e$. Finally, assume that $x \in V_{b_i}$ and $y \in V_{b_j}$ for $i \neq j$ and $x, y \notin \chi$. Then x and y cannot be adjacent because of the connectivity condition. This shows that $I_{b,\sigma}$ is indeed independent.

Now assume that $I_{b,\sigma}$ is not of maximum size and let $J \subseteq V_b$ be an independent set with $|J| > |I_{b,\sigma}|$ and $J \cap \chi = \sigma$. Because J and $I_{b,\sigma}$ are fixed to σ on χ there must be a b_i such that $|J \cap V_{b_i}| > |I_{b_i, \sigma_i}|$. This contradicts the choice of σ_i. So $I_{b,\sigma}$ is indeed of maximum size.

Because each block has at most $k|V|^k$ independent sets, all computations can be done in time $k|V|^{O(k)}$. $\square$

6.2 Parameterized complexity

While the algorithm in the last section is nice in that it is a polynomial time algorithm for fixed k, it is somewhat unsatisfying for some decomposition techniques: If we can compute the composition quickly, we would ideally want to be able to compute the star size efficiently, too. Naturally we cannot expect a polynomial time algorithm independent of k, because independent set is **NP**-complete, but we can hope for at least fixed parameter tractability with respect to k. We will show that this is indeed possible for some width measures, in particular tree decompositions and hingetree decompositions. On the other hand we show that this can likely not be extended to more general decomposition techniques, because independent set parameterized by hypertree width is **W[1]**-hard.

PROPOSITION 6.3. *Given a hypergraph $\mathcal{H}$ computing a maximum independent set in $\mathcal{H}$ is fixed parameter tractable parameterized by the treewidth of $\mathcal{H}$.*

This can be seen either by applying Courcelle's Theorem of by straightforward dynamic programming. Interestingly, one can show the same result also for bounded hingetree width, which is a decomposition technique in which the bags are of unbounded size. This unbounded size makes the dynamic programming in the proof far more involved than for treewidth.

PROPOSITION 6.4. *Given a hypergraph $\mathcal{H}$ computing a maximum independent set in $\mathcal{H}$ is fixed parameter tractable parameterized by the hingetree width of $\mathcal{H}$.*

PROOF. First observe that minimum width hingetree decompositions can be computed in polynomial time [20], so we simply assume that a decomposition is given in the rest of the proof.

The proof has some similarity with that of Proposition 6.2, so we use some notation from there. For guarded block (λ, χ) we will again compute maximum independent sets containing prescribed vertices. The difference is, that we can take these prescribed sets to be of size 1: because of the hingetree condition, only one vertex of a block may be reused in any independent set in the parent. The second idea is that we can use equivalence classes of vertices in the computations of independent sets in the considered guarded blocks, which limits the number of independent sets we have to consider. We now describe the computation in detail.

Let $\Xi = (\mathcal{T}, (\lambda_t)_{t \in T}, (\chi_t)_{t \in T})$ be a hingetree decomposition of $\mathcal{H}$ of width k. Let $b = (\lambda, \chi)$ be a guarded block of Ξ and let $b' = (\lambda', \chi')$ be its parent. As before, let $\mathcal{T}_b$ be the subtree of $\mathcal{T}$ with b as its root and $V_b := \chi(\mathcal{T}_b)$. Set $\mathcal{H}_b := (V_b, E_b)$ with $E_b := \bigcup \lambda^*$ with the union being over all guarded blocks in $\mathcal{T}_b$. The main idea is to iteratively compute, for all vertices $v \in \chi' \cap \chi$, a maximum independent set $J_{v,b}$ in $\mathcal{H}_b = (V_b, E_b)$ containing v. Furthermore, we also compute an independent set $J_{\emptyset, b}$ that contains no vertices

of $\chi' \cap \chi$. Note that, since $\chi \subseteq \bigcup_{e \in \lambda} e$, there are no isolated vertices in χ and the size of a maximum independent set is bounded by k in each block.

For a node $b = (\lambda, \chi)$, we organize the vertices in χ into at most 2^k equivalence classes by defining v and u to be equivalent if they lie in the same subset of edges of λ. The equivalence class of v is denoted by $\mathbf{c}(v)$. For each class, a representant is fixed. We denote by $\bar{v}$, the representant of the equivalence class of v and by $\bar{\chi} \subseteq \chi$, the restriction of χ on these at most 2^k representants.

Let first b be a leaf. We first compute independent sets on $\bar{\chi}$. Observe that the independent sets are invariant under the choice of representants. For each equivalence class $\mathbf{c}(v)$, we compute $J_{\bar{v},b} \subseteq \bar{\chi}$ as a maximum independent set containing $\bar{v}$. Computing the classes and a choice of maximum independent sets containing each $\bar{v}$ can be done in time $k2^{k^2}$ because independent sets cannot be bigger than k. Clearly, $J_{v,b}$, a maximum independent set containing v, can be easily computed from the set $J_{\bar{v},b}$. Thus, one can compute all the $J_{v,b}$ in time $k2^{k^2}n$. The computation of $J_{\emptyset,b}$ can be done on representants, too, by simply excluding the vertices from $\chi' \cap \chi$.

Let b now be an inner vertex and $b_1, b_2, ..., b_m$ be its children with $b_i = (\lambda_i, \chi_i)$, $i \in [m]$. We again consider equivalence classes on χ. Fix $v \in \chi$ and compute the list $L_{\bar{v},b}$ of all independent sets $\sigma \subseteq \bar{\chi}$ containing $\bar{v}$. Fix now $\sigma \in L_{\bar{v},b}$. We first compute a set $J_{v,b}^\sigma$ as a maximum independent set of $\mathcal{H}_b$ containing v and whose vertices in χ have the representants σ. We will distinguish for a given vertex $\bar{u} \in \sigma$ if it is the representant of a vertex belonging to the block of some (or several) children of b or if it represents vertices of $\chi \setminus (\bigcup_{i=1}^m \chi_i)$ only. Therefore we partition σ into σ', σ'' accordingly:

- $\sigma := \sigma' \cup \sigma''$
- $\sigma' := \bar{\chi} \cap \{\bar{u} \mid u \in \bigcup_{i=1}^m \chi_i\}$.
- $\sigma'' := \bar{\chi} \setminus \{\bar{u} \mid u \in \bigcup_{i=1}^m \chi_i\}$

Set $\sigma' := \{\bar{u}_1, ..., \bar{u}_h\}$ with $h \leq m$. Let us examine the consequences of $\mathcal{T}$ being a hingetree decomposition. We have that, for all $i \in [m]$, there exists $e_i \in \lambda$, such that $\chi \cap \chi_i \subseteq e_i$. Thus, since σ is an independent set in $\bar{\chi} \subseteq \chi$, at most one vertex in σ' is a representant of a vertex in χ_i. Thus

$$\forall u \neq u' \in \sigma : \chi_i \cap \mathbf{c}(u) = \emptyset \vee \chi_i \cap \mathbf{c}(u') = \emptyset. \quad (1)$$

We denote by $S_i = \{j \mid \mathbf{c}(u_i) \cap \chi_j \neq \emptyset\}$ and by $S = [m] \setminus \bigcup S_i$. By (1) the sets $S_1, ..., S_h, S$ form a partition of $[m]$. To construct $J_{v,b}^\sigma$, we now determine for each $i \leq h$, which vertex u of $\mathbf{c}(u_i)$ can contribute the most, by taking the union of all the maximum independent sets J_{u,b_j}, $j \in S_i$, it induces at the level of the children of b.

For each fixed $u \in \mathbf{c}(u_i)$, let

$$I_{i,u} = \{u\} \cup \bigcup_{j \in S_i} J_{u,b_j},$$

where we set $J_{u,b_j} := J_{\emptyset,b_j}$ if $u \notin \chi_j$. Let then $I_i = I_{i,u}$ for some $u \in \mathbf{c}(u_i)$ for which the size of $I_{i,u}$ is maximal.

The set $J_{v,b}^\sigma$ is now obtained as follows depending on whether $\bar{v} \in \sigma''$ or $\bar{v} \in \sigma'$. If $\bar{v} \in \sigma''$, we claim that $J_{v,b}^\sigma$ can be chosen as

$$J_{v,b}^\sigma := \{v\} \cup (\sigma'' \setminus \{\bar{v}\}) \cup \bigcup_{i=1}^h I_i \cup \bigcup_{i \in S} J_{\emptyset,b_i}.$$

If $\bar{v} \in \sigma'$, say $\bar{v} = u_1$, we claim that $J_{v,b}^\sigma$ can be chosen as

$$J_{v,b}^\sigma := \sigma'' \cup \bigcup_{j \in S_1 : v \in \chi_j} J_{v,b_j} \cup \bigcup_{j \in S_1 : v \notin \chi_j} J_{\emptyset,b_j} \cup \bigcup_{i=2}^h I_i \cup \bigcup_{i \in S} J_{\emptyset,b_i}.$$

The set $J_{v,b}$ is taken as one of the sets $J_{v,b}^\sigma$ of maximal size for a $\sigma \in L_{v,b}$. To compute $J_{\emptyset,b}$, the arguments are similar.

We first show that all $J_{v,b}$ are indeed independent sets in $\mathcal{H}_b$. Clearly, it is enough to prove this for any $J_{v,b}^\sigma$. There will be no reason to distinguish whether $\bar{v} \in \sigma''$ or $\bar{v} \in \sigma'$, because our arguments will apply to all $J_{v,b}^\sigma$ independent of the choice of a distinguished element v. We will make extensive use of the two following facts.

- Let $j, j' \in [m]$ and $I \subseteq V_{b_j}, I' \subseteq V_{b_{j'}}$ independent sets of $\mathcal{H}_{b_j}$ and $\mathcal{H}_{b_{j'}}$ respectively. By the connectivity condition for tree decomposition we have

  $$I \cap I' \subseteq \chi_j \cap \chi_{j'} \cap \chi.$$

  This permits to investigate the intersection of two independent sets I, I' by looking at their restriction on χ.

- Let now $I \subseteq V_{b_j}$ be an independent set of $\mathcal{H}_{b_j}$. Then, I remains an independent set in $\mathcal{H}_b$. Indeed, suppose there is a $e \in E_b \setminus E_{b_j}$ containing two vertices $y_1, y_2 \in I$. Since all edges must belong to a guard, there exists a node $b^* = (\lambda^*, \chi^*)$ such that $e \in \lambda^*$. Then, since in a hingetree decomposition we have $\chi^* = \bigcup \lambda^*$, then $\{y_1, y_2\} \subseteq e \subseteq \chi^*$. But then, by the connectivity condition it follows that $\{y_1, y_2\} \subseteq \chi$. Hence, by the intersection property of hingetree decomposition, there exists $e_j \in \chi_j$ such that

  $$\{y_1, y_2\} \subseteq \chi \cap \chi_j \cap e_j$$

  which implies that y_1 and y_2 are adjacent in $\mathcal{H}_{b_j}$. Contradiction.

We now start the proof that $J_{v,b}^\sigma$ is independent incrementally. Let $i \in [h]$, $u \in \mathbf{c}(u_i)$ and $j \in S_i$ and consider the set $I := J_{u,b_j}$. By induction, the set I is independent in $\mathcal{H}_{b_j}$. By the hingetree condition, there exists $e_j \in \lambda_j$ such that $\chi \cap \chi_j \subseteq e_j$. By the connectivity condition, this implies $\chi \cap I \subseteq e_j$. Then, since I is an independent set, no two vertices of χ can belong to I i.e. $|\chi \cap I| \leq 1$. The connectivity condition also implies that, for $j' \neq j$, $V_{b_{j'}} \cap I \subseteq \chi \cap \chi_j$, hence $|V_{b_{j'}} \cap I| \leq 1$ and I is an independent set of $\mathcal{H}_b$. Finally, the set $I_i = \bigcup_{j \in S_i} J_{u,b_j}$ is also an independent set of $\mathcal{H}_b$, since for any distinct $j, j' \in S_i$:

$$J_{u,b_j} \cap J_{u,b_{j'}} \subseteq \chi_j \cap \chi_{j'} \cap \chi \subseteq e_j.$$

Hence $J_{u,b_j} \cap J_{u,b_{j'}}$ contains at most one vertex (which is in χ and could then only be u).

Let now $i, i' \in [m]$ be distinct. By the arguments above, I_i (resp. $I_{i'}$) contains at most one element u (resp. u') such that $u \in \mathbf{c}(u_i)$ (resp. $u' \in \mathbf{c}(u_{i'})$). By Equation 1, we have that the two classes are distinct and that $u_i \neq u_{i'}$. But $u_i, u_{i'} \in \sigma$ and σ is independent in χ. Hence, $u_i, u_{i'}$ cannot be adjacent in $\mathcal{H}_b$. Consequently,

$$\bigcup_{i=1}^h I_i$$

is an independent set in $\mathcal{H}_b$.

Let $j \in S$. $J_{\emptyset,b_j}$ is independent in $\mathcal{H}_{b_j}$ and $J_{\emptyset,b_j} \subseteq V_{b_j} \backslash \chi$. Hence, $J_{\emptyset,b_j}$ is independent in $\mathcal{H}_b$. This also implies that, given $j' \in [m]$ distinct from j, $J_{\emptyset,b_j} \cap V_{b_{j'}} = \emptyset$. Thus,

$$\bigcup_{i=1}^{h} I_i \cup \bigcup_{i \in S} J_{\emptyset,b_i}.$$

is independent in $\mathcal{H}_b$.

Finally, by construction, for all $i \in [h]$, $I_i \cap \chi = \{u\}$ with $\bar{u} = \bar{u}_i \in \sigma'$. Also $\sigma = \sigma' \cup \sigma''$ is independent in χ hence in $\mathcal{H}_b$. No vertices $y_1 \in I_i$ and $y_2 \in \sigma''$ can be adjacent because, again, this would imply that $\{y_1, y_2\} \subseteq \chi$ and contradict the fact that $\bar{y}_1, \bar{y}_2$ are independent in σ. Thus $J_{v,b}^{\sigma}$ is independent.

We now prove that $J_{v,b}$ is of maximum size. Observe that it suffices to show this again for each $J_{v,b}^{\sigma}$. Each maximum independent set J of $\mathcal{H}_b$ that contains v and whose vertices in χ have exactly the representants σ can be expressed as $\tau \cup J_1 \cup J_2 \cup ... \cup J_m$. Here $\tau \subseteq \chi$ is an independent set of b containing v and whose representants are σ. Furthermore, J_i is an independent set of $\mathcal{H}_b$ that contains only vertices of V_{b_i}. The set J_i may only contain one vertex u_i from $\chi \cap \chi_i$. But then exchanging J_i for J_{u_i,b_i} may only increase the size of the independent set, so we can assume that I has the form $\tau \cup J_{u_1,b_1} \cup J_{u_2,b_2} \cup ... \cup J_{u_m,b_m}$ where u_i may also stand for $\emptyset$.

Assume now that $J_{v,b}^{\sigma}$ is not maximum, i.e. there is an independent set J containing v whose vertices in χ have the representants σ and J is bigger than $J_{v,b}^{\sigma}$. Then one of four following things must happen:

- There is an i such that $v \in \chi_i$ and $J \cap V_{b_i}$ is bigger than J_{v,b_i}. But this case cannot occur by induction.

- $v = u_1$ and there is a $j \in S_1$ such that $v \notin \chi_j$ and $|J \cap V_{b_j}| > |J_{\emptyset,b_j}|$. By induction we know that $J_{\emptyset,b_j}$ is optimal under all independent sets of $\mathcal{H}_{b_j}$ not containing any vertex of $\chi_j \cap \chi$, so there must be a vertex $u \in J \cap \chi \cap \chi_j$. Since J is independent, v and u share no edge in λ and then $\bar{v} \neq \bar{u}$. Since $j \in S_1$, it holds that $\mathbf{c}(v) \cap \chi_j \neq \emptyset$ and by Equation 1, $\mathbf{c}(u) \cap \chi_j = \emptyset$. Contradiction.

- There is an $i \in S$ such that $J \cap V_{b_i}$ is bigger than $J_{\emptyset,b_i}$. But from $i \in S$ it follows by definition that $\chi \cap \chi_i \cap J = \emptyset$, so this case can not occur by induction, either.

- There is an $i \in [h]$ such that $|J \cap (\bigcup_{j \in S_i} V_j)| > |I_i|$. We claim that $(\bigcup_{j \in S_i} \chi_j) \cap \chi \cap J$ contains only one vertex. Assume there are two such vertices x and y. By definition, $\bar{x}, \bar{y} \in \bar{\tau}$. Since J is independent, $\bar{x}$ and $\bar{y}$ are not adjacent in $\bar{\chi}$ and $\bar{x} \neq \bar{y}$. At least one of these, say y, must be in $\mathbf{c}(u_i)$, because $\bar{u}_i \in \bar{\tau}$ by definition. Let $x \in V_{j'}$ with $j' \in S_i$, then there is a vertex $w \in \mathbf{c}(u_i) = \mathbf{c}(y)$ in $\chi_{j'} \cap \chi \subseteq e_j$ by definition of S_i. But then $\bar{x}$ and $\bar{y}$ are adjacent in $\bar{\chi}$ which is a contradiction.

  So there is exactly one vertex u in $(\bigcup_{j \in S_i} \chi_j) \cap \chi \cap J$. But then $|J \cap (\bigcup_{j \in S_i} V_j)| > I_{i,u}$. Thus either there must be a $j \in S_i$ with $u \in V_j$ such that $|J \cap V_j| > |J_{u,b_j}|$ or there must be a $j \in S_i$ with $u \notin V_j$ such

that $|J \cap V_j| > |J_{\emptyset,b_j}|$. The former clearly contradicts the optimality of J_{u,b_j}, while the latter leads to a contradiction completely analogously to the second item above.

Because only $k2^{k^2}n^2$ sets have to be considered for each guarded block, this results in an algorithm with runtime $k2^{k^2}|V|^{O(1)}$. $\square$

LEMMA 6.5. *Computing maximum independent sets on hypergraphs is* $\mathbf{W}[1]$-*hard parameterized by generalized hypertree width.*

PROOF. We will show a reduction from p-IndependentSet which is the following problem: Given a graph G and an integer k which is the parameter, decide if G has an independent set of size k. Because p-IndependentSet is well known to be $\mathbf{W}[1]$-hard, this suffices to establish $\mathbf{W}[1]$-hardness of independent set parameterized by hypertree width.

So let $G = (V, E)$ be a graph and let k be a positive integer. We construct a hypergraph $\mathcal{H} = (V', E')$ in the following way: For each vertex v the hypergraph $\mathcal{H}$ has k vertices $v_1, \ldots, v_k$. For $i = 1, \ldots, k$ we have an edge $V_i := \{v_i \mid v \in V\}$ in E'. Furthermore, for each $v \in V$ we add an edge $H_v := \{v_i \mid i \in [k]\}$. Finally we add the edge sets $E_{ij} := \{v_i u_j \mid uv \in E\}$ for $i, j \in [k]$. $\mathcal{H}$ has no other vertices or edges. The construction is illustrated in Figure 3.

We claim that G has an independent set of size k if and only if $\mathcal{H}$ has an independent set of size k. Indeed, if G has an independent set $v^1, \ldots, v^k$, then $v_1^1 \ldots v_k^k$ is easily seen to be an independent set of size k in $\mathcal{H}$. Now assume that $\mathcal{H}$ has an independent set I of size k. Then for each $v \in I$ we can choose a vertex $\pi(v) \in V$ such that $v \in H_{\pi(v)}$. Furthermore for distinct $v, u \in I$ the corresponding vertices $\pi(v), \pi(u)$ have to be distinct, too, so $\pi(I) \subseteq V$ has size k. Finally, we claim that $\pi(I)$ is independent in G. Assume this is not true, then there are vertices $\pi(v), \pi(u)$ such that $\pi(v)\pi(u) \in E$. But then $vu \in E'$ by construction which is a contradiction. So, indeed G has an independent set of size k if and only if $\mathcal{H}$ has one.

We now show that $\mathcal{H}$ has generalized hypertree width at most k by constructing a generalized hypertree decomposition $(\mathcal{T}, (\lambda_t)_{t \in T}, (\chi_t)_{t \in T})$ of $\mathcal{H}$ of width k. The tree $\mathcal{T}$ only consists of one single vertex v, the block of v is $\chi_v := V'$ and the guard is $\lambda_t := \{V_1, \ldots, V_k\}$. It is easily seen that this is indeed a hypertree decomposition of width k.

Observing that the construction of $\mathcal{H}$ from G can be done in time polynomial in $|V|$ and k completes the proof. $\square$

6.3 Approximation

We have seen that computing maximum independent sets of hypergraphs with decompositions of width k can be done in polynomial time for fixed width k and that for some decompositions it is even fixed parameter tractable with respect to k. Still, the exponential influence of k is troubling. In this section we will show that we can get rid of it if we are willing to sacrifice the optimality of the solution. We give a k-approximation algorithm for computing maximum independent sets of graphs with generalized hypertree width k assuming that a decomposition is given. We start by formulating a lemma.

LEMMA 6.6. *Let* $\mathcal{H}$ *be a hypergraph with a generalized hypertree decomposition* $\Xi = (\mathcal{T}, (\lambda_t)_{t \in T}, (\chi_t)_{t \in T})$ *of width* k.
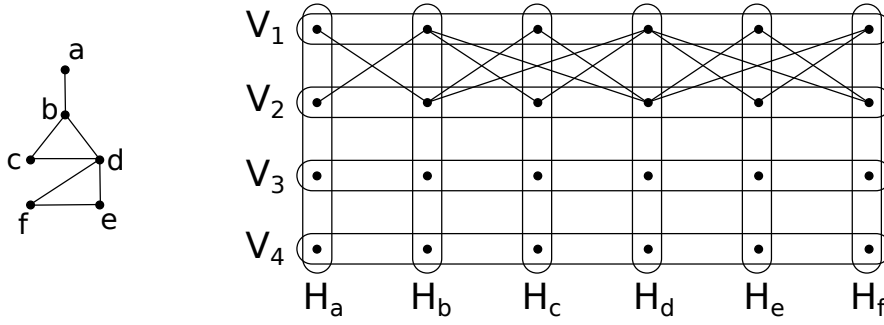
Figure 3: We illustrate the construction for Lemma 6.5 by an example. A graph G on the left with the associated hypergraph $\mathcal{H}$ for $k = 4$ on the right. To keep the illustration more transparent the edge sets E_{ij} are not shown except for $E_{1,2}$ and $E_{2,1}$.

Let $\mathcal{H}' = (V, E')$ where $E' := \{\chi_t \mid t \in T\}$. Let ℓ be the size of a maximum independent set in $\mathcal{H}$ and let ℓ' be the size of a maximum independent set in $\mathcal{H}'$. Then

$$\frac{\ell}{k} \leq \ell' \leq \ell.$$

Before we prove Lemma 6.6 we will show how to get the approximation algorithm from it.

OBSERVATION 6.7. *Every independent set of $\mathcal{H}'$ is also an independent set of $\mathcal{H}$.*

PROOF. Each pair of independent vertices x, y in $\mathcal{H}'$ is by definition only in different blocks χ_t in $\mathcal{H}$. For each edge $e \in E$ there must by definition of generalized hypertree decompositions be a block χ such than $e \subseteq \chi$. Thus no edge $e \in E$ can contain both x and y, so x and y are independent in $\mathcal{H}$, too. $\square$

COROLLARY 6.8. *There is a polynomial time algorithm that given a hypergraph $\mathcal{H}$ and a generalized hypertree decomposition of width k computes an independent set of size ℓ of $\mathcal{H}$ such that $|I| \geq \frac{\ell}{k}$ where ℓ is the size of a maximum independent set of $\mathcal{H}$.*

PROOF. Observe that $\mathcal{H}'$ is acyclic. With Lemma 4.2 we compute a maximum independent set I of $\mathcal{H}'$ whose size by Lemma 6.6 only differs by a factor $\frac{1}{k}$ from ℓ. By Observation 6.7 we have that I is also an independent set of $\mathcal{H}$. $\square$

PROOF OF LEMMA 6.6. The second inequality follows directly from Observation 6.7.

For the first inequality consider a maximum independent set I of $\mathcal{H}$. Observe that a set I' is an independent set of $\mathcal{H}'$ if and only if it is an independent set of its primal graph $\mathcal{H}'_P$, so it suffices to show the same result for $\mathcal{H}'_P$.

CLAIM 1. *$\mathcal{H}'_P[I]$ has treewidth at most $k - 1$.*

PROOF. We construct a tree decomposition from Ξ. To do so consider $\Xi[I]$ which for each guarded block (λ, χ) of Ξ contains (λ', χ') where $\lambda' := \{e \cap I \mid e \in \lambda, e \cap I \neq \emptyset\}$ and $\chi' := \chi \cap I$. The set I is independent, so each guard of $\Xi'[I]$ is a set of singletons and if follows $|\chi'| \leq |\lambda'|$ for each guarded block (λ', χ').

Let $\mathcal{T}[I]$ be the tree of $\Xi[I]$ induced by $\mathcal{T}$ in the obvious way. Then the blocks $\chi' = \chi \cap I$ still fulfill the connectedness

condition. Furthermore, for each edge uv in $\mathcal{H}'[I]$ there is a guarded block (λ', χ') such that $u, v \in \chi'$. Thus $\Xi[I]$ is a tree decomposition of $\mathcal{H}'_P[I]$. But we have that $|\chi'| \leq |\lambda'| \leq |\lambda| \leq k$ and thus the tree decomposition is of width at most $k - 1$. $\square$

CLAIM 2. *$\mathcal{H}'_P[I]$ has an independent set I' of size at least $\frac{|I|}{k}$.*

PROOF. From Claim 1 it follows that $\mathcal{H}'[I]$ and all of its subgraphs have a vertex of degree at most k (see e.g. [12, p. 265]). We construct I' iteratively by choosing a vertex of minimum degree and deleting it and its neighbors from the graph. In each round we delete at most k vertices, so we can choose a vertex in at least $\frac{|I|}{k}$ rounds. Obviously the chosen vertices are independent. $\square$

Every independent set of $\mathcal{H}_P[I]$ is also an independent set of $\mathcal{H}_P$ which completes the proof. $\square$

7. FRACTIONAL HYPERTREE WIDTH

In this section we extend the main results of the paper to fractional hypertree width, which is the most general known notion that leads to tractable Boolean CQ [18]. In particular it is strictly more general than generalized hypertree width.

DEFINITION 7.1. *Let $\mathcal{H} = (V, E)$ be a hypergraph. A fractional edge cover of a vertex set $S \subseteq V$ is a mapping $\psi : E \to [0, 1]$ such that for every $v \in V$ we have $\sum_{e \in E : v \in e} \psi(e) \geq 1$. The weight of ψ is $\sum_{e \in E} \psi(e)$. The fractional edge cover number of S, denoted by $\rho^*_{\mathcal{H}}(S)$ is the minimum weight taken over all fractional edge covers of S.*

A fractional hypertree decomposition of $\mathcal{H}$ is a triple $(\mathcal{T}, (\chi_t)_{t \in T}, (\psi_t)_{t \in T})$ where $\mathcal{T} = (T, F)$ is a tree, and $\chi_t \subseteq V$ and ψ_t is a fractional edge cover of χ_t for every $t \in T$ satisfying the following properties:

1. *For every $v \in V$ the set $\{t \in T \mid v \in \chi_t\}$ induces a subtree of $\mathcal{T}$.*

2. *For every $e \in E$ there is a $t \in T$ such that $e \subseteq \chi_t$.*

The width of a fractional hypertree decomposition $(\mathcal{T}, (\chi_t)_{t \in T}, (\psi_t)_{t_i n T})$ is $\max_{t \in T}(\rho^*_{\mathcal{H}}(\chi_t))$. The fractional hypertree width of $\mathcal{H}$ is the minimum width over all fractional hypertree decompositions of $\mathcal{H}$.

We first formulate a version of Theorem 4.1 for fractional hypertree width.

THEOREM 7.2. *There is an algorithm that given a #CQ-instance Φ of quantified starsize ℓ and fractional hypertree width k counts the solutions of Φ in time $\|\Phi\|^{p(k,\ell)}$ for a polynomial p.*

Also S-star size or equivalently independent sets of bounded fractional hypertree width hypergraphs can be computed efficiently.

LEMMA 7.3. *There is an algorithm that given a hypergraph $\mathcal{H} = (V, E)$ of fractional hypertree width at most k computes a maximum independent set of $\mathcal{H}$ in time $|\mathcal{H}|^{k^{O(1)}}$.*

8. CONCLUSION

The results of this paper give a clear picture of tractability for counting solutions of conjunctive queries for structural classes that are known to have tractable decision problems. Essentially counting is tractable if and only if these classes are combined with quantified star size. So to find more general structural classes that allow tractable counting, progress for the corresponding decision question appears to be necessary.

Another way of generalizing the results of this paper would be extending the logic that the queries can be formulated in. Just recently Chen and Dalmau [7] have characterized the tractable classes of bounded arity QCSP which is essentially a version of CQ in which also universal quantifiers are allowed. They do this by introducing a new width measure for first order $\{\forall, \exists, \wedge\}$-formulas. We conjecture that their width measure also characterizes the tractable cases for #QCSP, i.e. tractable decision and counting coincide here. It would be interesting to see how far this can be pushed for the case of unbounded arity.

Another extension appears in a recent paper by Chen [5] where he considers existential formulas that may use conjunction and disjunction. This is particularly interesting, because it corresponds to the classical select-project-join queries with union that play an important role in database theory. One may wonder if Chen's results may be extended to counting, too.

9. REFERENCES

[1] I. Adler, G. Gottlob, and G. Grohe. Hypertree width and related hypergraph invariants. *Eur. J. Comb.*, 28(8):2167–2181, 2007.

[2] G. Bagan, A. Durand, and G. Grandjean. On Acyclic Conjunctive Queries and Constant Delay Enumeration. In *CSL'07, 16th Annual Conference of the EACSL*, volume 4646 of *LNCS*, pages 208–222. Springer, 2007.

[3] H. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 226–234. ACM, 1993.

[4] A. Bulatov, V. Dalmau, M. Grohe, and D. Marx. Enumerating homomorphisms. *J. Comput. Syst. Sci.*, 78(2):638–650, 2012.

[5] H. Chen. On the Complexity of Existential Positive Queries. *ArXiv e-prints*, June 2012.

[6] H. Chen and V. Dalmau. Beyond Hypertree Width: Decomposition Methods Without Decompositions. In *11th International Conference Principles and Practice of Constraint Programming*, CP '05, pages 167–181, 2005.

[7] H. Chen and V. Dalmau. Decomposing quantified conjunctive (or disjunctive) formulas. LICS, 2012.

[8] D. Cohen, P. Jeavons, and M. Gyssens. A unified theory of structural tractability for constraint satisfaction problems. *Journal of Computer and System Sciences*, 74(5):721 – 743, 2008.

[9] V. Dalmau and P. Jonsson. The complexity of counting homomorphisms seen from the other side. *Theor. Comput. Sci.*, 329(1-3):315–323, 2004.

[10] A. Durand and S. Mengel. The Complexity of Weighted Counting for Acyclic Conjunctive Queries. *Arxiv preprint arXiv:1110.4201*, 2011.

[11] J. Flum, M. Frick, and M. Grohe. Query Evaluation via Tree-Decompositions. *J. ACM*, 49(6):716–752, 2002.

[12] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag New York Inc, 2006.

[13] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. *Artif. Intell.*, 124(2):243–282, 2000.

[14] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *J. Comput. Syst. Sci.*, 64(3):579–627, 2002.

[15] G. Gottlob, Z. Miklós, and T. Schwentick. Generalized Hypertree Decompositions: NP-Hardness and Tractable Variants. *J. ACM*, 56(6), 2009.

[16] G. Greco and F. Scarcello. Structural Tractability of Enumerating CSP Solutions. In *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming*, CP '10, pages 236–251, 2010.

[17] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), 2007.

[18] M. Grohe and D. Marx. Constraint Solving via Fractional Edge Covers. In *17th annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, pages 289–298, New York, NY, USA, 2006. ACM.

[19] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 657–666. ACM, 2001.

[20] M. Gyssens, P. Jeavons, and D. Cohen. Decomposing Constraint Satisfaction Problems Using Database Techniques. *Artif. Intell.*, 66(1):57–89, 1994.

[21] L. Libkin. *Elements of Finite Model Theory*. EATCS Series. Springer, 2004.

[22] D. Marx. Approximating fractional hypertree width. *ACM Trans. Algorithms*, 6(2):29:1–29:17, Apr. 2010.

[23] Z. Miklós. *Understanding Tractable Decompositions for Constraint Satisfaction*. PhD thesis, University of Oxford, 2008.

[24] R. Pichler and A. Skritek. Tractable Counting of the Answers to Conjunctive Queries. In *AMW*, 2011.

[25] M. Yannakakis. Algorithms for Acyclic Database Schemes. In *VLDB*, pages 82–94, 1981.