Size and Treewidth Bounds for Conjunctive Queries

GEORG GOTTLOB and STEPHANIE TIEN LEE, University of Oxford, UK
GREGORY VALIANT and PAUL VALIANT, University of California, Berkeley

This article provides new worst-case bounds for the size and treewith of the result $Q(D)$ of a conjunctive query Q applied to a database D. We derive bounds for the result size $|Q(D)|$ in terms of structural properties of Q, both in the absence and in the presence of keys and functional dependencies. These bounds are based on a novel "coloring" of the query variables that associates a *coloring number* $C(Q)$ to each query Q. Intuitively, each color used represents some possible entropy of that variable. Using this coloring number, we derive tight bounds for the size of $Q(D)$ in case (i) no functional dependencies or keys are specified, and (ii) simple functional dependencies (keys) are given. These results generalize recent size-bounds for join queries obtained by Atserias et al. [2008]. In the case of arbitrary (compound) functional dependencies, we use tools from information theory to provide lower and upper bounds, establishing a close connection between size bounds and a basic question in information theory. Our new coloring scheme also allows us to precisely characterize (both in the absence of keys and with simple keys) the treewidth-preserving queries—the queries for which the treewidth of the output relation is bounded by a function of the treewidth of the input database. Finally, we give some results on the computational complexity of determining the size bounds, and of deciding whether the treewidth is preserved.

Categories and Subject Descriptors: F.2.0 [**Analysis of Algorithms and Problem Complexity**]: *Miscellaneous*; H.2.4 [**Database Management**]: Systems—*Relational databases, query processing*

General Terms: Algorithm, Design, Theory

Additional Key Words and Phrases: Database theory, conjunctive queries, size bounds, treewidth

ACM Reference Format:
Gottlob, G., Lee, S. T., Valiant, G., and Valiant, P. 2012. Size and treewidth bounds for conjunctive queries. J. ACM 59, 3, Article 16 (June 2012), 35 pages.
DOI = 10.1145/2220357.2220363 http://doi.acm.org/10.1145/2220357.2220363

16

1. INTRODUCTION

In this article we consider the general question of how large and how intricate the result of a conjunctive query can be relative to the input relations. We derive worst-case

The technical content of this article appeared as two works: Size and treewidth bounds for conjunctive queries [Gottlob et al. 2009] and Size bounds for conjunctive queries with general functional dependencies [Valiant and Valiant 2009].

G. Gottlob is a James Martin Senior Fellow.
G. Gottlob's work was supported by the EPSRC Grant EP/G055114/1 Constraint Satisfaction for Configuration: Logical Fundamentals, Algorithms, and Complexity. G. Valiant's work was supported by a National Science Foundation Graduate Research Fellowship. P. Valiant's work was supported by a National Science Foundation Mathematical Sciences Postdoctoral Research Fellowship.

Authors' addresses: G. Gottlob, University of Oxford, Room 358, Wolfson Building, Parks Road, Oxford OX1 3QD UK; email: georg.gottlob@cs.ox.ac.uk; S. T. Lee, 400 Howard Street, San Francisco, CA 94105; email: stlee@post.harvard.edu; G. Valiant and P. Valiant, University of California, Berkeley, Berkeley, CA 94720; email: {gregory.valiant, pvaliant}@gmail.com.

DOI 10.1145/2220357.2220363 http://doi.acm.org/10.1145/2220357.2220363

bounds for the size of the query result, which depend only on structural properties of the query itself. We also derive bounds for the treewidth of the query result—a natural measure of the intricacy of the database—and characterize those conjunctive queries that are treewidth-preserving.

Conjunctive queries are the most fundamental and most widely used database queries [Abiteboul et al. 1995; Chandra and Merlin 1977; Levy et al. 1995] . They correspond to project-select-join queries in the relational algebra, and to SQL queries of the form `SELECT` ... `FROM` ... `WHERE` ..., whose where-clause equates pairs of attributes of the relations contained in the from-clause. Conjunctive queries also correspond to nonrecursive datalog rules of the form

$$R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge R_{i_2}(u_2) \cdots \wedge R_{i_m}(u_m),$$

where R_{i_j} is a relation name of the underlying database (which we will frequently refer to as D), R_0 is the output relation, and where each argument u_j is a list of $|u_j|$ variables, (where $|u_j|$ is the arity of the corresponding relation R_{i_j}). Note that each variable can occur multiple times in one or more argument lists. Also, $i_1, \ldots, i_m$ are not necessarily distinct, and thus we allow a single relation R_i to appear several times in the query. Throughout this article, we adopt this datalog rule representation for conjunctive queries.

In general, it is clear that the result of a conjunctive query can be exponentially large in the input size. Even in the case of bounded arities, the result can be substantially larger than the input relations. In fact, in the worst case, the output size is r^k, where r is the number of tuples in the largest input relation and k is the arity of the output relation.

Queries with very large outputs are sometimes unavoidable, but in most cases they are either ill-posed or anyway undesirable, as they can be disruptive to a multi-user DBMS. It is thus useful to recognize such queries, whenever possible. Further, obtaining good worst-case bounds for conjunctive queries is relevant to view management [Levy et al. 1995] and data integration [Lenzerini 2002; Levy et al. 1995], as well as to data exchange [Fagin et al. 2003; Kolaitis 2005], where data is transferred from a source database to a target database according to schema mappings that are specified via conjunctive queries. In this latter context, good bounds on the result size of a conjunctive query may be used for estimating the amount of data that needs to be materialized at the target site.

In the area of query optimization, models for predicting the size of the output of a conjunctive query based on selectivity indices for relational operators have been developed [Chaudhuri 1998; Jarke and Koch 1984; Swami and Schiefer 1994]. The selectivity indices are obtained via sampling techniques (see, e.g., Olken and Rotem [1990] and Haas et al. [1996]) from existing database instances. Worst case bounds may be obtained by setting each selectivity index to 1, thus assuming the maximum selectivity for each operator. Unfortunately, the resulting bounds are then often trivial (akin to this r^k bound).

A new and very interesting characterization of the worst-case output size of *join queries* was developed by Atserias et al. [2008]. The join queries form a proper subclass of the conjunctive queries, and correspond to those conjunctive queries in which *all* query variables appear in the head atom $R_0(u_0)$. Their result is based on the notion of *fractional edge cover* [Grohe and Marx 2006], and the associated concept of *fractional edge-cover number* $\rho^*(Q)$ of a join query Q. In particular, in Grohe and Marx [2006] it was shown that

$$|Q(D)| \leq \mathrm{rmax}(D)^{\rho^*(Q)}, \tag{1}$$

where rmax(D) represents the number of tuples in the largest input relation among $R_1, \ldots, R_n$ in D. In Atserias et al. [2008], it was shown that this bound is essentially tight.

A substantial part of this article paper deals with the question of whether it is possible to find similar tight worst-case bounds for the output of general conjunctive queries. In particular, we asked this question in three different settings:

(1) for general conjunctive queries without functional dependencies (integrity constraints);
(2) for general conjunctive queries where simple functional dependencies (keys) have been specified on the underlying database D; and
(3) for general conjunctive queries in the setting where arbitrary functional dependencies have been specified on D.

We derive tight bounds similar to those in Eq. (1) for settings 1 and 2, and we give both lower and upper bounds for the third (most general) setting. Moreover, for the third setting, we give a precise characterization of the queries for which no size increase is possible.

A crucial step towards our results is the introduction of a new coloring scheme for query variables, and, accordingly, the association of a *color number* $C(Q)$ with each query Q. Roughly, a valid coloring assigns a set $\mathcal{L}(X)$ of colors to each query variable X, such that for each functional dependency $R[i]R[j] \rightarrow R[k]$, if variables X, Y, Z occur in the ith, jth, and kth positions, respectively, of an atom $R(u)$ in the body of query Q, then $\mathcal{L}(Z) \subseteq \mathcal{L}(X) \cup \mathcal{L}(Y)$. The *color number* $C(Q)$ of Q is the maximum over all valid colorings of Q of the quotient of the number of colors appearing in the output (i.e., head) variables of Q by the maximum number of colors appearing in the variables of any input (i.e., body) atom of Q. Intuitively, each color used represents some possible entropy of that variable.

In the case that functional dependencies are given (settings 2 and 3), rather than coloring the original query Q, we color *chase*(Q), which is the result of chasing the keys over the query. The *chase* [Aho et al. 1979a; Beeri and Vardi 1984; Deutsch et al. 2006; Fagin et al. 2003; Maier et al. 1979] is a well-known procedure for enforcing functional dependencies, and, in particular, keys. For example, given the query $Q = R(X, Y, Z) \leftarrow S(X, Y) \wedge S(X, Z)$, where the first attribute of the relation S is a key for S, or equivalently, where the functional dependency $S[1] \rightarrow S[2]$ has been specified, then *chase*(Q) is the query $R(X, Y, Y) \leftarrow S(X, Y)$.

We prove the following essentially tight worst-case bounds for the case without functional dependencies, and with simple functional dependencies (or keys):

$$|Q(D)| \leq \text{rmax}(D)^{C(\mathit{chase}(Q))}.$$

In these settings, the color number $C(\mathit{chase}(Q))$ can be computed in polynomial time via a polynomial-time chase followed by the solution of a polynomially sized linear program. As a corollary, we show that when $C(\mathit{chase}(Q))$ is bounded, and when each query variable occurs in the output variables u_0, we obtain a polynomial algorithm for query evaluation.

For composite keys and arbitrary functional dependencies (Setting 3), while our lower bound given by the color number holds, we illustrate that the color number no longer provides an upper bound on the worst-case size increase. Nevertheless, we precisely characterize the set of queries that admit no size increase.[1]

> Given a conjunctive query Q and set of arbitrary functional dependencies, there exists a database D compatible with Q and the functional dependencies with $|Q(D)| > \mathrm{rmax}(D)$, if and only if $C(\mathit{chase}(Q)) > 1$.

Beyond this characterization of queries that admit no size increase, in order to provide size bounds in this general setting we require machinery beyond the color number. We use tools from information theory developed to analyze the precise interactions of multivariate distributions, and construct a linear program with entropies as the variables and the exponent of the worst-case size increase as the solution. Functional dependencies can be encoded as constraints in the linear programs. The difficulty is determining which additional constraints must be added to the linear program to ensure that the solution is realizable as a database instance. This question, as it turns out, is crucially related to an old and ongoing investigation at the heart of information theory: "which entropy structures can be instantiated in multivariate distributions?" [Dougherty et al. 2007; Matúš 2007a, 2007b; Pippenger 1986; Zhang and Yeung 1997, 1998]. We cannot show that our upper bound is tight in this general setting, and believe that an explicit (even exponential-sized) characterization of the worst-case size increase is unlikely without significant advances in information theory.

The remainder of our results analyze how much more intricate the results of a query can be than the input database. An important measure for describing the inherent intricacy of a graph or finite structure (in our case, a database) is *treewidth* [Robertson and Seymour 1986a]. The treewidth $\mathrm{tw}(D)$ of a structure D corresponds in a precise sense to its degree of cyclicity. For example, each tree has treewidth 1, each cycle has treewidth 2, the clique K_k of k elements has treewidth $k-1$.

By Courcelle's theorem [Courcelle 1990], all Boolean queries that can be expressed in terms of monadic second-order logic can be answered in linear time on structures (databases) of bounded treewidth. This result was generalized to a large class of queries based on monadic *optimization problems* [Arnborg et al. 1991]. By these results, interesting queries that cannot be formulated in SQL or first order logic, and that are NP-hard on arbitrary structures, can be answered in linear time on structures of bounded treewidth. Queries in this class include, for example, questions related to network multicuts [Gottlob and Lee 2007], program flow graphs [Thorup 1998], or logic-based diagnosis [Gottlob et al. 2007]. One may not always issue such queries directly to the original database, but in some cases to a view that has been defined via a conjunctive query.

In this context, we consider the following questions.

— Given a set of relations D of bounded treewidth and a conjunctive query Q, how large is the treewidth of $Q(D)$?
— Is it possible to characterize the queries that *preserve bounded treewidth* in the sense that $\mathrm{tw}(Q(D)) = O(\mathrm{tw}(D))$?

We begin by considering a simple class of queries: *keyed joins*. A join expression $R \bowtie_{A=B} S$ is a *keyed join* if B is a key for S. (Here B may be a composite attribute.)

[1]This characterization of queries that admit no size increase was referred to as a characterization of "sparsity preserving" queries in our conference paper [Gottlob et al. 2009]; since the term "sparsity" is somewhat misleading, in this version we reframed the results of Gottlob et al. [2009] that pertained to database sparsity in terms of size bounds.

We derive the following bound on the treewidth of the result of a keyed join $R \bowtie_{A=B} S$ where S has arity j, $\mathrm{tw}(\langle R, S\rangle) = \omega$ and B is a key of S:

$$\mathrm{tw}(R \bowtie_{A=B} S) \leq j(\omega + 1) - 1.$$

Via a nontrivial example, we show that this bound is tight up to a constant factor. We also extend this bound to sequences of keyed joins and to conjunctive queries whose bodies consist of such sequences.

For the case of (fixed) queries Q over (variable) input databases D without keys, we use colorings to precisely characterize the treewith-preserving queries:

> *$tw(Q(D)) = O(tw(D))$ if, and only if $tw(Q(D)) \leq tw(D)$ if, and only if, there exists no valid coloring of Q with two colors having color number two. (See Definition* 3.2 *for the definition of color number.)*

In the setting with arbitrary (simple) keys that are not necessarily part of the join attributes, we also characterize the queries that are treewidth-preserving. The proof uses a reduction to the case without keys, leveraging both the bounds on the increase in treewidth after a sequence of keyed joins, and the above characterization of treewidth-preserving queries in the setting without keys.

This article is organized as follows. In Section 2, we state some useful definitions. In Section 3, we define the basic coloring scheme and the color number of a query, and discuss the connection between these concepts and the size bounds of Grohe and Marx [2006] and Atserias et al. [2008]. Section 4 proves our main size-bounds in the case without functional dependencies, and with simple keys or functional dependencies. Section 5 presents all our results on treewidth. Section 6 considers the setting where arbitrary functional dependencies are specified on the underlying database, and frames size bounds and the coloring scheme in terms of the entropy structures that are realizable in a distribution over the tuples of a query's output relation. Section 7 deals with the question of the complexity of computing the size bounds and of deciding whether the treewidth is bounded. Finally, some open problems and directions for future research are discussed in Section 8.

2. PRELIMINARIES AND BASIC DEFINITIONS

We assume familiarity with the theory of relational database systems, relational algebra, and conjunctive queries, and provide a cursory overview of the main notions mainly to clarify notation.

A *finite structure* or *database* $D = (\mathcal{U}_D, R_1, \ldots, R_n)$ consists of a finite universe $\mathcal{U}_D$ and relations $R_1, \ldots, R_n$ over $\mathcal{U}_D$. As already mentioned in the introduction, a *conjunctive query* has the form $Q = R(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$, where each u_j is a list of (not necessarily distinct) variables of length $|u_j|$. Each variable occurring in the query head $R_0(u_0)$ must also occur in the body of the query. The set of all variables occurring in Q is denoted by $var(Q)$. It is important to recall that a single relation R_i might appear several times in the query, and thus it may happen that $i_j = i_k$ for some pairs of indices j, k. The query Q may be applied to D provided that for all $j \in [m]$, index $i_j \in [n]$ and $arity(R_{i_j}) = |u_j|$. The answer $Q(D)$ of query Q over database D consists of the structure $(\mathcal{U}_D, R_0)$ whose unique relation R_0 contains precisely all tuples $\theta(u_0)$ where $\theta : var(Q) \rightarrow \mathcal{U}_D$ is a substitution such that for each atom $R_{i_j}(u_j)$ appearing in the query body, $\theta(u_j) \in R_{i_j}$. For ease of notation, we define $\mathrm{rmax}(D)$ to be the number of tuples in the largest relation among $R_1, \ldots, R_n$ in D, and define $|Q(D)|$ to be the number of tuples in the output relation.

A *(simple) attribute* of a relation R identifies a column of R. We refer to the attribute in the ith *position* of relation R by $R[i]$. An *attribute list* consists of a list (without repetition) of attributes of a relation R. A *compound attribute* is an attribute list with at least two attributes. A list consisting of a unique attribute A is identified with A. The list of all attributes of R is denoted by $attr(R)$. If V is a list of attributes of R and $t \in R$ a tuple of R, then the V-value of t, denoted by $t[V]$ consists of the tuple obtained as the ordered list of all values in V-positions of t.

If A and B are (possibly compound) attributes of R, then a *functional dependency* $A \rightarrow B$ on relation R expresses that for each pair of tuples $t, t' \in R$, $t[A] = t'[A]$ implies that $t[B] = t'[B]$. If A and B are single attributes, then the functional depencency $A \rightarrow B$ is called a *simple functional dependency*. A (possibly compound) attribute K of R is a *key* iff $K \rightarrow attr(R)$ holds. Such a key is called a *simple key* if K is a simple attribute, otherwise it is called a *compound key*.[2] An argument position in an atom that corresponds to a simple key attribute is referred to as a *keyed position*.

Throughout, for clarity and succinctness of exposition, when given a query and set of relations, we admit the slight abuse of notation and also refer to functional dependencies *between query variables*: that is, given a functional dependency $R[i] \rightarrow R[j]$, if the query in question has an atom $R(u)$ in the body of the query with variables X and Y occurring in positions i and j, respectively, we may refer to the functional dependency as $X \rightarrow Y$. This interpretation is consistent, in the sense that in any output tuple, the value in a position corresponding to X uniquely determines the value in a position corresponding to Y. Note that this representation of functional dependencies as dependencies between query variables does not contain the full information of the actual set of function dependencies; for example, given a query body $R(X, Y) \wedge S(X, Y)$, the functional dependency between variables, $X \rightarrow Y$, means that either $R[1] \rightarrow R[2]$, or $S[1] \rightarrow S[2]$, or both. Nevertheless, it is the structure of the dependencies between the query variables which will be relevant to our size bounds, and it will be convenient to refer to and reason about these dependencies between query variables using the notation of functional dependencies.

Let $D = (\mathcal{U}_D, R_1, \ldots, R_n)$ be a structure. The *Gaifman graph* $G(D)$ of D is the graph $(\mathcal{U}_D, E)$, where $\{a, b\} \in E$ iff a and b are two distinct elements from $\mathcal{U}_D$ which appear jointly in some tuple of a relation of D.

Given a graph $\mathcal{G} = (V, E)$, a *tree decomposition* [Robertson and Seymour 1986a] of $\mathcal{G}$ is a pair (T, λ), where $T = (V', E')$ is a tree, and λ a labeling function $\lambda : V' \rightarrow 2^V$ such that: (i) for all $v \in V$, there exists $b \in V'$ such that $v \in \lambda(b)$; (ii) for all edges $e \in E$, there exists $b \in V'$ such that $\lambda(b) \supseteq e$; (iii) for every $v \in V$, the set $\{b \in V' \mid v \in \lambda(b)\}$ induces a connected subtree in T. So as to avoid possible confusion between the vertex set of $\mathcal{G}$ and the vertices of the tree T, we will refer to the vertices of the tree as *bags*, and will say that a bag $b \in V'$ *contains* the set $\lambda(b)$ of vertices of $\mathcal{G}$.

The *width* of a tree decomposition (T, λ) with $T = (V', E')$ is the integer value $\max\{|\lambda(b)| - 1 \mid b \in V'\}$. The *treewidth* of a graph $\mathcal{G} = (V, E)$, denoted $\mathrm{tw}(\mathcal{G})$, is the minimum width of all tree decompositions. Given a finite structure $D = (\mathcal{U}_D, R_1, \ldots, R_n)$, the treewidth of D is defined to be the treewidth of its Gaifman graph, formally $\mathrm{tw}(D) = \mathrm{tw}(G(D))$.

In some proofs, it will be more convenient to work with the equivalent definition of treewidth in terms of *elimination orderings*. Given graph $\mathcal{G} = (V, E)$, an *elimination ordering* $\pi = v_1, \ldots, v_n$, of the vertex set induces a sequence of graphs $\mathcal{G} = \mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_n$, where $\mathcal{G}_i$ is obtained from $\mathcal{G}_{i-1}$ by adding edges so as to make the neighbor set of v_i in $\mathcal{G}_{i-1}$ a clique, and then removing vertex v_i from the graph (thus $\mathcal{G}_i$ is a graph on $\{v_{i+1}, \ldots, v_n\}$ and $\mathcal{G}_n$ is the empty graph). The *elimination width* of an ordering π is the largest clique present in one of the graphs, $\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_n$, and the treewidth of $\mathcal{G}$ is one less than the minimum elimination width achieved by any elimination ordering.

[2]Note: We do not require compound keys to be minimal.

The following motivating example illustrates the treewidth of a database, and shows that the size and treewidth of the result of a conjunctive query can be significantly larger that of the input database.

Example 2.1. Consider the relation

$$R(A, B) = \{\langle 1, 1\rangle, \langle 1, 2\rangle, \ldots, \langle 1, n\rangle\}$$

that has treewidth 1. The relation $R' = R \bowtie_{A=A} R$, which is equivalently obtained as the result of the conjunctive query

$$R'(X, Y, Z) \leftarrow R(X, Y) \wedge R(X, Z),$$

has n^2 tuples where each of the n elements appears in a tuple with each of the other elements. Thus, the Gaifman graph is the complete graph on n vertices, which has treewidth $n - 1$. Thus, we see that the treewidth of R' can be arbitrarily large, while the treewidth of R is still 1.

In the case where a given relation appears more than once in the query, there may be some additional implied dependencies, beyond those that can be derived from the set of functional dependencies. The following example illustrates such an instance, and motivates the *chase* operation which we define in this section.

Example 2.2. Consider the query

$$R_0(WXYZ) \leftarrow R_1(WXY) \wedge R_1(WWW) \wedge R_2(YZ),$$

together with the first position of R_1 being a key for R_1.

The information that the first position of R_1 is a key for R_1, and the atom $R_1(WWW)$ imply that in any output tuple, the values in positions corresponding to W, X, and Y must all be equal. Thus, in addition to the key information, there is also the further restriction on the output tuples that the values in positions X and Y must be the same as the value in position W, and thus there can be at most $|R_2|$ tuples in the output.

This example motivates the *chase* procedure that reduces the set of variables so as to eliminate any implied dependencies on the *values* taken by the variables beyond those implied by the given set of functional dependencies. [Maier et al. 1979].

Definition 2.3. Given a conjunctive query

$$Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m),$$

we define *chase*(Q) to be the result of iteratively performing the following replacements:

— Given two atoms $R_{i_j}(u_j)$ and $R_{i_k}(u_k)$ where $i_j = i_k$ and a simple functional dependency for that relation, $R_{i_j}[q] \rightarrow R_{i_j}[r]$, if the variables in the qth positions of u_j and u_k are identical, then for every instance (anywhere in the query) of the variable occurring in the rth position of u_k, we replace that variable with the variable occurring in the rth position of u_j.

This definition extends naturally to compound functional dependencies: for a dependency $R[i]R[j] \rightarrow R[k]$, if the relation R occurs twice in the query with the same variable list in positions i and j in both atoms, then all occurrences of the variable in position k of one of the atoms are replaced by the variable occurring in position k of the other atom. The analogous definition applies to functional dependencies involving more than two positions.

The query obtained by performing the chase operation is dependent on the order in which atoms are selected for this replacement procedure—throughout, so as to make *chase*(Q) well defined, we assume that the chase is performed according to some (arbitrary) fixed ordering. The following fact confirms the intuition that the substitutions in Definition 2.3 do not affect the result of the query.

FACT 2.4 [AHO ET AL. 1979B; BEERI AND VARDI 1984; MAIER ET AL. 1979]. Let Q be a conjunctive query and $Q' = \mathit{chase}(Q)$. For every D to which Q may be applied, we have $Q(D) = Q'(D)$.

The algorithm for computing *chase*(Q) runs in polynomial time, even in the case of arbitrary functional dependencies, (see, e.g., Aho et al. [1979b] for an $O(n^4)$ algorithm, where n is the total size of the input consisting of the conjunctive query Q together with a set of functional dependencies).

3. THE COLORING

In this section we define a coloring scheme for assigning colors to the variables appearing in a conjunctive query. This coloring underlies both our bounds on the size of the results of the query, and our characterizations of queries that have bounded treewidth.

Definition 3.1. Given a conjunctive query $Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$, and set of functional dependencies, a *valid coloring* of Q with c colors assigns to each variable $X \in \mathit{var}(Q)$ a label $\mathcal{L}(X) \subseteq \{1, \ldots, c\}$ consisting of zero or more colors, that satisfies the following properties:

— For each functional dependency $X_1 \cdots X_k \rightarrow Y$,

$$\mathcal{L}(Y) \subseteq \bigcup_{i=1,\ldots,k} \mathcal{L}(X_i).$$

— There exists a variables $X \in \mathit{var}(Q)$ with $\mathcal{L}(X) \neq \emptyset$.

Definition 3.2. The *color number* of a coloring $\mathcal{L}$ for query $Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$, is the ratio of the total number of colors appearing in the output variables u_0, to the maximum number of colors appearing in any given u_j, for $j \geq 1$; and the *color number* of query Q, denoted $C(Q)$, is the maximum color number attained by any valid coloring of Q. Formally:

$$C(Q) := \max_{\mathcal{L}} \frac{|\cup_{X \in u_0} \mathcal{L}(X)|}{\max_{i \geq 1} |\cup_{X \in u_i} \mathcal{L}(X)|},$$

where $\mathcal{L}$ ranges over valid colorings of Q.

The following two examples illustrate the definition of color number.

Example 3.3. Let Q be the query

$$S(X, Y, Z) \leftarrow R(X, Y) \wedge R(X, Z) \wedge R(Y, Z).$$

Assume no keys are asserted. Then $C(Q) = 3/2$, which is attained with three different colors, one for each $\mathcal{L}(X)$, $\mathcal{L}(Y)$, and $\mathcal{L}(Z)$.

Example 3.4. Consider the query of Example 2.2: Let

$$Q = R_0(WXYZ) \leftarrow R_1(WXY) \wedge R_1(WWW) \wedge R_2(YZ),$$

with the functional dependencies implied by the first position of R_1 being a key for R_1, thus $W \rightarrow X$ and $W \rightarrow Y$. The labeling

$$\mathcal{L}(W) = \{1\},\ \mathcal{L}(X) = \mathcal{L}(Y) = \emptyset,\ \mathcal{L}(Z) = \{2\},$$

is a valid coloring, with color number 2, and in fact $C(Q) = 2$.

Now consider $Q' = \mathit{chase}(Q) = R_0(WWWZ) \leftarrow R_1(WWW) \wedge R_2(WZ)$, and note that since all the variables appearing in the output relation also appear together in one of the atoms of the query body, $C(Q') = 1 < C(Q)$. In general, $C(\mathit{chase}(Q)) \leq C(Q)$, because, trivially, any valid coloring of $\mathit{chase}(Q)$ is also a valid coloring of Q.

3.1. Color Number and Fractional Edge Coverings

The color number of a query seems to be a new concept; nevertheless, in the special case of a query $Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$ without any keys or functional dependencies, colorings of a specific form are related to fractional edge covering of the hypergraph associated to query Q (see Definition 3.5) via linear programming duality. Our main size bounds of Section 4 build upon the results of Grohe and Marx [2006] and Atserias et al. [2008], in which the size increase for join queries (queries without functional dependencies (or keys), where all variables appear in the output atom) is bounded by the minimal fractional edge cover number; thus, it will be useful to establish this connection between the color number and the minimal fractional edge cover number. Demonstrating this connection relies on observing that the lack of functional dependencies implies that there is always a maximal coloring with an especially simple structure, and then leveraging this structure to show that the color number of such queries is given as the solution to a linear program, whose dual describes the minimal fractional edge covering. We now make this high-level overview rigorous.

Definition 3.5. Given query $Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$, the *minimal fractional edge cover number* of Q, denoted by $\rho^*(Q)$, is given by the following linear program:

$$\begin{array}{ll} \text{minimize} & \sum_{j=1}^{m} y_j \\ \text{subject to} & \sum_{j:X\in u_j} y_j \geq 1 \qquad \forall X \in \mathit{var}(Q), \\ & y_j \geq 0. \end{array}$$

This definition should be interpreted as the linear program relaxation of the minimal edge covering of the hypergraph corresponding to Q in which $\mathit{var}(Q)$ are the graph nodes, and each u_j defines a hyperedge.

The following proposition shows that for queries with no functional dependencies, the color number is given by the solution to a simple linear program.

PROPOSITION 3.6. *Given query $Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \ldots \wedge R_{i_m}(u_m)$ with no functional dependencies, the color number, $C(Q)$, is given as the solution to the following linear program:*
Assume without loss of generality that $\mathit{var}(Q) = \{X_1, \ldots, X_n\}$,

$$\begin{array}{ll} \textit{maximize} & \sum_{i:X_i\in u_0} x_i \\ \textit{subject to} & \sum_{i:X_i\in u_j} x_i \leq 1 \qquad \forall j \geq 1 \\ & x_i \geq 0. \end{array}$$

While colorings are combinatorial objects, it is worth stressing that the above linear program is *not* a proper relaxation of the quantity in question; the linear program will have a rational solution (whose bit–length is polynomial in $|Q|$), and any rational solution p/q to this linear program with $x_i{=}r_i/q$ can be transformed into a valid coloring $\mathcal{L}$ with p colors such that $|\mathcal{L}(X_i)| = r_i$. The constraints of the linear program then imply $\max_j |\cup_{X\in u_j} \mathcal{L}(X)| \leq q$.

PROOF OF PROPOSITION 3.6. First, note that there is always an optimal coloring with the property that for all $X \notin u_0$, $\mathcal{L}(X) = \emptyset$. Next, observe that it suffices to assume that no color appears in the label of more than a single variable. To see this, if a color appears in the labels of at least two variables, by removing it from the label of one of the labels, the numerator of the expression for the color number remains unchanged whereas the denominator can only decrease. Thus, the color number of such a query is given by the following expression, where $x_i \in \mathbb{N}$ denotes the number of colors assigned to variable $X_i \in u_0$ in an optimal coloring:

$$\max_{x_i \in \mathbb{N}} \frac{\sum_{i:X_i \in u_0} x_i}{\max_{j \geq 1} \sum_{i:X_i \in u_j} x_i}.$$

Pushing the normalization factor into constraints on the x_i's, the above expression is equivalent to the following linear program (which is maximized over $x_i \in \mathbb{R}$):

$$\begin{array}{rl} \text{maximize} & \sum_{i:X_i \in u_0} x_i \\ \text{subject to} & \sum_{i:X_i \in u_j} x_i \leq 1 \ \ \forall j \geq 1 \\ & x_i \geq 0. \end{array}$$

□

From Proposition 3.6 and linear programming duality, for a query Q with no functional dependencies, the color number $C(Q)$ is also given by the following linear program, which is dual to that of Proposition 3.6:

$$\begin{array}{rl} \text{minimize} & \sum_j y_j \\ \text{subject to} & \sum_{j:X \in u_j} y_j \geq 1 \ \ \forall X \in u_0 \\ & y_j \geq 0. \end{array}$$

This linear program can be recognized as expressing the minimal fractional edge cover of the hypergraph associated to query Q', which is obtained from Q by simply removing all variables that don't appear in u_0 from all atoms.

4. SIZE BOUNDS

In this section, we present our main size bounds. We consider conjunctive queries without functional dependencies, and with simple keys (or simple functional dependencies). Whenever we use the word "key" in this section we mean "simple key". Essentially, the color number of a query will be a tight bound on the exponent relating the sizes of the input and the output database. We start this section by considering the special case in which the given query has no keyed relations, and use the connection between color number and minimal fractional edge cover number described in the previous section to relate our characterization to previous results regarding size bounds. We will then reduce the general case with keys to this special case.

4.1. Size Bounds without Keys

We begin by considering the case when no functional dependencies are specified. While this case has essentially been considered [Atserias et al. 2008; Grohe and Marx 2006], our approach will allow us to extend the results to the case with specified output variables and keys.

PROPOSITION 4.1. *Given a query $Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$ without specification of functional dependencies or keys, for any database D,*

$$|Q(D)| \leq rmax(D)^{C(Q)},$$

where $rmax(D)$ is the size of the largest relation among $R_1, \ldots, R_n$ in D. Furthermore, this bound is essentially tight: for any integer $N > 0$, there exists a database D with

$rmax(D) \leq rep(Q) \cdot N$, and $|Q(D)| = N^{C(Q)}$, where $rep(Q)$ is defined to be the maximum number of times any specific relation R_i appears in Q.

COROLLARY 4.2. *Given a query Q, as in Proposition 4.1, without functional dependencies or keys, if, for all databases D compatible with Q, the number of tuples in the result $Q(D)$ is at most the number of tuples in the input relations, then there exists an $i \in [m]$ such that the variables in u_0 are a (not necessarily proper) subset of the variables of u_i.*

Our proof of Proposition 4.1 will follow easily from the connection between the color number $C(Q)$ and the minimal fractional edge cover number $\rho^*(Q)$ demonstrated in Section 3.1, together with the following proposition given in [Atserias et al. 2008; Grohe and Marx 2006].

PROPOSITION 4.3 (LEMMAS 2 AND 3 OF ATSERIAS ET AL. [2008]). *Given a query $Q = R_0(u_0) \leftarrow R_1(u_1) \wedge \cdots \wedge R_m(u_m)$ without specification of FDs or keys, where u_0 contains all variables in $var(Q)$, then for any database D,*

$$|Q(D)| \leq rmax(D)^{\rho^*(Q)},$$

where $rmax(D)$ is the size of the largest relation among $R_1, \ldots, R_n$ in D.

Furthermore, this bound is tight: for any integer $N > 0$, there exists a database D with $rmax(D) = N$, and $|Q(D)| = N^{\rho^*(Q)}$.

The upper bound is proved in Grohe and Marx [2006], using a clever argument in which Shearer's Lemma is employed to exploit the submodularity of the entropy function to yield a combinatorial statement. The tightness of the bound is via a simple construction, and we give a generalization of this construction in the proof of Proposition 4.5 in the following section.

PROOF OF PROPOSITION 4.1. For the upper bound, first note that the exponent of the size increase of $Q(D)$ is at most that of the query $Q' = R_0(u_0) \leftarrow S_1(u_1) \wedge \cdots \wedge S_m(u_m)$, which is identical to Q except that each relation is distinct, because for any database D compatible with Q, there is a database D' compatible with Q' with $\text{rmax}(D) = \text{rmax}(D')$, and $Q(D) = Q'(D')$. Such a database D' is obtained from D by simply making multiple copies of the relations that occur multiple times in query Q. Next, given a query Q' in which each relation occurs at most once, we claim that the size increase affected by Q' is at most that of the related query Q'' which is identical to Q' except that all variables not appearing in the head u_0 have been removed. To see why this is the case, given any database D' with relations $S_1, \ldots, S_m$ that is compatible with Q', consider the database D'' with relations $T_1, \ldots, T_m$ where $T_i(D'')$ consists of projecting $S_i(D')$ onto the attributes that occur in positions corresponding to the variables in u_i that also occur in u_0. Thus, $Q'(D') = Q''(D'')$, and $\text{rmax}(D'') \leq \text{rmax}(D')$. Proposition 4.3 applies to Q'', since it consists of a total join (no projections), and each relation occurs exactly once, and thus by Proposition 3.6 and the comments at the end of Section 3.1 that show that for such queries $C(Q) = \rho^*(Q)$, we have

$$|Q''(D'')| \leq \text{rmax}(D'')^{\rho^*(Q'')} = \text{rmax}(D'')^{C(Q'')}.$$

To conclude the argument, note that trivially, $C(Q) = C(Q')$, and as was mentioned at the end of Section 3.1, there is always a maximal coloring (of a query without any FDs) in which variables not appearing in u_0 are not assigned any colors, and thus $C(Q') = C(Q'')$.

The lower bound follows by noting that if relation R_i appears j_i times in Q, then given a database D' consistent with the query Q' as defined above, one can construct a database D by letting each $R_i(D)$ consist of the union of the j_i sets of tuples of the j_i corresponding relations in D', in which case $Q(D) = Q'(D')$, and $\mathrm{rmax}(D) \leq rep(Q)\mathrm{rmax}(D')$. □

While Proposition 4.1 is similar to the size bounds based on the fractional edge cover number, and coincides with it in case all query variables occur in the head's atom, our characterization in terms of colorings provides a more concrete and intuitive connection with database instances; each color represents a degree of freedom of the output. This more intuitive characterization of the fractional edge cover in terms of colorings allows us to extend Proposition 4.1 to the case with simple keys and general conjunctive queries that include a projection. Furthermore, while the color number will not be directly applicable to bounding the possible increase in treewidth, a related property of the colorings will allow us to characterize those queries with bounded blowup in treewidth.

4.2. Size Bounds with Simple Functional Dependencies

We apply the intuition of the color number provided in the previous section with the hope of yielding a generalization of Proposition 4.1.

THEOREM 4.4. *Given a query $Q = R(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$ and set of simple keys,*

$$|Q(D)| \leq rmax(D)^{C(chase(Q))}.$$

Furthermore, this bound is essentially tight: for any $N > 0$, there exists a database D with $\frac{rmax(D)}{rep(Q)} > N$, and

$$|Q(D)| \geq \left(\frac{rmax(D)}{rep(Q)}\right)^{C(chase(Q))},$$

where $rep(Q)$ is the maximum number of times any single relation appears in Q.

The upper bound does not follow trivially from Proposition 4.1 because the color number in the presence of functional dependency information may be significantly smaller than if the key information is ignored.

We start by proving the tightness of the bound via a construction, which applies to the case that we are given an arbitrary set of functional dependencies (not just simple dependencies). This construction is an extension of that given in Atserias et al. [2008], and provides some insight into the connections between colorings of variables and the entropy of different sets of variables (under the uniform distribution over the output tuples). This insight will be useful in the remainder of the paper.

PROPOSITION 4.5. *Given a query $Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \ldots \wedge R_{i_m}(u_m)$ with a set of arbitrary functional dependencies, and any valid coloring $\mathcal{L}$ of $chase(Q)$ with color number $C(chase(Q))$, for any $M \in \mathbb{N}$ there exists a database D satisfying:*

— $|Q(D)| = M^{|\cup_{X \in u_0} \mathcal{L}(X)|}$,

— $rmax(D) \leq rep(Q) \cdot M^{\frac{|\cup_{X \in u_0} \mathcal{L}(X)|}{C(chase(Q))}}$.

PROOF. We first give the construction in the case that $rep(Q) = 1$, in which case $Q = chase(Q)$. Let d be the total number of colors used in the coloring defined by labeling $\mathcal{L}$. Consider a table T of arity d, with attributes $C_1, \ldots, C_d$, corresponding to each of the d colors. We construct the table T so that each attribute C_i takes M values, $v_{i,1}, \ldots, v_{i,M}$, and T consists of the total join (cartesian product) of these attribute values, and thus has M^d d-tuples, and the projection $\pi_{C_{i_1},\ldots,C_{i_k}}(T)$ of T onto any k attributes $C_{i_1}, \ldots, C_{i_k}$ has size M^k.

We now use T to construct the desired database D. For a given relation R, where the atom $R(u)$ occurs in the body of Q, assume without loss of generality that the variables $X_1, \ldots, X_\ell$ occur in u, and that in the given coloring of $chase(Q)$, $\bigcup_{i=1,\ldots,\ell} \mathcal{L}(X_i) = \{1, \ldots, q\}$. We populate $R(D)$ with M^q tuples, which will be derived from the M^q tuples in $\pi_{C_1,\ldots,C_q}(T)$ as follows: the attribute[s] corresponding to variable X_i with colors $\mathcal{L}(X_i) = \{i_1, \ldots, i_{|\mathcal{L}(X_i)|}\}$ will take $M^{|\mathcal{L}(X_i)|}$ values of the form $v_{(i_1,h_1),\ldots,(i_{|\mathcal{L}(X_i)|},h_{|\mathcal{L}(X_i)|})}$ for $h_k \in [M]$. For each tuple $\langle v_{1,j_1}, \ldots, v_{q,j_q}\rangle$ of $\pi_{C_1,\ldots,C_q}(T)$, we add the tuple to $R(D)$ where the position in the tuple corresponding to variable X_i takes the value $v_{(i_1,j_{i_1}),(i_2,j_{i_2}),\ldots,(i_{|\mathcal{L}(X_i)|},j_{i_{|\mathcal{L}(X_i)|}})}$. In the case that $\mathcal{L}(X_i) = \emptyset$, in every tuple, the corresponding position takes a special null symbol, $v_\emptyset$, and if there are no colors assigned to any variable in atom u, we add the single tuple, all of whose positions take value $v_\emptyset$.

By construction, for each atom $R(u)$ occurring in Q, we have $|R(D)| = M^{|\bigcup_{X\in u} \mathcal{L}(X)|}$. Additionally, because D is formed from the table T that is the complete product table, for each of the $|Q(D)| = M^{|\bigcup_{X\in u_0} \mathcal{L}(X)|}$ tuples in T that correspond to possible assignments to the attributes of T corresponding to colors in $\bigcup_{X\in u_0} \mathcal{L}(X)$, there will be the associated tuple in $Q(D)$, and since this association between such tuples in T and in $Q(D)$ is bijective, $|Q(D)| = M^{|\bigcup_{X\in u_0} \mathcal{L}(X)|}$, as claimed.

To see that the constructed database D obeys all functional dependencies, consider a dependency $X_1 \cdots X_i \to X_j$, and let relation R have positions corresponding to $X_1, \ldots, X_i$ and X_j. Since $\mathcal{L}$ is a valid coloring, $\mathcal{L}(X_j) \subseteq \bigcup_{h\in[i]} \mathcal{L}(X_h)$, and thus by construction, in any tuple of $R(D)$ the subscripts of the values in positions $X_1, \ldots, X_i$ uniquely determine the values in the associated tuple of T taken by attributes (of T) corresponding to the colors in $\bigcup_{h\in[i]} \mathcal{L}(X_h)$ and thus uniquely determine the subscripts of the value in position X_j.

To conclude, we consider the case where $rep(Q) > 1$. In this case, given $chase(Q) = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$, we populate a database D' in the fashion described above, corresponding to the query $Q' = R_0(u_0) \leftarrow R'_1(u_1) \wedge \cdots \wedge R'_m(u_m)$, which is identical to $chase(Q)$, except each relation is assumed to be distinct. If relation R in Q was replaced by relations $R'_1, \ldots, R'_k$ in Q', we simply let the tuples of $R(D)$ be the union $\bigcup_{j=1,\ldots,k} R'_j(D')$. Note that all functional dependencies will be preserved, since the coloring of Q', by assumption respects all the dependencies of $chase(Q)$. This concatenation increases the size of $\mathrm{rmax}(D)$ by at most a factor of $rep(Q)$, from which the claim follows. ☐

To complete our proof of Theorem 4.4, we now show that the upper bound holds. To prove the upper bound, we will transform $chase(Q)$ and the set of simple functional dependencies into a query Q' over a modified set of relations that has no functional dependencies. This transformation will have the property that $C(chase(Q)) = C(Q')$, and the transformation preserves the relationship between the size of the original structure and the size of the output. To obtain Q', we first let the query Q^* be the same as $chase(Q)$, except have each atom refer to a unique relation. Thus Q^* can be written as $R_0(u_0) \leftarrow R'_1(u_1) \wedge \cdots \wedge R'_m(u_m)$. To obtain Q' from Q^*, we fix some ordering of the variables $var(Q)$, $X_1, \ldots, X_{|var(Q)|}$, and ‘remove’ the functional dependencies in $|var(Q)|$ ‘rounds’, where the ith round consists of applying the following procedure (at most $|var(Q) - 1|$ times), resulting in removing all functional dependencies with X_i on the left side:

For each functional dependency with X_i on the left side, $X_i \to X_j$,

— replace each occurrence of X_i in any of the query atoms that does not already contain variable X_j, with the list X_i, X_j.

— For every functional dependency with X_i on the right side: $X_k \to X_i$, we add the dependency $X_k \to X_j$,
— we remove the functional dependency $X_i \to X_j$.

Although any given round might increase the total number of functional dependencies remaining, when the functional dependency $X_i \to X_j$ is 'removed', the only dependencies that can be added are of the form $X_{i'} \to X_{j'}$ with $i' > i$, and thus after i rounds, all remaining functional dependencies will be of the form $X_j \to X_k$, with $j > i$.

The following example illustrates this procedure:

Example 4.6. Consider $chase(Q) = Q^* = R_0(X_1) \leftarrow R_1(X_1X_2X_3) \wedge R_2(X_1X_4) \wedge R_3(X_5X_1)$, where the first attribute of each relation is a key. Variable X_1 is a key for X_2, X_3, in R_1, and thus applying the procedure to the functional dependency $X_1 \to X_2$ yields the query $R_0(X_1X_2) \leftarrow R_1(X_1X_2X_3) \wedge R_2(X_1X_2X_4) \wedge R_3(X_5X_1X_2)$, with the additional functional dependency $X_5 \to X_2$.

Performing the procedure to the dependencies $X_1 \to X_3$ and then to $X_1 \to X_4$ yields the query $R_0(X_1X_4X_3X_2) \leftarrow R_1(X_1X_2X_3) \wedge R_2(X_1X_3X_2X_4) \wedge R_3(X_5X_1X_4X_3X_2)$, with the new additional functional dependencies $X_5 \to X_3, X_4$.

Performing the procedure to the remaining four functional dependencies, $X_5 \to X_1, X_2, X_3, X_4$ has no effect, and thus the resulting query is $Q' = R'_0(X_1X_4X_3X_2) \leftarrow R'_1(X_1X_4X_2X_3) \wedge R'_2(X_1X_3X_2X_4) \wedge R'_3(X_5X_1X_4X_3X_2)$, with no functional dependencies.

We now argue that the above transformation does not alter the color number; that is, $C(chase(Q)) = C(Q')$, where Q' is obtained from $chase(Q)$ by the above procedure:

LEMMA 4.7. *$C(chase(Q)) = C(Q')$, where Q' is obtained from $chase(Q)$ by removing all functional dependencies as prescribed in this procedure.*

PROOF. We first show that $C(chase(Q)) \geq C(Q')$. Since the initial transformation to Q^* does not change the color number, it suffices to consider the effect of a single application of the procedure. Given a query Q_1 and set of simple functional dependencies, let Q_2 denote the query resulting from applying the prescribed procedure to remove the functional dependency $X \to Y$. Given a valid coloring of Q_2 defined by labeling $\mathcal{L}_2$, we define the coloring $\mathcal{L}_1$ of Q_1 as follows:

$$\mathcal{L}_1(X) := \mathcal{L}_2(X) \cup \mathcal{L}_2(Y),$$

and for all other variables $W \in var(Q_1)$, $\mathcal{L}_1(W) := \mathcal{L}_2(W)$. $\mathcal{L}_1$ is clearly a valid coloring because it respects the functional dependency $X \to Y$ by construction, and all other functional dependencies associated to Q_1 are also present in Q_2, and are thus respected by the labeling $\mathcal{L}_1$. To see that the color number of $\mathcal{L}_1$ for Q_1 is the same as that of $\mathcal{L}_2$ for Q_2, note that in any atom $R(u)$ of Q_2, either variable X is not present, or both X and Y are present: in either case, the total number of colors used in the coloring of variables in u in $\mathcal{L}_2$ is identical to that used by $\mathcal{L}_1$ to color the associated atom in Q_1. Thus, we conclude that $C(Q_1) \geq C(Q_2)$, and thus $C(chase(Q)) \geq C(Q')$.

For the other direction, let Q_1 and Q_2 be the query, and associated set of functional dependencies before, and after *all* functional dependencies with X on the left hand side have been removed. That is, let Q_2 be obtained from Q_1 by applying the above procedure for all functional dependencies $X \to Y_1, \ldots, Y_j$. Given a valid coloring $\mathcal{L}_1$ of Q_1, consider the coloring $\mathcal{L}_2$ defined by $\mathcal{L}_2(X) := \mathcal{L}_1(X) \setminus \bigcup_{i \in [j]} \mathcal{L}_1(Y_i)$, and $\mathcal{L}_2(W) := \mathcal{L}_1(W)$, for all variables $W \neq X$. To see that $\mathcal{L}_2$ is a valid coloring, note that the functional dependencies in Q_2 that were not present in Q_1 are of the form $Z \to Y_i$, where the functional dependency $Z \to X$ was present in Q_1; these functional dependencies are clearly respected by $\mathcal{L}_2$, since $\mathcal{L}_2(Z) = \mathcal{L}_1(Z) \supseteq \mathcal{L}_1(X) \supseteq \mathcal{L}_1(Y_i) = \mathcal{L}_2(Y_i)$.

Additionally, any functional dependency of Q_1 that remains in Q_2 must have a variable $W \neq X$ on the left hand side, and thus since $\mathcal{L}_2(W) = \mathcal{L}_1(W)$, it will be respected by coloring $\mathcal{L}_2$. Finally, we argue that the color number of $\mathcal{L}_2$ is equal to that of $\mathcal{L}_1$. Since $\mathcal{L}_1(X) \supseteq \bigcup_{i \in [j]} \mathcal{L}_1(Y_i)$, we have that $\mathcal{L}_1(X) = \mathcal{L}_2(X) \cup \left(\bigcup_{i \in [j]} \mathcal{L}_2(Y_i)\right)$; together with the fact that for any atom of Q_1 that contained variable X, the corresponding atom of Q_2 will contain the variables $X, Y_1, \ldots, Y_j$, we have that the color numbers of $\mathcal{L}_1$ and $\mathcal{L}_2$ are equal, from which it follows that $C(Q') \geq C(chase(Q))$, as desired. □

We are now ready to complete our proof of Theorem 4.4.

PROOF OF THEOREM 4.4. Proposition 4.5 establishes that the possible blowup is at least the color number. From Lemma 4.7, together with the size bound of Proposition 4.1 which applies to queries with no functional dependencies, such as the query Q' (yielded from $chase(Q)$ by removing all functional dependencies according to the procedure previously described), we have established that $C(chase(Q)) = C(Q')$, and that the exponent of the maximum possible size increase of query Q' is at most $C(Q')$. To establish our theorem, all that remains is to show that the potential size increase of query Q, together with its set of simple functional dependencies is at most that of Q'. To this end, it suffices to prove that this maximum size increase does not decrease for a single application of the procedure to remove the functional dependency $X \to Y$.

Given query Q_1 and its associated set of functional dependencies, let Q_2, and an associated set of functional dependencies be obtained from Q_1 by applying the prescribed procedure to remove the dependency $X \to Y$. Given a database D_1 upon which Q_1 can be evaluated, we shall construct database D_2 for Q_2 such that for every relation in D_1, the size of the corresponding relation in D_2 is identical, and additionally, $|Q_1(D_1)| = |Q_2(D_2)|$.

Since $X \to Y$ in Q_1, in database D_1, for each possible value x in a position corresponding to variable X, one can define a unique value y_x in positions corresponding to variable Y. To construct D_2, we start with D_1, and for every tuple of the relations of D_2 that has a position corresponding to variable X, we add an extra attribute corresponding to variable Y, and then for each tuple, we uniquely populate the extra attribute with the value $y(x)$ corresponding to the value x that the tuple has in a position corresponding to variable X. This generates a database compatible with Q_2, preserves the number of input tuples in each relation, and clearly has the property that $|Q_2(D_2)| = |Q_1(D_1)|$, as claimed. □

Finally, the following corollary to Theorem 4.4 captures the fact that if $C(chase(Q))$ is bounded, not only is the size of $|Q(D)|$ at most polynomial in $|D|$, but, in the case that all variables are output variables, $Q(D)$ can be computed in polynomial time via a join-project plan.

COROLLARY 4.8. *Given a conjunctive query*

$$Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m),$$

with simple functional dependencies, such that u_0 contains all the variables in $var(Q)$, for any database D compatible with Q, there exists a join-project plan for calculating $Q(D)$ in time

$$O\left(|var(Q)|^2 \cdot |Q|^2 \cdot rmax(D)^{C(chase(Q))+1}\right).$$

Furthermore, such a plan can be calculated in time $O(|Q|^3 + |Q|^2|var(Q)|^2)$, plus the $O(|Q|^4)$ time to compute $chase(Q)$, where $|Q|$ denotes the total length of query Q together with the associated set of functional dependencies.

The proof will follow easily from the proof of Theorem 4.4 together with Theorem 15 of Atserias et al. [2008], which we restate here for convenience.

THEOREM 4.9 [ATSERIAS ET AL. 2008]. *For every join query Q of m relations, and database D, there is a join-project plan for $Q(D)$ that can be evaluated in time $O(|Q|^2 rmax(D)^{\rho^*(Q)+1})$. Additionally, such a join-project plan can be computed in time $|Q|^2$.*

The following fact from Flum et al. [2002] is also helpful.

FACT 4.10. *Let $||R(D)||$ denote $|R(D)| \cdot arity(R)$. The join $R \bowtie S$ can be computed in time $O(||R(D)|| + ||S(D)|| + ||R(D) \bowtie S(D)||)$.*

PROOF OF COROLLARY 4.8. Given Q and D, from the comment after Fact 2.4, $chase(Q)$ can be computed in time $O(|Q|^4)$. The iterative procedure to remove functional dependencies described in the proof of Theorem 4.4 runs in time $O(|Q|^3)$, yielding a query Q' that is a join-project query without any functional dependencies, with $|Q'| \leq |Q| \cdot |var(Q)|$. Furthermore, a database D' is yielded via at most $m|var(Q)|$ keyed joins, which, by Fact 4.10, takes time at most $O(m|var(Q)|^2 \text{rmax}(D))$. Since u_0 contains all variables, from the proof of Theorem 4.4 it follows that $Q(D) = Q'(D')$, and $\text{rmax}(D) = \text{rmax}(D')$. The corollary now follows from the above theorem. □

It is worth noting that the condition that all variables appear in the output is necessary in the above corollary; in the absence of this condition, it is easy to construct examples of databases and queries for which $C(Q) = 1$ but for which evaluating the query is equivalent to solving an NP-hard problem.

5. TREEWIDTH

In this section, we consider the problem of deciding whether the result of a general conjunctive query together with a set of simple functional dependencies (or simple keys) can have treewidth unbounded by any function of the treewidth of the inputs and the query. While the characterization for bounded treewidth queries will be different than our characterization of queries that preserve the number of tuples, the characterization will still be based on properties of the set of valid colorings of the query.

We begin this section by providing a tight bound on the possible increase in treewidth after a single keyed join operation. Note that while the number of tuples in the result of a single keyed join operation can be no more than the number of input tuples, it still might be the case that the treewidth of the result is larger—nevertheless, we provide a bound on the possible increase. We then give our characterization of queries without functional dependencies that induce a bounded increase in treewidth. Finally, we leverage our results on the effect of keyed joins on the treewidth to give a characterization of queries with simple functional dependencies that induce a bounded increase in treewidth.

Throughout this section, when referring to join operations we will use the query notation $R \bowtie_{A=B} S$, as opposed to the more general datalog representation used elsewhere in this paper.

5.1. Treewidth of Keyed Joins

We present tight bounds on the blowup of the treewidth of the result of a single keyed join operation. The bound follows easily from the definition of treewidth, and the example illustrating the tightness of the bound is a nontrivial construction in which the Gaifman graphs of the initial relations are grids with some extra nodes and edges, and the Gaifman graph of the result contains a quadratically larger grid. We begin with the construction.

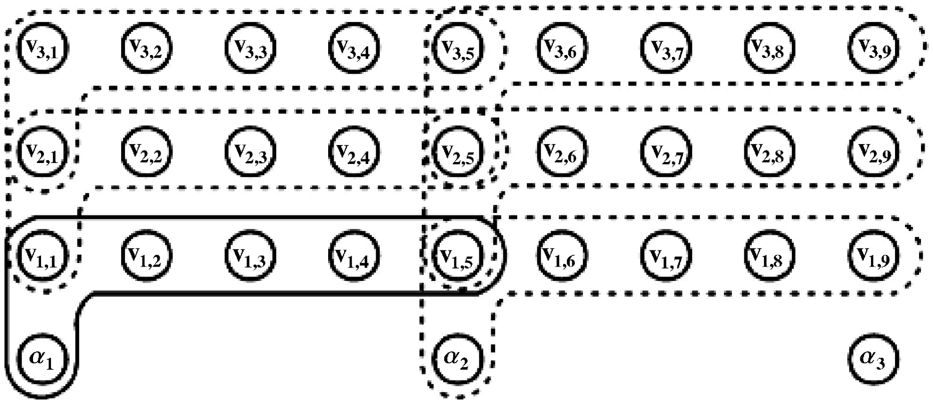

Fig. 1. The structure of G in the case $m = 4$, with the partitions into $S_{i,j}$ indicated: the set outlined with the solid line corresponds to the ordered set $S_{1,1} := (\alpha_1, v_{1,1}, v_{1,2}, v_{1,3}, v_{1,4}, v_{1,5})$.

The following standard fact will be useful, and follows immediately from the fact that the treewidth of an $n \times n$ rectangular grid is n, for $n \geq 2$ (see, e.g., Result 1.4 in Robertson and Seymour [1986b]).

FACT 5.1. *For positive integers m, n with $m + n \geq 3$, the treewidth of an $n \times m$ rectangular grid is* $\min(n, m)$.

PROPOSITION 5.2. *For any positive n, m with $m \leq n - 2$, there exists a relation R of arity $m + 2$ whose Gaifman graph has treewidth n, such that after a single keyed join operation, $R \between R$, the resulting treewidth is at least nm.*

PROOF. Consider an $(nm + 1) \times nm$ rectangular lattice, together with n additional vertices $\{\alpha_1, \ldots, \alpha_n\}$. We partition the $O(n^2m^2)$ vertices into ordered sets $S_{i,j}$ of size $m + 2$, for $1 \leq i \leq nm$, and $1 \leq j \leq n$ as follows. For $i = 1$, let $S_{i,j}$ consist of vertices $\alpha_j, v_{1,m(j-1)+1}, v_{1,m(j-1)+2}, \ldots, v_{1,mj+1}$. For $i \geq 2$, let $S_{i,j}$ consist of the $m + 2$ vertices

$$v_{i-1,m(j-1)+1}, v_{i,m(j-1)+1}, v_{i,m(j-1)+2}, \ldots, v_{i,m(j-1)+m+1}.$$

For each (i, j) pair, we add $\binom{m+2}{2}$ edges so as to make $S_{i,j}$ a complete graph. Let G denote the resulting graph. The structure of G is suggested in Figure 1, in which the partition of the nodes into the groups $S_{i,j}$ is depicted.

The validity of the construction is now intuitively clear: from the point of view of treewidth, G behaves like an $n \times nm$ grid and thus (from Fact 5.1) has treewidth n; after a single keyed join operation, all edges between vertices in the set $S_{i,j} \cup S_{i-1,j}$ will be present, and thus the resulting graph will contain the $nm \times nm$ grid, and will thus have treewidth at least nm. The following lemmas make this intuition rigorous. □

LEMMA 5.3. *The treewidth of G is n.*

PROOF. The treewidth of G is at least n, since G contains the $n \times nm$ grid as a subgraph. To see that the treewidth of G is also bounded by n, we will exhibit a (partial) elimination ordering that reduces G to an $n \times nm$ graph without creating any cliques of size more than $m + 2 \leq n$. Define the set $V = \{v_{i,j} : i, j \in \{1, \ldots, mn\}$ and $j \neq 1 + km\}$, for any integer k. Every vertex in V has degree $m + 2$, with all its neighbors forming a clique. Thus, by iteratively eliminating all vertices in V (in any ordering), we preserve the invariant that each remaining node has degree at most $m + 2$, with a neighbor set that is a clique. After eliminating all nodes in V, we then remove the nodes $v_{i,mn+1}$, and α_i (which all have a single neighbor), leaving us with the $n \times nm$ grid, as desired. □

We now define the database instance associated with the graph G and sets $S_{i,j}$. Define the relation R, with attributes $A_1, \ldots, A_{m+2}$, and populate a database D as follows: for each ordered set $S_{i,j}$ include the $(m+2)$-tuple whose ℓth attribute A_ℓ has value equal to the vertex label of the ℓth vertex in $S_{i,j}$.

LEMMA 5.4. *The treewidth of $R(D)$ is n, and the treewidth of the result of the keyed join $R(D) \bowtie_{A_1=A_2} R(D)$, is at least nm.*

PROOF. $R(D)$ consists of the n^2m tuples
$(\alpha_j, v_{1,m(j-1)+1}, v_{1,m(j-1)+2}, \ldots, v_{1,mj+1})$, and

$$(v_{i-1,m(j-1)+1}, v_{i,m(j-1)+1}, v_{i,m(j-1)+2}, \ldots, v_{i,mj+1}),$$

for $2 \leq i \leq nm$, $1 \leq j \leq n$, and thus has Gaifman graph G. From Lemma 5.3, $\mathrm{tw}(R(D)) = n$.

Let G^G be the Gaifman graph of $R(D) \bowtie_{A_1=A_2} R(D)$. For all i, k, $1 \leq i \leq nm$, $1 \leq k \leq nm$, the edge $(v_{i,k}, v_{i,k+1})$ is in G, and thus is also in G^G. Furthermore, for $1 \leq i \leq nm-1$, $1 \leq k \leq nm+1$, the edge $(v_{i,k}, v_{i+1,k})$ is also in G^G as a result of the join operation. Thus G^G contains the grid on $nm+1 \times nm$ vertices as a subgraph, and thus has treewidth at least nm. □

The following theorem shows that the construction of Figure 1 yields the worst-case increase in treewidth.

THEOREM 5.5. *Let R, S be relations, over a database D where S has arity j, and $tw(\langle R(D), S(D)\rangle) = \omega$. If B is a key in $S(D)$, then,*

$$tw\left(R(D) \bowtie_{A=B} S(D)\right) \leq j(\omega+1) - 1.$$

Furthermore, this bound is tight to a constant factor, assuming $\omega \geq j \geq 1$.

The tightness of the bound follows from Proposition 5.2. The proof of the bound depends on the following observation that follows immediately from the definition of tree decomposition:

Observation 5.6. Let $G = (V, E)$ be a graph with tree decomposition $\mathcal{T} = (T = (V', E'), \lambda)$, with $\lambda : V' \to 2^V$. Fix $v, w \in V'$ and consider the path $p_{v,w}$ between them in T. Let $W \subseteq \lambda(v)$ and for each $v' \in V'$, define the associated set $S_{v'}$ as follows:

$$S_{v'} = \begin{cases} \lambda(v') \cup W & \text{if } v' \text{ lies on } p_{v,w} \\ \lambda(v') & \text{otherwise.} \end{cases}$$

Then $\mathcal{T}' := (T, \lambda')$, where for all $v' \in V'$, we define $\lambda'(v') := S_{v'}$ is also valid tree decomposition of graph G.

PROOF OF THEOREM 5.5. The proof is by explicitly constructing a tree decomposition for $R(D) \bowtie_{A=B} S(D)$, from a tree decomposition of $\langle R(D), S(D)\rangle$. We start by considering a tree decomposition $\mathcal{T}_0 = (T = (V', E'), \lambda_0)$ of $\langle R(D), S(D)\rangle$, and iteratively construct tree decompositions $\mathcal{T}_1 = (T = (V', E'), \lambda_1), \ldots, \mathcal{T}_k = (T = (V', E'), \lambda_k)$, for $k \leq |R(D)| \cdot |S(D)|$, with one decomposition corresponding to each of the $k \leq |R(D)| \cdot |S(D)|$ pairs of tuples t_i, u_i with $t_i \in R(D)$ and $u_i \in S(D)$ satisfying $t_i(A) = u_i(B)$.

To obtain λ_i from λ_{i-1}, note that there are vertices in the tree, $v, v' \in V'$ such that $\lambda_0(v)$ contains all the attribute values that occur in tuple t_i, and $\lambda_0(v')$ contains all values that occur in u_i. Let $v = v_0, v_1, \ldots, v_m = v'$ be the (unique) path in T connecting

v and v', and let W consist of the set of all values that occur in tuple u_i other than the value $u_i(B)$. We define $\lambda_i(v_\ell) := \lambda_{i-1}(v_\ell) \cup W$ for $i \in [m]$, and for all other nodes $w \in V'$, set $\lambda_i(w) := \lambda_{i-1}(w)$.

By Observation 5.6, each $\mathcal{T}_i$ is a valid tree decomposition. Additionally, $\mathcal{T}_k$ is a tree decomposition for $R(D) \bowtie_{A=B} S(D)$, since by construction, for every output tuple, there is some $v \in V'$ such that $\lambda_k(v)$ contains all values that appear in the tuple.

We now analyze $\max_{v\in V'} |\lambda_k(v)|$. Let $\mathcal{T}_0$ be a minimal tree decomposition in that $\max_{w\in V'} |\lambda_0(w)| = \omega + 1$. If $\mathcal{T}_{i-1}$ is a valid tree decomposition, when $\mathcal{T}_i$ is formed by augmenting the 'bags' along path $v = v_0, v_1, \ldots, v_m = v'$, corresponding to the tuples t_i, u_i, it must be the case that for $\ell \in [m]$, $\lambda_{i-1}(v_\ell)$ contains the value $u_i(B)$, and thus $|\lambda_i(v_\ell)| \leq |\lambda_{i-1}(v_\ell)| + j - 1$. Finally, since B is a key for $S(D)$, for each value c, there is at most one tuple $t \in S(D)$ such that $t(B) = c$, and thus in our construction, for each node $w \in V'$, the values in at most $|\lambda_0(w)| \leq \omega+1$ tuples $t \in S(D)$ will be added to yield $\lambda_k(w)$, and since each such tuple of arity j contributes at most $j-1$ new values that were not present in $\lambda_0(w)$, we have that $|\lambda_k(w)| \leq (j-1)(\omega+1) + (\omega+1)$, as desired. □

The following proposition extends the above bounds to a sequence of keyed joins. While the proposition is intuitively clear, the proof involves some bookkeeping.

PROPOSITION 5.7. *Consider a query Q of the form*

$$\left(\cdots\left((R_1 \bowtie_{B_1=A_2} R_2) \bowtie_{B_2=A_3} R_3\right) \bowtie \ldots \bowtie_{B_{n-1}=A_n} R_n\right),$$

such that for $i \geq 2$ the attribute A_i occurs in a keyed position of R_i and let $\ell = \max_i(arity(R_i))$.

$$tw(Q(D)) \leq \ell^{n-1}\left(1 + \max\left(tw(\langle R_1(D), \ldots, R_n(D)\rangle), 2\right)\right) - 1.$$

The proof of this proposition follows from $n-1$ applications of the following lemma:

LEMMA 5.8. *For relations S_1, S_2, S_3, and a join query $S_1 \bowtie_{B_1=A_2} S_2$, where A_2 is in a keyed position of S_2, let $\ell = arity(S_2)$. Then, for any database D consistent with the query,*

$$tw\left(\langle S_1(D) \bowtie_{B_1=A_2} S_2(D), S_3(D)\rangle\right) \leq \ell\left(1 + \max\left(tw(\langle S_1(D), S_2(D), S_3(D)\rangle), 2\right)\right) - 1$$

PROOF. Let S_1, S_2, S_3 have respective arities s_1, s_2, s_3, and let B_1 refer to the ith position of S_1, and A_2 the jth position of S_2. We now construct a database D', and populate relations $S'_1(D'), S'_2(D')$ so as to contain all tuples of $S_1(D), S_2(D)$, respectively, and have the additional property that

$$G\left(\langle S'_1(D') \bowtie_{B_1=A_2} S'_2(D'), S_3(D)\rangle\right) = G\left(\langle S'_1(D') \bowtie_{B_1=A_2} S'_2(D')\rangle\right).$$

Assume that $s_1 \geq 3$, (otherwise we simply add dummy attributes to each tuple, and set their values to be the same as the value taken by the last attribute of S_1 in each tuple, and note that this does not affect the treewidths). Also, assume, without loss of generality, that $i \geq 3$. Let $S'_1(D')$ consist of all the tuples of $S_1(D)$. Additionally, for every pair of values c, c' that appear together in the same tuple of $S_3(D)$, we add the tuple $(c, c', \alpha_{\{c,c'\}}, \ldots, \alpha_{\{c,c'\}})$ to $S'_1(D')$, where the value $\alpha_{\{c,c'\}}$ appears only in that one tuple of $S'_1(D')$, and does not appear anywhere else in any of the relations. Thus, $arity(S'_1) = arity(S_1)$. Let $S'_2(D')$ consist of all the tuples of $S_2(D)$, and, as above, for every pair of values c, c' that appear together in a tuple of $S_3(D)$, add the tuple $(\alpha_{\{c,c'\}}, \ldots, \alpha_{\{c,c'\}})$ to $S'_2(D')$.

From the construction, it is clear that the jth position of $S'_2(D')$ is a key for $S'_2(D')$, and that the Gaifman graphs satisfy

$$G\left(\langle S'_1(D') \bowtie_{B_1=A_2} S'_2(D'), S_3(D)\rangle\right) = G\left(\langle S'_1(D') \bowtie_{B_1=A_2} S'_2(D')\rangle\right).$$

Furthermore, for each pair c, c', because the value $\alpha_{\{c,c'\}}$ only appears in the same tuple as two other values (namely c and c') a valid tree decomposition of $G(\langle S'_1(D'), S'_2(D')\rangle)$ can be made from any tree decomposition $\mathcal{T} = ((V', E'), \lambda)$ of $G(\langle S_1(D), S_2(D), S_3(D)\rangle)$ by adding adding a new vertex $v_{c,c'}$ to V' for each pair c, c', defining the label $\lambda(v_{c,c'}) := \{c, c', \alpha_{\{c,c'\}}\}$, and adding a single edge between node $v_{c,c'}$ and a node $w \in V'$ satisfying $\lambda(w) \supseteq \{c, c'\}$. Note that such a node w must exist because $\mathcal{T}$ is a valid tree decomposition, and values c, c' appear together in a tuple of $S_3(D)$. To conclude, note that in the process of modifying $\mathcal{T}$ to create a tree decomposition for $G(\langle S'_1(D'), S'_2(D')\rangle)$, each new vertex has a label of size at most 3, and thus

$$\mathrm{tw}(\langle S'_1(D'), S'_2(D')\rangle) \leq \max\left(\mathrm{tw}(\langle S_1(D), S_2(D), S_3(D)\rangle), 2\right).$$

Combining this with Theorem 5.5 we have the following:

$$\begin{aligned}
\mathrm{tw}\left(\langle S_1(D) \bowtie_{B_1=A_2} S_2(D), S_3(D)\rangle\right) &\leq \mathrm{tw}(S'_1(D') \bowtie_{B_1=A_2} S'_2(D')) \\
&\leq \ell\,\left(1 + \mathrm{tw}(\langle S'_1(D'), S'_2(D')\rangle)\right) - 1 \\
&\qquad\qquad \text{(from Theorem 5.5)} \\
&\leq \ell\,\left(1 + \max\left(\mathrm{tw}(\langle S_1(D), S_2(D), S_3(D)\rangle), 2\right)\right) - 1. \qquad \square
\end{aligned}$$

5.2. Bounding Treewidth without Functional Dependencies

In this section, we present a necessary and sufficient condition for the results of a general conjunctive query without key information to have treewidth bounded by some function of the input treewidth.

PROPOSITION 5.9. *Given a conjunctive query*

$$Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \ldots \wedge R_{i_m}(u_m),$$

with $i_j \in [n]$ and no functional dependencies, $tw(Q(D))$ can not be bounded as any function of $tw\left(\langle R_1(D), \ldots, R_n(D)\rangle\right)$, m, and $|u_i|$ if there exists a valid coloring of Q with 2 colors and color number 2. Furthermore, if there is no such coloring, then

$$tw(Q(D)) \leq tw\left(\langle R_1(D), \ldots, R_n(D)\rangle\right).$$

PROOF. We first argue that given a valid coloring defined by $\mathcal{L}$ with 2 colors and color number 2, for any $M \in \mathbb{N}$, the method of populating the relations given in the proof of Proposition 4.5 yields a database D with $|R_i(D)| \leq M$, such that $\mathrm{tw}\left(\langle R_1(D), \ldots, R_n(D)\rangle\right) \leq 1$, and $\mathrm{tw}(Q(D)) \geq M$. Indeed, for each u_j with $j \geq 1$, since the color number is 2 and there are at most 2 colors, $|\cup_{X\in u_j} \mathcal{L}(X)| \leq 1$; thus the Gaifman graph $G\left(\langle R_1(D), \ldots, R_n(D)\rangle\right)$ consists of a tree, with all values other than the null value $v_\emptyset$ either having no neighbors, or having a single neighbor $v_\emptyset$.

In the result of the query, each of the M values taken by a variable X with $\mathcal{L}(X) = \{1\}$ appears together with each of the M values taken by variables Y with $\mathcal{L}(Y) = \{2\}$ (since $|\cup_{X\in u_0} \mathcal{L}(X)| = 2$), and thus $G(Q(D))$ will contain the complete graph on M vertices, and will thus have treewidth at least $M-1$, and, in particular, can be made arbitrarily larger than the original treewidth.

For the other direction, if there is no valid coloring with 2 colors and color number 2, then for each pair of variables X, Y that appear in u_0, there must be some atom containing both X and Y. Thus every edge in the Gaifman graph $G(Q(D))$ will exist in the Gaifman graph $G(R_i(D))$, for some $i \in [n]$, and thus $\mathrm{tw}(Q(D)) \leq \mathrm{tw}\left(\langle R_1(D), \ldots, R_n(D)\rangle\right)$, as claimed. $\square$

5.3. Treewidth and Simple Functional Dependencies

We now extend the characterization of queries that preserve bounded treewidth to the setting with simple keys/functional dependencies via a reduction to the results of the previous section (Proposition 5.9). Our reduction analyzes the iterative process of removing simple functional dependencies given in the proof of Theorem 4.4, together with the bounds of Lemma 5.8 for the increase in treewidth resulting from a keyed join operation.

THEOREM 5.10. *Given a conjunctive query*

$$Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m),$$

with $i_j \in [n]$, and a set of arbitrary functional dependencies, $tw(Q(D))$ can not be bounded as any function of $tw(\langle R_1, \ldots, R_n \rangle)$, m, and $|u_i|$ if there exists a valid coloring of chase(Q) with 2 colors and color number 2. Furthermore, if all functional dependencies are simple, and there is no such coloring, then for any database D,

$$tw(Q(D)) \leq 2^{m|var(Q)|^2}\left(1 + \max\left(tw(\langle R_1(D), \ldots, R_n(D)\rangle), 2\right)\right) - 1.$$

PROOF. First observe that if there is a valid coloring with 2 colors and color number 2, the proof in the case without functional dependencies (Proposition 5.9) is still valid, and yields an instance with unbounded blowup in treewidth.

To show that the upper bound holds in the case that all functional dependencies are simple, and there is no coloring with 2 colors and color number 2, we will analyze the transformation of query Q into Q' given in the proof of Theorem 4.4. By Lemma 4.7, there is a coloring of *chase*(Q) with 2 colors and color number 2 if, and only if there is such a coloring of Q'.

Let Q_i be some intermediate query obtained in the process of eliminating functional dependencies, and Q_{i+1} derived from Q_i by eliminating the dependency $X \to Y$. Additionally, let D_i be a database compatible with Q_i, with relations $R_1^i(D_i), \ldots, R_n^i(D_i)$. We construct D_{i+1} as was done in the proof of Theorem 4.4, by appending an extra attribute to each relation R_j^i for which the corresponding atom contains variable X, and populating that atom via the uniquely determined value $y(x)$, determined by the functional dependency $X \to Y$ and the value x in a position corresponding to variable X. Now, define the relation $S(D_i)$, consisting of all tuples of the form $(x, y(x))$, with one tuple for each distinct value x that occurs in a position corresponding to variable X in any relation of D_i. For each j, either $R_j^{i+1}(D_{i+1}) = R_j^i(D_i)$, or $R_j^{i+1}(D_{i+1}) = R_j^i(D_i) \bowtie_{X=X} S(D_i)$, where the join is keyed. By Lemma 5.8, we have the following:

$$\begin{aligned} \text{tw}\left(\langle R_1^{i+1}(D_{i+1}), \ldots, \right. & \left. R_m^{i+1}(D_{i+1})\rangle\right) \\ &= \text{tw}\left(\langle R_1^i(D_i) \bowtie_{X=X} S(D_i), \ldots, R_m^i(D_i) \bowtie_{X=X} S(D_i))\rangle\right), \\ &\leq 2^m\left(1 + \max\left(\text{tw}\left(\langle R_1^i(D_i), \ldots, R_m^i(D_i), S(D_i)\rangle\right), 2\right)\right) - 1. \end{aligned}$$

Finally, noting that $\text{tw}(Q_{i+1}(D_{i+1})) \geq \text{tw}(Q_i(D_i))$, because the Gaifman graph of $Q_{i+1}(D_{i+1})$ contains that of $Q_i(D_i)$, from repeated application of these bounds for each of the at most $|var(Q)^2|$ functional dependencies that might occur, and Proposition 5.9, we have:

$$\begin{aligned} \text{tw}(Q(D)) &\leq \text{tw}(Q'(D')) \\ &\leq \text{tw}(\langle R_1'(D'), \ldots, R_n'(D')) \qquad \text{by Proposition 5.9,} \\ &\leq 2^{m|var(Q)|^2}\left(1 + \max\left(\text{tw}(\langle R_1(D), \ldots, R_n(Q)\rangle), 2\right)\right) - 1. \end{aligned}$$

□

6. EXTENSIONS TO GENERAL FUNCTIONAL DEPENDENCIES

In this section, we attempt to extend the size bounds of Section 4 to conjunctive queries with arbitrary functional dependencies. Proposition 4.5 proves that, even for arbitrary functional dependencies, the color number gives a lower-bound on the exponent of the maximum possible size increase of the results of a conjunctive query. Unfortunately, the matching upper bound does not hold in the setting with general functional dependencies; as we will show, however, even in this most general setting, the color number provides a characterization of those conjunctive queries that admit no size increase. Specifically, the number of tuples in the result of a query (with a set of arbitrary functional dependencies) can be larger than the number of tuples in the input relations if, and only if $C(chase(Q)) > 1$.

Beyond this characterization, in order to provide size bounds in this general setting we shall require some further machinery. The tools we have introduced thus-far have enabled us to analyze settings in which the worst-case size increase is realizable via databases that have particularly simple structures (in a manner which we will precisely characterize in Section 6.4). However, to establish size bounds for queries together with general functional dependencies, we must develop a more intimate understanding of the constraints that functional dependencies impose on database instances. Specifically, we shall use tools from information theory developed to analyze the precise interactions of multivariate distributions.

Our general size bound will take the form of a linear program with entropies as the variables, and the exponent of the worst-case size increase as the solution. It will be clear how functional dependencies can be encoded as constraints in the linear program. The difficulty is determining which additional constraints must be added to the linear program to ensure that the solution is realizable as a database instance. This question, as it turns out, is crucially related to an old and ongoing investigation at the heart of information theory: which entropy structures can be instantiated in multivariate distributions?

While we believe that providing an explicit characterization of the worst-case size increase is impossible without significant advances in information theory, in this section we provide an explicit upper bound, to go with the lower bound of $C(chase(Q))$ proven in Proposition 4.5. The upper bound can be viewed as a natural complement to the color-number bound, but as we exhibit, is not tight, and in fact has a super-constant gap. Finally, we provide an alternative characterization of the color number in terms of the solution to a linear program. This alternative characterization provides some insight into the entropy structure of the worst-case database instances in the settings without functional dependencies, and with simple functional dependencies.

We start this section by proving that the results of a conjunctive query with an arbitrary set of functional dependencies can be larger than those of the inputs if, and only if $C(chase(Q)) > 1$. In Section 6.2 we establish the connection between entropy and worst-case size increases. In Section 6.3, we provide definitions of the basic information theory quantities, and define the Shannon information inequalities. In Section 6.4, we prove our linear program size bound, provide an alternative definition for the color number in terms of entropies, and exhibit a construction showing a super-constant gap between our upper and lower bounds.

6.1. Characterization of Size-Preserving Queries

THEOREM 6.1. *Given a query $Q = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$ with arbitrary functional dependencies, there exists a database D such that $|Q(D)| > rmax(D)$ if and only if $C\left(chase(Q)\right) > 1$. Furthermore, $C(chase(Q)) > 1$ implies that $C(chase(Q)) \geq \frac{m}{m-1}$.*

PROOF. By Proposition 4.5, if $C(chase(Q)) > 1$, then there exists a database D with $|Q(D)| > \text{rmax}(D)$, thus it suffices to prove the other direction.

Consider a query Q and set of functional dependencies for which there is an instance with $|Q(D)| > \text{rmax}(D)$. For the remainder of the proof, for ease of notation, assume that $Q = chase(Q)$ (otherwise we simply apply these arguments to $chase(Q)$). Given such an instance, we will explicitly construct a valid coloring with color number greater than 1. First, let $Q' = R_0(var(Q)) \leftarrow R_{i_1}(u_1)\wedge, \dots, \wedge R_{i_m}(u_m)$ be the query identical to Q except whose output relation contains all the variables. Define the table T of arity $|var(Q)|$, such that each tuple in T is contained in $Q'(D)$, and such that there is a bijection between tuples of T and tuples of $Q(D)$; thus $|T| = |Q(D)|$. Table T is simply the extension of $Q(D)$ to include attributes corresponding to the variables in $var(Q)$ that are not in u_0. Note that $\text{rmax}(D) \geq \max_{i\in[m]} |\pi_{u_i}(T)|$, where $\pi_{u_i}(T)$ denotes the projection of T onto the positions corresponding to variables in u_i.

We will now use table T to construct a coloring $\mathcal{L}$ with color number at least $\frac{m}{m-1}$. The construction of the coloring proceeds as follows: for each $i \in [m]$, by the pigeonhole principle there must be some subset of the tuples, $S_i \subset T$, with $|S_i| = 2$ such that $|\pi_{u_i}(S_i)| = 1$. For each variable $X \in var(Q)$, we "mark" it with an i if $|\pi_X(S_i)| = 2$. (Note that no $X \in u_i$ will be marked with an i, since, by definition, $|\pi_{u_i}(S_i)| = 1$.) After completing these markings for all $X \in var(Q)$, we simply assign colors according to the markings: if variable X has not been marked, then $\mathcal{L}(X) := \emptyset$. Otherwise, if X has been marked with indices $j_1, \dots, j_\ell$, we set $\mathcal{L}(X) := \{j_1, \dots, j_\ell\}$.

We now argue that this coloring is valid. Consider a functional dependency $X_1 \dots X_h \to Y$. If $j \in \mathcal{L}(Y)$, then $|\pi_Y(S_j)| = 2$, and thus in the two tuples of S_j, position Y takes on two distinct values. The functional dependency implies that at least one of $X_1, \dots, X_h$ must also take on two distinct values in these tuples, and thus at least one of these variables will also be marked with a j, and therefore $j \in \bigcup_{i\in[h]} \mathcal{L}(X_i)$, and the functional dependency is obeyed by the coloring defined by $\mathcal{L}$.

Finally, since for all $i \in [m]$, $|\pi_{u_0}(T)| = |T| > |\pi_{u_i}(T)|$, for each $i \in [m]$, i will mark at least one of the variables X in u_0, and thus $|\bigcup_{X\in u_0} \mathcal{L}(X)| = m$. As noted above, for each $i \in [m]$, $i \notin \bigcup_{X\in u_i} \mathcal{L}(X)$, and thus $\mathcal{L}$ is a valid coloring with m colors appearing in the final projection variables, with at most $m-1$ appearing together in any u_i, thus $C(Q) \geq \frac{m}{m-1}$, as desired. □

6.2. Entropy Size Bounds

To see the connection between entropy and worst-case size increases, consider a conjunctive query $Q = R_0(u_0) \leftarrow R_1(u_1) \wedge \dots \wedge R_n(u_n)$, and database D, such that $|Q(D)| = \text{rmax}(D)^c$. Let $Q' = R'_0(var(Q)) \leftarrow R_1(u_1) \wedge \dots \wedge R_n(u_n)$, and define the distribution $\mathcal{D}$ over the tuples of $Q'(D)$ to be such that the marginal distribution $\mathcal{D}_{u_0}$ over the values of the $|u_0|$-tuples corresponding to variables in u_0 is the uniform distribution. Note that such a choice for $\mathcal{D}$ is not necessarily unique, unless $u_0 = var(Q)$. Let $H_{\mathcal{D}}(u_i)$ denote the entropy of the marginal distribution of the values in positions labeled by the variables of u_i. Observe that for any $i \in \{1, \dots, n\}$,

$$\frac{H_{\mathcal{D}}(u_0)}{H_{\mathcal{D}}(u_i)} \geq \frac{H_{\mathcal{D}}(u_0)}{H_{unif_i}(u_i)} = \frac{\log(|Q(D)|)}{\log(|R_i(D)|)} \geq c, \tag{2}$$

where $unif_i$ is the uniform distribution over the tuples of $R_i(D)$. This provides the motivation for the form that our linear programs will take: maximizing the entropy $H_{\mathcal{D}}(u_0)$ while bounding the entropies of each $H_{\mathcal{D}}(u_i)$.

6.3. Conditional Entropy and Information Measures

In this section, we state the basic definitions of *conditional entropy* and *information measures*, and then state some facts about Shannon and non-Shannon information inequalities, which we will use in the following sections.

Definition 6.2. For discrete random variables X, Y with respective supports $\mathcal{X}, \mathcal{Y}$, the conditional entropy of X given Y, denoted by $H(X|Y)$ is given by

$$H(X|Y) := \sum_{y\in\mathcal{Y}} p(y)H(X|Y=y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\left(p(x|y)\right).$$

The following fact follows from these definition:

FACT 6.3. *For discrete random variables X, Y with respective supports $\mathcal{X}, \mathcal{Y}$,*

$$H(X,Y) = H(X) + H(Y|X).$$

Definition 6.4. For discrete random variables X, Y, as shown, the *mutual information* between X and Y is

$$I(X;Y) := \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}.$$

The following fact follows from the previous definition:

FACT 6.5. *For discrete random variables X, Y,*

$$I(X;Y) = I(Y;X) = H(X) + H(Y) - H(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

Definition 6.6. For discrete random variables $X_1, \ldots, X_n$ with respective supports $\mathcal{X}_1, \ldots, \mathcal{X}_n$, and $n \geq 3$, we recursively define their mutual information as

$$I(X_1;\cdots;X_n) = I(X_1;\cdots;X_{n-1}) - I(X_1;\cdots;X_{n-1}|X_n),$$

where the *conditional mutual information* is defined as

$$I(X_1;\cdots;X_{n-1}|X_n) = \sum_{x_n\in\mathcal{X}_n} p(x_n)\, I(X_1;\cdots;X_{n-1}|X_n = x_n),$$

and for $n = 2$ the mutual information is as defined in Definition 6.4.

Unsurprisingly, the above information measures have a set-theoretic structure, and can be represented in an *information diagram*, from which basic relations between information measures can be easily read off. Figure 2 illustrates a general information diagram for three variables. The following basic facts follow from the previous definitions, and can easily be seen by considering the associated information diagram. (We refer the reader to Chapter 3 of Yeung [2008] for proofs of these facts and a rigorous definition of the set-theoretic structure of information measures.)

FACT 6.7. *For discrete random variables $X_1, \ldots, X_n$, and any disjoint sets K, $K' \subseteq [n]$,:*

$$H(X_K|X_{K'}) = \sum_{S:S\cap K\neq\emptyset,S\cap K'=\emptyset} I(S|X_{[n]-S}),$$

$$I(K|X_{K'}) = \sum_{S:S\supseteq K,S\cap K'=\emptyset} I(S|[n]-S),$$

where $I(S|X_{S'})$ denotes $I(X_1;\cdots;X_j|X_{S'})$, for $S = [j]$. Note that we avoid the notation $I(X_S|X_{S'})$, which has the interpretation of $I(X_1,\ldots,X_j|X_{S'}) = H(X_S|X_{S'})$.

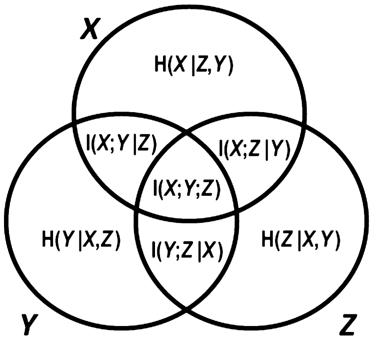

Fig. 2. The generic information diagram of X, Y, Z. Note that the set-theoretic properties of these information measures allows various information equalities to be read off from such a diagram; for example, $I(X; Y) = I(X; Y; Z) + I(X; Y|Z)$, and $H(Z) = I(X; Y; Z) + I(X; Z|Y) + I(Y; Z|X) + H(Z|X, Y)$.

We now define the basic information inequalities.

Definition 6.8. For discrete random variables $X_1, \ldots, X_n$ as above, and for a subset $K \subseteq [n]$, denote the tuple of all X_i for $i \in K$ by X_K, the *Shannon information inequalities* consist of all inequalities of the form

$$H(X_i|X_{[n]-\{i\}}) \geq 0,$$

for all $i \in [n]$, and

$$I(X_i; X_j|X_K) \geq 0,$$

for all $i \neq j \in n$ and $K \subseteq [n] - \{i, j\}$.

We note that, as above, the mutual information expressions can be reexpressed in terms of entropies. For example, $I(X_i; X_j|X_K) = H(X_i|X_K) - H(X_i|X_j, X_K) = H(X_i, X_K) + H(X_j, X_K) - H(X_K) - H(X_i, X_j, X_K)$. (See Yeung [2008], Chapter 14 for further discussion of the Shannon inequalities.)

The Shannon information inequalities are well understood and were initially hypothesized to essentially capture the space of valid entropy configurations. However, in a breakthrough work in 1998, Zhang and Yeung [1998] showed that there are fundamental constraints on this space that are not captured by the Shannon inequalities, even for four random variables. This accounts for the lack of tightness in our upper bound.

6.4 Size Bounds for General Functional Dependencies

We are now equipped to give our linear programming upper bound for the worst-case size increase. Throughout this section, we admit a slight abuse of notation, and refer to the entropy of a set of attributes of a database, interpreted in the natural way: given a database table with attribute set $A = \{X_1, \ldots, X_k\}$, some fixed probability distribution $\mathcal{D}$ over the tuples of the table, and two subsets $S, S' \subseteq A$, we refer to the conditional entropy $H_{\mathcal{D}}(S|S')$ where S, S' respectively are interpreted to be the discrete random variables whose possible values consist of the $|S|$, respectively $|S'|-$tuples of values that the corresponding variables have in the tuples of the database table, with probabilities given according to $\mathcal{D}$.

PROPOSITION 6.9. *Given a query $Q = chase(Q) = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$, with $i_j \in [n]$ and $var(Q) = \{X_1, \ldots, X_k\}$, and a set of arbitrary functional dependencies, for any database D,*

$$|Q(D)| \leq rmax(D)^{s(Q)},$$

where $rmax(D) := \max_{i \in n} |R_i(D)|$, and $s(Q)$ is the solution to the following linear program:

$$\begin{array}{lll} \textit{maximize} & h(u_0) & \\ \textit{subject to} & h(u_j) \leq 1 & \forall j \in [m] \\ & h(x_t | x_{j_1}, \ldots, x_{j_\ell}) = 0 & \textit{for each f.d. } X_{j_1} \ldots X_{j_\ell} \to X_t \\ & h(x_j | x_{[k]-\{j\}}) \geq 0 & \forall j \in [k] \\ & I(x_j; x_\ell | x_S) \geq 0 & \forall j, \ell \in [k] \textit{ and } S \subseteq [k] - \{j, \ell\}, \end{array}$$

where the variables of the linear program are the (unconditional) entropies $h(x_S)$ for all $S \subseteq [k]$, and the expressions involving mutual information or conditional entropies appearing in the constraints are implicitly considered to stand in for the corresponding linear expressions of these variables (as described in Section 6.3).

PROOF. Let $\mathcal{D}$ denote the distribution over $|var(Q)|$-tuples whose marginal distribution $\mathcal{D}_{u_0}$ over the variables in u_0 is uniform, as defined in Section 6.2. To see that the value of the above linear program provides an upper bound on $\frac{\log(|Q(D)|)}{\log(|R_i(D)|)}$, note that for any set $S \subseteq [k]$, the quantity $\frac{H_{\mathcal{D}}(S)}{\max_{i \in [m]} H_{\mathcal{D}}(u_i)}$ must satisfy the analogs of all the constraints that the corresponding variable $h(S)$ is subject to in the linear program, including the last two sets of constraints that represent the Shannon information inequalities. Thus, the assignments $h(S) := \frac{H_{\mathcal{D}}(S)}{\max_{i \in [m]} H_{\mathcal{D}}(u_i)}$ yields a point in the feasible region of the linear program, and thus by Eq. (2) the value of the solution to the linear program must be at least $\frac{\log(|Q(D)|)}{\log(|R_i(D)|)}$. □

In order to make the size bound given by the solution to the linear program of Proposition 6.9 tight, we would need to add additional constraints so as to enforce the *non-Shannon* information inequalities. Unfortunately, it was recently shown that even for just four variables, there are infinitely many independent such inequalities [Matúš 2007a].

We now reexamine the color number in an effort to better understand the types of entropy structures that it can capture. As the following proposition shows, the color number can be defined via the linear program of Proposition 6.9 with some additional constraints on the entropies. In particular, we require extra constraints that enforce that all mutual information measures be nonnegative. Note that the Shannon inequalities imply that all mutual information measures of two variables be nonnegative; however, as Figure 3 depicts, the mutual information of more than two variables can be negative.

PROPOSITION 6.10. *Given a query $Q = chase(Q) = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$, with $var(Q) = \{X_1, \ldots, X_k\}$, and a set of arbitrary functional dependencies, $C(Q)$ is equal to the solution to the following linear program:*

$$\begin{array}{lll} \textit{maximize} & h(u_0) & \\ \textit{subject to} & h(u_i) \leq 1 & \forall i \geq 1 \\ & h(x_t | x_{j_1}, \ldots, x_{j_\ell}) = 0 & \textit{for each f.d. } X_{j_1} \ldots X_{j_\ell} \to X_t \\ & I(x_{j_1}; , \ldots ; x_{j_\ell} | x_{[k]-\{j_1, \ldots, j_\ell\}}) \geq 0 & \forall \textit{ sets } \{j_1, \ldots, j_\ell\} = S \subseteq [k], \end{array}$$

where the variables of the linear program are the (unconditional) entropies $h(x_S)$ for all $S \subseteq [k]$, and the expressions involving mutual information or conditional entropies appearing in the constraints are implicitly considered to stand in for the corresponding linear expressions of these variables (as described in Section 6.3).

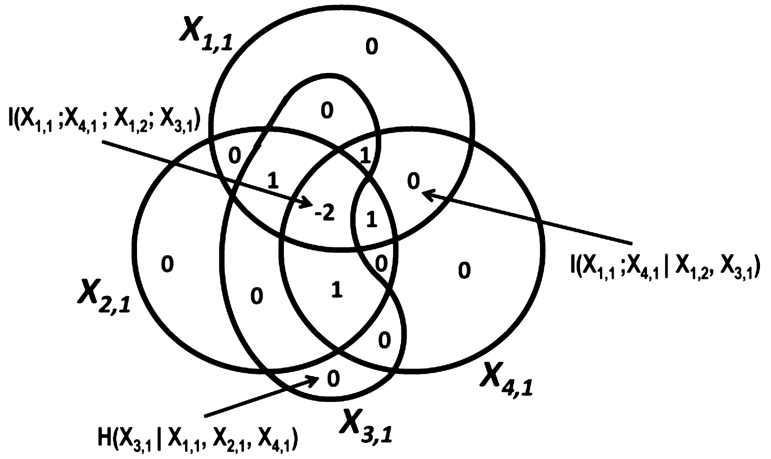
X1,1
X2,1
X3,1
X4,1
I(X1,1 ;X4,1 ; X1,2; X3,1)
I(X1,1 ;X4,1 | X1,2, X3,1)
H(X3,1 | X1,1, X2,1, X4,1)
0
0
0
1
1
-2
1
0
0
0
0
1
0
0
0

Fig. 3. The information diagram of $X_{1,1}, \ldots, X_{4,1}$ in our construction for $k = 4$. Note that any set of size 2 or more contains all the entropy of all four variables. The negative mutual information $I(X_{1,1}; X_{1,2}; X_{1,3}; X_{1,4}) = -2$ suggests that no valid coloring can closely approximate the entropy structure; this intuition is leveraged in our construction to yield a super-constant gap between the color number and the exponent of the worst-case size increase.

PROOF. We first show that given any valid coloring achieving color number $C(Q)$, we can find a feasible point for the linear program with value $C(Q)$. Given a valid coloring defined by $\mathcal{L}$, such that $\max_{i\in[m]} |\bigcup_{X\in u_i} \mathcal{L}(X)| \leq r$, for every set $S \subseteq [k]$, we set

$$I(S|x_{[k]-S}) = \frac{|\bigcap_{i\in S} \mathcal{L}(X_i) - \bigcup_{i\notin S} \mathcal{L}(X_i)|}{r},$$

where $I(S|x_{[k]-S})$ denotes $I(x_{j_1}; \cdots ; x_{j_\ell}|x_{[k]-S})$, with $S = \{j_1, \ldots, j_\ell\}$. Note that these 2^k mutual information values are sufficient to determine the values of all variables in the linear program. In particular, these 2^k mutual information measures are the values that would appear in an information diagram. From Fact 6.7, for any disjoint sets $T, T' \subseteq [k]$, we will now express $I(T|x_{T'})$ in terms of the color labels. We note that for distinct sets S_1, S_2, the corresponding sets of labels $\bigcap_{i\in S_j} \mathcal{L}(X_i) - \bigcup_{i\notin S_j} \mathcal{L}(X_i)$ will be disjoint, because these sets consist of exactly those colors appearing in the labels of each element of S_j and not in any of the labels of elements not in S_j. Thus, the sum in Fact 6.7 may be expressed in terms of the size of the union of these sets for S containing T and disjoint from T'. This union consists of exactly those colors appearing in the labels of each element of T and not in any of the labels of elements of T', yielding:

$$I(T|x_{T'}) = \frac{|\bigcap_{i\in T} \mathcal{L}(X_i) - \bigcup_{i\in T'} \mathcal{L}(X_i)|}{r}.$$

It is now easy to see that this construction yields a feasible point for the linear program. First, observe that all the information inequalities are trivially satisfied, since for every set $S \subseteq [k]$, $I(S|x_{[k]-S}) \geq 0$ in our construction. To see that the equality constraints given by the functional dependencies are observed, note that the dependency $X_1, \ldots, X_j \to X_{j+1}$ implies that $\mathcal{L}(X_{j+1}) - \bigcup_{i\in[j]} \mathcal{L}(X_i) = \emptyset$, and thus in this assignment, $I(x_{j+1}|x_{[j]}) = 0$, as desired. (Note that, by definition, $h(x_{j+1}|x_{[j]}) = I(x_{j+1}|x_{[j]})$.) Finally, to see that the first set of constraints are observed, note that for any $j \leq k$, $h(x_{[j]}) = \sum_{S \text{ s.t. } S\cap[j]\neq\emptyset} I(S|x_{[j]-S})$, which, by our construction, is precisely $\frac{|\bigcup_{i\in[j]} \mathcal{L}(X_i)|}{r}$, which is bounded by 1 whenever S is the index set of an atom occurring in the query body, and which will equal $C(Q)$ when S is the index set of u_0 by the definition of the color number.

For the other direction, given a rational feasible point for the linear program with objective function value v, where all variables have values r_i/q, for integers r_i, q, with q being the common denominator, we will construct a coloring with color number $C(Q)$. The final set of constraints of the linear program imply that for any set $S \subseteq [k]$, $I(S|x_{[k]-S}) = \frac{r_S}{q} \geq 0$. Furthermore, since our feasible point is rational, $r_S \in \mathbb{N}$. To populate our coloring, we begin with the empty coloring, and then for each $S \subseteq [k]$, we add $q \cdot I(S|x_{[k]-S})$ unique colors to the labels of all X_i for which $i \in S$. To see that this coloring obeys the functional dependencies, note that for $X_1 \dots X_j \to X_{j+1}$, we have that $I(x_{j+1}|X_{[j]}) = 0$, and thus by Fact 6.7, for any $S \subseteq [k] - [j]$ such that $j+1 \in S$, $I(S|X_{[k]-S}) = 0$, from which it follows that in our construction $\mathcal{L}(X_{j+1}) \subseteq \bigcup_{i \in [j]} \mathcal{L}(X_i)$. Finally, to see that the color number is at least the value v, of the linear program, note that by Fact 6.7, a total of

$$\sum_{S \subseteq [k] \text{ s.t. } S \cap K \neq \emptyset} q \cdot I(S|X_{[k]-S}) = q \cdot h(X_S)$$

unique colors is assigned to each set X_S, and thus the color number is at least $h(u_0)$, as desired. □

Leveraging the understanding of the entropy structures that are compatible with the color number given by the previous proposition, we now show that there is a super-constant gap between the true worst-case size increase, and the color number (in the case of general functional dependencies). We show this by exhibiting a family of queries, and associated databases whose color numbers fall short of their true size increase by a superconstant factor (in the exponent). This family is a generalization of a construction suggested to us by Daniel Marx.

PROPOSITION 6.11. *For any constant $\alpha \in \mathbb{R}$, there exists a conjunctive query Q and set of functional dependencies, and database D such that*

$$|Q(D)| > rmax(D)^{\alpha\, C(chase(Q))}.$$

PROOF. We shall construct a family of queries, and associated databases whose color numbers fall short of the true size increase by a superconstant factor. In particular, for any even integer k, and prime number $N > k$, we will construct a query Q with a set of functional dependencies, and database D such that $\text{rmax}(D) = N^{k/2}$, $|Q(D)| = N^{k^2/4}$, yet $C(chase(Q)) = 2$.

Fix an even integer k, and consider the following query Q over $k^2/2$ variables $X_{i,j}$, for $i \in \{1, \dots, k\}$, and $j \in \{1, \dots, k/2\}$:

$$Q = R(X_{1,1}, \dots, X_{i,j}, \dots, X_{k,k/2}) \leftarrow \bigwedge_{j=1}^{k/2} R_i(X_{1,j}, \dots, X_{k,j}) \wedge \bigwedge_{i=1}^{k} T_i(X_{i,1}, \dots, X_{i,k/2}).$$

Additionally, for each $j \in \{1, \dots, k/2\}$ we impose the following functional dependencies: given any set $S \subseteq \{X_{1,j}, \dots, X_{k,j}\}$, with $|S| \geq k/2$, for any $i \in [k]$,

$$S \to X_{i,j}.$$

Intuitively, this construction has $k/2$ groups of k variables (indexed by $j \in [k/2]$, above), such that amongst any group, any set of $k/2$ of those variables suffice to recover the remaining $k/2$ variables in that group. The information diagram of one group of the construction in the case $k = 4$ is depicted in Figure 3.

Given any prime integer $N > k$, we will construct a database D such that for all $i \in [k]$, $j \in [k/2]$, $|R_j(D)| = N^{k/2} = |T_i(D)|$. Additionally, the values assigned to positions labeled by $X_{i,j}$ and $X_{i',j'}$ will be disjoint whenever $j \neq j'$; i.e. the values assigned to each of the $k/2$ groups are disjoint. Each of the $N^{k/2}$ tuples of $R_j(D)$ will be constructed so as to be Shamir $(k/2, k)$ secret shares [Shamir 1979]. That is, given the values of any $k/2$ attributes $X_{1,j}, \ldots, X_{k/2,j}$, the values of the remaining $k/2$ attributes can be uniquely determined, and for $S \subseteq \{X_{1,j}, \ldots, X_{k,j}\}$,

$$|\pi_S(R_i(D))| = N^{\min(|S|, \frac{k}{2})}.$$

Explicitly, let $\{p_1, \ldots, p_{N^{k/2}}\}$ be the set of all $N^{k/2}$ polynomials of degree at most $\frac{k}{2} - 1$ over the finite field of N elements. We populate the mth tuple of $R_j(D)$ by, for each $i \in [k]$, letting the ith position (corresponding to variable $X_{i,j}$) take value $p_m(i-1)$, namely the evaluation of the mth polynomial at the value $x = i - 1$. Additionally, we add a marker j to each value, so that the set of symbols used to populate $R_j(D)$ and $R_{j'}(D)$ are disjoint for $j \neq j'$; for example, whenever we want to put a "7" in a tuple of $R_j(D)$, we instead put the symbol "$7j$", and if we want to put a "7" in a tuple of $R_\ell(D)$, we put the symbol "7ℓ".

Since each polynomial p_m is uniquely defined by its evaluation on any $k/2$ distinct points, the constructed relations $R_j(D)$ satisfy the functional dependencies, and additionally, for any $S \subseteq \{X_{1,j}, \ldots, X_{k,j}\}$, $|\pi_S(R_j(D))| = N^{\min(|S|, \frac{k}{2})}$.

We now define the $T_i(D)$. Let

$$Q' = R'(X_{1,1}, \ldots, X_{i,j}, \ldots, X_{k,k/2}) \leftarrow \bigwedge_{j=1}^{k/2} R_j(X_{1,j}, \ldots, X_{k,j}),$$

and for each $i \in [k]$, let $T_i(D) := \pi_{X_{i,1},\ldots,X_{i,k/2}}(Q'(D))$ be the projection of the total join (the complete cartesian product) of the $R_j(D)$s onto the attribute list $(X_{i,1}, ..., X_{i,k/2})$. Thus, $|T_i(D)| = N^{k/2}$. We remark that given the above construction, the addition of the T_is to the query does not affect the result of the query. The sole purpose of the T_is in the query is to allow us to bound the color number—as we will argue, at least one of the T_is must have attributes that have "lots" of colors, thereby ensuring that $C(Q)$ is small.

Since $Q(D)$ consists of the complete join (the complete cartesian product) of each R_j, $|Q(D)| = \left(N^{k/2}\right)^{k/2} = N^{k^2/4}$, whereas the size of the largest input relation is $\text{rmax}(D) = N^{k/2}$. We now show that $C(chase(Q)) = C(Q) \leq 2$, which will complete our proof of the proposition.

First, observe that it suffices to consider the case that for $j \neq j'$, $\mathcal{L}(X_{i,j}) \cap \mathcal{L}(X_{i',j'}) = \emptyset$, because, assuming otherwise, for any color c that lies in the intersection, by removing the color c from the labels $\mathcal{L}(X_{i'',j})$ for all $i'' \in [k]$, we still have a valid coloring (since there are no functional dependencies involving both $X_{i,j}$ and $X_{i',j'}$ for $j \neq j'$), and the new color number can only have increased or remained the same. Let $r_j = |\bigcup_{i=1}^{k} \mathcal{L}(X_{i,j})|$, and $t_i = |\bigcup_{j=1}^{k/2} \mathcal{L}(X_{i,j})| = \sum_{j=1}^{k/2} |\mathcal{L}(X_{i,j})|$ denote the number of colors assigned to the variables of each atom occurring in the query body. Thus, in an optimal coloring defined by such a labeling $\mathcal{L}$, we have

$$|\bigcup_{i,j} \mathcal{L}(X_{i,j})| = \sum_{i=1}^{k/2} |\bigcup_{j=1}^{k} \mathcal{L}(X_{j,i})| = \sum_{i=1}^{k/2} r_i.$$

Next, observe that each element of $\mathcal{L}(X_{i,j})$, must occur in the labels of at least $k/2$ other variables $X_{i',j}$; if this were not the case, then there would exist a set $S \subseteq \{X_{1,j}, \ldots, X_{k,j}\}$

of size $|S| \geq k/2$, such that $\mathcal{L}(X_{i,j}) \not\subseteq \bigcup_{X_{i',j} \in S} \mathcal{L}(X_{i',j})$, which violates one of the functional dependencies. Thus, it follows that

$$\sum_{i=1}^{k} |\mathcal{L}(X_{i,j})| \geq \frac{k}{2} r_j.$$

To conclude, putting the above equations together, we have

$$\sum_{i=1}^{k} t_i = \sum_{i,j} |\mathcal{L}(X_{i,j})| \geq \frac{k}{2} \sum_{j=1}^{k/2} r_j,$$

and thus there must be at least one $i \in [k]$ such that $t_i \geq \frac{(k/2)\sum_{j=1}^{k/2} r_j}{k} = \frac{1}{2}\sum_{j=1}^{k/2} r_j$, and thus $C(Q) \leq 2$. □

As a final remark, we note that the jump in difficulty of establishing tight size bounds occurs when the left-hand sides of functional dependencies go from having single variables, to having 2 variables. It is not hard to show that any size bounds for the case where functional dependencies have left-hand sides with at most two variables can be extended to work for arbitrary functional dependencies, via the following fact.

FACT 6.12. *Given a query $Q = chase(Q)$ and set of functional dependencies, there exists a query Q' with the following properties:*

— *each functional dependency of Q' has at most two variables on its left-hand side,*
— $Q' = chase(Q')$,
— *the set of functional dependencies of Q' is at most polynomially larger than that of Q,*
— *the description of Q' is at most polynomially larger than that of Q,*
— *the worst-case size increase of Q and Q' are identical.*
— $C(Q) = C(Q')$.

PROOF. We shall iteratively remove functional dependencies from Q that have 3 or more variables occurring on their left-hand sides, via the addition of a (polynomial number) of additional variables, relations, and functional dependencies.

Given a functional dependency $X_1 \cdots X_k \to Y$, we add a relation $R(X_1, X_2, Z)$, with the new variable Z, together with the functional dependencies $X_1X_2 \to Z, Z \to X_1, Z \to X_2$. We then add the relation $R'(Z, X_3, \cdots, X_k, Y)$, together with the functional dependency $ZX_3 \cdots X_k \to Y$. Finally, we remove the functional dependency $X_1 \cdots X_k \to Y$ from the set of functional dependencies.

Iteratively applying the previous procedure until there are no more functional dependencies (other than implied ones) with more than two variables on their left-hand sides clearly results in a query Q' with at most a polynomially longer description, and polynomially more functional dependencies. Additionally, since all new relations are distinct, and all original functional dependencies are implied by the new set of functional dependencies, $chase(Q') = Q'$. To see that the size increase of Q' is the same as that of Q, note after each single iteration of the previous procedure, the size increase must remain unchanged, as the values taken by variables X_1, X_2 dictate that taken by Z, and vice-versa, defining a $1:1$ mapping between tuples of $Q(D)$ and tuples of the result of the query generated after one step of the procedure. To conclude, there is a natural mapping between valid colorings of Q, and the query obtained after one step of the above procedure, namely for the step described previously, $\mathcal{L}(Z) \leftrightarrow \mathcal{L}(X_1) \cup \mathcal{L}(X_2)$. □

7. COMPLEXITY CONSIDERATIONS

In this section, we consider the computational complexity of computing the bounds on the size increase, and deciding whether the treewidth must have a bounded blowup.

PROPOSITION 7.1. *Given a conjunctive query Q with a set of simple functional dependencies, the maximum size increase, $C(chase(Q))$, can be computed in time polynomial in $|Q|$. Additionally, whether or not the treewidth is bounded (whether $chase(Q)$ has a coloring with 2 colors achieving color number 2) can be decided in polynomial time.*

PROOF. First, note that $chase(Q)$ can be computed in polynomial [Aho et al. 1979b]. The proof of Theorem 4.4 gives a polynomial-time reduction from $chase(Q)$ to Q', a query without functional dependencies and the property that there is a coloring of Q' with k colors achieving color number r if, and only if there is such a coloring of $chase(Q)$. Our claim follows from noting that $C(Q')$ is given as the solution to a polynomial-sized LP, and from noting that deciding whether there is a coloring with 2 colors and color number 2 is equivalent to deciding whether every pair of output variables occurs together in some atom, which can be checked efficiently. □

As our next theorem shows, even for queries Q with arbitrary functional dependencies, it is possible to efficiently decide whether $C(Q) > 1$. The proof reduces the question at hand to the satisfiability of a sequence of tractable SAT instances—one for relation in the query body.

THEOREM 7.2. *Given a conjunctive query Q with an arbitrary set of functional dependencies, it can be decided in polynomial time whether the results of Q can be larger than the input relations, or, equivalently, whether $C(chase(Q)) > 1$.*

PROOF. Since $chase(Q)$ can be computed in polynomial time, it suffices to consider the case that $Q = chase(Q) = R_0(u_0) \leftarrow R_{i_1}(u_1) \wedge \cdots \wedge R_{i_m}(u_m)$. First observe that a necessary and sufficient condition for $C(Q) > 1$ is the existence of some coloring $\mathcal{L}$ such that for each $i \in [m]$, there is a color c such that $c \in \bigcup_{X_j \in u_0} \mathcal{L}(X_j)$, but $c \notin \bigcup_{X_j \in u_i} \mathcal{L}(X_j)$. We will represent this condition as a set of m tractable SAT expressions, one for each atom occurring in the query body, as follows: Our set of SAT variables will be $\{x_1, \ldots, x_{|var(Q)|}\}$, in natural correspondence with the set of query variables $var(Q) = \{X_1, \ldots, X_{|var(Q)|}\}$.

From Fact 6.12, it suffices to prove our theorem in the case that all functional dependencies have at most two variables on their left-hand sides. Given p functional dependencies $X_{j_1}X_{k_1} \rightarrow X_{h_1}, \ldots, X_{j_p}X_{k_p} \rightarrow X_{h_p}$, our SAT expression for atom u_i will have the form

$$SAT_i = \bigwedge_{X_j \in u_i} \neg x_j \wedge \left(\bigvee_{X_j \in u_0} x_j \right) \wedge (x_{j_1} \vee x_{k_1} \vee \neg x_{h_1}) \wedge \cdots \wedge (x_{j_p} \vee x_{k_p} \vee \neg x_{h_p}).$$

Any satisfying assignment of SAT_i yields a valid coloring of Q that uses exactly 1 color, and has the property that no variable in u_i has a color, but at least one variable in u_0 has a color; such a coloring is given by assigning all variables that are set to $FALSE$ to not have the color, and all variables set to $TRUE$ to have the color. To see this, note that the first part of SAT_i ensures that no variable occurring in u_i can be $TRUE$ in a satisfying assignment; the second part of SAT_i ensures that at least one variable in the output projection will be colored, and the third part of SAT_i ensures that the functional dependencies are respected. Since any set of valid colorings can be combined to yield a valid coloring (by letting $\mathcal{L}_{1,2}(X_i) := \mathcal{L}_1(X_i) \cup \mathcal{L}_2(X_i)$), it follows that if, for all $i = 1, \ldots, m$, SAT_i is satisfiable, then there exists a coloring with m colors, yielding $C(Q) \geq \frac{m}{m-1} > 1$. Conversely, if, for some i, SAT_i is not satisfiable, then there is no valid coloring of the variables in which a color appears in the label of some output variable but not in the label of a variable of u_i, in which case $C(Q) = 1$.

What remains is to verify that SAT_i can be solved efficiently. Since SAT_i is a conjunction of clauses where each clause contains at most one negated literal, it is a (dual-)Horn formula, whose satisfiability can be decided via a linear-time algorithm (see, e.g., Dowling and Gallier [1984]). □

While we do not have a characterization of those queries with arbitrary functional dependencies that have bounded treewidth, we did show that if such a query has a coloring with 2 colors achieving color number 2, then its treewidth is not bounded (Theorem 5.10). We now show that it can be NP-hard to decide whether there exists such a 2-coloring. The proof is a reduction from 3-SAT, and, intuitively, constructs disjunctions from dependencies of the form $XY \to Z$.

PROPOSITION 7.3. *Given a conjunctive query Q, and set of functional dependencies, it is NP-complete to decide if there is a valid coloring with* 2 *colors that achieves color number* 2*, even if each functional dependency has at most two variables on its left-hand side.*

PROOF. We reduce from 3-SAT. Given a 3-SAT expression E, over the literals $x_1, \ldots, x_n, \bar{x_1}, \ldots, \bar{x_n}$, consider the query $Q(A, B) \leftarrow V_1 \wedge \cdots, V_n \wedge C_1 \wedge \cdots \wedge C_k$, where, for each variable x_i in expression E we have one expression

$$V_i := R_{i,1}(X_i, \bar{X}_i, A) \wedge R_{i,2}(Y_i, \bar{Y}_i, B) \wedge R_{i,3}(X_i, Y_i) \wedge R_{i,4}(\bar{X}_i, \bar{Y}_i),$$

and for each clause in E we have an atom C_i of the form

$$C_i := S_i(X_j, X_k, X_m, A),$$

with the clause $(x_1 \vee \bar{x_2} \vee x_3)$ of E being mapped to $S(X_1, \bar{X}_2, X_3, A)$, for example. To complete the setup, we define the following functional dependencies that fall into two types:

— $X_i\bar{X}_i \to A$, and $Y_i\bar{Y}_i \to B$,
— and the functional dependencies implies by having the first three attributes of each S_i be a (compound) key for the fourth attribute of S_i.

We claim that the 3-SAT expression E is satisfiable if, and only if, the associated query Q has a coloring with 2 colors and color number greater than 1.

Given a satisfying assignment to the 3-SAT expression E, consider the labeling $\mathcal{L}$ defined as follows: set $\mathcal{L}(A) = \{1\}$, $\mathcal{L}(B) = \{2\}$, and for each variable x_i, if x_i is $TRUE$, then set $\mathcal{L}(X_i) = \{1\}$, $\mathcal{L}(\bar{X}_i) = \emptyset$, $\mathcal{L}(Y_i) = \emptyset$, $\mathcal{L}(\bar{Y}_i) = \{2\}$. If x_i is $FALSE$, then set $\mathcal{L}(X_i) = \emptyset$, $\mathcal{L}(\bar{X}_i) = \{1\}$, $\mathcal{L}(Y_i) = \{2\}$, $\mathcal{L}(\bar{Y}_i) = \emptyset$. Such a coloring clearly satisfies all functional dependencies of the first kind and has color number 2; since E is satisfied, for every clause, at least one of the variables in the clause must be $TRUE$, and thus the corresponding variable will have label $\{1\} = \mathcal{L}(A)$, and thus all dependencies of the second type will be satisfied.

For the other direction, assume that there is a coloring $\mathcal{L}$ with 2 colors and color number 2. Without loss of generality we can assume that $1 \in \mathcal{L}(A)\backslash\mathcal{L}(B)$ and $2 \in \mathcal{L}(B)\backslash\mathcal{L}(A)$. The functional dependencies imply that $\mathcal{L}(X_i) \cup \mathcal{L}(\bar{X}_i) \subseteq \mathcal{L}(A)$ and $\mathcal{L}(Y_i) \cup \mathcal{L}(\bar{Y}_i) \subseteq \mathcal{L}(B)$.

We argue that for all i, we must have exactly one of $\mathcal{L}(X_i)$ and $\mathcal{L}(\bar{X}_i)$ containing 1, and neither containing 2, and exactly one of $\mathcal{L}(Y_i)$ and $\mathcal{L}(\bar{Y}_i)$ with a label containing 2, and neither containing 1. Indeed, from the first type of functional dependencies, the only other possible option is that both X_i and $\bar{X}_i$ have color sets $\{1\}$, in which case in order for the color number to be 2, neither Y_i nor $\bar{Y}_i$ can be labeled 2, but this contradicts the functional dependency $Y_i\bar{Y}_i \rightarrow B$.

Note that if we replace each $C_i = S_i(X_j, X_k, X_m, A)$ by $S_i'(X_j, A_i', A) \wedge S_i''(X_k, X_m, A_i')$, and let the first two attributes of each relation be a (compound) key for the third attribute, we have an equivalent query where all functional dependencies have at most two variables on the left sides. □

8. FUTURE DIRECTIONS

We view the main contributions of this work as establishing tight worst-case size bounds and characterizing treewidth-preserving queries in the setting in which either no keys or simple keys are specified, and establishing the connection between worst-case size bounds and multivariate entropy structures in the setting with general functional dependencies, allowing the tools of information theory to be leveraged towards database analysis. We see three main lines of future work. The first direction is characterizing the conjunctive queries that have a bounded increase in treewidth, in the setting with arbitrary functional dependencies.

The other two direction are prompted by the connection between entropy structures and size bounds. One natural question in the setting of general functional dependencies is to investigate whether one can explicitly characterize the worst-case size increase, even if that characterization is exponentially large. It is also conceivable that, while exactly characterizing the size increase might not be possible, one can explicitly (and possibly even efficiently) compute an approximation of the worst-case size increase. This seems like a deep and challenging question, and such a result would likely involve a significant advance in the understanding of the structure of non-Shannon type information inequalities.

The final direction is investigating which types of entropy structures arise from databases and their associated queries in practice. Such an investigation would help determine where practical instances lie on the spectrum between the basic color number bounds and the more intricate bounds of Proposition 6.9. Such database measures as treewidth were introduced with corresponding goals in mind, and have proved effective at succinctly capturing the ease with which certain database operations can be done. We propose the following measure of the entropy structure of a database and associated query, in the hope that it will succinctly capture this new facet of database complexity, as suggested by the results of Section 6.4.

Definition 8.1. The *knitted complexity* of a database with respect to a query is the ratio of the sum of the absolute values of the mutual informations of all subsets of the query variables, to the sum of the (signed) mutual informations of all subsets of the query variables.

ACKNOWLEDGMENTS

We are deeply grateful to Daniel Marx, who first pointed out to us that the color number does not provide an upper bound on the worst-case size increase in the setting in which general functional dependencies are specified.

REFERENCES

ABITEBOUL, S., HULL, R., AND VIANU, V. 1995. *Foundations of Databases*. Addison-Wesley.

AHO, A. V., BEERI, C., AND ULLMAN, J. D. 1979a. The theory of joins in relational databases. *ACM Trans. Datab. Syst. 4,* 3, 297–314.

AHO, A. V., SAGIV, Y., AND ULLMAN, J. D. 1979b. Equivalence of relational expressions. *SIAM J. Comput. 8,* 2, 218–246.

ARNBORG, S., LAGERGREN, J., AND SEESE, D. 1991. Easy problems for tree-decomposable graphs. *J. Algor. 12,* 2, 308–340.

ATSERIAS, A., GROHE, M., AND MARX, D. 2008. Size bounds and query plans for relational joins. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.

BEERI, C. AND VARDI, M. Y. 1984. A proof procedure for data dependencies. *J. ACM 31,* 4, 718–741.

CHANDRA, A. K. AND MERLIN, P. M. 1977. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing (STOC)*.

CHAUDHURI, S. 1998. An overview of query optimization in relational systems. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*.

COURCELLE, B. 1990. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput. 85,* 1, 12–75.

DEUTSCH, A., POPA, L., AND TANNEN, V. 2006. Query reformulation with constraints. *ACM SIGMOD Record 35,* 1, 65–73.

DOUGHERTY, R., FREILING, C., AND ZEGER, K. 2007. Networks, matroids, and non-shannon information inequalities. *IEEE Trans. Inf. Theory 53,* 6, 1949–1969.

DOWLING, W. F. AND GALLIER, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. Logic Prog. 3,* 1, 267–284.

FAGIN, R., KOLAITIS, P. G., MILLER, R. J., AND POPA, L. 2003. Data exchange: Semantics and query answering. In *Proceedings of the 9th International Conference on Database Theory (ICDT)*.

FLUM, J., FRICK, M., AND GROHE, M. 2002. Query evaluation via tree-decompositions. *J. ACM 49,* 6, 716–752.

GOTTLOB, G. AND LEE, S. T. 2007. A logical approach to multicut problems. *Inf. Proc. Letters 103,* 4, 136–141.

GOTTLOB, G., LEE, S. T., AND VALIANT, G. 2009. Size and treewidth bounds for conjunctive queries. In *Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*.

GOTTLOB, G., PICHLER, R., AND WEI, F. 2007. Efficient datalog abduction through bounded treewidth. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*. AAAI Press, 1626–1631.

GROHE, M. AND MARX, D. 2006. Constraint solving via fractional edge covers. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

HAAS, P. J., NAUGHTON, J. F., SESHADRI, S., AND SWAMI, A. N. 1996. Selectivity and cost estimation for joins based on random sampling. *J. Comput. Syst. Sci. 52,* 3, 550–569.

JARKE, M. AND KOCH, J. 1984. Query optimization in database systems. *ACM Comput. Surv. 16,* 2, 111–152.

KOLAITIS, P. 2005. Schema mappings, data exchange, and metadata management. In *Proceedings of the 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*.

LENZERINI, M. 2002. Data integration: A theoretical perspective. In *Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*.

LEVY, A. Y., MENDELZON, A. O., AND SAGIV, Y. 1995. Answering queries using views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*.

MAIER, D., MENDELZON, A. O., AND SAGIV, Y. 1979. Testing implications of data dependencies. *ACM Trans. Datab. Syst. 4,* 4, 455–469.

MATÚŠ, F. 2007a. Infinitely many information inequalities. In *Proceedings of the IEEE International Symposium on Information Theory*.

MATÚŠ, F. 2007b. Two constructions on limits of entropy functions. *IEEE Trans. Inf. Theory 53,* 1, 320–330.

OLKEN, F. AND ROTEM, D. 1990. Random sampling from database files: A survey. In *Proceedings of the 5th International Workshop on Statistical and Scientific Data Management*.

PIPPENGER, N. 1986. What are the laws of information theory? In *Proceedings of the Special Problems on Communication and Computation Conference*.

ROBERTSON, N. AND SEYMOUR, P. D. 1986a. Graph minors II: Algorithmic aspects of tree-width. *J. Algor. 7,* 3, 309–322.

ROBERTSON, N. AND SEYMOUR, P. D. 1986b. Graph minors. V. Excluding a planar graph. *J. Combinat. Theory 41,* 1, 92–114.

SHAMIR, A. 1979. How to share a secret. *Commun. ACM 22,* 11, 612–613.

SWAMI, A. N. AND SCHIEFER, K. B. 1994. On the estimation of join result sizes. In *Proceedings of the 4th International Conference on Extending Database Technology - Advances in Database Technology (EDBT'94)*.

THORUP, M. 1998. Structured programs have small tree-width and good register allocation. *Inf. Comput. 142*, 318–332.

VALIANT, G. AND VALIANT, P. 2009. Size bounds for conjunctive queries with general functional dependencies. http://arxiv.org/abs/0909.2030.

YEUNG, R. W. 2008. *Information Theory and Network Coding*. Springer Publishing Company, Incorporated.

ZHANG, Z. AND YEUNG, R. W. 1997. A non-shannon-type conditional inequality of information quantities. *IEEE Trans. Inf. Theory 43,* 6, 1982–1986.

ZHANG, Z. AND YEUNG, R. W. 1998. On characterization of entropy function via information inequalities. *IEEE Trans. Inf. Theory 44,* 4, 1440–1452.

Received April 2010; revised July 2011, January 2012; accepted February 2012