Entropy Bounds for Conjunctive Queries with Functional Dependencies

Tomasz Gogacz*[1] and Szymon Toruńczyk†[2]

1 University of Edinburgh, Edinburgh, UK
2 University of Warsaw, Warsaw, Poland

Abstract

We study the problem of finding the worst-case size of the result $Q(\mathbb{D})$ of a fixed conjunctive query Q applied to a database $\mathbb{D}$ satisfying given functional dependencies. We provide a characterization of this bound in terms of entropy vectors, and in terms of finite groups. In particular, we show that an upper bound provided by Gottlob, Lee, Valiant and Valiant [9] is tight, and that a correspondence of Chan and Yeung [5] is preserved in the presence of functional dependencies. However, tightness of a weaker upper bound provided by Gottlob et al., which would have immediate applications to evaluation of join queries [11], remains open. Our result shows that the problem of computing the worst-case size bound, in the general case, is closely related to difficult problems from information theory.

1998 ACM Subject Classification H.2.4 Query Processing

Keywords and phrases database theory, conjunctive queries, size bounds, entropy, finite groups, entropy cone

Digital Object Identifier 10.4230/LIPIcs.ICDT.2017.15

1 Introduction

Given a conjunctive query Q we would like to determine the least bound $\alpha \in \mathbb{R}$ such that for every database $\mathbb{D}$, the inequality

$$|Q(\mathbb{D})| \;\leq\; c \cdot |\mathbb{D}|^{\alpha} \tag{1}$$

holds, for some multiplicative factor c depending only on Q. Above, $|\mathbb{D}|$ denotes the size of the largest table in the database $\mathbb{D}$, and $|Q(\mathbb{D})|$ denotes the size of the result of the query applied to $\mathbb{D}$. In the general problem which is the main focus of this paper, we may additionally impose some functional dependencies. Define $\alpha(Q)$ as the infimum of all values α for which there exists a multiplicative factor c so that (1) holds for all databases $\mathbb{D}$ satisfying the given functional dependencies. Note that defining $|\mathbb{D}|$ as the sum of the sizes of the tables in $\mathbb{D}$ would result in the same value of $\alpha(Q)$, since this would increase $|\mathbb{D}|$ by at most a constant factor (the number of relations in $\mathbb{D}$), and this constant can be accommodated by c in (1).

For example, if $Q_1(x, y, z) = R(x, y) \wedge S(y, z)$ and there are no functional dependencies, then it is not difficult to see that $|Q_1(\mathbb{D})| \leq |R(\mathbb{D})| \cdot |S(\mathbb{D})| \leq |\mathbb{D}|^2$, so in 1 we can take $\alpha = 2$, and it is not difficult to see that $\alpha(Q_1) = 2$. Now consider the natural join query $Q_2(x, y, z) = R(x, y) \wedge S(y, z) \wedge T(z, x)$. A trivial bound gives $\alpha(Q_2) \leq 3$. However, since

* Supported by VADA EPSRC grant M025268.
† Funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ERC-2014-CoG 648276 AUTAR).

$Q_2(\mathbb{D}) \subseteq Q_1(\mathbb{D})$ for every $\mathbb{D}$, it follows that $\alpha(Q_2) \leq 2$. With some effort, one can show that in the absence of functional dependencies, $\alpha(Q_2) = 3/2$. This follows from a result due to Atserias, Grohe and Marx [1], which characterizes $\alpha(Q)$ when the query Q is a natural join query (without projections, self-joins and attribute renamings), and there are no functional dependencies, which we recall now.

Consider the hypergraph whose vertices are the variables appearing in Q, and for each relation name R in Q there is a hyperedge containing those variables which appear in R. A *fractional edge packing* of a hypergraph assigns a positive rational number to each of its vertices, so that for every hyperedge, the numbers assigned to the adjacent vertices sum up to at most 1. The total weight of a fractional edge packing is the sum of the numbers assigned to all the vertices. Define $\mathrm{AGM}(Q)$ as the largest possible total weight of a fractional edge packing of the hypergraph associated to Q. The AGM bound then states that $\mathrm{AGM}(Q) = \alpha(Q)$, in the absence of functional dependencies. For example, the hypergraph obtained from the query Q_2 is the triangle, and assigning $1/2$ to each vertex gives a fractional edge packing with total weight $3/2$, which is the largest possible. Hence, $\alpha(Q_2) = \mathrm{AGM}(Q_2) = 3/2$. Note that $\mathrm{AGM}(Q)$ can be computed using linear programming.

In this paper, we make progress towards characterizing the value $\alpha(Q)$ in the presence of functional dependencies. In particular, we show that $\alpha(Q)$ can be characterized in two ways: as an entropy bound $H(Q)$, and in terms of a number $\mathrm{GC}(Q)$ derived from systems of finite groups. The bound $\alpha(Q) \leq H(Q)$ was observed by Gottlob et al. [9]. We provide a matching lower bound $\alpha(Q) \geq H(Q)$ based on a construction using finite groups. Unfortunately, we do not know how to compute the value $\alpha(Q)$. Our results indicate that the problem is closely connected to notorious problems from information theory.

Note that the paper [9] also gives a weaker bound $\alpha(Q) \leq H(Q) \leq h(Q)$, where $h(Q)$ is obtained by relaxing the entropy cone to the polymatroid defined by Shannon inequalities (see Corollary 8). The question whether $\alpha(Q) = h(Q)$ remains open. Note that $h(Q)$ can be computed using linear programming. Moreover, $\alpha(Q) = h(Q)$ would prove optimality of an algorithm computing $Q(\mathbb{D})$, recently provided by Khamis et al. [11].

Throughout most of the paper, we focus on natural join queries. A gentle introduction to entropy and natural join queries, and the relationship between them is given in the preliminaries (Section 2). In the following section, Section 3, we state our main results and their consequences. The main result is proved by first characterizing the value $H(Q)$ (Section 4), then adapting upper bounds and the tightness proof from the work of [5] to the setting with functional dependencies (Sections 5 and 6). In Section 7 we discuss how to treat general conjunctive queries and in Section 8 we show some preliminary results concerning the computation of the result $Q(\mathbb{D})$.

Comparison with previous work. The work of Chan and Yeung [5] establishes a two-way relationship between entropy vectors and group vectors. Our construction of databases from group systems, presented in Section 5, although reminiscent of their construction of entropy vectors from group systems, was independent of that work and was motivated by the coloring construction of Gottlob et al. [9] and their example showing its suboptimality. Nevertheless, it is only fair to say that the development of Section 5 is a straightforward adaptation of a construction of Chan and Yeung [5] in the database setting. As a consequence, we extend their correspondence to a three-way correspondence between group vectors, entropy vectors, and vectors induced from databases. The easier direction, from databases to entropy vectors, was implicit in the paper by Gottlob et al. [9], however, the other direction was missing. For completeness, we present all proofs.

More importantly, however, we demonstrate that in all directions, the correspondences preserve functional dependencies. Some of these preservation results are rather obvious: if a group vector satisfies certain functional dependencies, then the corresponding entropy vector satisfies them too – this has been observed in Proposition 3 of [4] – and so does the corresponding database (see Proposition 18 below). In Section 6 we analyse the construction of group vectors from entropy vectors from [14] and demonstrate that it preserves functional dependencies. Also, to prove Theorem 5, we need to prove that functional dependencies behave well with respect to topological closure (cf. Lemma 12). Reassuming, the main contribution of this paper can be seen as a detailed study of the preservation of functional dependencies in the framework proposed by Chan and Yeung.

2 Preliminaries

To fix notation, we recall some notions concerning databases and entropy. We assume a fixed *schema* Σ, which specifies a finite set of *attributes* $\mathbf{V}(\Sigma)$, a finite set of *relation names*, and for each relation name R, a finite set $\mathbf{V}(R) \subseteq \mathbf{V}(\Sigma)$ of attributes of R. If X is a set of attributes, then a *row* with attributes X is a function r assigning to each $x \in X$ some value $r[x]$. If r is a row with attributes X and $Y \subseteq X$ then by $r[Y]$ we denote the restriction of r to Y. A *table* with attributes X is a finite set of rows with attributes X. A *database* $\mathbb{D}$ over Σ specifies for each relation name R in Σ a table with attributes $\mathbf{V}(R)$. A *natural join query* is a set Q of relation names in Σ; we denote $\mathbf{V}(Q) = \bigcup_{R \in Q} \mathbf{V}(R)$. Such a query can be applied to a database $\mathbb{D}$, yielding as result the table $Q(\mathbb{D})$ consisting of those rows r with attributes $\mathbf{V}(Q)$ such that $r[\mathbf{V}(R)] \in R(\mathbb{D})$ for every $R \in Q$.

We say that a database $\mathbb{D}$ *satisfies* a *functional dependency* $R : X \mapsto x$ – where X is a set of attributes and x is a single attribute – if for any two rows u, v of $R(\mathbb{D})$, $u[X] = v[X]$ implies $u[x] = v[x]$.

For the rest of this paper, fix a schema Σ and a set of functional dependencies $\mathcal{F}$. Every database $\mathbb{D}$ is assumed to be over this schema, and to satisfy $\mathcal{F}$. Define $\alpha(Q)$ as the smallest value α for which there exists a constant c such that (1) holds for all databases $\mathbb{D}$ over Σ which satisfy the functional dependencies in $\mathcal{F}$. For convenience, we define $|\mathbb{D}|$ to be the maximal size of a relation in $\mathbb{D}$.

▶ Remark. Observe that since Q is a natural join query and in the definition of $\alpha(Q)$ we are interested in maximizing $|Q(\mathbb{D})|$ while keeping $|\mathbb{D}|$ bounded, we may assume that Σ contains only the relation symbols which appear in Q (as $\mathbb{D}$ would have the remaining tables empty anyway) and furthermore, that if $R : X \mapsto x$ is a functional dependency in $\mathcal{F}$, then also $S : X \mapsto x$ is a functional dependency in $\mathcal{F}$, for every relation name S such that $X \subseteq \mathbf{V}(S)$. Therefore, we may simply write that $\mathcal{F}$ contains the functional dependency $X \mapsto x$ instead of writing $R : X \mapsto x$.

For a database $\mathbb{D}$ (over Σ, satisfying $\mathcal{F}$), denote

$$\alpha(Q, \mathbb{D}) = \frac{\log |Q(\mathbb{D})|}{\log |\mathbb{D}|}. \tag{α}$$

Convention. Throughout this paper we will define several real-valued parameters of the form $\gamma(Q, x)$, where Q is a query and x is some object. For a fixed query Q, we denote by $\sup_x \gamma(Q, x)$ the supremum, and by $\limsup_x \gamma(Q, x)$ the limit superior over all values x, for which the value $\gamma(Q, x)$ is defined. In particular, $\limsup_x \gamma(Q, x)$ is the smallest value s in $\mathbb{R} \cup \{-\infty, +\infty\}$ such that for every real $\varepsilon > 0$, there are only finitely many x's such that $\gamma(Q, x)$ is defined and larger than $s + \varepsilon$.

Limit superior vs. supremum. The following lemma will simplify several formulations and proofs throughout this paper.

▶ **Lemma 1.** *Let Q be a natural join query Q. Then $\alpha(Q) = \limsup_{\mathbb{D}} \alpha(Q, \mathbb{D}) = \sup_{\mathbb{D}} \alpha(Q, \mathbb{D})$.*

Proof. We omit the easy proof that $\alpha(Q) = \limsup_{\mathbb{D}}(Q, \mathbb{D})$, and prove only the second equality. To show that $\sup_{\mathbb{D}} \alpha(Q, \mathbb{D}) \leq \limsup_{\mathbb{D}} \alpha(Q, \mathbb{D})$, we use the following construction. For a database $\mathbb{D}$ and a natural number n, let $\mathbb{D}^n$ be the database defined so that the rows of $R(\mathbb{D}^n)$ are n-tuples of rows of $R(\mathbb{D})$, and for such a row $r = (r_1, \ldots, r_n)$, we define $r[x] = (r_1[x], \ldots, r_n[x])$ for an attribute $x \in \mathbf{V}(R)$. It is easy to check that $\mathbb{D}^n$ satisfies the same functional dependencies as $\mathbb{D}$, and that $|\mathbb{D}^n| = |\mathbb{D}|^n$ and $|Q(\mathbb{D})| = |Q(\mathbb{D})|^n$. In particular, $\alpha(Q, \mathbb{D}^n) = \alpha(Q, \mathbb{D})$. It follows that if $|\mathbb{D}| > 1$, then by choosing n arbitrarily large, we have arbitrarily large databases $\mathbb{D}^n$ with $\alpha(Q, \mathbb{D}^n) = \alpha(Q, \mathbb{D})$. Therefore, $\limsup_{\mathbb{D}} \alpha(Q, \mathbb{D}) \geq \sup_{\mathbb{D}} \alpha(Q, \mathbb{D})$, the other inequality being obvious. ◀

Entropy. In this paper, we only consider random variables taking finitely many values. Formally, a random variable X is a measurable function $X : \Omega \to V$ from a fixed probability space $(\Omega, \mathbb{P})$ of events to a finite set V. In this paper, however, it is not harmful to assume that Ω is a finite probability space, in which case every function $X : \Omega \to V$ is a random variable. By $\mathrm{Im}(X) \subseteq V$ we denote the set of values v such that $\mathbb{P}[X = v] > 0$, where $\mathbb{P}[X = v]$ is a shorthand for $\mathbb{P}[\{\omega \in \Omega : X(\omega) = v\}]$.

For a random variable X taking values in a finite set V, define the *entropy* of X as $H(X) = -\sum_{v \in \mathrm{Im}(X)} p_v \log p_v$, where $p_v = \mathbb{P}[X = v]$. Clearly, the entropy of X only depends on the distribution of X. Also, the maximal possible entropy of a random variable with values in a finite set V is equal to $\log |V|$, and is attained by the uniform distribution on V, as follows from Jensen's inequality applied to the convex function $-\log(x)$.

Upper bound. We recall an upper bound on $\alpha(Q, D)$, observed by Gottlob et al. [9]. Fix a natural join query Q. Let U be a random variable U taking as values rows with attributes $\mathbf{V}(\Sigma)$. For a set of attributes $X \subseteq \mathbf{V}(\Sigma)$, define $U[X]$ to be the random variable whose value is the restriction of the value of U to the set of attributes X. In particular, $U[X]$ is a random variable whose values are rows with attributes X, and $\mathrm{Im}(U[\mathbf{V}(R)])$ is a table with attributes $\mathbf{V}(R)$. We say that U satisfies a functional dependency $Y \mapsto x$ if the table $\mathrm{Im}(U)$ satisfies the functional dependency $Y \mapsto x$. We write Yx to denote the set $Y \cup \{x\}$. We say that a vector $h \in \mathbb{R}^{P(\mathbf{V}(Q))}$ satisfies a functional dependency $Y \mapsto x$ if $h(Y) = h(Yx)$. The following lemma is immediate.

▶ **Lemma 2.** *U satisfies a functional dependency $Y \mapsto x$ iff the vector h^U satisfies $Y \mapsto x$.*

We remark that in information theory, $h^U(Y) = h^U(Yx)$ can be expressed using conditional entropy as $H(\ U[x] \mid U[Y]\ ) = 0$ or $h^U(x \mid Y) = 0$.

For a random variable U which satisfies every functional dependency in $\mathcal{F}$, define

$$H(Q, U) = \frac{H(U[\mathbf{V}(Q)])}{\max_{R \in \Sigma} H(U[\mathbf{V}(R)])}. \tag{H}$$

The following is an observation from [9].

▶ **Lemma 3.** *Let Q be a natural join query. For a database $\mathbb{D}$, let $U_{\mathbb{D}}$ be a random variable which picks a row of $Q(\mathbb{D})$ uniformly at random. Then $\alpha(Q, \mathbb{D}) \leq H(Q, U_{\mathbb{D}})$.*

Proof. By definition of a natural join query, the values of $U_{\mathbb{D}}[R]$ are rows of $R(\mathbb{D})$. Then $H(Q, U_{\mathbb{D}}) \geq \alpha(Q, \mathbb{D})$ since $H(U_{\mathbb{D}}) = \log |Q(\mathbb{D})|$ and $H(U_{\mathbb{D}}[R]) \leq \log |R(\mathbb{D})|$ by the fact that the uniform distribution maximizes entropy. ◀

Entropy cone. Fix a finite set X. Let $U = (U_x)_{x\in X}$ be a family of random variables indexed by X. For $Y \subseteq X$, let $U[Y]$ denote the joint random variable $(U_y)_{y\in Y}$. The random variable $U[Y]$ can be seen as a random variable taking as values tuples indexed by Y. Consider the real-valued function h^U from subsets of X, such that $h^U(Y) = H(U[Y])$ for $Y \subseteq X$; the function h^U can be also seen as a vector in $\mathbb{R}^{P(X)}$. Vectors of the form $h^U \in \mathbb{R}^{P(X)}$, where U is a family of random variables indexed by X, are called *entropy vectors* (or *entropic vectors*) with ground set X [16].

The set of all entropy vectors with ground set X forms a subset of $\mathbb{R}^{P(X)}$, denoted Γ^*_X. Its topological closure $\overline{\Gamma^*_X}$ is a convex cone. The sets Γ^*_X and $\overline{\Gamma^*_X}$ are well studied, however, to date, they lack effective descriptions when $|X| \geq 4$. It is known that the closed cone $\overline{\Gamma^*_X}$ is a polyhedron if $|X| \leq 3$, and is not a polyhedron if $|X| > 3$ (i.e. it is not described by finitely many linear inequalities). Entropy vectors $h \in \Gamma^*_X$ (and hence, by continuity, also all $h \in \overline{\Gamma^*_X}$) satisfy the submodularity property, expressing *Shannon's inequality* for information:

$$h(Y \cup Z) + h(Y \cap Z) \leq h(Y) + h(Z) \quad \text{for } Y, Z \subseteq X. \tag{2}$$

Groups, actions and cosets. We use basic notions from algebra. We refer the reader to [12] for background. If G is a group and H is its subgroup, then a (left) *coset* of H in G is a subset of G of the form $gH = \{gh : h \in H\} \in G/H$. Let G/H denote the *coset space*, i.e., the set $G/H = \{gH : g \in G\}$ of all cosets of H in G.

For a finite group G and a set X, recall that a (left) action of G on X is a mapping $G \times X \to X$ denoted $(g, x) \mapsto g \cdot x$, such that $(g \cdot h) \cdot x = g \cdot (h \cdot x)$ for $g, h \in G$ and $x \in X$, and $e \cdot x = x$ for $x \in X$ and e the identity element of G. We say that the action is *transitive* if for every $x, y \in X$ there is $g \in G$ such that $g \cdot x = y$. The *stabiliser* of a point $x \in X$ (for a given action of G on X) is the subset $\{g \in G : g \cdot x = x\}$ of G; this is in fact a subgroup of G.

Note that if H is a subgroup of G, then G acts transitively on G/H, where the action is given by $g' \cdot gH = (g'g)H$ for $g', g \in G$; transitivity of this action follows from the fact that for $g, g' \in G$, $(g'g)^{-1}gH = g'H$. Every transitive action of G on some set X arises in this way, i.e., is isomorphic to the action of G on G/H, from some subgroup H of G (namely, one can take H as the stabilizer of any element $x \in X$).

3 Main result and consequences

In this section, we give a brief overview of the main result of the paper, and its consequences.

Main result. For a relation name R in a schema Σ, by $\mathbf{V}(R)$ we denote the set of attributes of R. For $X \subseteq \mathbf{V}(R)$ and $x \in \mathbf{V}(R)$, we write $R : X \mapsto x$ to denote the functional dependency (fd) requiring that in R, the values of attributes X determine the value of the attribute x. The value $\alpha(Q)$ is defined as in the introduction, taking into account all databases $\mathbb{D}$ over the schema Σ which satisfy a given set of functional dependencies $\mathcal{F}$. We associate another value $H(Q)$ to a query Q, as follows.

▶ **Definition 4.** Fix a schema Σ and a set of functional dependencies $\mathcal{F}$. Let Q be a natural join query with attributes X. Let $H(Q)$ be the maximal[1] value of $h(X)$, for h ranging over $\overline{\Gamma^*_X}$, satisfying:

[1] Note that the supremum of $h(X)$ is attained by some entropy vector h, since the function $h \mapsto h(X)$ is continuous, and the set of entropy vectors satisfying the given constraints is compact, as we use $\overline{\Gamma^*_X}$. This will not necessarily hold in other places throughout this paper.

$$\begin{cases} h(\mathbf{V}(R)) \leq 1 & \text{for } R \in Q, \\ h(Z \cup \{z\}) = h(Z) & \text{for every fd } R : Z \mapsto z \text{ in } \mathcal{F}. \end{cases}.$$

The main result of this paper is the following.

▶ **Theorem 5.** *Let Q be a natural join query. Then $\alpha(Q) = H(Q)$.*

▶ Remark. The inequality $\alpha(Q) \leq H(Q)$ is straightforward, and was observed in [9] (see Lemma 3). However, as we discuss in Remark 4, the inequality $\alpha(Q) \geq H(Q)$ is more involved.

Symmetric databases. The proof of the AGM bound (cf. [1] or Section 3.1 below) shows that in the absence of functional dependencies, the databases $\mathbb{D}$ for which the size-increase $\frac{\log |Q(\mathbb{D})|}{\log |\mathbb{D}|}$ achieves the bound $\alpha(Q)$ are of a very simple, specific form: each table is a full Cartesian product. It follows from [9] that in the presence of functional dependencies, this is no longer the case: databases of this form, satisfying the given functional dependencies, are arbitrarily far from reaching the value of $\alpha(Q)$.

In this paper, we improve the construction of worst-case databases in the presence of functional dependencies, by constructing databases which are arbitrarily close to achieving the bound $\alpha(Q)$. Interestingly, these databases have a very symmetric structure, and their construction uses finite groups. Our construction of databases from groups is very similar to a construction from [5].

For a fixed group G, by a *G-symmetric* database we mean a database $\mathbb{D}$ together with an action of G on the set of all values appearing in all tables of $\mathbb{D}$, such that the componentwise action of G on (the rows of) each table $R(\mathbb{D})$ is transitive (the componentwise action is given by $(g \cdot r)[x] = g \cdot (r[x])$ for $g \in G, r \in R(\mathbb{D})$ and $x \in \mathbf{V}(R)$). A *symmetric* database is a database $\mathbb{D}$ which is G-symmetric for some finite group G. Intuitively, in each table of a symmetric database, all rows are the same, up to permutation. Symmetric databases are very regular. For example, if a database $\mathbb{D}$ represents a graph G (i.e., it has one relation E, which is symmetric) and if $\mathbb{D}$ is symmetric, then the graph G is regular (all vertices have the same degree), vertex transitive (for any two vertices there is an automorphism of G mapping the first one to the second) and edge transitive (for any two edges there is an automorphism of G mapping the first one to the second).

The second main result of this paper is the following.

▶ **Theorem 6.** *Fix a schema Σ, functional dependencies $\mathcal{F}$, and a natural join query Q. Then, for each $\varepsilon > 0$ there are arbitrarily large symmetric databases $\mathbb{D}$ with $\frac{\log |Q(\mathbb{D})|}{\log |\mathbb{D}|} > \alpha(Q) - \varepsilon$.*

This can be seen as a structure result for worst-case databases. Apart from that, symmetric databases are essential in our proof of the lower bound given in Theorem 5.

Simple statistics. We state without proof a result generalizing Theorem 5, whose proof can be obtained in a similar way. In Theorem 5, intuitively, we were interested in the worst-case size of $Q(\mathbb{D})$, assuming the maximal table of $\mathbb{D}$ has a given size (which is manifested by the inequality $h(\mathbf{V}(R)) \leq 1$ for all R). Knowing the precise sizes of all tables in $\mathbb{D}$ may lead to better bounds on $Q(\mathbb{D})$, as we show below. For a database $\mathbb{D}$ over the schema of Q, its *simple log-statistics* is the vector $\mathrm{sls}(\mathbb{D}) = (\log |R(\mathbb{D})|)_{R \in Q}$.

▶ **Theorem 7.** *Fix a schema Σ and a set of functional dependencies $\mathcal{F}$. Let Q be a natural join query with attributes X, and let $s = (s_R)_{R \in Q}$ be a vector of nonnegative real numbers.*

Let $\beta(Q,s)$ be the maximal value of $h(X)$, for h ranging over $\overline{\Gamma^*_X}$ and satisfying:

$$\begin{cases} h(\mathbf{V}(R)) \leq s_R & \textit{for } R \in Q, \\ h(Z \cup \{z\}) = h(Z) & \textit{for every fd } R : Z \mapsto z \textit{ in } \mathcal{F}. \end{cases}$$

Then $\sup_{\mathbb{D}} \log|Q(\mathbb{D})| = \limsup_{\mathbb{D}} \log|Q(\mathbb{D})| = \beta(Q,s)$, where $\mathbb{D}$ ranges over all finite databases satisfying the functional dependencies $\mathcal{F}$ and such that $\mathrm{sls}(\mathbb{D}) \leq s$ componentwise.

Theorem 5 is an immediate consequence of Theorem 7, obtained by setting $s_R = \log|\mathbb{D}|$ for each $R \in Q$, and using the fact that $\overline{\Gamma^*_X}$ is a cone, since we can divide the vector obtained by Theorem 7 by any positive number (in our case, $\log|\mathbb{D}|$) and still get a vector from $\overline{\Gamma^*_X}$. It is straightforward to verify that this new vector is a solution postulated by Theorem 5.

In this paper, we present in detail only the proof of Theorem 5. The proof of Theorem 7 proceeds similarly, and will be presented in the full version of the paper [8].

3.1 Consequences

We now show how some results known previously can be obtained as consequences of Theorem 5 (in fact, of the easier, upper bound $\alpha(Q) \leq \sup_U H(Q,U)$ presented in Section 2.) The results from Section 3.1 are not used in this paper, and are presented only for completeness.

Relaxing the condition in Definition 4 that $h \in \overline{\Gamma^*_X}$ to the condition that h is submodular gives the following upper bound on $\alpha(Q)$, which is equivalent to a bound in [9]. For a query Q and functional dependencies $\mathcal{F}$, consider the maximal[2] value of $h(X)$ for h ranging over $\mathbb{R}^{P(X)}$, satisfying:

$$h(\emptyset) = 0 \tag{3}$$
$$h(Y) \leq h(Z) \qquad \text{for } Y \subseteq Z \subseteq X, \tag{4}$$
$$h(Y \cup Z) + h(Y \cap Z) \leq h(Y) + h(Z) \qquad \text{for } Y, Z \subseteq X, \tag{5}$$
$$h(\mathbf{V}(R)) \leq 1 \qquad \text{for } R \in Q, \tag{6}$$
$$h(Z \cup \{z\}) = h(Z) \qquad \text{for every fd } R : Z \mapsto z \text{ in } \mathcal{F}. \tag{7}$$

Denote the above optimum by $h(Q)$. Since the conditions describing $h(Q)$ are a relaxation of the conditions describing $H(Q)$, from $\alpha(Q) \leq H(Q)$ we get the following.

▶ **Corollary 8** ([9]). *For a conjunctive query Q and functional dependencies $\mathcal{F}$, $\alpha(Q) \leq h(Q)$.*

Note that the bound $h(Q)$ can be computed by linear programming, where the number of variables is exponential in the size of Q. Unfortunately, the question whether $\alpha(Q) = h(Q)$ for every query Q and functional dependencies, stated in [9], remains open. Recently, Khamis, Ngo and Suciu [11] provided an algorithm for computing $Q(\mathbb{D})$ in the presence of functional dependencies, in time $\tilde{O}(|\mathbb{D}|^{h(Q)})$ (where $\tilde{O}$ hides polylogarithmic factors), which is optimal assuming the tightness of the above bound.

We now show how the AGM bound follows from Corollary 8. Note that the original proof [1] of the AGM bound used Shearer's Lemma from information theory and strong duality of linear programs. The proof presented below only uses Shannon information inequalities, manifested in Corollary 8. A function $h \in \mathbb{R}^{P(X)}$, is *modular* if it satisfies $h(Y) = \sum_{y \in Y} h(\{y\})$ for $Y \subseteq X$.

[2] The maximum is attained, as follows by a simple compactness argument.

▶ **Lemma 9.** *In absence of functional dependencies, the optimum $h(Q)$ is attained by a modular function $h \in \mathbb{R}^{P(X)}$.*

Proof. Let $h \in \mathbb{R}^{P(X)}$ be a function satisfying conditions (3)-(6), with maximal possible value of $h(X)$. In particular, h is submodular. We show that h can be in fact taken modular.

Denote by P_h the set[3] of functions $r : X \to \mathbb{R}$ satisfying $\hat{r}(Y) \leq h(Y)$ for $Y \subseteq X$, where $\hat{r}(Y)$ is shorthand for $\sum_{x \in Y} r(x)$. Consider the linear optimization problem of finding $r \in P_h$ which maximizes the value $\hat{r}(X) = \sum_{x \in X} r(x)$. It follows from general principles (see e.g. [13], Section 3) that the so-called "greedy algorithm" gives an optimal solution r to this problem, defined by

$$r(x_i) = h(\{x_1, \ldots, x_i\}) - h(\{x_1, \ldots, x_{i-1}\}) \qquad \text{for } i = 1, \ldots, n,$$

where $n = |X|$ and $x_1, x_2, \ldots, x_n$ is a fixed enumeration of the elements of X. In our special case it is sufficient and not difficult to check that conditions (3) and (5) imply $\hat{r}(Y) \leq h(Y)$ for all $Y \subseteq X$. Optimality of $\hat{r}$ then follows from the fact that $\hat{r}(X) = \sum_{i=1}^{n} r(x_i) = h(X) - h(\emptyset) = h(X)$. By modularity, $\hat{r}$ satisfies conditions (3)-(5), and it also satisfies condition (6) because $\hat{r}(Y) \leq h(Y)$ for all Y. Since $\hat{r}(X) = h(X)$, this proves the lemma. ◀

Since modular functions $h \in \mathbb{R}^{P(X)}$ are uniquely determined by their values for singletons, and automatically satisfy conditions (3)-(5), we conclude that in the absence of functional dependencies, $h(Q)$ is equal to the maximal value of $\sum_{x \in X} r(x)$ for $r \in \mathbb{R}^X$ satisfying $\sum_{x \in \mathbf{V}(R)} r(x) \leq 1$ for $R \in Q$. Such functions r are by definition real-valued edge packings of the hypergraph associated to Q, and the maximal possible total weight of such a packing is equal[4] to $\mathrm{AGM}(Q)$. Hence we get the following.

▶ **Corollary 10.** *In the absence of functional dependencies, $\alpha(Q) \leq h(Q) = \mathrm{AGM}(Q)$.*

We remark that [1] also prove a matching lower bound $\alpha(Q) \geq h(Q)$ in the absence of functional dependencies, thus proving $h(Q) = \alpha(Q) = \mathrm{AGM}(Q)$.

We remark that Theorem 7 can be used to derive the following, more precise variant of the AGM bound. A *fractional vertex covering* of a hypergraph is an assignment of rational weights to hyperedges, so that for every vertex, the sum of the weights assigned to the adjacent hyperedges is at least 1. By strong duality for linear programming, the smallest total weight of a fractional vertex covering is equal to the largest total weight of a fractional edge packing.

▶ **Corollary 11** ([1]). *In the absence of functional dependencies, $|Q(\mathbb{D})| \leq \prod_{R \in Q} |R(\mathbb{D})|^{w_R}$, where $(w_R)_{R \in Q}$ is any fractional vertex covering of the hypergraph associated to Q.*

To deduce the above corollary from Theorem 7, repeat the reasoning used above when deriving Corollary 10 from Theorem 5, by relaxing the condition $h \in \overline{\Gamma_X^*}$ in Theorem 7 to submodularity of h, and apply strong duality for linear programming. We leave the details to the interested reader.

[3] Called the polymatroid associated with the submodular function h.

[4] It follows from general principles of linear programming that a linear program with rational coefficients is optimized by a fractional solution.

Geometric inequalities. As noted elsewhere [15, 7, 2], Corollary 11 provides an upper bound on the size of a finite set in multi-dimensional space, in terms of the sizes of its projections. It implies the discrete versions of many inequalities from geometry and analysis, such as Hölder's inequality, Cauchy-Schwartz inequality, Loomis-Whitney inequality, Bollobás-Thomason inequality, Friedgut's inequality.

4 Characterization of $H(Q)$

In this section, we relate the value $H(Q)$ from Definition 4 in terms of $H(Q, U)$, as $\sup_U H(Q, U)$ (cf. Proposition 14). Recall that according to our convention, the supremum ranges over all random variables U which satisfy the given functional dependencies. Note that the inequality $H(Q) \geq \sup_U H(Q, U)$ follows from Definition 4 and the fact that the entropy vector h_U associated to U also satisfies the functional dependencies by Lemma 2. The remaining inequality $H(Q) \leq \sup_U H(Q, U)$ requires a more careful explanation. This is because $H(Q)$ is defined as a supremum for $h \in \overline{\Gamma^*_{\mathbf{V}(Q)}}$ which are limits of a sequence $h_1, h_2, \ldots$ of entropy vectors, and such that h (and not $h_1, h_2, \ldots$) satisfies the given functional dependencies. Therefore, to prove Proposition 14, we need the following.

▶ **Lemma 12.** *Let $h \in \overline{\Gamma^*_{\mathbf{V}(Q)}}$ be a vector satisfying functional dependencies $\mathcal{F}$. Then there exists a sequence of random variables $\{V_n\}_{n\in\mathbb{N}}$ such that $h = \lim_{n\to\infty} h^{V_n}$, and each V_n satisfies $\mathcal{F}$.*

Proof. Lemma 12 will be proven by induction on the size of $\mathcal{F}$. Let $\{U_n\}_{n\in\mathbb{N}}$ be a sequence of random variables such that $h = \lim_{n\to\infty} h^{U_n}$ satisfies a functional dependency $X \mapsto y$, and moreover each random variable U_n satisfies a set of functional dependencies $\mathcal{F}$. We construct a sequence of random variables $\{V_n\}_{n\in\mathbb{N}}$ such that each random variable V_n satisfies $\mathcal{F} \cup \{X \mapsto y\}$, and $\lim_{n\to\infty} h^{V_n} = h$.

Let us focus on a single random variable $U = U_n$. We partition the table $\mathrm{Im}(U)$ into sets $A_1, A_2, \ldots$, so that each A_i satisfies the functional dependency $X \mapsto y$, as follows. For each row $r \in \mathrm{Im}(U[X])$ proceed as follows. Choose a row $r_1 \in \mathrm{Im}(U[Xy])$ extending r, which is attained by the random variable $U[Xy]$ with the greatest probability. Add to the set A_1 all rows $r_1' \in \mathrm{Im}(U)$ which extend r_1. After that choose a row $r_2 \in \mathrm{Im}(U[Xy])$ extending r, which is attained by the random variable $U[Xy]$ with the second greatest probability. Add to the set A_2 all rows $r_2' \in \mathrm{Im}(U)$ which extend r_2. And so on.

At the end of this process, observe that each table A_i satisfies the dependency $X \mapsto y$.

Let A be a random variable which indicates to which set A_i the tuple from $\mathrm{Im}(U[X])$ belongs. In other words, $A = i \iff U \in A_i$. We omit the proof of the following lemma.

▶ **Lemma 13.** $H(A) \leq 2 \cdot H(U[y]|U[X])$

Let V be the following modification of U. Each attribute value in the set A_i gets a new prefix $A_i\_$ attached to its name, so the sets of possible attribute values in sets A_i and A_j are disjoint for $i \neq j$. After such a procedure, the entropy of the attributes could be increased, but not more than by $H(A)$.

We will show that setting V_n to V gives the desired result. The random variable V satisfies the dependencies $\mathcal{F} \cup \{X \mapsto y\}$ on each set A_i by definition of the set A_i. As presented during the construction, for any set of attributes Z, the entropy of $V[Z]$ can differ from $U[Z]$ at most by $2 \cdot H(U[y] \mid U[X])$. Since $\lim_{n\to\infty} H(U_n[y] \mid U_n[X]) = 0$ we get that $\lim_{n\to\infty} h^{U_n} = \lim_{n\to\infty} h^{V_n}$. ◀

Lemma 12 proves the following.

▶ **Proposition 14.** *Let Q be a natural join query. Then $H(Q) = \sup_U H(Q, U)$.*

From Lemma 3 and Proposition 14 we get the following.

▶ **Lemma 15.** *In the presence of functional dependencies, $\alpha(Q) \leq H(Q)$.*

Therefore, to prove Theorem 5, it remains to show that $\alpha(Q) \geq H(Q)$.

▶ Remark. To prove $\alpha(Q) \geq H(Q)$, it would be enough to construct, for a given random variable U a database $\mathbb{D}$ satisfying the same functional dependencies, and such that $\alpha(Q, \mathbb{D}) \geq H(Q, U)$. It is not clear how to construct such a database, even if U attains each row with rational probability (this is without loss of generality – see Lemma 22), or even with uniform distribution (this assumption would require justification, since $H(Q) \leq \alpha(Q)$ implies that for every entropy vector h there is a database $\mathbb{D}$ which achieves the same size-increase; we can then consider the uniform distribution $U_{\mathbb{D}}$ on the results $Q(\mathbb{D})$, as in Lemma 3). Our proof of Theorem 5 does not follow such a direct approach, but rather constructs a database from a system of finite groups constructed from the random variable U.

In Sections 5 and 6 we introduce a new parameter $\mathrm{GC}(Q)$ and prove the inequalities $\alpha(Q) \geq \mathrm{GC}(Q)$ and $\mathrm{GC}(Q) \geq H(Q)$. This, together with Lemma 15, will finish the proof of Theorem 5.

5 Lower bounds

In this section, we prove a lower bound $\alpha(Q) \geq \mathrm{GC}(Q)$. This bound will be obtained by a series of more and more refined constructions of databases. We start from recalling a construction using colorings due to Gottlob et al. [9]. We then improve this bound to vector space colorings, and finally, to group systems. In the entire Section 5, fix a natural join query Q over a schema Σ, and a set of functional dependencies $\mathcal{F}$.

5.1 Colorings

A *coloring* of Q is a function f assigning finite sets to $\mathbf{V}(Q)$. We say that f satisfies a functional dependency $X \mapsto x$ if $f(x) \subseteq f(X)$, where $f(X)$ denotes $\bigcup_{y \in X} f(y)$. For a coloring f of Q which satisfies all functional dependencies in $\mathcal{F}$, define

$$C(Q, f) = \frac{|f(\mathbf{V}(Q))|}{\max_{R \in \Sigma} |f(\mathbf{V}(R))|}, \tag{C}$$

and let $C(Q) = \sup_f C(Q, f)$.

The following proposition is proved in [9], and amounts to constructing a database $\mathbb{D}$ from a given coloring f. In the following section we will extend this construction to *vector space colorings.*

▶ **Proposition 16** ([9]). *Let Q be a natural join query. Then $\alpha(Q) \geq C(Q)$.*

It is shown in [9] that the value $C(Q)$ can be computed by a linear program. In the case without functional dependencies, this program is dual to the program for $\mathrm{AGM}(Q)$, so $C(Q) = \mathrm{AGM}(Q) = \alpha(Q)$.

5.2 Vector space colorings

In the presence of functional dependencies, there are queries Q for which $\alpha(Q) > C(Q)$, as shown in the paper [9], by elaborating an example proposed by Dániel Marx, and using Shamir's secret sharing scheme. Inspired by that construction and Blakley's secret sharing scheme, in this section we define a new parameter $\mathrm{VC}_{\mathbb{K}}(Q)$ based on vector spaces, and generalized later in Section 5.3 to $\mathrm{GC}(Q)$ based on arbitrary groups. We study the relations between these parameters and $C(Q)$. We note that the development of Section 5.2 and the inequalities involving the parameter $\mathrm{VC}_{\mathbb{K}}(Q)$ proved in Section 5.3 are not needed for proving $\alpha(Q) \geq \mathrm{GC}(Q)$ and our main result (Theorem 5), but we present them in order to relate our parameters $\mathrm{VC}_{\mathbb{K}}(Q)$ and later $\mathrm{GC}(Q)$ to the coloring number $C(Q)$, and also to provide a gradual introduction to the most general parameter $\mathrm{GC}(Q)$ defined in Section 5.3.

We consider vector spaces over a fixed finite field $\mathbb{K}$. If V is a vector space, X is a set, and V_x is a subspace of V for $x \in X$, then by $\sum_{x\in X} V_x$ we denote the smallest subspace of V containing every V_x, for $x \in X$ (we refer to [10] for background on vector spaces).

A *vector space coloring* of Q is a pair $\mathcal{V} = (V, (V_x)_{x\in\mathbf{V}(Q)})$, where V is a vector space and $(V_x)_{x\in\mathbf{V}(Q)}$ is a family of its subspaces. For such a coloring, define $V_Y = \sum_{x\in Y} V_x$ for $Y \subseteq \mathbf{V}(Q)$. We say that $\mathcal{V}$ *satisfies* a functional dependency $Y \mapsto x$ if $V_x \subseteq V_Y$. For a vector space coloring $\mathcal{V}$ satisfying all the functional dependencies in $\mathcal{F}$, define

$$\mathrm{VC}_{\mathbb{K}}(Q,\mathcal{V}) = \frac{\dim(V_{\mathbf{V}(Q)})}{\max_{R\in\Sigma}\dim(V_{\mathbf{V}(R)})}, \tag{VC}$$

Let $\mathrm{VC}_{\mathbb{K}}(Q) = \sup_{\mathcal{V}} \mathrm{VC}_{\mathbb{K}}(Q,\mathcal{V})$.

▶ **Proposition 17.** *Let Q be a natural join query. Then* $\mathrm{VC}_{\mathbb{K}}(Q) \geq C(Q)$.

Proof. The inequality $\mathrm{VC}_{\mathbb{K}}(Q) \geq C(Q)$ is obtained by defining for a coloring f a vector space coloring $\mathcal{V}$ with $V = \mathbb{K}^{Colors}$, $V_x = \mathbb{K}^{f(x)} \subseteq \mathbb{K}^{Colors}$, where $Colors = \bigcup_{x\in\mathbf{V}(Q)} f(x)$, and $\mathbb{K}^{f(x)}$ embeds into $\mathbb{K}^{Colors}$ in the natural way, by extending a vector with zeros on coordinates in $Colors - f(x)$. It is easy to see that $\mathcal{V}$ satisfies the same functional dependencies as f, and that $\mathrm{VC}_{\mathbb{K}}(Q,\mathcal{V}) = C(Q,f)$. This proves $\mathrm{VC}_{\mathbb{K}}(Q) \geq C(Q)$. ◀

5.3 Group systems

We relax the notion of a vector space coloring, by considering finite groups, as follows. Let G be a finite group and $(G_x)_{x\in\mathbf{V}(Q)}$ be a family of its subgroups. For a set of attributes $X \subseteq \mathbf{V}(Q)$, denote by G_X the group $G_X = \bigcap_{x\in X} G_x$, and by G/G_X the space of (left) cosets, $\{gG_X : g \in G\}$. We call the pair $\mathcal{G} = (G, (G_x)_{x\in X})$ a *group system* for Q, and say that it satisfies a functional dependency $X \mapsto x$ if $G_X \subseteq G_x$ (note the duality with respect to vector space colorings, with $\cup$ replaced by $\cap$ and $\subseteq$ replaced by $\supseteq$). For a group system $\mathcal{G}$ satisfying all the functional dependencies in $\mathcal{F}$, define

$$\mathrm{GC}(Q,\mathcal{G}) = \frac{\log|G/G_{\mathbf{V}(Q)}|}{\max_{R\in\Sigma}(\log|G/G_{\mathbf{V}(R)}|)}, \tag{GC}$$

and let $\mathrm{GC}(Q) = \sup_{\mathcal{G}} \mathrm{GC}(Q,\mathcal{G})$.

▶ **Proposition 18.** *Let Q be a natural join query. Then*

$$\alpha(Q) \geq \mathrm{GC}(Q) \geq \mathrm{VC}_{\mathbb{K}}(Q). \tag{8}$$

Proof. Call a group system $\mathcal{G}$ a *vector space system* if it consists of a vector space V, treated as a group with addition, and its subspaces $(V_x)_{x\in X}$, treated as subgroups of V. The second inequality in (8) is an immediate consequence of the following lemma, which is an application of vector space duality (see e.g. [10]).

▶ **Lemma 19.** *For every vector space coloring $\mathcal{V}$ there is a vector space system $\mathcal{V}^*$ satisfying the same functional dependencies as $\mathcal{V}$, and such that* $\mathrm{VC}_{\mathbb{K}}(Q,\mathcal{V}) = \mathrm{GC}(Q,\mathcal{V}^*)$.

Proof. If V is a vector space, let V^* denote its algebraic dual, i.e., the space of all linear functions from V to $\mathbb{K}$ (with addition and multiplication by scalars defined coordinatewisely). If L is a subspace of V, then let $L^\perp \subseteq V^*$ denote the set of functionals $f \in V^*$ which vanish on L, i.e., $f(L) \subseteq \{0\}$. For a vector space coloring $\mathcal{V} = (V,(V_x)_{x\in\mathbf{V}(Q)})$ let $\mathcal{V}^* = (V^*,(V_x^\perp)_{x\in\mathbf{V}(Q)})$. Using the standard facts $(L_1\cap L_2)^\perp = L_1^\perp + L_2^\perp$ and $L_1 \subseteq L_2$ iff $L_1^\perp \supseteq L_2^\perp$, one easily checks that $\mathcal{V}^*$ satisfies the same functional dependencies as $\mathcal{V}$. Furthermore, from $\dim V = \dim V^*$ and

$$\dim(L^\perp) = \dim_V - \dim L = \log|V|/\log|\mathbb{K}| - \log|L|/\log|\mathbb{K}| = \log|V/L|/\log|\mathbb{K}|,$$

$\mathrm{VC}_{\mathbb{K}}(Q,\mathcal{V}) = \mathrm{GC}(Q,\mathcal{V}^*)$ follows easily. ◀

It remains to prove the inequality $\alpha(Q) \geq \mathrm{GC}(Q)$. To this end, from a group system $\mathcal{G}$ we construct a database $\mathbb{D}$ satisfying the same functional dependencies, and such that

$$\alpha(Q,\mathbb{D}) = \mathrm{GC}(Q,\mathcal{G}). \tag{9}$$

For a relation name R and an element $g \in G$, let r_g be the row such that $r_g[x] = gG_x \in G/G_x$ for every attribute $x \in \mathbf{V}(R)$. Define $\mathbb{D}$ by setting $R(\mathbb{D}) = \{r_g : g \in G\}$, for every relation name R. The following lemma implies immediately that the database $\mathbb{D}$ satisfies the same functional dependencies as $\mathcal{G}$.

▶ **Lemma 20.** *Fix a group G, a family $(G_x)_{x\in X}$ of subgroups of G, and a subgroup $G_0 \subseteq G$ such that $G_0 \supseteq \bigcap_{x\in X} G_x$. For any given $g \in G$, the cosets $(gG_x)_{x\in X}$ determine gG_0, i.e., for every two elements $g,h \in G$, if $gG_x = hG_x$ for all $x \in X$, then $gG_0 = hG_0$.*

Proof. If G_1 is a subgroup of G_0 then $gG_1 \subseteq gG_0$, and gG_0 is determined by gG_1 as follows: $gG_0 = \{k\cdot h : h \in gG_1, h \in G_0\}$. Applying this observation to $G_1 = \bigcap_{x\in X} G_x$ yields that gG_0 is determined by $g(\bigcap_{x\in X} G_x) = \bigcap_{x\in X}(gG_x)$. ◀

Next we show (9). For each relation name R, consider the mapping $f_R : G \to R(\mathbb{D})$, where $f_R(g) = r_g$. Clearly, the mapping is onto $R(\mathbb{D})$. To compute the size of its image, we analyse the kernel of f_R and observe that $\{h : r_g = r_h\} = gG_{\mathbf{V}(R)}$ for every $g \in G$. In particular, $|R(\mathbb{D})| = |G/G_{\mathbf{V}(R)}|$. Similarly, we verify that $|Q(\mathbb{D})| = |G/G_{\mathbf{V}(Q)}|$. Equation (9) then follows. By Lemma 1, this proves $\alpha(Q) \geq \mathrm{GC}(Q)$. ◀

▶ Remark. The database $\mathbb{D}$ constructed in the above proof is G-symmetric. Indeed, the values appearing in the database are cosets the form gG_x (where $x \in \mathbf{V}(Q)$) and G acts (from the left) on such cosets in the obvious way. Moreover, the action of G on each table $R(\mathbb{D})$ is isomorphic to the action of G on $G/G_{\mathbf{V}(R)}$, in particular, it is transitive. Also, if $\mathbb{D}$ is G-symmetric and $\mathbb{D}^n$ is defined as in the proof of Lemma 1, then $\mathbb{D}^n$ is G^n-symmetric.

6 Tightness

Lemma 15 and Proposition 18 show that $H(Q) \geq \alpha(Q) \geq \mathrm{GC}(Q)$ for every natural join query Q, in the presence of functional dependencies. In this section, we close the circle by proving the following.

▶ **Proposition 21.** *Let Q be a natural join query. Then* $\mathrm{GC}(Q) \geq H(Q)$.

Together with Proposition 18 and Lemma 15, this proves that $H(Q) = \alpha(Q) = \mathrm{GC}(Q)$. As noted in Proposition 14, this gives Theorem 5. Moreover, from the observation in Remark 5.3, it follows that the bound $\alpha(Q)$ can be approximated by (arbitrarily large) symmetric databases, proving Theorem 6.

All the necessary ideas to prove the proposition are present in the proof of the result of Chan and Yeung [5]. However, we cannot directly apply that theorem, since we need to keep track of the functional dependencies. In the rest of Section 6, we present a self-contained proof of Proposition 21, following [14].

For the rest of this section, fix a random variable U, taking values in a finite set of rows with attributes X, and satisfying the functional dependencies $\mathcal{F}$.

We say that a random variable V is *rational* if for every value $v \in \mathrm{Im}(V)$, the probability that V achieves v is a rational number. The following lemma follows easily from the density of rationals among the real numbers.

▶ **Lemma 22.** *For every number $\varepsilon > 0$ there exists a rational random variable V satisfying the same functional dependencies as U, and such that $\|h^V - h^U\| < \varepsilon$ with respect to the euclidean norm on $\mathbb{R}^{P(X)}$.*

To prove Proposition 21, we proceed as follows. For each rational random variable U satisfying the given functional dependencies, we will find a sequence of group systems $\mathcal{G}_k$ satisfying the functional dependencies $\mathcal{F}$, and such that

$$\lim_{k\to\infty} \mathrm{GC}(Q, \mathcal{G}_k) = H(Q, U). \tag{10}$$

From that, Proposition 21 follows:

$$H(Q) \overset{Prop. 14}{=} \sup_U H(Q, U) \overset{Lem. 22}{=} \sup_{U \text{ rational}} H(Q, U) \overset{(10)}{\leq} \sup_{\mathcal{G}} \mathrm{GC}(Q, \mathcal{G}) = \mathrm{GC}(Q).$$

From now on, let U be a rational random variable satisfying the functional dependencies $\mathcal{F}$. We will show that there exists a sequence of group systems witnessing (10).

Let $q \in \mathbb{N}$ be a natural number such that for each row $r \in \mathrm{Im}(U)$ the probability $\mathbb{P}[U = r]$ can be represented as a rational number with denominator q. For $k = q, 2q, 3q, \ldots$ let A_k be a matrix whose columns are indexed by attributes, containing exactly $k \cdot \mathbb{P}[U = r]$ copies of the row r, for every row $r \in \mathrm{Im}(U)$. Notice that $k \cdot \mathbb{P}[U = r]$ is always a natural number, and that A_k has exactly k rows in total.

Let G^k denote the group of all permutations of the rows of A_k. For a set of attributes $Y \subseteq X$, let G^k_Y denote the subgroup of G^k which stabilizes the submatrix $A_k[Y]$ of A_k, i.e.,

$$G^k_Y = \{\sigma \in G^k \mid \sigma(r)[y] = r[y] \text{ for each } r \in A_k \text{ and } y \in Y\}.$$

Denote $G^k_{\{x\}}$ by G^k_x. In particular, $G^k_Y = \bigcap_{y \in Y} G^k_y$. The following lemma is immediate.

▶ **Lemma 23.** *Suppose that U satisfies the functional dependency $Y \mapsto x$. Then $G^k_Y \subseteq G^k_x$.*

We define $\mathcal{G}_k$ as $(G^k, (G^k_x)_{x\in X})$. By Lemma 23, $\mathcal{G}_k$ is a group system satisfying the functional dependencies $\mathcal{F}$. It remains to prove (10).

Fix a set of attributes $Y \subseteq X$. For a row $r \in \mathrm{Im}(U[Y])$, let p_r denote $\mathbb{P}[U[Y] = r]$. Since r occurs exactly $k \cdot p_r$ times as a row of $A_k[Y]$, it follows that G^k_Y is isomorphic to the product of symmetric groups $\prod_{r\in\mathrm{Im}(U[Y])} S_{k\cdot p_r}$ (where $S_{k\cdot p_r}$ is the symmetric group of all permutations of the copies of the row r). In particular, $|G^k_Y| = \prod_{r\in\mathrm{Im}(U[Y])}(k \cdot p_r)!$. Using Stirling's approximation we get the following ($a_k \sim b_k$ signifies $\lim_{k\to\infty} a_k/b_k = 1$):

$$\log\left(\frac{|G^k|}{|G^k_Y|}\right) = \log\left(\frac{k!}{\prod_{r\in\mathrm{Im}(U[Y])}(k\cdot p_r)!}\right) \sim \left(k\log(k) - \sum_{r\in\mathrm{Im}(U[Y])}(k\cdot p_r)\log(k\cdot p_r)\right) \sim$$
$$- \sum_{r\in\mathrm{Im}(U[Y])}(k\cdot p_r)(\log(k\cdot p_r - \log k)) \ \sim\ (-k)\cdot \sum_{r\in\mathrm{Im}(U[Y])} p_r\log p_r \ \sim\ k\cdot H(U[Y]).$$

In particular,

$$\mathrm{GC}(Q, \mathcal{G}^k) = \frac{\log(|G^k|/|G^k_X|)}{\max_{R\in Q}\log(|G^k|/|G^k_{\mathbf{V}(R)}|)} \xrightarrow{k\to\infty} \frac{H(U)}{\max_{R\in Q} H(U[\mathbf{V}(R)])} = H(Q, U).$$

This yields (10), proving Proposition 21, which together with Lemma 15 and Proposition 18 gives $H(Q) = \mathrm{GC}(Q) = \alpha(Q)$ and finishes the proof of Theorem 5. The more general Theorem 7 is proved similarly. Moreover, Theorem 6 follows from Remark 5.3.

7 General conjunctive queries

The previous sections concern natural join queries: conjunctive queries without existential quantifiers (or projections), in which the variables name coincides with the name of the attribute of the relation in which it appears (in particular, the same variable name cannot occur in the scope of one conjunct, and there are no equalities). In this section, we discuss how to treat arbitrary conjunctive queries.

7.1 Set semantics

Define a *natural conjunctive query* to be a query of the form $Q = \exists Y\ Q'$, where Q' is a natural join query over the schema Σ, and $Y \subseteq \mathbf{V}(\Sigma)$. The set of *free variables* of Q is $\mathbf{V}(Q) = \mathbf{V}(\Sigma) - Y$. For a database $\mathbb{D}$ over the schema Σ, define $Q(\mathbb{D})$ as $T[\mathbf{V}(Q)]$, where $T = Q'(\mathbb{D})$. In other words, $Q(\mathbb{D})$ is the table $Q'(\mathbb{D})$ restricted to the free variables of Q. This is the so-called *set-semantics*, since the result $T[\mathbf{V}(Q)]$ is a set of rows, i.e., each row occurs either 0 or 1 times.

As explained in [9] for each conjunctive query Q there exists a natural conjunctive query S, such that $\alpha(Q) = \alpha(S)$. Such S can be constructed in a purely syntactical way from Q by the chase procedure. Because of this, we only consider natural conjunctive queries.

The definition of $\alpha(Q)$ can be lifted without modification to natural conjunctive queries. The generalization of Theorem 5 has the expected form:

▶ **Theorem 24.** *Fix a relational schema Σ and a set of functional dependencies $\mathcal{F}$. Let Q be a natural conjunctive query over the schema Σ. Then $\alpha(Q)$ is equal to the maximal value of $v_{\mathbf{V}(Q)}$, for v ranging over $\overline{\Gamma^*_{\mathbf{V}(\Sigma)}}$ and satisfying:*

$$\begin{cases} v_{\mathbf{V}(R)} \le 1 & \text{for } R \in Q,\\ v_{Z\cup\{z\}} = v_Z & \text{for every fd } Z \mapsto z \text{ in } \mathcal{F}.\end{cases}$$

The proof of the lower bound is exactly the same as the proof of Theorem 5. However the proof of the upper bound has to be modified slightly, as described below.

For a query $Q = \exists Y.\ Q'$, define $H(Q,U)$ and $H(Q)$ as in Section 2. We then have the following analogue of Lemma 15, proving the upper bound.

▶ **Lemma 25.** *For a natural conjunctive query* $Q = \exists Y.\ Q'$, $\alpha(Q) \leq H(Q)$.

Proof. For a database $\mathbb{D}$, let $U_{\mathbb{D}}$ be the random variable with values in $Q'(\mathbb{D})$, described as the result r' of the following process: first choose uniformly at random a row $r \in Q(\mathbb{D})$, and then, choose uniformly at random a row $r' \in Q'(\mathbb{D})$ such that $r'[\mathbf{V}(Q)] = r$.

By definition, the random variable $U_{\mathbb{D}}[\mathbf{V}(Q)]$ is uniformly distributed on $Q(\mathbb{D})$. Now we get that:

$$\begin{aligned}\alpha(Q,\mathbb{D}) &= \frac{\log |Q(\mathbb{D})|}{\max_{R\in Q'}\ \log |R(\mathbb{D})|} = \frac{H(U_{\mathbb{D}}[\mathbf{V}(Q)])}{\max_{R\in Q'}\ \log |R(\mathbb{D})|} \\ &\leq \frac{H(U_{\mathbb{D}}[\mathbf{V}(Q)])}{\max_{R\in Q'}\ H(U_{\mathbb{D}}[\mathbf{V}(R)])} = H(Q, U_{\mathbb{D}}).\end{aligned}$$

By a similar argument as in the proof of Lemma 15, we derive that $\alpha(Q) \leq H(Q)$. ◀

8 Evaluation

In this section, we give some rudimentary results bounding the worst-case complexity of computing the result of a query $Q(\mathbb{D})$, for a given database $\mathbb{D}$. In the absence of functional dependencies, it was originally shown in [1] that $Q(\mathbb{D})$ can be computed from $\mathbb{D}$ in time proportional to $|\mathbb{D}|^{\alpha(Q)+1}$, which was later improved to $|\mathbb{D}|^{\alpha(Q)}$ by Ngo et al. [15]. In the bounds presented below, there is an additional factor $|\mathbb{D}|^m$, where m is a parameter depending on the functional dependencies, defined below. Recently, Khamis et al. [11] showed how to compute the result in time $\tilde{O}(|\mathbb{D}|^{h(Q)})$ in the presence of functional dependencies, i.e., in almost optimal time assuming that the bound $\alpha(Q) \leq h(Q)$ of Gottlob et al. [9] is tight.

Let $\mathcal{F}$ be a set of functional dependencies over attributes X. A *minimal component* C is an inclusion-minimal nonempty set of attributes $C \subseteq X$ with the property that whenever $Y \mapsto x$ is a functional dependency with $Y \cap C$ nonempty, then $x \in C$. For a set $X' \subseteq X$, define $\mathcal{F}[X']$ to be the set of functional dependencies over X' which consists of those functional dependencies $Y \mapsto x$ from $\mathcal{F}$, such that $Y \subseteq X'$.

We say that a set of attributes $S \subseteq X$ *spans* $\mathcal{F}$, if the smallest subset $\bar{S}$ of X containing S and closed under functional dependencies (i.e., $Y \mapsto x$ and $Y \subseteq \bar{S}$ implies $x \in \bar{S}$) is equal to X. We say that $\mathcal{F}$ has *width* m if it has a set of size at most m which spans it. We inductively define the *iterative width* by saying that $\mathcal{F}$ has iterative width m if either the set of attributes is empty, or for every its minimal component C, $\mathcal{F}[C]$ has width m, and if M is the set of attributes which belong to the minimal components, then $\mathcal{F}[X - M]$ also has iterative width m.

▶ **Example 26.** If $\mathcal{F}$ is an empty set of functional dependencies over a nonempty set of attributes, then $\mathcal{F}$ has iterative width 1. The set of dependencies $x \mapsto y, y \mapsto z, z \mapsto x$ has width 1, since it is spanned by $\{x\}$. It also has iterative width 1. Finally, let $X = \{x, y, z\}$ and $\mathcal{F}$ consist of the functional dependency $\{x, y\} \mapsto z$. Then $\mathcal{F}$ has width 2, since the set $\{x, y\}$ spans it. The only minimal component is $\{z\}$, and $\mathcal{F}[\{z\}]$ and $\mathcal{F}[\{x, y\}]$ have width 1 and 2, respectively. Hence $\mathcal{F}$ has iterative width 2.

Let $\mathbb{D}$ be a database and let C be a minimal component. For a row r over attributes X, denote by r/C the row with attributes X such that $r/C[x] = r[x]$ for $x \in X - C$ and $r/C[x] = x$ for $x \in X \cap C$. Therefore, r/C is obtained by replacing each value of an attribute in C by a placeholder, storing the name of the attribute. Denote by $\mathbb{D}/C$ the database obtained from $\mathbb{D}$ by replacing in each table R, every row r by the row r/C. Clearly, we have the following.

▶ **Lemma 27.** *The database $\mathbb{D}/C$ can be computed from $\mathbb{D}$ in linear time.*

The following lemma is immediate, by the fact that C is a minimal component.

▶ **Lemma 28.** *The database $\mathbb{D}/C$ satisfies all the functional dependencies of Q.*

Note, however, that the database $\mathbb{D}/C$ usually satisfies more functional dependencies than $\mathbb{D}$, namely, it satisfies all functional dependencies $\emptyset \mapsto x$, for $x \in C$.

▶ **Lemma 29.** *Let C be a minimal component, and suppose that $\mathcal{F}[C]$ has width m. Then, for a given database $\mathbb{D}$, the result $Q(\mathbb{D})$ can be computed from $Q(\mathbb{D}/C)$ in time $O(|Q(\mathbb{D}/C)| \cdot |\mathbb{D}|^m)$.*

Very roughly, the algorithm proceeds by computing, for all $s \in Q(\mathbb{D}/C)$ and all rows r over a spanning set for C and with admissible values, the row compatible with s and r (if it exists), and adding it to the result. We omit the details, due to lack of space.

By iteratively applying Lemma 29, we obtain the main result of this section.

▶ **Proposition 30.** *Fix a natural join query Q and functional dependencies $\mathcal{F}$ of iterative width m. There is an algorithm which for a given database $\mathbb{D}$ computes $Q(\mathbb{D})$ in time $O(|\mathbb{D}|^{\alpha(Q)+m})$.*

Observe that when the set $\mathcal{F}$ of functional dependencies is empty, Proposition 30 gives the algorithm from [1], whose running time is $O(|\mathbb{D}|^{\alpha(Q)+1})$, since then $\mathcal{F}$ has iterative width 1.

9 Conclusion and future work

We characterized the worst-case size-increase for the evaluation of conjunctive queries, in two ways: in terms of entropy and in terms of finite groups. Our construction improves a construction from [9]. We also presented a rudimentary result concerning the evaluation of natural join queries.

We see several directions of a possible future work. The first direction is to try to prove tightness of the bound $\alpha(Q) \leq h(Q)$ from Corollary 8, due to [9]. This would yield computability of $\alpha(Q)$, as $h(Q)$ can be computed using linear programming. Moreover, together with the recent results of Khamis et al. [11], this would give an almost optimal $\tilde{O}(|\mathbb{D}|^{\alpha(Q)})$ algorithm for computing the results of joins.

Computing $H(Q)$ directly looks hard, and probably would require a deeper understanding of entropy, and the entropy cone in particular. By comparison, we note that in cryptography and information theory the seemingly similar optimization problem of finding the optimal *information rate* in *access structures* [3] (a security system which can be used as a secret sharing scheme) is considered notoriously difficult [6]. This demonstrates that optimization problems over the entropy cone can be very difficult.

The fractional edge cover was useful in the analysis of the Hypercube algorithm [2], an algorithm for parallel evaluation of queries. Perhaps some ideas from the current paper can also be applied in the parallel setting.

Acknowledgements. We are grateful to Dan Suciu for introducing the problem of computing the worst-case size bound increase to us, to Adam Witkowski for fruitful discussions, and to anonymous reviewers.

References

1 Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. *SIAM J. Comput.*, 42(4):1737–1767, 2013. doi:10.1137/110859440.

2 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 273–284, 2013. doi:10.1145/2463664.2465224.

3 Ernest F. Brickell and Daniel M. Davenport. On the classification of ideal secret sharing schemes. *Journal of Cryptology*, 4(2):123–134, 1991. doi:10.1007/BF00196772.

4 Terence H. Chan. Group characterizable entropy functions. *CoRR*, abs/cs/0702064, 2007. URL: http://arxiv.org/abs/cs/0702064.

5 Terence H. Chan and Raymond W. Yeung. On a relation between information inequalities and group theory. *IEEE Transactions on Information Theory*, 48(7):1992–1995, 2002.

6 Oriol Farràs, Jessica Ruth Metcalf-Burton, Carles Padró, and Leonor Vázquez. On the optimization of bipartite secret sharing schemes. *Des. Codes Cryptography*, 63(2):255–271, May 2012. doi:10.1007/s10623-011-9552-7.

7 Ehud Friedgut. Hypergraphs, entropy, and inequalities. *The American Mathematical Monthly*, 111(9):749–760, 2004. URL: http://www.jstor.org/stable/4145187.

8 Tomasz Gogacz and Szymon Toruńczyk. Entropy bounds for conjunctive queries with functional dependencies. *CoRR*, abs/1512.01808, 2015. URL: http://arxiv.org/abs/1512.01808.

9 Georg Gottlob, Stephanie Tien Lee, Gregory Valiant, and Paul Valiant. Size and treewidth bounds for conjunctive queries. *J. ACM*, 59(3):16, 2012. doi:10.1145/2220357.2220363.

10 Paul Halmos. *Finite-Dimensional Vector Spaces*. Springer, 1974.

11 Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. Computing join queries with functional dependencies. In Tova Milo and Wang-Chiew Tan, editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 327–342. ACM, 2016. doi:10.1145/2902251.2902289.

12 Serge Lang. *Algebra*. Addison-Wesley, Menlo Park Cal, 1993. URL: http://opac.inria.fr/record=b1081613.

13 Laszló Lovász. Submodular functions and convexity. *Mathematical programming: the state of the art*, 1983. URL: http://www.cs.elte.hu/~{}lovasz/scans/submodular.pdf.

14 Desmond S. Lun. A relationship between information inequalities and group theory, 2002.

15 Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. Worst-case optimal join algorithms: [extended abstract]. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 37–48, 2012. doi:10.1145/2213556.2213565.

16 Zhen Zhang and Raymond W. Yeung. On characterization of entropy function via information inequalities. *IEEE Transactions on Information Theory*, 44(4):1440–1452, 1998. doi:10.1109/18.681320.