What do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog have to do with one another?

Mahmoud Abo Khamis
LogicBlox Inc.

Hung Q. Ngo
LogicBlox Inc.

Dan Suciu
LogicBlox Inc. and
University of Washington

ABSTRACT

Recent works on bounding the output size of a conjunctive query with functional dependencies and degree bounds have shown a deep connection between fundamental questions in information theory and database theory. We prove analogous output bounds for *disjunctive datalog rules*, and answer several open questions regarding the tightness and looseness of these bounds along the way. The bounds are intimately related to Shannon-type information inequalities. We devise the notion of a "proof sequence" of a specific class of Shannon-type information inequalities called "Shannon flow inequalities". We then show how a proof sequence can be used as symbolic instructions to guide an algorithm called PANDA, which answers disjunctive datalog rules within the size bound predicted. We show that PANDA can be used as a black-box to devise algorithms matching precisely the fractional hypertree width and the submodular width runtimes for aggregate and conjunctive queries *with* functional dependencies and degree bounds.

Our results improve upon known results in three ways. First, our bounds and algorithms are for the much more general class of disjunctive datalog rules, of which conjunctive queries are a special case. Second, the runtime of PANDA matches precisely the submodular width bound, while the previous algorithm by Marx has a runtime that is polynomial in this bound. Third, our bounds and algorithms work for queries with input cardinality bounds, functional dependencies, *and* degree bounds.

Overall, our results showed a deep connection between three seemingly unrelated lines of research; and, our results on proof sequences for Shannon flow inequalities might be of independent interest.

Keywords

Submodular width; Disjunctive datalog; Shannon-type inequalities; Entropy; Functional dependencies; Restricted access patterns; Degree bounds; Join algorithms

PODS'17, May 14 - 19, 2017, Chicago, IL, USA

ISBN 978-1-4503-4198-1/17/05…$15.00

DOI: http://dx.doi.org/10.1145/3034786.3056105

1. INTRODUCTION

In this paper we answer four major questions that resulted from four different research threads, and establish new connections between those threads.

1.1 Size Bound for Full Conjunctive Queries

Grohe and Marx [30], Atserias, Grohe, and Marx [13], and Gottlob, Lee, Valiant and Valiant [27] developed a deep connection between the output size bound of a conjunctive query with (or without) functional dependencies (FD) and information theory. Our first problem is to extend this bound to degree constraints, and to study whether the bound is tight.

We associate a full conjunctive query Q to a hypergraph $\mathcal{H} = ([n], \mathcal{E})$, $\mathcal{E} \subseteq 2^{[n]}$. The query's variables are A_i, $i \in [n]$. Its atoms are $R_F, F \in \mathcal{E}$. The query is:

$$Q(\mathbf{A}_{[n]}) \coloneq \bigwedge_{F\in\mathcal{E}} R_F(\mathbf{A}_F), \tag{1}$$

where $\mathbf{A}_J$ denotes the set $\{A_j \mid j \in J\}$, for $J \subseteq [n]$. Our goal is to compute an upper bound on the output size, when the input database satisfies a set of *degree constraints*. Define $\deg_F(\mathbf{A}_Y|\mathbf{A}_X) \stackrel{\text{def}}{=} \max_{\mathbf{t}} |\Pi_{\mathbf{A}_Y}(\sigma_{\mathbf{A}_X=\mathbf{t}}(R_F))|$; then, a degree constraint is an assertion of the form $\deg_F(\mathbf{A}_Y|\mathbf{A}_X) \le N_{Y|X}$, for $X \subset Y \subseteq F$, A *cardinality constraint* is an assertion of the form $|R_F| \le N_F$, for some $F \in \mathcal{E}$; it is exactly the degree constraint $\deg_F(\mathbf{A}_F|\emptyset) \le N_{F|\emptyset} \stackrel{\text{def}}{=} N_F$. A *functional dependency* $\mathbf{A}_X \to \mathbf{A}_Y$ is a degree constraint with $N_{X\cup Y|X} = 1$. Thus, degree constraints strictly generalize both cardinality constraints and FDs.

The first output size upper bound was pioneered in [13,30], who established a tight bound, for cardinality constraints only, known today as the AGM bound. Extensions to FDs and degree constraints were discussed in [27] and [3], respectively, who left open the question whether these bounds are tight. Handling queries with degree constraints has a strong practical motivation. Armbrust et al. [10–12], described a new approach to query evaluation, called *scale-independent query processing*, which guarantees a fixed runtime even when the size of the database increases without bound; this guarantee is provided by asking developers to write explicit degree constraints, then using heuristics to derive upper bounds on the query output. Thus, improved upper bounds on the size of the query answer have immediate applications to scale-independent query processing. Several complexity results on the associated decision problem ("is the output size of the query bounded?") were considered in [15–17].

Our first question is whether the upper bounds for FDs or degree constraints are tight. To set the technical context for this question, we briefly describe how the bound was derived. Fix an input database $\mathbf{D}$ and consider the joint distribution on random variables $\mathbf{A}_{[n]}$ where each output tuple $\mathbf{t} \in Q(\mathbf{D})$ is selected uniformly with probability $1/|Q(\mathbf{D})|$. Let $h(\mathbf{A}_S)$ denote the marginal entropy on the variables $\mathbf{A}_S$. Then[1], by uniformity $h(\mathbf{A}_{[n]}) = \log|Q(\mathbf{D})|$, and $h(\mathbf{A}_Y|\mathbf{A}_X) \leq \log N_{Y|X}$ for every degree constraint. A function $h : 2^{\mathbf{A}_{[n]}} \to \mathbb{R}_+$ is said to be *entropic* if there is a joint distribution on $\mathbf{A}_{[n]}$ such that $h(\mathbf{A}_S)$ is the marginal entropy on $\mathbf{A}_S$, $S \subseteq [n]$. The *entropic bound* of a query is $\log|Q| \leq \max_h h(\mathbf{A}_{[n]})$, where h ranges over all entropic functions satisfying the given constraints. Recently, Gogacz and Toruńczyk [25] showed that the entropic bound is tight even in the presence of FDs. However, they did not address general degree constraints.

The problem with the entropic bound is that we do not know how to compute it (except for the special case when all degree constraints are cardinality constraints), because the entropic cone is characterized by infinitely many *non-Shannon-type inequalities* [33, 38]. To overcome this limitation, Gottlob et al. [27] replace entropic functions (which are difficult) with polymatroids (which are easier). A *polymatroid* is a set function $h : 2^{[n]} \to \mathbb{R}_+$ that is non-negative, monotone, and submodular, with $h(\emptyset) = 0$ (see Sec 2). Every entropic function h is also a polymatroid, if we write $h(S)$ for $h(\mathbf{A}_S)$. (We will use $h(S)$ and $h(\mathbf{A}_S)$ interchangeably in this paper, depending on context.) Linear inequalities satisfied by all polymatroids are called *Shannon-type inequalities* [38]. The *polymatroid bound* of a full conjunctive query is $\log|Q| \leq \max_h h(\mathbf{A}_{[n]})$, where h ranges over all polymatroids satisfying the given constraints. The polymatroid bound, while at least as large as the entropic bound, is known to be tight for cardinality constraints, because the AGM bound is a polymatroid bound for cardinality constraints (see Prop. B.1) and it is tight [13]. The polymatroid bound is also tight for cardinality constraints with certain sets of FDs [3]. We ask whether it is tight in more general settings:

Question 1. Is the polymatroid bound tight for general degree constraints? Or, at least for queries with both cardinality and FD constraints?

Example 1.1. Consider the query:

$$Q(A_1, A_2, A_3, A_4) \text{ :- } R_{12}(A_1, A_2), R_{23}(A_2, A_3), R_{34}(A_3, A_4), R_{41}(A_4, A_1) \quad (2)$$

Assuming all input relations have size $\leq N$, then (a) the AGM bound is $|Q| \leq N^2$, (b) if we add the degree constraints $\deg_{12}(A_1A_2|A_1) \leq D$ and $\deg_{12}(A_1A_2|A_2) \leq D$ for some integer $D \leq \sqrt{N}$ then $|Q| \leq D \cdot N^{3/2}$, and (c) if we replace the degree constraints with FDs $A_1 \to A_2$ and $A_2 \to A_1$ the bound reduces further to $|Q| \leq N^{3/2}$. These bounds can be proven using only Shannon inequalities, thus they are polymatroid bounds. They are also asymptotically tight (see [4]).

Answer 1. The polymatroid bound is *not* tight! By adding a variable to a non-Shannon inequality by Zhang-Yeung [39] and constructing accordingly a database instance, we prove in Appendix A the following.

[1]All logs are in base 2, unless otherwise stated.

Theorem 1.2. *For any integer $s > 0$, there exists a query Q of size $\Theta(s)$, with cardinality and FD constraints, such that the ratio between the polymatroid bound and the entropic bound is $\geq N^s$, where N is the size of the database.*

1.2 Size Bound for Disjunctive Datalog Rules

Disjunctive datalog [9, 21] is a powerful extension of datalog. In this paper we are interested in a single disjunctive-datalog rule:

$$P : \quad \bigvee_{B \in \mathcal{B}} T_B(\mathbf{A}_B) \text{ :- } \bigwedge_{F \in \mathcal{E}} R_F(\mathbf{A}_F) \quad (3)$$

The body is similar to that of a conjunctive query, while the head is a disjunction of output predicates T_B, which we call *targets*. Given a database instance $\mathbf{D}$, a *model* of P is a tuple $\mathbf{T} = (T_B)_{B \in \mathcal{B}}$ of relations, one for each target, such that the logical implication indicated by the rule holds. More precisely, for any tuple $\mathbf{t}$, if $\pi_F(\mathbf{t}) \in R_F$ for every input relation R_F, then there exists a target $T_B \in \mathbf{T}$ such that $\pi_B(\mathbf{t}) \in T_B$. We write $\mathbf{T} \models P$ to denote the fact that $\mathbf{T}$ is a model. Define the size of a model to be $\max_B |T_B|$, and define the *output size* of P to be the minimum size over all models:

$$|P(\mathbf{D})| \stackrel{\text{def}}{=} \min_{\mathbf{T}:\mathbf{T} \models P} \max_{B \in \mathcal{B}} |T_B| \quad (4)$$

Our second question is to find an output size upper bound for a disjunctive datalog rule. If the rule has a single target then it becomes a conjunctive query: a model is any superset of the answer, and the output size is the standard size of the query's answer. We thus expect the upper bound to come in two flavors, entropic and polymatroid, as for full conjunctive queries.

Question 2. Find the entropic and polymatroid output size bounds of a disjunctive datalog rule, under given cardinality constraints, and more generally under degree constraints. Determine if it is tight.

Example 1.3. Consider the disjunctive datalog rule:

$$P : \; T_{123}(A_1, A_2, A_3) \vee T_{234}(A_2, A_3, A_4) \text{ :- } R_{12}(A_1, A_2), R_{23}(A_2, A_3), R_{34}(A_3, A_4), R_{41}(A_4, A_1)$$

A model of size N^3 can be obtained by populating the target T_{123} with all triples obtained from the active domain, but we will show that $|P(\mathbf{D})| \leq N^{3/2}$, $\forall \mathbf{D}$.

Answer 2. To describe the answer to the second question, we recall some standard notations [38]. We identify set-functions $h : 2^{[n]} \to \mathbb{R}_+$ with vectors in $\mathbb{R}_+^{2^n}$, and we use both $h(\mathbf{A}_S)$ and $h(S)$ to denote h_S, where $A_1, A_2, \ldots$ are (random) variables. (The reason being, $h(\mathbf{A}_S)$ is more apt for marginal entropies, and $h(S)$ is more apt for polymatroids.) The sets $\Gamma_n^* \subset \overline{\Gamma}_n^* \subset \Gamma_n \subset \mathbb{R}_+^{2^n}$ denote the set of entropic functions, its topological closure, and the set of polymatroids. We encode degree constraints by a set DC of triples $(X, Y, N_{Y|X})$, specifying $\deg_F(\mathbf{A}_Y|\mathbf{A}_X) \leq N_{Y|X}$. Define

$$\mathsf{HDC} \stackrel{\text{def}}{=} \left\{ h \mid \bigwedge_{(X,Y,N_{Y|X}) \in \mathsf{DC}} h(Y|X) \leq \log N_{Y|X} \right\} \quad (5)$$

to be the collection of set functions satisfying those constraints, where $h(Y|X) \stackrel{\text{def}}{=} h(Y) - h(X)$. Fix a closed subset

$\mathcal{F} \subseteq \mathbb{R}_+^{2^n}$. Define the log-*size-bound* of a disjunctive datalog rule P w.r.t. $\mathcal{F}$ to be the quantity:

$$\mathsf{LogSizeBound}_{\mathcal{F}}(P) \stackrel{\text{def}}{=} \max_{h \in \mathcal{F}} \min_{B \in \mathcal{B}} h(B). \tag{6}$$

The following is our second result, whose proof can be found Appendix A and [4].

Theorem 1.4. *Let P be any disjunctive datalog rule (3), and DC be given degree constraints.*

(i) For any database instance $\mathbf{D}$ satisfying all constraints in DC, the following holds:

$$\log|P(\mathbf{D})| \leq \underbrace{\mathsf{LogSizeBound}_{\overline{\Gamma}_n^* \cap \mathsf{HDC}}(P)}_{\textit{entropic bound}} \tag{7}$$

$$\leq \underbrace{\mathsf{LogSizeBound}_{\Gamma_n \cap \mathsf{HDC}}(P)}_{\textit{polymatroid bound}} \tag{8}$$

(ii) The entropic bound above is asymptotically tight.

Eq.(7) and (8) generalize the entropic and polymatroid bounds from full conjunctive queries to arbitrary disjunctive datalog rules (see Prop. B.1). The tightness result (*ii*) generalizes the main result in [25], which states that the entropic bound is asymptotically tight for full conjunctive queries under FDs.

1.3 Worst-case Optimal Algorithms

A *worst-case optimal algorithm* is an algorithm for computing a query in time bounded by its size bound. Such algorithms are known for full conjunctive queries under cardinality constraints [1, 34, 35, 37] and FDs [3]. Our next problem is finding a worst-case optimal algorithm for a disjunctive datalog rule P, under degree constraints. More precisely, given an input database $\mathbf{D}$ satisfying the given constraints, compute a model $\mathbf{T}$ in time no larger than the worst-case bound for $|P(\mathbf{D})|$ under those constraints. Notice that we allow the algorithm to compute *any* model. A conjunctive query Q is a single-target disjunctive datalog rule P_Q. If Q is full, then from any model $\mathbf{T}$ of P_Q we can answer Q by semijoin-reducing $\mathbf{T}$ with each input relation. Thus, any algorithm evaluating disjunctive rules can also be used to answer a full conjunctive query. However, this does not hold for non-full queries. For example, if Q is Boolean, then P_Q has a single target $T_\emptyset()$, and can be answered trivially by simply returning $T_\emptyset = \{()\}$, since this is always a model, but this does not help us answer Q. Our third problem is:

Question 3. Design a worst-case optimal algorithm for disjunctive datalog rules under degree constraints.

Answer 3. Details are presented in Sections 3 and 4. We summarize the ideas here. We present an algorithm called PANDA (**P**roof-**A**ssisted e**N**tropic **D**egree-**A**ware), which computes a model of a disjunctive datalog rule P within the runtime predicted by the bound (8). PANDA is derived using a novel principle that we introduced in [3]. First, one has to provide "evidence", called *proof sequence*, that the polymatroid bound is correct. Second, each step in the sequence is interpreted as a relational operator (one of: join, horizontal partition, union), leading to a model of P.

The polymatroid bound (8) is difficult to handle. While the feasible region $\Gamma_n \cap \mathsf{HDC}$ is polyhedral, the objective of (6) is non-linear. We start by proving in Lemma 3.2 that it is equivalent to a linear program: there exist constants $\lambda_B \geq 0$, for $B \in \mathcal{B}$ for which:

$$\max_{h \in \Gamma_n \cap \mathsf{HDC}} \min_{B \in \mathcal{B}} h(B) = \max_{h \in \Gamma_n \cap \mathsf{HDC}} \sum_{B \in \mathcal{B}} \lambda_B h(B) \tag{9}$$

The RHS of (9) is simpler to deal with. To prove an upper bound for it, one has to prove an inequality of the following form, which we call a *Shannon-flow inequality*:

$$\sum_{B \in \mathcal{B}} \lambda_B \cdot h(B) \leq \sum_{(X,Y,N_{Y|X}) \in \mathsf{DC}} \delta_{Y|X} \cdot h(Y|X) \tag{10}$$

This is a (vast) generalization of Shearer's lemma [18]. The inequality (10) implies $\log|P| \leq \sum \delta_{Y|X} \log N_{Y|X}$, and we show in the paper how to choose the coefficients $\delta_{Y|X}$ such that the last expression is precisely (8). Thus, the first task is to prove (i.e. provide evidence for) the inequality (10).

A key technical result in the paper is Theorem 3.8 which, stated informally, says that Eq.(10) can be proved using a sequence of rules of one of the following four types, where $X \subseteq Y$:

$$\begin{aligned}
&\textit{Submodularity} && h(Y|X) \to h(Y \cup Z|X \cup Z)\\
&\textit{Monotonicity} && h(Y) \to h(X),\\
&\textit{Composition} && h(X) + h(Y|X) \to h(Y),\\
&\textit{Decomposition} && h(Y) \to h(X) + h(Y|X).
\end{aligned}$$

To explain the theorem, assume for the sake of discussion that all coefficients in (10) are natural numbers. Then both sides can be seen as bags of terms, and the theorem says that there exists a sequence of rewritings that transform the bag on the RHS to the bag on the LHS. Obviously, if such a sequence exists, then Eq.(10) holds, because each rewriting replaces a term (or sum of two terms) with a smaller or equal term (or sum of two terms). The converse statement is non-obvious. For example in our prior work [3] we found that, without the decomposition rule, the remaining three rules are *not* complete: there exists a Shannon-flow inequality without a proof sequence.

Finally, PANDA consists of interpreting each step in the proof sequence as a relational operation, leading to:

Theorem 1.5. *PANDA computes a model of a disjunctive datalog rule P under degree constraints DC in time*

$$\tilde{O}\left(N + \text{poly}(\log N) \cdot \prod_{(X,Y,N_{Y|X}) \in \mathsf{DC}} N_{Y|X}^{\delta_{Y|X}}\right),$$

where $\sum_{(X,Y,N_{Y|X}) \in \mathsf{DC}} \delta_{Y|X} \log N_{Y|X} = \mathsf{LogSizeBound}_{\Gamma_n \cap \mathsf{HDC}}(P)$.

1.4 Towards Optimal Algorithms

What is an optimal runtime to compute a given conjunctive query? A common belief is that its cost is of the form $\tilde{O}(N^d + |\mathsf{output}|)$, where N is the size of the input database, N^d represents the "intrinsic" cost of the query, and $|\mathsf{output}|$ is the unavoidable cost of reporting the output.[2] Worst-case optimal algorithms are not optimal in this sense. They are only good for inputs whose intrinsic cost is about the same

[2]In this paper, the big-O notation is in data-complexity, hiding a factor that is query-dependent and data-independent. The big-$\tilde{O}$ hides a single log-factor in data-complexity.

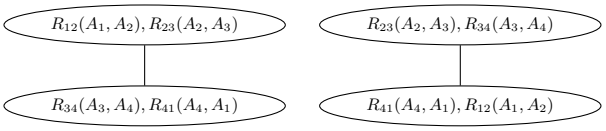

Figure 1: Two tree decompositions for the query in Example 1.1. Normally, each tree node is labeled with a set of variables, e.g. $\chi(t) = \{A_1, A_2, A_3\}$; for convenience we also show the atoms contained in those variables, i.e. $R_{12}(A_1, A_2), R_{23}(A_2, A_3)$.

as the worst-case output size. As described below, there are algorithms whose runtimes are more output-sensitive. If the query is Boolean, then the output size is always 1, and the cost is totally dominated by the intrinsic cost of the query; for simplicity we discuss here only Boolean queries, but our discussion extends to other conjunctive and aggregate queries [2]. Thus, an optimal algorithm should compute a Boolean query in time $\tilde{O}(N^d)$, with an exponent d as small as possible. Generalizing to degree constraints, it should compute the query in time $\tilde{O}(\prod N_{Y|X}^{\delta_{Y|X}})$, where $N_{Y|X}$ are the degree bounds, and the product is minimized.

In search of a yardstick for optimality, we borrow from the long history of research on fixed-parameter tractability. A class $\mathcal{C}$ of Boolean conjunctive queries (equivalently, CSP instances) is *fixed-parameter tractable* (FPT, with parameter $\mathcal{H}$, the query's hypergraph) if there is an algorithm solving every $\mathcal{C}$-instance in time $f(|\mathcal{H}|) \cdot N^d$ for some fixed constant d, where f is any computable function, and N is the data size. In a beautiful paper, Marx [32] showed that $\mathcal{C}$ is FPT iff it has a bounded submodular width. His results suggest to us using the submodular width, $\mathsf{subw}(Q)$, as a yardstick for optimality. In order to prove that bounded submodular width implies FPT, Marx described a query evaluation algorithm that runs in time $O(\text{poly}(N^{\mathsf{subw}(Q)}))$.[3] We define an algorithm to be *optimal* if its runtime is $\tilde{O}(N^{\mathsf{subw}(Q)})$. While no lower bounds are known to date to rule out faster algorithms for a specific query, Marx's dichotomy theorem ruled out faster algorithms for any recursively enumerable class of queries (see [4, 32]). Our fourth problem is:

Question 4. Design an algorithm evaluating a Boolean conjunctive query Q in $\tilde{O}(N^{\mathsf{subw}(Q)})$-time. Extend the notion of submodular width, and the algorithm, to handle arbitrary degree constraints, to full conjunctive and aggregate queries.

We briefly review the notion of submodular width, and its relationship to other width notions. Note that all known width parameters considered only cardinality constraints. A polymatroid h is *edge-dominated* if $h(F) \le 1, \forall F \in \mathcal{E}$. Edge domination is a normalized version of cardinality constraints. The submodular width is defined to be $\mathsf{subw}(Q) \stackrel{\text{def}}{=} \max_h \min_{(T,\chi)} \max_{t \in V(T)} h(\chi(t))$, where h ranges over edge-dominated polymatroids, and (T, χ) over tree decompositions of Q (see Defn. 2.2). Prior notions of width such as tree-width [23], generalized- [28] and fractional- hypertree width [30] (see [26] for a nice survey) are defined by first defining the width of a tree decomposition, then choosing the decomposition that minimizes this width. Thus, there is always a *best* tree decomposition (T, χ), and a query evaluation algorithm runs on that (T, χ); e.g., the fractional hypertree width is $\mathsf{fhtw}(Q) \stackrel{\text{def}}{=} \min_{(T,\chi)} \max_{t \in V(T)} \rho^*(\chi(t))$, where ρ^* is the fractional edge cover number of the set $\chi(t)$. In the submodular width, we are allowed to choose the tree T *after* we see the polymatroid h. Marx showed that $\mathsf{subw}(Q) \le \mathsf{fhtw}(Q)$, for all Q, and there are classes of queries for which the gap is unbounded. (See also a simple example in [4].)

Example 1.6. Fig. 1 shows the only non-trivial tree decompositions of the query Q in Example 1.1. Each tree has a fhtw of 2, hence $\mathsf{fhtw}(Q) = 2$. In contrast, we will show later that $\mathsf{subw}(Q) = 3/2$. For the Boolean version of the query, an algorithm with runtime $O(N^{\mathsf{subw}(Q)})$ will match the best known $O(N^{3/2})$-time algorithm for detecting a 4-cycle in a graph with N edges described by Alon et al. [8].

Answer 4. Our answer is presented in Section 5. Briefly, we generalize the notion of submodular width $\mathsf{subw}(Q)$ to account for arbitrary degree constraints, and call it *degree-aware submodular width*, $\mathsf{da\text{-}subw}(Q)$. In fact, we describe a very general framework that captures virtually all previously defined width-parameters, *and* extends them to degree constraints. We then show how to use PANDA to compute a query in time whose exponent is $\mathsf{da\text{-}subw}(Q)$, using the same earlier principle: from a proof of the bound of $\mathsf{da\text{-}subw}(Q)$, we derive an algorithm that computes Q in that bound:

Theorem 1.7. *PANDA computes any full, Boolean conjunctive, or aggregate query Q in time $\tilde{O}(N + \text{poly}(\log N) \cdot 2^{\mathsf{da\text{-}subw}(Q)} + |\mathsf{output}|)$.*

2. BACKGROUND

Throughout the paper, we use the following convention. The non-negative reals, rationals, and integers are denoted by $\mathbb{R}_+, \mathbb{Q}_+$, and $\mathbb{N}$ respectively. Uppercase A_i denotes a variable/attribute, and lowercase a_i denotes a value in the discrete domain $\mathsf{Dom}(A_i)$ of the variable. For any subset $S \subseteq [n]$, define $\mathbf{A}_S = (A_i)_{i \in S}$, $\mathbf{a}_S = (a_i)_{i \in S} \in \prod_{i \in S} \mathsf{Dom}(A_i)$. In particular, $\mathbf{A}_S$ is a tuple of variables and $\mathbf{a}_S$ is a tuple of specific values with support S. Occasionally we use $\mathbf{t}_S$ to denote a tuple with support S.

We will work on multi-hypergraphs $\mathcal{H} = ([n], \mathcal{E})$ (i.e. a hyperedge may occur multiple times in $\mathcal{E}$). A function $f : 2^{[n]} \to \mathbb{R}_+$ is called a (non-negative) *set function* on $\mathcal{H}$. A set function f on $\mathcal{H}$ is *modular* if $f(S) = \sum_{v \in S} f(\{v\})$ for all $S \subseteq [n]$, is *monotone* if $f(X) \le f(Y)$ whenever $X \subseteq Y$, is *subadditive* if $f(X \cup Y) \le f(X) + f(Y)$ for all $X, Y \subseteq [n]$, and is *submodular* if $f(X \cup Y) + f(X \cap Y) \le f(X) + f(Y)$ for all $X, Y \subseteq [n]$. Unless specified otherwise, we will only consider *non-negative* and *monotone* set functions f for which $f(\emptyset) = 0$; this assumption will be implicit in the entire paper. Let M_n, SA_n, and Γ_n denote the set of all (non-negative and monotone) modular, subadditive, and submodular set functions on $[n]$, respectively.

Definition 2.1. Given a hypergraph $\mathcal{H} = ([n], \mathcal{E})$, define the following two set functions:

$$\mathsf{ED} \stackrel{\text{def}}{=} \{h \mid h : 2^{[n]} \to \mathbb{R}_+, h(F) \le 1, \forall F \in \mathcal{E}\} \quad (11)$$

[3] It is not clear what the exact runtime of Marx's algorithm is. His theorem states that it is $O(\text{poly}(N^{\mathsf{subw}(Q)}))$. Our best interpretation of Lemma 4.3 and Lemma 4.5 of [32] is that Marx's algorithm runs in time at least $O(N^{2 \cdot \mathsf{subw}(Q)})$.

$$\mathsf{VD} \stackrel{\text{def}}{=} \{h \mid h : 2^{[n]} \to \mathbb{R}_+, h(\{v\}) \le 1, \forall v \in [n]\} \quad (12)$$

ED stands for *edge-dominated* and VD stands for *vertex-dominated*.

Given a set function h and a scalar s, we will use $s \cdot h$ to denote h scaled by s. (Specifically, we will be interested in $\log N \cdot \mathsf{ED}$ and $\log N \cdot \mathsf{VD}$.)

For any two finite sets S and T, let S^T denote the collection of all maps $f : T \to S$. Such a map f is also viewed as a vector whose coordinates are indexed by members of T and whose coordinate values are members of S.

For the sake of clarity, this section focuses on a full conjunctive query $Q(\mathbf{A}_{[n]}) = \bigwedge_{F\in\mathcal{E}} R_F(A_F)$, whose hypergraph is $\mathcal{H} = ([n], \mathcal{E})$, as discussed in Section 1. For each $F \in \mathcal{E}$, let $N_F \stackrel{\text{def}}{=} |R_F|$, where $N_F \in \mathbb{N} \cup \{\infty\}$. We set $N_F = \infty$ if R_F is not a materialized predicate, a negation of a predicate, or if its size is not known. Throughout this paper, let

$$N = \max_{F\in\mathcal{E}, |R_F|<\infty} N_F. \quad (13)$$

(The full version [4] explains how we deal with Boolean, count, or aggregate queries.)

2.1 Queries without FDs nor degree-bounds

Bounds on the worst-case output size. Bounding the worst-case output size $|Q|$ of a natural join query Q is a well-studied problem. There is a hierarchy of such bounds.

A trivial bound is the *vertex bound* $|Q| \le \mathsf{VB}(Q) \stackrel{\text{def}}{=} N^n$. A slightly less trivial bound is the *integral edge cover* bound, described as follows. Define the edge cover polytope

$$\mathsf{EC} \stackrel{\text{def}}{=} \left\{ \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \in \mathbb{R}_+^{\mathcal{E}}, \sum_{\substack{F\in\mathcal{E}\\ v\in F}} \lambda_F \ge 1, \forall v \in [n] \right\}.$$

The integral edge cover bound is

$$\log|Q| \le \rho(Q, (N_F)_{F\in\mathcal{E}}) \stackrel{\text{def}}{=} \log\left(\prod_{F\in\mathcal{E}} N_F^{\lambda_F}\right) \quad (14)$$

where $\boldsymbol{\lambda} = \operatorname{argmin}\left\{\sum_{F\in\mathcal{E}} \lambda_F \log N_F \mid \boldsymbol{\lambda} \in \mathsf{EC} \cap \{0,1\}^{\mathcal{E}}\right\}$. This bound is dependent on the input relations' sizes. Often, to state a bound that is independent of the input size, researchers use a cruder approximation of the bound: $\rho(Q) = \rho(Q, (N_F = 1)_{F\in\mathcal{E}})$, which is called the *integral edge cover number* of $\mathcal{H}$.

Building on earlier works [7, 18, 24, 29], Atserias, Grohe, and Marx [13] observed that the rational relaxation of (14) still holds, leading to the *AGM-bound*:

$$\log|Q| \le \log \mathsf{AGM}(Q, (N_F)_{F\in\mathcal{E}}) \stackrel{\text{def}}{=} \log\left(\prod_{F\in\mathcal{E}} N_F^{\lambda_F}\right) \quad (15)$$

where $\boldsymbol{\lambda} = \operatorname{argmin}\left\{\sum_{F\in\mathcal{E}} \lambda_F \log N_F \mid \boldsymbol{\lambda} \in \mathsf{EC}\right\}$. The relaxation $\rho^*(Q) \stackrel{\text{def}}{=} \log \mathsf{AGM}(Q, (N_F = 1)_{F\in\mathcal{E}})$ is called the *fractional edge cover number*.

One remarkable property of the AGM-bound is that it is asymptotically tight. There are known algorithms [1, 34, 35, 37] with runtime $\tilde{O}(\mathsf{AGM}(Q))$: they are *worst-case optimal*.

Tree decompositions and their widths. Tree decompositions capture conditional independence among variables in a query, facilitating dynamic-programming. We refer the reader to the recent survey by Gottlob et al. [26] for more details on historical contexts, technical descriptions, and open problems thereof. We are necessarily brief in this section.

Definition 2.2. A *tree decomposition* of a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ is a pair (T, χ), where T is a tree and $\chi : V(T) \to 2^{[n]}$ maps each node t of the tree to a subset $\chi(t)$ of vertices such that (1) Every hyperedge $F \in \mathcal{E}$ is a subset of some $\chi(t)$, $t \in V(T)$, (2) For every vertex $v \in [n]$, the set $\{t \mid v \in \chi(t)\}$ is a non-empty (connected) sub-tree of T. Somewhat confusingly, the sets $\chi(t)$ are often called the *bags* of the tree decomposition.

The common method of defining width parameters is the framework introduced by Adler [5]. Let $\mathcal{H} = ([n], \mathcal{E})$ be a hypergraph. Let $g : 2^{[n]} \to \mathbb{R}_+$ be a function that assigns a non-negative real number to each subset of $[n]$. Then, the *g-width* of a tree decomposition (T, χ) is $\max_{t\in V(T)} g(\chi(t))$. The *$g$-width of $\mathcal{H}$* is the *minimum* g-width over all tree decompositions of $\mathcal{H}$. Note that the g-width of a hypergraph is a *minimax* function.

For any subset of vertices $B \subseteq [n]$, define $s(B) = |B| - 1$, $\rho(B)$ the integral edge cover number of the set B using edges in $\mathcal{H}$, and $\rho^*(B)$ the fractional edge cover number. Then, the *treewidth* of $\mathcal{H}$, denoted by $\mathsf{tw}(\mathcal{H})$, is the s-width of $\mathcal{H}$. The *generalized hypertree width* of $\mathcal{H}$, denoted by $\mathsf{ghtw}(\mathcal{H})$ is the ρ-width of $\mathcal{H}$. And, the *fractional hypertree width* of $\mathcal{H}$, denoted by $\mathsf{fhtw}(\mathcal{H})$, is the ρ^*-width of $\mathcal{H}$. Very recently, Fischl et al. [22] showed that, checking whether a given hypergraph has a fractional hypertree width or a generalized hypertree width at most 2 is NP-hard, settling two important open questions.

The above three width parameters are based on the same idea: we decompose the query into sub-queries in a dynamic-programming algorithm, and the runtime is dominated by the worst bag size bound of the tree decomposition. It is known [2, 6, 14, 19] that a vast number of problems in graphical model inference, database query computation, constraint satisfaction, and logic can be solved using this strategy. However, this tree-decomposition-first strategy has a drawback that once we stick with a tree decomposition we are forced to suffer the worst-case instance for that tree decomposition. Marx [31, 32] had a wonderful observation: we can partition the data first, and then use a different tree decomposition for each part of the data, then we can in some cases significantly improve the runtime. This idea leads to the notions of *adaptive width* and *submodular width* of a query, where in essence data partitioning and query decomposition are used in an interleaving way.

Definition 2.3. Recall the notion of edge domination from Defn. 11. Then, the *adaptive width* $\mathsf{adw}(\mathcal{H})$ of $\mathcal{H}$ and the *submodular width* $\mathsf{subw}(\mathcal{H})$ of $\mathcal{H}$ are defined by

$$\mathsf{adw}(\mathcal{H}) \stackrel{\text{def}}{=} \max_{f\in\mathsf{ED}\cap\mathsf{M}_n} \min_{(T,\chi)} \max_{t\in V(T)} f(\chi(t)), \quad (16)$$

$$\mathsf{subw}(\mathcal{H}) \stackrel{\text{def}}{=} \max_{f\in\mathsf{ED}\cap\Gamma_n} \min_{(T,\chi)} \max_{t\in V(T)} f(\chi(t)). \quad (17)$$

2.2 FD and degree constraints

The above series of bounds and width parameters were based on a single class of statistics on the input relations: their sizes. In practice we very often encounter queries with functional dependency (FD) and degree bound information. The FDs come from two main sources: primary keys and

builtin predicates (such as $A_1 + A_2 = A_3$). The degree constraints come from more refined statistics of the input (materialized) predicates.

Definition 2.4. A *degree constraint* is a triple $(X, Y, N_{Y|X})$ where $X \subset Y \subseteq [n]$ and $N_{Y|X} \in \mathbb{N} \cup \{\infty\}$. A relation R_F is said to *guard* the degree constraint $(X, Y, N_{Y|X})$ if $X \subset Y \subseteq F$ and for every tuple $\mathbf{t}_X$ we have

$$\deg_{R_F}(Y|\mathbf{t}_X) \stackrel{\text{def}}{=} |\Pi_Y(\sigma_{\mathbf{A}_X=\mathbf{t}_X}(R_F)| \le N_{Y|X}. \tag{18}$$

The quantity on the left hand side is called the *degree of* $\mathbf{t}_X$ with respect to Y in relation R_F. Note that a relation may guard multiple degree constraints.

To avoid writing $\log_2$ in many places, define $n_{Y|X} \stackrel{\text{def}}{=} \log_2 N_{Y|X}$. We use DC to denote a set of degree constraints. A cardinality constraint is a triple $(\emptyset, F, N_F)$. We use CC to denote a set of cardinality constraints. Similar to HDC defined by (5), let HCC denote the set of functions h satisfying cardinality constraints CC.

For example, consider an input relation $R(A_1, A_2, A_3)$ satisfying the following conditions: for every value a_1 in the active domain of A_1, there are at most D different values $a_2 \in \mathsf{Dom}(A_2)$ such that $(a_1, a_2) \in \Pi_{A_1A_2}(R)$. Then, R guards the degree constraint $(\{A_1\}, \{A_1, A_2\}, D)$.

The output size of $Q(\mathbf{A}_{[n]})$ can be bounded by

$$\log_2 |Q| \le \underbrace{\max_{h \in \mathsf{HDC} \cap \bar{\Gamma}_n^*} h([n])}_{\mathsf{DAEB}(Q)} \le \underbrace{\max_{h \in \mathsf{HDC} \cap \Gamma_n} h([n])}_{\mathsf{DAPB}(Q)} \tag{19}$$

(DAEB and DAPB are "degree-aware" entropic and polymatroid bounds, respectively. Note that (19) is a special case of (7) and (8).) The CSMA algorithm from [3] can solve a join query Q with known degree constraints in time $\tilde{O}(N + \text{poly}(\log N) \cdot 2^{\mathsf{DAPB}(Q)})$.

3. SHANNON FLOW INEQUALITIES

The PANDA algorithm is built on the notion of a "proof sequence" for a class of Shannon-type inequalities called the *Shannon flow inequalities*.

Definition 3.1. Let $\mathcal{B} \subseteq 2^{[n]}$ denote a collection of subsets of $[n]$. Let $\mathcal{C} \subseteq 2^{[n]} \times 2^{[n]}$ denote a collection of pairs (X, Y) such that $\emptyset \ne X \subset Y \subseteq [n]$. Let $\boldsymbol{\lambda}_{\mathcal{B}} = (\lambda_F)_{F \in \mathcal{B}} \in \mathbb{Q}_+^{\mathcal{B}}$ and $\boldsymbol{\delta}_{\mathcal{C}} = (\delta_{Y|X})_{(X,Y) \in \mathcal{C}} \in \mathbb{Q}_+^{\mathcal{C}}$ denote two vectors of non-negative rationals. For any polymatroid h, let $h(Y|X)$ denote $h(Y) - h(X)$.[4] If the following inequality

$$\sum_{B \in \mathcal{B}} \lambda_B \cdot h(B) \le \sum_{(X,Y) \in \mathcal{C}} \delta_{Y|X} \cdot h(Y|X) \tag{20}$$

holds for all $h \in \Gamma_n$ (i.e. for all polymatroids), then it is called a *Shannon flow inequality*. The set $\mathcal{B}$ is called the set of *targets* of the flow inequality.

Section 3.1 motivates the study of these inequalities. Section 3.2 explains why they are called "flow" inequalities.

3.1 Motivations

Fix a disjunctive datalog rule P of the form (3) with degree constraints DC. Abusing notations, we write $(X, Y) \in \mathsf{DC}$ whenever $(X, Y, N_{Y|X}) \in \mathsf{DC}$. In particular the set DC can play the role of the generic set $\mathcal{C}$ in the definition of Shannon flow inequality. To explain where the Shannon flow inequalities come from, we study the (log) polymatroid bound (8) for P, which was defined by (6) with $\mathcal{F}$ chosen to be $\Gamma_n \cap \mathsf{HDC}$. Appendix C.1 contains the proof of the following reformulation: the maximin optimization problem (6) (with $\mathcal{F} = \Gamma_n \cap \mathsf{HDC}$) has precisely the same objective value as a linear program.

Lemma 3.2. *There exists a non-negative vector* $\boldsymbol{\lambda} = (\lambda_B)_{B \in \mathcal{B}}$, *with* $\|\boldsymbol{\lambda}\|_1 = 1$, *such that*

$$\mathsf{LogSizeBound}_{\Gamma_n \cap \mathsf{HDC}}(P) = \max_{h \in \Gamma_n \cap \mathsf{HDC}} \sum_{B \in \mathcal{B}} \lambda_B \cdot h(B). \tag{21}$$

This linear program along with Farkas lemma give rise to Shannon flow inequalities. We first need the dual LP of (21). Associate a dual variable $\delta_{Y|X}$ to each degree constraint, a variable $\sigma_{I,J}$ to each submodularity constraint, and a variable $\mu_{X,Y}$ to each monotonicity constraint. For any $Z \in 2^{[n]}$, define the quantity

$$\mathsf{inflow}(Z) \stackrel{\text{def}}{=} \sum_{X:(X,Z) \in \mathsf{DC}} \delta_{Z|X} - \sum_{Y:(Z,Y) \in \mathsf{DC}} \delta_{Y|Z} + \sum_{\substack{I \perp J \\ I \cap J = Z}} \sigma_{I,J}$$
$$+ \sum_{\substack{I \perp J \\ I \cup J = Z}} \sigma_{I,J} - \sum_{J: J \perp Z} \sigma_{Z,J} - \sum_{X: X \subset Z} \mu_{X,Z} + \sum_{Y: Z \subset Y} \mu_{Z,Y}.$$

Here, $I \perp J$ means $I \not\subseteq J$ and $J \not\subseteq I$. Note that the function $\mathsf{inflow} : 2^{[n]} \to \mathbb{Q}_+$ is also a function of the dual variables $(\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$. However, we do not explicitly write down this dependency to avoid heavy-loading notations. The dual of the RHS of (21) is

$$\begin{aligned} \min \quad & \textstyle\sum_{(X,Y) \in \mathsf{DC}} n_{Y|X} \cdot \delta_{Y|X} && \\ \text{s.t.} \quad & \mathsf{inflow}(B) && \ge \lambda_B, \ \forall B \in \mathcal{B} \\ & \mathsf{inflow}(Z) && \ge 0, \emptyset \ne Z \subseteq [n]. \\ & (\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu}) && \ge \mathbf{0}. \end{aligned} \tag{22}$$

(Recall that $n_{Y|X} \stackrel{\text{def}}{=} \log_2 N_{Y|X}$.) Let $\mathbf{h}^* = (h^*_Z)_{Z \subseteq [n]}$ denote an optimal solution to (21), then its objective value $h^*([n])$ is the same as the objective value of (22) due to strong duality. In particular, let $(\boldsymbol{\delta}^*, \boldsymbol{\sigma}^*, \boldsymbol{\mu}^*)$ denote an optimal solution to (22), then $\sum_{B \in \mathcal{B}} \lambda_B \cdot h^*(B) = \sum_{(X,Y) \in \mathsf{DC}} \delta^*_{Y|X} \cdot n_{Y|X}$. One way to characterize *any* dual feasible solution $(\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$, is to use Farkas' lemma [36], which in our context takes the following form (see Appendix C.1 for a proof):

Proposition 3.3. *Given* $\boldsymbol{\lambda}$ *and* $\boldsymbol{\delta}$, *the inequality*

$$\sum_{B \in \mathcal{B}} \lambda_B \cdot h(B) \le \sum_{(X,Y) \in DC} \delta_{Y|X} \cdot h(Y|X) \tag{23}$$

is a Shannon flow inequality if and only if there exist $\boldsymbol{\sigma}$ *and* $\boldsymbol{\mu}$ *such that* $(\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$ *is feasible to the dual LP* (22).

3.2 Proof sequences

A key observation from Abo Khamis et al. [3] was that we can turn a proof of a special case of inequality (23) into an algorithm. The proof has to be performed in a sequential manner; and this brings us to the concept of a proof sequence. In this paper, we refine the proof sequence notion from [3] in four significant ways. First, the definition

[4] The quantity $h(Y|X)$ is the polymatroid-analog of the conditional entropy $H[Y|X] = H[Y] - H[X]$.

of the proof sequence is different, allowing for a simpler algorithm (PANDA) than CSMA in [3]. Second, in [3] we left open whether proof sequences are a complete proof system, even for special Shannon-flow inequalities; the CSMA algorithm used a specific workaround to achieve optimality even without proving completeness of proof sequences. Our most important contribution here is to prove completeness of our new proof sequence. Third, we are able to bound the length of the proof sequence to be polynomial in the size of the linear program (22), as opposed to the doubly exponential length in [3]. Fourth, new technical ideas are introduced so that we can construct proof sequences for the much more general Shannon flow inequality (23) (as opposed to the special case of "output inequality" in [3]).

Definition 3.4. Let $\mathcal{P} \subseteq 2^{[n]} \times 2^{[n]}$ denote the set of all pairs (X, Y) such that $\emptyset \subseteq X \subset Y \subseteq [n]$. A vector $\mathbf{f} \in \mathbb{R}_+^{\mathcal{P}}$ has coordinates indexed by pairs $(X, Y) \in \mathcal{P}$. We denote the corresponding coordinate value of $\mathbf{f}$ by $f(Y|X)$. The vector $\mathbf{f}$ is called a *conditional polymatroid* iff there exists a polymatroid h such that $f(Y|X) = h(Y) - h(X)$; and, we say h defines the conditional polymatroid $\mathbf{f}$. Abusing notation somewhat, the conditional polymatroid defined by the polymatroid h is denoted by $\mathbf{h}$. In particular, $\mathbf{h} = (h(Y|X))_{(X,Y)\in\mathcal{P}}$. If h is a polymatroid then $h(\emptyset) = 0$, in which case we write $h(Y)$ instead of $h(Y|\emptyset)$.

To formally define the notion of a proof sequence, we rewrite the Shannon flow inequality (20) as an inequality on conditional polymatroids in the $\mathbb{Q}_+^{\mathcal{P}}$ space. We extend the vectors $\boldsymbol{\lambda}_{\mathcal{B}} \in \mathbb{Q}_+^{\mathcal{B}}$ and $\boldsymbol{\delta}_{\mathcal{C}} \in \mathbb{Q}_+^{\mathcal{C}}$ to become vectors $\boldsymbol{\lambda}, \boldsymbol{\delta}$ in the $\mathbb{Q}_+^{\mathcal{P}}$ space in the obvious way:

$$\lambda(Y|X) \stackrel{\text{def}}{=} \begin{cases} \lambda_{\mathcal{B}}(B) & \text{when } Y = B, X = \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

$$\delta(Y|X) \stackrel{\text{def}}{=} \begin{cases} \delta_{\mathcal{C}}(Y|X) & \text{when } (X, Y) \in \mathcal{C} \\ 0 & \text{otherwise.} \end{cases}$$

Then, inequality (20) can be written simply as $\langle \boldsymbol{\lambda}, \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}, \mathbf{h} \rangle$. Note the crucial fact that, even though $\boldsymbol{\lambda} \in \mathbb{Q}_+^{\mathcal{P}}$, for it to be part of a Shannon flow inequality only the entries $\lambda_{B|\emptyset}$ can be positive. We will often write λ_B instead of $\lambda_{B|\emptyset}$. These assumptions are implicit henceforth. Prop. 3.3 can now be written simply as:

Proposition 3.5. *Given any $\boldsymbol{\lambda}, \boldsymbol{\delta} \in \mathbb{Q}_+^{\mathcal{P}}$, where $\lambda_{Y|X} > 0$ implies $X = \emptyset$, the inequality $\langle \boldsymbol{\lambda}, \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}, \mathbf{h} \rangle$ is a Shannon flow inequality if and only if there exist $\boldsymbol{\sigma}$ and $\boldsymbol{\mu}$ such that $(\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$ satisfy the constraints*

$$\mathsf{inflow}(Z) \geq \lambda_Z, \emptyset \neq Z \subseteq [n] \text{ and } (\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu}) \geq \mathbf{0} \qquad (24)$$

The conditional polymatroids satisfy four basic laws:

$$\begin{aligned} h(I \cup J|J) - h(I|I \cap J) &\leq 0, \quad I \perp J \text{ (submodularity)} \\ -h(Y|\emptyset) + h(X|\emptyset) &\leq 0, \quad X \subset Y \text{ (monotonicity)} \\ h(Y|\emptyset) - h(Y|X) - h(X|\emptyset) &\leq 0, \quad X \subset Y \text{ (composition)} \\ -h(Y|\emptyset) + h(Y|X) + h(X|\emptyset) &\leq 0, \quad X \subset Y \text{ (decomposition)} \end{aligned}$$

For every $I \perp J$, define a vector $\mathbf{s}_{I,J} \in \mathbb{Q}_+^{\mathcal{P}}$, and for every $X \subset Y$, define three vectors $\mathbf{m}_{X,Y}, \mathbf{c}_{X,Y}, \mathbf{d}_{Y,X} \in \mathbb{Q}_+^{\mathcal{P}}$ such that the laws above can be written correspondingly in dot-product form:

$$\langle \mathbf{s}_{I,J}, \mathbf{h} \rangle \leq 0, \quad I \perp J \text{ (submodularity)} \qquad (25)$$

$$\langle \mathbf{m}_{X,Y}, \mathbf{h} \rangle \leq 0, \quad X \subset Y \text{ (monotonicity)} \qquad (26)$$

$$\langle \mathbf{c}_{X,Y}, \mathbf{h} \rangle \leq 0, \quad X \subset Y \text{ (composition)} \qquad (27)$$

$$\langle \mathbf{d}_{Y,X}, \mathbf{h} \rangle \leq 0, \quad X \subset Y \text{ (decomposition)} \qquad (28)$$

Definition 3.6. A *proof sequence* of a Shannon flow inequality $\langle \boldsymbol{\lambda}, \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}, \mathbf{h} \rangle$, is a sequence $(w_1 \mathbf{f}_1, \ldots, w_\ell \mathbf{f}_\ell)$ satisfying the following: (1) $\mathbf{f}_i \in \{ \mathbf{s}_{I,J}, \mathbf{m}_{X,Y}, \mathbf{c}_{X,Y}, \mathbf{d}_{Y,X} \}$ for all $i \in [\ell]$. The $\mathbf{f}_i$ are called *proof steps*. (2) $w_i \in \mathbb{R}_+$, $i \in [\ell]$ are the corresponding *weights* of the proof steps. (3) All the vectors $\boldsymbol{\delta}_0 \stackrel{\text{def}}{=} \boldsymbol{\delta}, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_\ell$ defined by $\boldsymbol{\delta}_i = \boldsymbol{\delta}_{i-1} + w_i \cdot \mathbf{f}_i$, $i \in [\ell]$ are non-negative. (4) Furthermore, $\boldsymbol{\delta}_\ell \geq \boldsymbol{\lambda}$.

Note that $\langle \boldsymbol{\delta}_{i-1}, \mathbf{h} \rangle \geq \langle \boldsymbol{\delta}_i, \mathbf{h} \rangle$ for every conditional polymatroid $\mathbf{h}$, and every $i \in [\ell]$. The proof step $\mathbf{s}_{I,J}$ is called a *submodularity step*, $\mathbf{m}_{X,Y}$ a *monotonicity step*, $\mathbf{d}_{Y,X}$ a *decomposition step*, and $\mathbf{c}_{X,Y}$ a *composition step*.

Definition 3.7. Let $\langle \boldsymbol{\lambda}, \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}, \mathbf{h} \rangle$ be a Shannon flow inequality. From Prop. 3.3 there exists $(\boldsymbol{\sigma}, \boldsymbol{\mu})$ such that $(\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$ belongs to the polyhedron (24). We call $(\boldsymbol{\sigma}, \boldsymbol{\mu})$ a *witness* for the Shannon flow inequality.

We present here one construction of a proof sequence for a Shannon flow inequality. See [4] for more advanced constructions of shorter sequences.

Theorem 3.8. *Let $\langle \boldsymbol{\lambda}, \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}, \mathbf{h} \rangle$ be a Shannon flow inequality with witness $(\boldsymbol{\sigma}, \boldsymbol{\mu})$. There exists a proof sequence for the inequality $\langle \boldsymbol{\lambda}, \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}, \mathbf{h} \rangle$ with length at most $D(3\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\mu}\|_1)$, where D is the minimum common denominator of all entries in $(\boldsymbol{\lambda}, \boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$.*

Proof. We induct on the quantity

$$\ell(\boldsymbol{\lambda}, \boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu}) \stackrel{\text{def}}{=} D\left(\|\boldsymbol{\lambda}\|_1 + 2\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\mu}\|_1\right),$$

which is an integer. The base case is when $\|\boldsymbol{\lambda}\|_1 = 0$, which is trivial because the inequality has a proof sequence of length 0. In the inductive step, assume $\|\boldsymbol{\lambda}\|_1 > 0$, meaning there must be some $B \subseteq [n]$ for which $\lambda_B > 0$. We will produce a Shannon flow inequality $\langle \boldsymbol{\lambda}', \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}', \mathbf{h} \rangle$ witnessed by $(\boldsymbol{\sigma}', \boldsymbol{\mu}')$ such that $\ell(\boldsymbol{\lambda}', \boldsymbol{\delta}', \boldsymbol{\sigma}', \boldsymbol{\mu}') < \ell(\boldsymbol{\lambda}, \boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$. From the induction hypothesis we obtain a proof sequence $\mathsf{ProofSeq}'$ for $\langle \boldsymbol{\lambda}', \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}', \mathbf{h} \rangle$. Finally the proof sequence $\mathsf{ProofSeq}$ for $\langle \boldsymbol{\lambda}, \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}, \mathbf{h} \rangle$ is constructed from $\mathsf{ProofSeq}'$ by appending to the beginning one or two proof steps.

From Prop. 3.5, we know $\sum_{\emptyset \neq W \subseteq [n]} \mathsf{inflow}(W) \geq \lambda_B > 0$. Consequently, there must exist $Z \neq \emptyset$ for which $\delta_{Z|\emptyset} > 0$; otherwise, all variables $\delta_{Y|X}, \sigma_{I,J}, \mu_{X,Y}$ contribute a non-positive amount to the sum $\sum_{\emptyset \neq W \subseteq [n]} \mathsf{inflow}(W)$. Let $w \stackrel{\text{def}}{=} 1/D$, and fix an arbitrary $Z \neq \emptyset$ where $\delta_{Z|\emptyset} > 0$. We initially set $(\boldsymbol{\lambda}', \boldsymbol{\delta}', \boldsymbol{\sigma}', \boldsymbol{\mu}') = (\boldsymbol{\lambda}, \boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$; then we modify $(\boldsymbol{\lambda}', \boldsymbol{\delta}', \boldsymbol{\sigma}', \boldsymbol{\mu}')$ slightly depending on the cases below.

Case 1: $\lambda_Z > 0$. Reduce both λ'_Z and $\delta'_{Z|\emptyset}$ by w. From Prop. 3.5, we can verify that $\langle \boldsymbol{\lambda}', \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}', \mathbf{h} \rangle$ is a Shannon flow inequality witnessed by $(\boldsymbol{\sigma}', \boldsymbol{\mu}') = (\boldsymbol{\sigma}, \boldsymbol{\mu})$. By induction hypothesis, $\langle \boldsymbol{\lambda}', \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}', \mathbf{h} \rangle$ has a proof sequence $\mathsf{ProofSeq}'$ of length at most $D(3\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\delta}'\|_1 + \|\boldsymbol{\mu}\|_1)$. Furthermore, the $\mathsf{ProofSeq}'$ *is* also a proof sequence for $\langle \boldsymbol{\lambda}, \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}, \mathbf{h} \rangle$.

Case 2: $\lambda_Z = 0$ and $\mathsf{inflow}(Z) > 0$. Reduce $\delta'_{Z|\emptyset}$ by w. Then, from Prop. 3.5, we can verify that $\langle \boldsymbol{\lambda}, \mathbf{h} \rangle \leq \langle \boldsymbol{\delta}', \mathbf{h} \rangle$ is a Shannon flow inequality witnessed by $(\boldsymbol{\sigma}, \boldsymbol{\mu})$. The inductive step is now identical to that of Case 1.

Case 3: $\mathsf{inflow}(Z) = 0$. Since $\delta_{Z|\emptyset} > 0$, there must be some dual variable that is contributing a negative amount

to $\mathsf{inflow}(Z)$. In particular, one of the following three cases must hold:

(1) There is some $X \subset Z$ such that $\mu_{X,Z} \geq w$. Define $\boldsymbol{\delta}' = \boldsymbol{\delta} + w \cdot \mathbf{m}_{X,Z}$ and reduce $\mu'_{X,Z}$ by w. Note that $\|\boldsymbol{\delta}'\|_1 = \|\boldsymbol{\delta}\|_1$, $\|\boldsymbol{\mu}'\|_1 = \|\boldsymbol{\mu}\|_1 - w$, and $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ is a Shannon flow inequality. witnessed by $(\boldsymbol{\sigma}, \boldsymbol{\mu}')$. By induction hypothesis, $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ has a proof sequence $\mathsf{ProofSeq}'$ of length at most $D(3\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\delta}'\|_1 + \|\boldsymbol{\mu}'\|_1)$. It follows that $\mathsf{ProofSeq} = (w \cdot \mathbf{m}_{X,Z}, \mathsf{ProofSeq}')$ is a proof sequence for $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}, \mathbf{h}\rangle$ of length at most $D(3\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\mu}\|_1)$.

(2) There is some $Y \supset Z$ such that $\delta_{Y|Z} \geq w$. Define $\boldsymbol{\delta}' = \boldsymbol{\delta} + w \cdot \mathbf{c}_{Z,Y}$. Note that $\|\boldsymbol{\delta}'\|_1 = \|\boldsymbol{\delta}\|_1 - w$ and $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ is a Shannon flow inequality witnessed by $(\boldsymbol{\sigma}, \boldsymbol{\mu})$. From the proof sequence $\mathsf{ProofSeq}'$ for $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ we obtain the proof sequence $\mathsf{ProofSeq} = (w \cdot \mathbf{c}_{Z,Y}, \mathsf{ProofSeq}')$ for $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}, \mathbf{h}\rangle$ of the desired length.

(3) There is some $J \perp Z$ such that $\sigma_{Z,J} \geq w$. Define $\boldsymbol{\delta}' = \boldsymbol{\delta} + w \cdot \mathbf{d}_{Z,Z\cap J} + w \cdot \mathbf{s}_{Z,J}$, and reduce $\sigma'_{Z,J}$ by w. In this case, $\|\boldsymbol{\delta}'\|_1 = \|\boldsymbol{\delta}\|_1 + w$, $\|\boldsymbol{\sigma}'\|_1 = \|\boldsymbol{\sigma}\|_1 - w$, and $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ is a Shannon flow inequality witnessed by $(\boldsymbol{\sigma}', \boldsymbol{\mu})$. By induction hypothesis, $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ has a proof sequence $\mathsf{ProofSeq}'$ of length at most $D(3\|\boldsymbol{\sigma}'\|_1 + \|\boldsymbol{\delta}'\|_1 + \|\boldsymbol{\mu}\|_1)$. It follows that $\mathsf{ProofSeq} = (w \cdot \mathbf{d}_{Z,Z\cap J}, w \cdot \mathbf{s}_{Z,J}, \mathsf{ProofSeq}')$ is a proof sequence for $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}, \mathbf{h}\rangle$ of length at most $D(3\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\mu}\|_1)$. □

The PANDA algorithm needs another technical lemma. (See Appendix C.2 for its proof.)

Lemma 3.9. *Let $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}, \mathbf{h}\rangle$ be a Shannon flow inequality with witness $(\boldsymbol{\sigma}, \boldsymbol{\mu})$. Let D be a common denominator of all entries in $(\boldsymbol{\lambda}, \boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$, and $w = 1/D$. Suppose $\|\boldsymbol{\lambda}\|_1 > 0$ and $\delta_{Y|\emptyset} > 0$. Then, there are two vectors $\boldsymbol{\lambda}', \boldsymbol{\delta}'$ satisfying the following conditions:*

(a) $\langle \boldsymbol{\lambda}', \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ is a Shannon flow inequality (with witness $(\boldsymbol{\sigma}', \boldsymbol{\mu}')$).

(b) $\boldsymbol{\lambda}' \leq \boldsymbol{\lambda}$ and $\boldsymbol{\delta}' \leq \boldsymbol{\delta}$ (component-wise comparisons).

(c) $\|\boldsymbol{\lambda}'\|_1 \geq \|\boldsymbol{\lambda}\|_1 - w$ and $\delta'_{Y|\emptyset} \leq \delta_{Y|\emptyset} - w$.

(d) D is a common denominator of all entries in the vector $(\boldsymbol{\lambda}', \boldsymbol{\delta}', \boldsymbol{\sigma}', \boldsymbol{\mu}')$.

4. THE PANDA ALGORITHM

This section presents an algorithm called PANDA that computes a model of a disjunctive datalog rule P in time predicted by its polymatroid bound (defined by (8)):

$$\tilde{O}(N + \mathrm{poly}(\log N) \cdot 2^{\mathsf{LogSizeBound}_{\Gamma_n \cap \mathsf{HDC}}(P)}).$$

(Recall the definition of N in (13).) The main result of this section is Theorem 1.5, whose proof is in Appendix D. We start with a simple example illustrating how one might turn a proof sequence into an algorithm.

Example 4.1. We illustrate PANDA on the disjunctive rule P in Example 1.3. First we prove $|P| \leq N^{3/2}$. This follows from

$$\begin{aligned}\log|P| &\leq \min(h(A_1A_2A_3), h(A_2A_3A_4)) \\ &\leq \frac{1}{2}(h(A_1A_2A_3) + h(A_2A_3A_4)),\end{aligned}$$

and from this Shannon-flow inequality:

$$h(A_1A_2A_3) + h(A_2A_3A_4) \leq h(A_1A_2) + h(A_2A_3) + h(A_3A_4)$$

The inequality implies $\log|P| \leq \frac{3}{2}\log N$, because each of $h(A_1A_2)$, $h(A_2A_3)$, $h(A_3A_4)$ is $\leq \log N$. It remains to prove the Shannon-flow inequality above, and for that we use this proof sequence:

$$\begin{aligned}h(A_1A_2) + h(A_2A_3) + h(A_3A_4) &\overset{(1)}{\to} \\ h(A_1A_2) + h(A_2A_3) + [h(A_4|A_3) + h(A_3)] &\overset{(2)}{\to} \\ h(A_1A_2|A_3) + h(A_2A_3) + [h(A_4|A_2A_3) + h(A_3)] &= \\ \big[h(A_1A_2|A_3) + h(A_3)\big] + \big[h(A_2A_3) + h(A_4|A_2A_3)\big] &\overset{(3)}{\to} \\ h(A_1A_2A_3) + h(A_2A_3A_4)\end{aligned}$$

PANDA interprets these steps as relational operators. (1) is a decomposition step: we partition $R_{34}(A_3, A_4)$ horizontally into $R'_3(A_3)$ and $R'_{34}(A_3, A_4)$, where R'_3 contains all values a_3 that are "heavy hitters", meaning that $|\sigma_{A_3=a_3}(R_{34})| \geq \sqrt{N}$, and R'_{34} consists of all pairs (a_3, a_4) with a_3 being "light hitters". (2) are two submodularity steps: PANDA does nothing, but keeps track that the term $h(A_1A_2|A_3)$ refers to $R_{12}(A_1, A_2)$ and $h(A_4|A_2A_3)$ refers to $R'_{34}(A_3, A_4)$. (3) are two composition steps, interpreted as joins. PANDA computes the first target $T_{123}(A_1, A_2, A_3) = R_{12}(A_1, A_2) \bowtie R'_3(A_3)$, and the second target $T_{234}(A_2, A_3, A_4) = R_{23}(A_2, A_3) \bowtie R'_{34}(A_3, A_4)$. Both joins take time $\tilde{O}(N^{3/2})$, because $|R'_3| \leq \sqrt{N}$ and $\deg_{R'_{34}}(A_3A_4|A_3) \leq \sqrt{N}$.

The above example has the nice property that the two terms $h(A_4|A_3)$ and $h(A_3)$ resulting from the decomposition (1) *diverged*, i.e. were used in different targets. This allowed PANDA to place each tuple from R_{34} in either R'_3 or R'_{34}: no need to place in both, since these relations are not joined later. However, we could neither prove nor disprove the divergence property in general. Instead, PANDA conservatively places each tuple in *both* relations, yet it must ensure $|R'_3(A_3)| \cdot \deg_{R'_{34}}(A_3A_4|A_3) \leq |R_{34}|$. For that it creates $\log N$ bins, with tuples whose degree is in $[2^i, 2^{i+1})$, for $i = 0, \ldots, \lfloor \log N \rfloor$, and processes each bin separately. This needs to be repeated at each non-divergent decomposition step, hence the additional $\mathrm{poly}(\log N)$ factor in the runtime.

In general, PANDA takes as input the collection of input relations $\mathcal{R}$, the degree constraints DC, a Shannon flow inequality and its proof sequence. The Shannon flow inequality is constructed by solving the optimization problem (6) where $\mathcal{F}$ is $\Gamma_n \cap \mathsf{HDC}$. From Lemma 3.2, we can find a vector $\boldsymbol{\lambda}_{\mathcal{B}}$ with $\|\boldsymbol{\lambda}\|_1 = 1$ such that the problem has the same optimal objective value as the linear program $\max_{h \in \Gamma_n \cap \mathsf{HDC}} \langle \boldsymbol{\lambda}, \mathbf{h}\rangle$. Recall from Section 3.2 that, when we extend $\boldsymbol{\lambda}_{\mathcal{B}}$ to the (conditional polymatroid) space $\boldsymbol{\lambda} \in \mathbb{Q}_+^{\mathcal{P}}$, only the entries $\lambda_{B|\emptyset}$ for $B \in \mathcal{B}$ can be positive. Let $(\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$ denote an optimal dual solution to this LP, then by strong duality

$$\sum_{(X,Y)\in \mathsf{DC}} n_{Y|X} \cdot \delta_{Y|X} = \mathsf{LogSizeBound}_{\Gamma_n \cap \mathsf{HDC}}(P) \qquad (29)$$

Moreover, from Proposition 3.3 we know $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}, \mathbf{h}\rangle$ is a Shannon flow inequality. For inductive purposes, we will assume a slightly more general condition that the guiding Shannon flow inequality satisfies $0 < \|\boldsymbol{\lambda}\|_1 \leq 1$. From Theorem 3.8, we obtain a proof sequence for the Shannon flow

inequality. Also for inductive purposes, the following invariant is maintained throughout:

Definition 4.2 (Degree-support invariant)**.** The guiding Shannon flow inequality is promised to satisfy the following invariant: if $\delta_{Y|X} > 0$ then there exist $Z \subseteq X$, $W \subseteq Y$ such that $W - Z = Y - X$ and $(Z, W, N_{W|Z}) \in \mathsf{DC}$. (Note that, if $X = \emptyset$ then $W = Y$ and $Z = \emptyset$.) The degree constraint $(Z, W, N_{W|Z})$ is said to *support* the positive $\delta_{Y|X}$. (If there are multiple constraints $(Z, W, N_{W|Z})$ supporting $\delta_{Y|X}$, then the one with the minimum $N_{W|Z}$ is always chosen to be *the* constraint that supports $\delta_{Y|X}$, where ties are broken arbitrarily.)

Invariant 4.2 is certainly satisfied at the very beginning. A very high-level description of the algorithm is as follows. Recall from Definition 3.6 that a proof sequence $\mathsf{ProofSeq}$ is a series of proof steps, which are used by PANDA as "symbolic instructions". For each instruction, PANDA does some computation, spawns a number of subproblem(s) all of which are disjunctive datalog rules, and creates new (intermediate) relations to become input of the subproblems if necessary. The output of the ith subproblem is a set of tables $T_B^{(i)}$ for $B \in \mathcal{B}$. The overall output is the set of tables $T_B = \bigcup_i T_B^{(i)}$, $B \in \mathcal{B}$; namely for each $B \in \mathcal{B}$ we take the union of the corresponding tables from the subproblems's outputs. The number of subproblems will be shown to be polylogarithmic in the input size.

We now walk the reader step-by-step through the algorithm. Define a "budget" quantity of

$$\mathsf{OBJ} \stackrel{\text{def}}{=} \sum_{(X,Y)} n(\delta_{Y|X}), \tag{30}$$

where

$$n(\delta_{Y|X}) \stackrel{\text{def}}{=} \begin{cases} \delta_{Y|X} \cdot n_{W|Z} & \text{if } \delta_{Y|X} > 0 \text{ and} \\ & (Z, W, N_{W|X}) \text{ supports it} \\ 0 & \text{if } \delta_{Y|X} = 0. \end{cases}$$

From (29), $\mathsf{OBJ} = \mathsf{LogSizeBound}_{\Gamma_n \cap \mathsf{HDC}}(P)$. We assume that the input size is within the budget, i.e. $N \le 2^{\mathsf{OBJ}}$. (We will see later how to enforce this assumption when it is not satisfied.) We will keep every step of the algorithm to run within the budget of $\tilde{O}(2^{\mathsf{OBJ}})$. Specifically, we will keep every intermediate relation the algorithm computes of size $\le 2^{\mathsf{OBJ}}$. In the base case, the algorithm stops as soon as there is a relation $R \in \mathcal{R}$ with attribute set $\mathbf{A}_B$ where $B \in \mathcal{B}$, in which case R is a target relation. Otherwise, the algorithm takes steps which are modeled after the proof steps. Let $\mathbf{f}$ be the first proof step (instruction) with weight w, i.e. $\mathsf{ProofSeq} = (w \cdot \mathbf{f}, \mathsf{ProofSeq}')$ where $\mathsf{ProofSeq}'$ contains the rest of the instructions.

Case 1: $\mathbf{f} = \mathbf{s}_{I,J}$ is a submodularity step. By definition of proof sequence, $\boldsymbol{\delta} + w \cdot \mathbf{s}_{I,J} \ge \mathbf{0}$, and thus $\delta_{I|I\cap J} \ge w > 0$. Let $(Z, W, N_{W|Z}) \in \mathsf{DC}$ be the degree constraint supporting $\delta_{I|I\cap J}$; then $Z \subseteq I \cap J$, $W \subseteq I$, and $W - Z = I - I \cap J$. The algorithm proceeds by setting $\boldsymbol{\delta}' = \boldsymbol{\delta} + w \cdot \mathbf{s}_{I,J}$. Note that $\delta'_{I\cup J|J}$ is now positive, and so it needs a supporting degree constraint. From the fact that $W - Z = I - I \cap J = I \cup J - J$, $(Z, W, N_{W|Z})$ can support $\delta'_{I\cup J|J}$. Since $\mathbf{f}$ was the next step in the proof sequence, $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \le \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ is a Shannon flow inequality with proof sequence $\mathsf{ProofSeq}'$.

Case 2: $\mathbf{f} = \mathbf{m}_{X,Y}$ is a monotonicity step. By definition of proof sequence, $\boldsymbol{\delta} + w \cdot \mathbf{m}_{X,Y} \ge \mathbf{0}$, and thus $\delta_{Y|\emptyset} \ge w > 0$. Let $(\emptyset, Y, N_{Y|\emptyset}) \in \mathsf{DC}$ be the degree constraint supporting $\delta_{Y|\emptyset}$, and $R \in \mathcal{R}$ be a guard for this degree constraint (hence $|\Pi_Y(R)| \le N_{Y|\emptyset}$). Then, we proceed by setting $\boldsymbol{\delta}' = \boldsymbol{\delta} + w \cdot \mathbf{m}_{X,Y}$. Note that $\delta'_{X|\emptyset}$ is positive, and so it needs a supporting degree constraint, which is the newly added degree constraint $(\emptyset, X, N_{X|\emptyset})$, guarded by R, where $N_{X|\emptyset} \stackrel{\text{def}}{=} |\Pi_X(R)| \le |R| \le 2^{\mathsf{OBJ}}$.

Case 3: $\mathbf{f} = \mathbf{d}_{Y,X}$ is a decomposition step with weight w. From $\boldsymbol{\delta} + w \cdot \mathbf{d}_{Y,X} \ge \mathbf{0}$, it follows that $\delta_{Y|\emptyset} \ge w > 0$. From the guarantee that $\delta_{Y|\emptyset}$ has a supporting degree constraint, it follows that there is a relation $R \in \mathcal{R}$ guarding $(\emptyset, Y, N_{Y|\emptyset})$, which means $|\Pi_Y(R)| \le N_{Y|\emptyset}$. We showed in [3] that R can be partitioned into at most $(k = 2\log_2 |R| \le 2 \cdot \mathsf{OBJ})$ sub-tables $R^{(1)}, \ldots, R^{(k)}$ such that $N^{(j)}_{Y|X} N^{(j)}_{X|\emptyset} \le N_{Y|\emptyset}$, for all $j \in [k]$, where

$$\begin{aligned} N^{(j)}_{Y|X} &\stackrel{\text{def}}{=} \max_{\mathbf{t}_X \in \Pi_X(R^{(j)})} \deg_{R^{(j)}}(Y|\mathbf{t}_X), \\ N^{(j)}_{X|\emptyset} &\stackrel{\text{def}}{=} |\Pi_X(R^{(j)})|. \end{aligned} \tag{31}$$

(See (18) in Definition 2.4.) For each of these sub-tables $R^{(j)}$ of R, we create a subproblem with the same input tables but with R replaced by $R^{(j)}$. The jth subproblem has degree constraints $\mathsf{DC}^{(j)}$ where

$$\mathsf{DC}^{(j)} = \mathsf{DC} \cup \left\{(\emptyset, X, N^{(j)}_{X|\emptyset}), (X, Y, N^{(j)}_{Y|X})\right\}.$$

The table $R^{(j)}$ guards both of the new degree constraints. Set $\boldsymbol{\delta} = \boldsymbol{\delta} + w \cdot \mathbf{d}_{Y,X}$. The jth subproblem is on the Shannon flow inequality $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \le \langle \boldsymbol{\delta}, \mathbf{h}\rangle$ with proof sequence $\mathsf{ProofSeq}'$.

Case 4: $\mathbf{f} = \mathbf{c}_{X,Y}$ is a composition step with weight w. By definition of proof sequence, $\boldsymbol{\delta} + w \cdot \mathbf{c}_{X,Y} \ge \mathbf{0}$, and thus $\delta_{Y|X} \ge w > 0$ and $\delta_{X|\emptyset} \ge w > 0$. From the facts that $\delta_{Y|X}$ and $\delta_{X|\emptyset}$ have supports, there must be two sets $Z \subseteq X$ and $W \subseteq Y$ for which $W - Z = Y - X$ and $(Z, W, N_{W|Z}) \in \mathsf{DC}$ which is guarded by an input relation R; and an input relation S for which $|\Pi_X(S)| \le N_{X|\emptyset}$. Note that $X \cup (W - Z) = X \cup (Y - X) = Y$. We consider two cases:

(Case 4a) If $N_{X|\emptyset} \cdot N_{W|Z} \le 2^{\mathsf{OBJ}}$, then we can compute the table $T(\mathbf{A}_Y) \stackrel{\text{def}}{=} \Pi_X(S) \bowtie \Pi_W(R)$ by going over all tuples in $\Pi_X(S)$ and expanding them using matching tuples in $\Pi_W(R)$. The runtime of the join is $\tilde{O}(N_{X|\emptyset} \cdot N_{W|Z}) = \tilde{O}(2^{\mathsf{OBJ}})$, and the size of T is $\le 2^{\mathsf{OBJ}}$. The Shannon flow inequality is modified by setting $\boldsymbol{\delta} = \boldsymbol{\delta} + w \cdot \mathbf{c}_{X,Y}$, with the the proof sequence $\mathsf{ProofSeq}'$, and the set of degree constraints is extended by adding the constraint $(\emptyset, Y, N_{Y|\emptyset} \stackrel{\text{def}}{=} |T|)$, guarded by T.

(Case 4b) If $N_{X|\emptyset} \cdot N_{W|Z} > 2^{\mathsf{OBJ}}$, then we *will not* perform this join. Instead, we *restart* the subproblem with a fresh inequality. In particular, set $\boldsymbol{\delta} = \boldsymbol{\delta} + w \cdot \mathbf{c}_{X,Y}$. Now we have $\delta_{Y|\emptyset} \ge w$. We restart the problem with the inequality $\langle \boldsymbol{\lambda}', \mathbf{h}\rangle \le \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ satisfying the conditions stated in Lemma 3.9. We show in the proof of Theorem 1.5 (in Appendix D) that in this case $\|\boldsymbol{\lambda}\|_1 > w$, and the new inequality is a Shannon flow inequality with a smaller upper bound on the proof sequence length.

5. DEGREE-AWARE WIDTH PARAMETERS

5.1 Minimax and maximin widths

We slightly reformulate existing width parameters under a common framework. Recall from Section 2 that there are

two classes of width parameters: the first class captures the class of algorithms which seek the best tree decomposition with the worst bag runtime, while the second class captures the class of algorithms which adapt the tree decomposition to the instance at hand. In the definitions below, the maximin width notion is from Marx [31,32].

Definition 5.1. Let $\mathcal{F}$ denote a topologically closed class of non-negative set functions on $[n]$. The *$\mathcal{F}$-minimax width* and *$\mathcal{F}$-maximin width* of a query Q are defined by

$$\mathsf{Minimaxwidth}_{\mathcal{F}}(Q) \stackrel{\text{def}}{=} \min_{(T,\chi)} \max_{t\in V(T)} \max_{h\in\mathcal{F}} h(\chi(t)), \quad (32)$$

$$\mathsf{Maximinwidth}_{\mathcal{F}}(Q) \stackrel{\text{def}}{=} \max_{h\in\mathcal{F}} \min_{(T,\chi)} \max_{t\in V(T)} h(\chi(t)). \quad (33)$$

These width notions are used by specializing $\mathcal{F}$ to capture two aspects of the input. The first aspect is either the entropic functions or some relaxation, coming from the chain of inclusion $\mathsf{M}_n \subset \overline{\Gamma}^*_n \subset \Gamma_n \subset \mathsf{SA}_n$. The second aspect models the granularity level of statistics we know from the input database instance, with the following inclusion chain

$$\mathsf{HDC} \subset \mathsf{HCC} \subset \mathsf{ED}\cdot\log N \subset \mathsf{VD}\cdot\log N. \quad (34)$$

Note that the bounds in the constraints HDC are not normalized as in the sets ED or VD in the traditional width parameters. This is because normalizing makes it less general than it can be, and it does not make practical sense to assume that all degree bounds are the same! (For example, the FD-based degree bounds are always 0, while the relation-size-based degree bounds are $\log_2 N_F$.) Consequently, we used the $\log_2 N$ scaled up versions of the traditional width parameters to compare with our new width parameters.

From these specializations, the minimax and maximin widths capture all width parameters we discussed in Section 2, summarized in the following proposition. (See [4] for the proof.)

Proposition 5.2. *Let Q be a conjunctive query with no FDs whose hypergraph is $\mathcal{H} = ([n], \mathcal{E})$. Then the followings hold (Recall notation from Section 2):*

$$1+\mathsf{tw}(\mathcal{H}) = \mathsf{Minimaxwidth}_{\mathcal{F}\cap\mathsf{VD}}(Q) = \mathsf{Maximinwidth}_{\mathcal{F}\cap\mathsf{VD}}(Q) \quad \forall\mathcal{F}\in\{\mathsf{M}_n, \overline{\Gamma}^*_n, \Gamma_n, \mathsf{SA}_n\} \quad (35)$$

$$\mathsf{ghtw}(\mathcal{H}) = \mathsf{Minimaxwidth}_{\mathsf{SA}_n\cap\mathsf{ED}}(Q) = \mathsf{Maximinwidth}_{\mathsf{SA}_n\cap\mathsf{ED}}(Q) \quad (36)$$

$$\mathsf{fhtw}(\mathcal{H}) = \mathsf{Minimaxwidth}_{\mathcal{F}\cap\mathsf{ED}}(Q) \quad \forall\mathcal{F}\in\{\mathsf{M}_n, \overline{\Gamma}^*_n, \Gamma_n\} \quad (37)$$

$$\mathsf{subw}(\mathcal{H}) = \mathsf{Maximinwidth}_{\Gamma_n\cap\mathsf{ED}}(Q) \quad (38)$$

$$\mathsf{adw}(\mathcal{H}) = \mathsf{Maximinwidth}_{\mathsf{M}_n\cap\mathsf{ED}}(Q). \quad (39)$$

While SA_n yields too large of an upperbound, and M_n only yields a lowerbound, depending on the constraints we want to impose, some parts of the hierarchy collapse. The following observation is straightforward:

Lemma 5.3. *If $\mathcal{G}\subseteq\mathcal{F}$ are two classes of functions,*

$$\mathsf{Minimaxwidth}_{\mathcal{G}}(\mathcal{H}) \le \mathsf{Minimaxwidth}_{\mathcal{F}}(\mathcal{H})$$
$$\mathsf{Maximinwidth}_{\mathcal{G}}(\mathcal{H}) \le \mathsf{Maximinwidth}_{\mathcal{F}}(\mathcal{H}).$$

For a fixed class $\mathcal{F}$ of functions, we have

$$\mathsf{Maximinwidth}_{\mathcal{F}}(\mathcal{H}) \le \mathsf{Minimaxwidth}_{\mathcal{F}}(\mathcal{H}).$$

5.2 New width parameters

Using the maximin and minimax formalism, we extend the traditional width parameters to handle general degree constraints. As shown by Theorem 1.2 we know that there is a gap between the polymatroid bound and the entropic bound; and hence it is natural to use $\overline{\Gamma}^*_n$ itself instead of some approximation of it.

Definition 5.4. We define the following width parameters for queries Q with degree constraints DC. The first two parameters are generalizations of fhtw and subw under degree constraints, and the last two are their entropic versions:

$$\mathsf{da\text{-}fhtw}(Q) \stackrel{\text{def}}{=} \mathsf{Minimaxwidth}_{\Gamma_n\cap\mathsf{HDC}}(Q) \quad (40)$$

$$\mathsf{da\text{-}subw}(Q) \stackrel{\text{def}}{=} \mathsf{Maximinwidth}_{\Gamma_n\cap\mathsf{HDC}}(Q) \quad (41)$$

$$\mathsf{eda\text{-}fhtw}(Q) \stackrel{\text{def}}{=} \mathsf{Minimaxwidth}_{\overline{\Gamma}^*_n\cap\mathsf{HDC}}(Q) \quad (42)$$

$$\mathsf{eda\text{-}subw}(Q) \stackrel{\text{def}}{=} \mathsf{Maximinwidth}_{\overline{\Gamma}^*_n\cap\mathsf{HDC}}(Q). \quad (43)$$

(da stands for "degree-aware", and eda for "entropic degree-aware".) The following relationships hold between these four quantities.

Proposition 5.5. *For any query Q with degree constraints*

$$\begin{array}{ccc} \mathsf{eda\text{-}subw}(Q) & \le & \mathsf{eda\text{-}fhtw}(Q) \\ \rotatebox{90}{$\le$} & & \rotatebox{90}{$\le$} \\ \mathsf{da\text{-}subw}(Q) & \le & \mathsf{da\text{-}fhtw}(Q). \end{array}$$

The quantities $\mathsf{eda\text{-}fhtw}(Q)$ *and* $\mathsf{da\text{-}subw}(Q)$ *are not comparable. The gap between the two sides of any of the above four inequalities can be made arbitrarily large by some input.*

Due to the fact that every non-negative modular set function is entropic, we have

Corollary 5.6. *When Q has only edge domination constraints* ED *(i.e. no FD nor proper degree bounds), we have* $\mathsf{adw}(Q) \le \mathsf{eda\text{-}subw}(Q)$.

Following Marx [31,32], for these queries $\mathsf{da\text{-}subw}(Q) = \mathsf{subw}(Q) = O(\mathsf{adw}^4(Q)) = O(\mathsf{eda\text{-}subw}^4(Q))$. Thus, when there is no FD nor proper degree bounds, if a class of queries has bounded eda-subw, then it has bounded da-subw. It is open whether or not the same relationship holds when Q has FDs and/or degree bounds.

We have mentioned quite a few known and proved new bounds in this paper. The bounds can be summarized systematically as follows. Each bound is identified by coordinates (X, Y, Z). The X-axis represents the entropy approximation that is being used: one starts from the desired target $\overline{\Gamma}^*_n$, then relaxes it to Γ_n and SA_n. The inclusion chain is $\overline{\Gamma}^*_n \subset \Gamma_n \subset \mathsf{SA}_n$. The Y-axis represents the constraints we can extract from the input database instance, where we can go from bounding domain sizes, relation sizes, to incorporating more refined degree bounds and functional dependencies. One chain of inclusion was given by (34). The Z-axis represents the level of sophistication of the query plan that is being considered in this bound. The simplest query plan just joins everything together without computing any tree decomposition – or, equivalently, this is the plan that uses the trivial tree decomposition with one bag containing all attributes. (Recall the bounds $\mathsf{DAEB}(Q)$ and $\mathsf{DAPB}(Q)$ from (19).) Then, one can get more sophisticated with computing a tree decomposition before computing the bags.

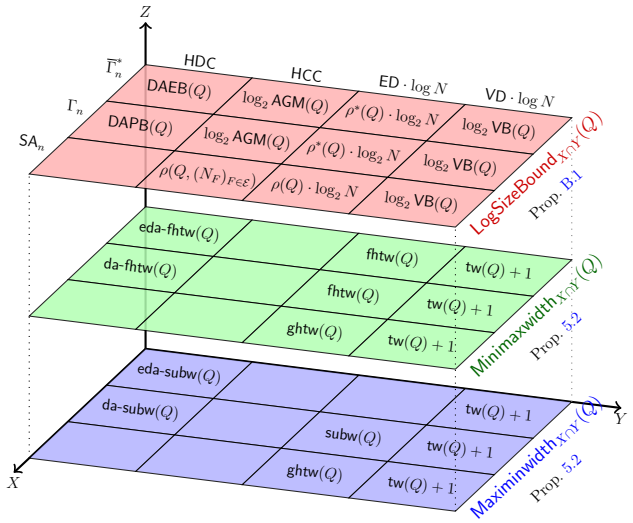

Figure 2: A hierarchy of bounds

And, lastly the query plan can also be adaptive to the input instance, yielding the submodular-width style of complexity. The bounds are summarized in Figure 2. If bound A has coordinates (X_1, Y_1, Z_1) that are smaller than the coordinates (X_2, Y_2, Z_2) of bound B (i.e. $X_1 \le X_2 \wedge Y_1 \le Y_2 \wedge Z_1 \le Z_2$), then bound $A \le$ bound B.

5.3 Achieving degree-aware width parameters

With increasing levels of complexity, the corollaries below explain how PANDA can be used to evaluate a (full or Boolean) conjunctive query achieving the degree-aware polymatroid size bound defined in (19), the degree-aware version of the fractional hypertree width defined in (40), and the degree-aware version of the submodular width defined in (41). Then, going beyond conjunctive queries, we show that PANDA can be used to solve aggregate queries by combining PANDA with the FAQ framework. The missing proofs can be found in the full version [4].

We start with two straightforward corollaries.

Corollary 5.7. *A full or Boolean conjunctive query Q with degree constraints can be solved by PANDA in time*

$$\tilde{O}(N + \text{poly}(\log N) \cdot 2^{\mathsf{DAPB}(Q)}).$$

Corollary 5.8. *A full or Boolean conjunctive query Q with degree constraints can be solved by PANDA in time*

$$\tilde{O}(N + \text{poly}(\log N) \cdot 2^{\mathsf{da\text{-}fhtw}(Q)} + |\mathsf{output}|).$$

The third corollary is on achieving the the degree-aware submodular width. Unlike the first two corollaries, proving this requires a couple of new ideas. In order to compute da-subw, even in a bruteforce manner, we need an auxiliary lemma, which is somewhat related to Neumann's minimax theorem [20].

Lemma 5.9. *Let A and B be two finite sets, and $f : A \times B \to \mathbb{R}$ be any function. Let B^A denote the set of all maps from A to B. Then, the following holds:*

$$\min_{a\in A}\max_{b\in B} f(a,b) = \max_{\beta\in B^A}\min_{a\in A} f(a,\beta(a)).$$

Intuitively, on the LHS we select for each $a \in A$ a neighbor b for which $f(a,b)$ is maximized; call such neighbor a's "representative". Then, we select the a with the least-weight representative. On the RHS, we have a "representative selector" β; we pick the a-value with the least-weight selected representative, and then maximize over all selectors.

Corollary 5.10. *A full or Boolean conjunctive query Q with degree constraints DC can be solved by PANDA in time*

$$\tilde{O}(N + \text{poly}(\log N) \cdot 2^{\mathsf{da\text{-}subw}(Q)} + |\mathsf{output}|).$$

Proof. We first apply Lemma 5.9 to reformulate (41). To this end, we need a few notations. Let $\mathcal{M}$ be the set of all maps $\beta : \mathsf{TD} \to 2^{[n]}$, such that $\beta(T,\chi) = \chi(t)$ for some $t \in V(T)$. In English, β is a "bag selector" map that picks out a bag from each tree decomposition (T,χ). Let $\mathbf{B}$ be the collection of images of all $\beta \in \mathcal{M}$, i.e.

$$\mathbf{B} = \{\mathcal{B} \mid \mathcal{B} = \mathsf{image}(\beta) \text{ for some } \beta \in \mathcal{M}\}. \tag{44}$$

Using Lemma 5.9, we can rewrite (41) as follows.

$$\begin{aligned}
\mathsf{da\text{-}subw}(\mathcal{H}) &= \max_{h\in\Gamma_n\cap\mathsf{HDC}}\ \min_{(T,\chi)\in\mathsf{TD}}\ \max_{t\in V(T)} h(\chi(t)) \\
\text{(Lemma 5.9)} &= \max_{h\in\Gamma_n\cap\mathsf{HDC}}\ \max_{\beta\in\mathcal{M}}\ \min_{(T,\chi)} h(\beta(T,\chi)) \\
&= \max_{h\in\Gamma_n\cap\mathsf{HDC}}\ \max_{\beta\in\mathcal{M}}\ \min_{B\in\mathsf{image}(\beta)} h(B) \\
&= \max_{\beta\in\mathcal{M}}\ \max_{h\in\Gamma_n\cap\mathsf{HDC}}\ \min_{B\in\mathsf{image}(\beta)} h(B) \\
&= \max_{\mathcal{B}\in\mathbf{B}}\ \max_{h\in\Gamma_n\cap\mathsf{HDC}}\ \min_{B\in\mathcal{B}} h(B) \\
\text{(Lemma 3.2)} &= \max_{\mathcal{B}\in\mathbf{B}}\ \underbrace{\max_{h\in\Gamma_n\cap\mathsf{HDC}}\left\{\sum_{B\in\mathcal{B}}\lambda_B h(B)\right\}}_{\text{Linear program (21)}}
\end{aligned} \tag{45}$$

In (45), for a fixed $\mathcal{B} \in \mathbf{B}$ the inner max is exactly LP (21) whose dual is (22). In particular, to compute the da-subw(Q), we can solve a collection of linear programs and take the maximum solution among them. Since there is a different linear program for each valid choice of $\mathcal{B}$, the total number of linear programs is $\le 2^{2^n}$.

In order to compute Q in the desired time, we mimic this strategy in the algorithm. For each $\mathcal{B} \in \mathbf{B}$, we solve the LP (21). Let $(\boldsymbol{\delta}^*, \boldsymbol{\sigma}^*, \boldsymbol{\mu}^*)$ denote a dual optimal solution. From Proposition 3.3, $\langle \boldsymbol{\lambda}, \mathbf{h}\rangle \le \langle \boldsymbol{\delta}^*, \mathbf{h}\rangle$ is a Shannon flow inequality. On this input PANDA computes a tuple $\mathbf{T}_\mathcal{B} = (T_B)_{B\in\mathcal{B}}$ of tables such that, for every $\mathbf{a} \in Q$ there exists a $B \in \mathcal{B}$ for which $\Pi_B(\mathbf{a}) \in T_B$.

Let $M = |\mathbf{B}|$ and suppose $\mathbf{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$. We prove the following claims:

Claim 1: for every $(B_1, \ldots, B_M) \in \prod_{i=1}^M \mathcal{B}_i$, there is a tree decomposition $(T,\chi) \in \mathsf{TD}(Q)$ such that, for every tree node $t \in V(T)$, $\chi(t) = B_j$ for some $j \in [M]$. Breaking ties arbitrarily, we call this tree decomposition *the* tree decomposition (of Q) associated with the tuple $(B_1, \ldots, B_M)$.

Claim 2: for any tuple $(B_1, \ldots, B_M) \in \prod_{i=1}^M \mathcal{B}_i$ with associated tree decomposition (T,χ), define

$$J(B_1, \ldots, B_M) \stackrel{\text{def}}{=} \bowtie_{t\in V(T)} T_{\chi(t)}.$$

Then,

$$Q \subseteq \left(\bigcup_{(B_1,\ldots,B_M)\in\prod_{i=1}^M\mathcal{B}_i} \bowtie_{j=1}^M T_{B_j}\right) \tag{46}$$

$$\subseteq \left(\bigcup_{(B_1,\ldots,B_M)\in\prod_{i=1}^M \mathcal{B}_i} J(B_1,\ldots,B_M) \right). \quad (47)$$

Assuming the claims, the query can be computed by running Yannakakis algorithm to compute all the relations $J(B_1,\ldots,B_M)$ within a runtime of $\tilde{O}(2^{\mathsf{da\text{-}subw}(Q)} + |J(B_1,\ldots,B_M)\cap Q|)$. If we apply Yannakakis's algorithm straight up on $J(B_1,\ldots,B_M)$, then we can attain the runtime $\tilde{O}(2^{\mathsf{da\text{-}subw}(Q)}+|J(B_1,\ldots,B_M)|)$, because every table T_{B_j} has size bounded by $2^{\mathsf{da\text{-}subw}(Q)}$. To reduce the runtime to $\tilde{O}(2^{\mathsf{da\text{-}subw}(Q)} + |J(B_1,\ldots,B_M)\cap Q|)$, we semijoin-reduce every table T_{B_j} with every input relation. There are $\prod_{i=1}^M |\mathcal{B}_i|$ such problems, which is a query-complexity quantity.

We next prove Claim 1. Fix a tuple $(B_1,\ldots,B_M) \in \prod_{i=1}^M \mathcal{B}_i$. Suppose to the contrary that for *every* tree decomposition (T,χ) there is a tree node $t\in V(T)$ such that $\chi(t)\neq B_j$ for every $j\in[M]$. Call the bag $\chi(t)$ a *missed* bag of the tree decomposition. Consider a bag selector $\beta : \mathsf{TD}(Q) \to 2^{[n]}$ where $\beta(T,\chi)$ is exactly the missed bag of the tree decomposition (T,χ). Note that by definition of $\mathbf{B}$ we have $\mathsf{image}(\beta) = \mathcal{B}_k$ for some $k\in[M]$. This is a contradiction because B_k must then be the missed bag of some tree decomposition, but it is not missed anymore.

Finally, we prove Claim 2. Consider an output tuple $\mathbf{a}\in Q$. For each $j\in[M]$, let B_j denote the bag for which $\Pi_{B_j}(\mathbf{a}) \in T_{B_j}$. Then, obviously $\mathbf{a}\in J(B_1,\ldots,B_M)$. This proves the first inclusion in (47). The second inclusion is obvious because the join $J(B_1,\ldots,B_M)$ drops some tables from the join $\bowtie_{j=1}^M T_{B_j}$. □

Example 5.11. We illustrate how PANDA computes the query Q in Example 1.1 in time $\tilde{O}(N^{\mathsf{subw}(Q)})$. We use Marx' original definition of submodular width which, recall, corresponds to using base N for the logarithm, hence the expression $N^{\mathsf{subw}(Q)}$ instead of our $2^{\mathsf{da\text{-}subw}(Q)}$. We prove first that $\mathsf{subw}(Q)\le 3/2$. Since Q has two tree decompositions, shown in Fig. 1, we need to prove:

$$\min(\max(h(A_1A_2A_3), h(A_3A_4A_1)), \max(h(A_2A_3A_4), h(A_4A_1A_2))) \le 3/2$$

where h is any edge-dominated polymatroid h. (The first max corresponds to the first tree, the second max to the second tree.) Using the distributivity law of min over max, we convert the inequality from $\min(\max)\le 3/2$ into $\max(\min)\le 3/2$, which is equivalent to the conjunction of four inequalities:

$$\begin{aligned}
\min(h(A_1A_2A_3), h(A_2A_3A_4)) &\le 3/2\\
\min(h(A_1A_2A_3), h(A_4A_1A_2)) &\le 3/2\\
\min(h(A_3A_4A_1), h(A_2A_3A_4)) &\le 3/2\\
\min(h(A_3A_4A_1), h(A_4A_1A_2)) &\le 3/2
\end{aligned}$$

The first inequality follows from $h(A_1A_2A_3)+h(A_2A_3A_4)\le h(A_1A_2)+h(A_2A_3)+h(A_3A_4)$ (Example 4.1) and the latter is ≤ 3 because h is edge-dominated. The others are similar. This concludes $\mathsf{subw}(Q)\le 3/2$.

At this point PANDA converts each of the four Shannon-flow inequalities that it has just proven, into a disjunctive datalog rule. For example, the first inequality becomes precisely the disjunctive datalog rule P in Example 1.3. Then, it evaluates each of these four rules, obtaining four target relations $T_{123}, T_{234}, T_{341}, T_{412}$ (since each target occurs in two rules, PANDA takes their union; it also semi-joins each target with the input relations, to remove spurious tuples, e.g. it semi-joins T_{123} with R_{12} and R_{23}). Each of the four targets has size $\le N^{3/2}$ and the runtime so far is $\tilde{O}(N^{3/2})$. Finally, PANDA runs Yannakakis' algorithm for acyclic queries on the first tree (in essence, joining T_{123} with T_{341}) then separately on the second tree, and returns the union of the two results. We need to prove that it returns the correct output, and for that we use the following subtle argument. Let $\mathbf{a} = (a_1,a_2,a_3,a_4)$ be any tuple. We show that, if $\pi_F(\mathbf{a})\in R_F$ for every input relation R_F, then at least one of the two trees contains (the projection of) $\mathbf{a}$ in both its targets. Suppose the contrary: some tuple $\mathbf{a}$ fails this property. For example, $\pi_{123}(\mathbf{a})\notin T_{123}$ in the first tree, and $\pi_{412}(\mathbf{a})\notin T_{412}$ in the second tree. That implies that $T_{123}\vee T_{412}$ is not a model of the disjunctive datalog rule corresponding to the second inequality above, contradiction. Thus, each tuple $\mathbf{a}$ is fully included in some tree, and by taking the union PANDA returns all answers to Q.

Our final corollary concerns aggregate queries. By an aggregate query we mean a FAQ-query under one semiring (also called SumProd or FAQ-SS, see [2, 6]).

Corollary 5.12. *An* FAQ-SS *query Q without free variables and with degree constraints DC can be solved by PANDA in time*

$$\tilde{O}(N + \mathrm{poly}(\log N)\cdot 2^{\mathsf{da\text{-}subw}(Q)} + |\mathsf{output}|).$$

The last two corollaries prove Theorem 1.7. It is possible to handle FAQ-SS queries *with* free variables and degree constraints using our framework and algorithm; however, the result is slightly messy to state and explain, and thus we skip stating this simple observation here.

6. CONCLUDING REMARKS

Our negative answer to question 1 leads to a natural question: can we design an algorithm whose runtime matches the entropic bound under the presence of FDs or degree constraints? Worst-case optimal join algorithms [1, 34, 35, 37] were able to achieve this when there are no FDs. And, as shown in [3] there are classes of queries with FDs for which the answer is positive (using the chain algorithm). A natural direction is to extend the class of queries with FDs where the entropic bound can be met, beyond what was shown in [3]. Along the same line, the next natural open question is to design algorithms to evaluate disjunctive datalog queries matching the bound $\mathsf{LogSizeBound}_{\Gamma_n^*\cap\mathsf{HDC}}(P)$. From there, the possibility of achieving eda-subw and/or eda-fhtw is within reach. We already have an example where PANDA was able to achieve eda-subw and eda-fhtw: the 4-cycle example. In general, the inner-most column of Figure 2 contains open algorithmic questions: we do not know of algorithms meeting bounds involving both $\overline{\Gamma}_n^*$ and HDC. Another big open question is to remove the polylog factor from the runtime of PANDA.

Acknowledgements

This work is partly supported by NSF grants 1535565 and 1614738, and by DARPA under agreement #FA8750-15-2-0009. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright thereon.

7. REFERENCES

[1] M. Abo Khamis, H. Q. Ngo, C. Ré, and A. Rudra. Joins via geometric resolutions: Worst-case and beyond. In *PODS 2015*, pages 213–228, New York, NY, USA, 2015. ACM.

[2] M. Abo Khamis, H. Q. Ngo, and A. Rudra. FAQ: questions asked frequently. In *PODS 2016*, pages 13–28, 2016.

[3] M. Abo Khamis, H. Q. Ngo, and D. Suciu. Computing join queries with functional dependencies. In *PODS 2016*, pages 327–342, 2016.

[4] M. Abo Khamis, H. Q. Ngo, and D. Suciu. What do shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another? *CoRR*, abs/1612.02503, 2016.

[5] I. Adler. *Width functions for hypertree decompositions.* 2006. Ph.D. Dissertation, Albert-Ludwigs-Universität Freiburg. 2006.

[6] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[7] N. Alon. On the number of subgraphs of prescribed type of graphs with a given number of edges. *Israel J. Math.*, 38(1-2):116–130, 1981.

[8] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

[9] M. Alviano, W. Faber, N. Leone, S. Perri, G. Pfeifer, and G. Terracina. The disjunctive datalog system DLV. In O. de Moor, G. Gottlob, T. Furche, and A. J. Sellers, editors, *Datalog Reloaded - First International Workshop, Datalog 2010, Oxford, UK, March 16-19, 2010. Revised Selected Papers*, volume 6702 of *Lecture Notes in Computer Science*, pages 282–301. Springer, 2010.

[10] M. Armbrust, K. Curtis, T. Kraska, A. Fox, M. J. Franklin, and D. A. Patterson. PIQL: success-tolerant query processing in the cloud. *PVLDB*, 5(3):181–192, 2011.

[11] M. Armbrust, A. Fox, D. A. Patterson, N. Lanham, B. Trushkowsky, J. Trutna, and H. Oh. SCADS: scale-independent storage for social computing applications. In *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2009, Online Proceedings*, 2009.

[12] M. Armbrust, E. Liang, T. Kraska, A. Fox, M. J. Franklin, and D. A. Patterson. Generalized scale independence through incremental precomputation. In *SIGMOD 2013*, pages 625–636, 2013.

[13] A. Atserias, M. Grohe, and D. Marx. Size bounds and query plans for relational joins. In *FOCS*, pages 739–748. IEEE Computer Society, 2008.

[14] N. Bakibayev, T. Kociský, D. Olteanu, and J. Zavodny. Aggregation and ordering in factorised databases. *PVLDB*, 6(14):1990–2001, 2013.

[15] M. Benedikt, J. Leblay, and E. Tsamoura. Querying with access patterns and integrity constraints. *PVLDB*, 8(6):690–701, 2015.

[16] M. Benedikt, B. ten Cate, and E. Tsamoura. Generating plans from proofs. *ACM Trans. Database Syst.*, 40(4):22, 2016.

[17] Y. Cao, W. Fan, T. Wo, and W. Yu. Bounded conjunctive queries. *PVLDB*, 7(12):1231–1242, 2014.

[18] F. R. K. Chung, R. L. Graham, P. Frankl, and J. B. Shearer. Some intersection theorems for ordered sets and graphs. *J. Combin. Theory Ser. A*, 43(1):23–37, 1986.

[19] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artif. Intell.*, 113(1-2):41–85, 1999.

[20] D. Du and P. Pardalos. *Minimax and Applications.* Nonconvex Optimization and Its Applications. Springer US, 1995.

[21] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM Trans. Database Syst.*, 22(3):364–418, 1997.

[22] W. Fischl, G. Gottlob, and R. Pichler. General and Fractional Hypertree Decompositions: Hard and Easy Cases. *ArXiv e-prints*, Nov. 2016.

[23] E. C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In H. E. Shrobe, T. G. Dietterich, and W. R. Swartout, editors, *Proceedings of the 8th National Conference on Artificial Intelligence. Boston, Massachusetts, July 29 - August 3, 1990, 2 Volumes.*, pages 4–9. AAAI Press / The MIT Press, 1990.

[24] E. Friedgut and J. Kahn. On the number of copies of one hypergraph in another. *Israel J. Math.*, 105:251–256, 1998.

[25] T. Gogacz and S. Toruńczyk. Entropy bounds for conjunctive queries with functional dependencies. In *Proc. 20th International Conference on Database Theory (ICDT)*, 2017.

[26] G. Gottlob, G. Greco, N. Leone, and F. Scarcello. Hypertree decompositions: Questions and answers. In *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 57–74, 2016.

[27] G. Gottlob, S. T. Lee, G. Valiant, and P. Valiant. Size and treewidth bounds for conjunctive queries. *J. ACM*, 59(3):16, 2012.

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

[29] M. Grohe and D. Marx. Constraint solving via fractional edge covers. In *SODA*, pages 289–298. ACM Press, 2006.

[30] M. Grohe and D. Marx. Constraint solving via fractional edge covers. *ACM Transactions on Algorithms*, 11(1):4, 2014.

[31] D. Marx. Tractable structures for constraint satisfaction with truth tables. *Theory Comput. Syst.*, 48(3):444–464, 2011.

[32] D. Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM*, 60(6):Art. 42, 51, 2013.

[33] F. Matus. Infinitely many information inequalities. In *2007 IEEE International Symposium on Information Theory*, pages 41–44. IEEE, 2007.

[34] H. Q. Ngo, E. Porat, C. Ré, and A. Rudra. Worst-case optimal join algorithms: [extended abstract]. In *PODS*, pages 37–48, 2012.

[35] H. Q. Ngo, C. Ré, and A. Rudra. Skew strikes back: new developments in the theory of join algorithms. *SIGMOD Record*, 42(4):5–16, 2013.

[36] A. Schrijver. *Theory of linear and integer programming*. Wiley-Interscience Series in Discrete Mathematics. John Wiley & Sons Ltd., Chichester, 1986. A Wiley-Interscience Publication.

[37] T. L. Veldhuizen. Triejoin: A simple, worst-case optimal join algorithm. In N. Schweikardt, V. Christophides, and V. Leroy, editors, *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014.*, pages 96–106. OpenProceedings.org, 2014.

[38] R. W. Yeung. *Information Theory and Network Coding*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[39] Z. Zhang and R. W. Yeung. On characterization of entropy function via information inequalities. *IEEE Transactions on Information Theory*, 44(4):1440–1452, 1998.

APPENDIX

A. MISSING DETAILS FROM SECTION 1

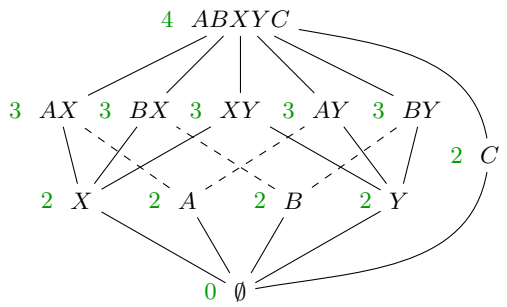

Figure 3: A polymatroid h (shown in green).

Proof of Theorem 1.2. Consider the following query:

$$Q(A,B,X,Y,C) \text{ :- } K(A,B,X,Y,C),$$
$$R(X,Y), S(A,X), T(A,Y), U(B,X), V(B,Y), W(C)$$

with given cardinality constraints $|R|, |S|, |T|, |U|, |V| \leq N^3$, $|W| \leq N^2$ (none on K), and the following keys in K: AB, AXY, BXY, AC, XC, YC. We show that the entropic bound is $\leq N^{43/11}$ while the polymatroid bound is N^4. This statement follows from two claims. First, the following non-Shannon inequality holds for all entropic functions:

$$\begin{aligned}11h(ABXYC) \leq 3h(XY) + 3h(AX) + 3h(AY) + h(BX) \\ + h(BY) + 5h(C) + (h(ABXYC|AB) + 4h(ABXYC|AXY) \\ + h(ABXYC|BXY)) + (h(ABXYC|AC) + 2h(ABXYC|XC) \\ + 2h(ABXYC|YC)). \quad (48)\end{aligned}$$

The first claim implies $11 \log |Q| \leq 11 \log N^3 + 5 \log N^2 = 43 \log N$. Second, there exists a polymatroid h satisfying all cardinality and FD constraints such that $h(ABXYC) = 4 \log N$. The two claims show a ratio of $N^{1/11}$ between the two bounds.

We now prove the first claim. Let $I(\mathbf{X}; \mathbf{Y}|\mathbf{Z}) := H(\mathbf{XZ}) + H(\mathbf{YZ}) - H(\mathbf{XYZ}) - H(\mathbf{Z})$ denote the conditional mutual information between random variables $\mathbf{X}$ and $\mathbf{Y}$ conditioned on random variables $\mathbf{Z}$. In a breakthrough paper in information theory, Zhang and Yeung [39] proved that $\overline{\Gamma}_4^* \subsetneq \Gamma_4$ by proving that the following inequality is a non-Shannon-type inequality (see [38, Th.15.7]):

$$2I(X;Y) \leq I(A;B) + I(A;XY) + 3I(X;Y|A) + I(X;Y|B).$$

We expand and rearrange it, obtaining

$$\begin{aligned}h(AB) + 4h(AXY) + h(BXY) \leq 3h(XY) + 3h(AX) \\ + 3h(AY) + h(BX) + h(BY) - (h(A) + 2h(X) + 2h(Y))\end{aligned}$$

Add $h(ABXYC|AB) + 4h(ABXYC|AXY) + h(ABXYC|BXY)$ to both sides:

$$\begin{aligned}6h(ABXYC) \leq 3h(XY) + 3h(AX) + 3h(AY) + h(BX) + h(BY) \\ - (h(A) + 2h(X) + 2h(Y)) + (h(ABXYC|AB) + \\ 4h(ABXYC|AXY) + h(ABXYC|BXY)) \quad (49)\end{aligned}$$

From these three Shannon-type inequalities $h(A) + h(C) \geq h(AC)$, $h(X) + h(C) \geq h(XC)$, and $h(Y) + h(C) \geq h(YC)$ we derive the following:

$$\begin{aligned}5h(ABXYC) \leq h(A) + 2h(X) + 2h(Y) + 5h(C) + \\ (h(ABXYC|AC) + 2h(ABXYC|XC) + 2h(ABXYC|YC)) \quad (50)\end{aligned}$$

By adding Eq.(49) and (50) we obtain (48).

The second claim is shown in Fig. 3. The figure shows a polymatroid h on the five variables A, B, X, Y, C. For each missing set of variables $\mathbf{Z}$, $h(\mathbf{Z}) \stackrel{\text{def}}{=} h(\mathbf{Z}^+)$, where $\mathbf{Z}^+$ is the smallest set shown in the figure that contains $\mathbf{Z}$: in other words, the figure shows the closed sets of a closure on $\{A, B, X, Y, C\}$. For example, $h(AB) \stackrel{\text{def}}{=} h(AB^+) = h(ABXYC) = 4$, etc. One can check that h satisfies all cardinality constraints and functional dependencies (using base N for the logarithm): for example $h(XY) = 3$, $h(AX) = 3$, and $AB \to XYC$ (because $h(AB) = h(ABXYC)$) etc.

Finally, to construct a query Q with an amplified gap between the two bounds, consider a query Q which is a cross-product of $11s$ independent copies of ZY. □

Due to space limitation, We prove only the first part of Theorem 1.4, leaving the second part in the full version [4].

Proof of part (i) of Theorem 1.4. Since $\overline{\Gamma}_n^* \subseteq \Gamma_n$, the second inequality is trivial. To show the first inequality, let T denote the set of all tuples $\mathbf{t}$ satisfying the body of P; construct a set $\overline{T}$ of tuples as follows. We scan though tuples $\mathbf{t} \in T$ one at a time, and either add $\mathbf{t}$ to $\overline{T}$ or ignore $\mathbf{t}$. To decide whether to add $\mathbf{t}$ to $\overline{T}$, we also keep a collection of tables $\mathbf{T} = (\overline{T}_B)_{B \in \mathcal{B}}$. These tables shall form a model of the disjunctive datalog rule P. Initially $\overline{T}$ and all the $\overline{T}_B$ are empty. Consider the next tuple $\mathbf{t}$ taken from T. If $\Pi_B(\mathbf{t}) \in \overline{T}_B$ for *any* $B \in \mathcal{B}$, then we ignore $\mathbf{t}$. Otherwise, we add $\Pi_B(\mathbf{t})$ to $\overline{T}_B$ for *every* $B \in \mathcal{B}$, and add $\mathbf{t}$ to $\overline{T}$. In the end, obviously the collection $(\overline{T}_B)_{B \in \mathcal{B}}$ is a model of the disjunctive datalog rule. Furthermore, by construction the tuples $\mathbf{t} \in \overline{T}$ satisfy the property that: for every two different tuples $\mathbf{t}, \mathbf{t}' \in \overline{T}$, every $B \in \mathcal{B}$, we have $\Pi_B(\mathbf{t}) \neq \Pi_B(\mathbf{t}')$ and both $\Pi_B(\mathbf{t})$ and $\Pi_B(\mathbf{t}')$ are in $\overline{T}_B$.

Now, construct a joint probability distribution on n variables by picking uniformly a tuple from $\overline{T}$. Let $\overline{h}$ denote the entropy function of this distribution, then, from the above property

$$\log_2 |\overline{T}| = \overline{h}([n]) = \overline{h}(B), \forall B \in \mathcal{B}. \tag{51}$$

Furthermore, $\overline{h} \in \overline{\Gamma}_n^*$ by definition, and $\overline{h} \in \mathsf{HDC}$ because T was a set of tuples satisfying all input degree constraints. Consequently,

$$\begin{aligned}
\min_{(T_B)_{B\in\mathcal{B}}} \max_{B\in\mathcal{B}} \log_2 |T_B| &\le \max_{B\in\mathcal{B}} \log_2 |\overline{T}_B| \\
\text{(due to (51))} &= \max_{B\in\mathcal{B}} \overline{h}(B) \\
\text{(due to (51))} &= \min_{B\in\mathcal{B}} \overline{h}(B) \\
\text{(because } \overline{h} \in \overline{\Gamma}_n^* \cap \mathsf{HDC}) &\le \max_{h\in\overline{\Gamma}_n^*\cap\mathsf{HDC}} \min_{B\in\mathcal{B}} h(B) \\
&= \mathsf{LogSizeBound}_{\overline{\Gamma}_n^*\cap\mathsf{HDC}}(P).
\end{aligned}$$

□

B. MISSING DETAILS FROM SECTION 2

Here we revisit known output size bounds for conjunctive queries without FDs nor degree bounds, and we reformulate them from the perspective of our framework. Recall that a full conjunctive query is a special case of a disjunctive datalog rule: It is a disjunctive datalog rule with only one target $B = [n]$. Hence for a full conjunctive query Q, the log-size-bound given by (6) simplifies to

$$\mathsf{LogSizeBound}_{\mathcal{F}}(Q) \stackrel{\text{def}}{=} \max_{h\in\mathcal{F}} h([n]).$$

The full version [4] proves the following simple proposition which recaps the major known bounds under one umbrella. Since the proposition is restricted to full conjunctive queries rather than the more general disjunctive datalog rules, it makes stronger claims about size bounds: In particular, some bounds collapse part of the hierarchy of function classes: $\mathsf{M}_n \subset \overline{\Gamma}_n^* \subset \Gamma_n \subset \mathsf{SA}_n$. (Review notation in Section 2.)

Proposition B.1. *Let Q be a full conjunctive query with no FDs whose hypergraph is $\mathcal{H} = ([n], \mathcal{E})$. Then the followings hold:*

$$\log_2 \mathsf{VB}(Q) = \mathsf{LogSizeBound}_{\mathcal{F}\cap(\mathsf{VD}\cdot\log N)}(Q), \quad \forall \mathcal{F} \in \{\mathsf{M}_n, \overline{\Gamma}_n^*, \Gamma_n, \mathsf{SA}_n\} \tag{52}$$

$$\rho(Q, (N_F)_{F\in\mathcal{E}}) = \mathsf{LogSizeBound}_{\mathsf{SA}_n\cap\mathit{HCC}}(Q) \tag{53}$$

$$\rho^*(Q) \cdot \log_2 N = \mathsf{LogSizeBound}_{\mathcal{F}\cap(\mathsf{ED}\cdot\log N)}(Q), \quad \forall \mathcal{F} \in \{\mathsf{M}_n, \overline{\Gamma}_n^*, \Gamma_n\} \tag{54}$$

$$\log_2 \mathit{AGM}(Q) = \mathsf{LogSizeBound}_{\mathcal{F}\cap\mathit{HCC}}(Q), \quad \forall \mathcal{F} \in \{\mathsf{M}_n, \overline{\Gamma}_n^*, \Gamma_n\} \tag{55}$$

The top part of Figure 2 depicts all the bounds in Proposition B.1. Regarding the AGM-bound, the fact that the size bound on Γ_n is equal to the size bound on M_n allows us to compute the bound efficiently (polynomial time in query complexity). By contrast, computing the integral edge cover bound is NP-hard in query complexity, despite being the worse bound: over the more relaxed function class SA_n. From the above proposition and the chains of inclusion $M_n \subset \overline{\Gamma}_n^* \subset \Gamma_n \subset \mathsf{SA}_n$ and $\mathsf{HCC} \subset \mathsf{VD} \cdot \log N$, we have $\mathsf{VB}(Q) \ge 2^{\rho(Q,(N_F)_{F\in\mathcal{E}})} \ge \mathsf{AGM}(Q)$ for any Q.

C. MISSING DETAILS FROM SECTION 3

C.1 Details from Section 3.1

The bound $\mathsf{LogSizeBound}_{\Gamma_n\cap\mathsf{HDC}}(P)$ is the optimal objective value of the following optimization problem:

$$\begin{array}{rll}
\max & \min_{B\in\mathcal{B}} h(B) & \\
s.t. & h(Y) - h(X) \le n_{Y|X}, & (X, Y, N_{Y|X}) \in \mathsf{DC} \\
& h(I\cup J|J) - h(I|I\cap J) \le 0, & I \perp J \\
& h(Y) - h(X) \ge 0, & \emptyset \ne X \subset Y \subseteq [n] \\
& h(Z) \ge 0, & \emptyset \ne Z \subseteq [n].
\end{array} \tag{56}$$

(Recall that implicitly we have $h(\emptyset) = 0$, and that $n_{Y|X} \stackrel{\text{def}}{=} \log_2 N_{Y|X}$.) Here, $X \perp Y$ means $X \not\subseteq Y$ and $Y \not\subseteq X$. The optimization problem above is not easy to handle. Lemma 3.2 allows us to reformulate it into an LP. Lemma 3.2 is a special case of the following lemma.

Lemma C.1 (A generalization of Lemma 3.2)**.** *Let $\mathbf{A} \in \mathbb{Q}^{\ell\times m}, \mathbf{b} \in \mathbb{R}^\ell$, and $\mathbf{C} \in \mathbb{Q}_+^{m\times p}$ be a matrix with columns $\mathbf{c}_1, \ldots, \mathbf{c}_p$. Consider the following maximin optimization problem:*

$$\max\{\min_{k\in[p]} \mathbf{c}_k^T \mathbf{x} \mid \mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge \mathbf{0}\} \tag{57}$$

If problem (57)'s objective value is positive and bounded, then there exists a vector $\boldsymbol{\lambda} \in \mathbb{Q}_+^p$ satisfying the following conditions:

(a) $\|\boldsymbol{\lambda}\|_1 = 1$

(b) The problem (57) has the same optimal objective value as the following linear program:

$$\max\{(\mathbf{C}\boldsymbol{\lambda})^T \mathbf{x} \mid \mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge \mathbf{0}\} \tag{58}$$

Proof of Lemma C.1. The dual of (58) can be written as

$$\min\{\mathbf{b}^T \mathbf{y} \mid \mathbf{A}^T \mathbf{y} \ge \mathbf{C}\boldsymbol{\lambda}, \mathbf{y} \ge \mathbf{0}\} \tag{59}$$

Let $\mathbf{1}_p \in \mathbb{R}^p$ be the all-1 vector. The LP (57) can be rewritten as

$$\max\{w \mid \mathbf{A}\mathbf{x} \le \mathbf{b}, \mathbf{1}_p w - \mathbf{C}^T \mathbf{x} \le \mathbf{0}, \mathbf{x} \ge \mathbf{0}, \quad w \ge 0\} \tag{60}$$

The dual of (60) is

$$\min\{\mathbf{b}^T \mathbf{y} \mid \mathbf{A}^T \mathbf{y} \ge \mathbf{C}\mathbf{z}, \mathbf{1}_p^T \mathbf{z} \ge 1, \mathbf{y} \ge \mathbf{0}, \quad \mathbf{z} \ge 0\} \tag{61}$$

Let $(w^*, \mathbf{x}^*)$ and $(\mathbf{z}^*, \mathbf{y}^*)$ be a pair of primal-optimal and dual-optimal solutions to (60) and (61). Define $\boldsymbol{\lambda} = \mathbf{z}^*$. To show that (57) is equivalent to (58), we show that (60) is equivalent to (58). In particular, we claim that $\mathbf{x}^*$ and $\mathbf{y}^*$ are a pair of primal-optimal and dual-optimal solution to (60) and (59), and that the objective values of (58) and (60) are identical. The fact that $\mathbf{x}^*$ and $\mathbf{y}^*$ are feasible to (58) and (59) is trivial. We are left to verify that they have the same objective value. Due to complementary slackness of the (60) and (61) pair, note that $w^* > 0$ implies $\mathbf{1}_p^T \mathbf{z}^* = 1$ and $(\mathbf{1}_p^T w^* - \mathbf{C}^T \mathbf{x}^*)^T \mathbf{z}^* = 0$. It follows that $\mathbf{b}^T \mathbf{y}^* = w^* = (\mathbf{1}_p^T \mathbf{z}^*) \cdot w^* = (\mathbf{C}^T \mathbf{x}^*)^T \cdot \mathbf{z}^* = (\mathbf{C}\mathbf{z}^*)^T \cdot \mathbf{x}^* = (\mathbf{C}\boldsymbol{\lambda})^T \cdot \mathbf{x}^*$. □

Proof of Proposition 3.3. The proposition is a simple consequence of Farkas' lemma. There are *many* variants of Farkas' lemma [36]. We use a version whose proof we also reproduce here because the proof is very short.

Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ be a matrix and $\mathbf{c} \in \mathbb{R}^n$ be a vector. Let $P = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \le \mathbf{0}, \mathbf{x} \ge \mathbf{0}\}$ be a polyhedron and $D =$

$\{\mathbf{y} \mid \mathbf{A}^T\mathbf{y} \geq \mathbf{c}, \mathbf{y} \geq \mathbf{0}\}$ be a dual polyhedron. Then, a variant of Farkas' lemma states that D is non-empty if and only if there is *no* $\mathbf{x} \in P$ such that $\mathbf{c}^T\mathbf{x} > 0$. To see this, note that the system $\{\mathbf{c}^T\mathbf{x} > 0, \mathbf{x} \in P\}$ is infeasible iff $\max\{\mathbf{c}^T\mathbf{x} \mid \mathbf{x} \in P\} = 0$, which by strong duality is equivalent to $\min\{\mathbf{0}^T\mathbf{y} \mid \mathbf{y} \in D\}$ is feasible, which is the same as D is non-empty.

Now, to see why the above variant of Farkas' lemma implies Proposition 3.3, we note that (23) holds for all polymatroids iff $\{(\boldsymbol{\lambda}_{\mathcal{B}} - \boldsymbol{\delta}_{\mathsf{DC}})^T\mathbf{h} > 0 \mid \mathbf{h} \in \Gamma_n\}$ is infeasible; now we are in the exact setting of the above variant of Farkas' lemma and the rest follows trivially. □

C.2 Details from Section 3.2

Definition C.2 (Tight witness)**.** A witness is said to be *tight* if $\mathsf{inflow}(Z) = \lambda_Z$ for all Z.

We remark that, if $\mathsf{inflow}(Z) > \lambda_Z$, we can always increase $\mu_{\emptyset,Z}$ by the amount $\mathsf{inflow}(Z) - \lambda_Z$ so that $\mathsf{inflow}(Z) = \lambda_Z$. In particular, it is easy to turn any witness into a tight witness. The proof of Lemma 3.9 below follows the same strategy as in the proof of Theorem 3.8, but with some subtle differences.

Proof of Lemma 3.9. W.L.O.G. we can assume that $(\boldsymbol{\sigma}, \boldsymbol{\mu})$ is a tight witness. We construct $(\boldsymbol{\lambda}', \boldsymbol{\delta}', \boldsymbol{\sigma}', \boldsymbol{\mu}')$ from $(\boldsymbol{\lambda}, \boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$ as follows. Initially we set $(\boldsymbol{\lambda}', \boldsymbol{\delta}', \boldsymbol{\sigma}', \boldsymbol{\mu}') = (\boldsymbol{\lambda}, \boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$. Let $\mathsf{inflow}'(Z)$ denote the quantity $\mathsf{inflow}(Z)$ measured on the vector $(\boldsymbol{\lambda}', \boldsymbol{\delta}', \boldsymbol{\sigma}', \boldsymbol{\mu}')$. Due to tightness of the witness, at this point $\mathsf{inflow}'(Z) - \lambda'_Z = 0$ for every Z.

Now, we start disturbing the flow balance equations starting from $Z = Y$ by setting $\delta'_{Z|\emptyset} = \delta'_{Z|\emptyset} - w$ which means $\mathsf{inflow}'(Z)$ was reduced by w. Note that Z is the only point for which $\mathsf{inflow}'(Z) - \lambda'_Z \neq 0$ (it is negative). If $\lambda'_Z > 0$, then we simply reduce λ'_Z by w and terminate. If $\lambda'_Z = 0$, then either (1) there is some $X \subset Z$ such that $\mu'_{X,Z} \geq w$, (2) there is some $Y \supset Z$ such that $\delta'_{Y|Z} \geq w$, or (3) there is some $J \perp Z$ such that $\sigma'_{Z,J} \geq w$. Cases (1) and (2) are handled in a similar way to the proof of Theorem 3.8 while case (3) is handled differently. In particular, if (1) holds, then we reduce $\mu'_{X,Z}$ by w and set $Z = X$. If (2) holds, then we reduce $\delta'_{Y|Z}$ by w and set $Z = Y$. If (3) holds, then we reduce $\sigma'_{Z,J}$ by w, *increase* $\mu'_{Z\cap J,J}$ by w, and set $Z = Z \cup J$. In all three cases, (the new) Z is the only point where $\mathsf{inflow}'(Z) - \lambda'_Z$ has a deficit, and the process continues if the new Z is not $\emptyset$.

The above process terminates because every time we move Z to a new deficit point, the quantity $2\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\mu}\|_1$ is reduced by w. When the process terminates, all quantities $\mathsf{inflow}'(Z) - \lambda'(Z) = 0$ and thus $\langle \boldsymbol{\lambda}', \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ is a Shannon flow inequality witnessed by $(\boldsymbol{\sigma}', \boldsymbol{\mu}')$ by Proposition 3.5. Property (b) holds because we only reduce the $\boldsymbol{\sigma}'$ and $\boldsymbol{\delta}'$ entries. Properties (c) and (d) hold trivially. □

D. MISSING DETAILS FROM SECTION 4

Proof of Theorem 1.5. Consider an input proof sequence of length $\ell \leq D(3\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\mu}\|_1)$, thanks to Theorem 3.8. If we do not hit Case 4b, then there will be at most ℓ steps in the algorithm, where each step either takes $\tilde{O}(2^{\mathsf{OBJ}})$ time or spawns $O(\mathsf{OBJ})$ subproblems, for a total of $\tilde{O}(\mathrm{poly}(\mathsf{OBJ}) \cdot 2^{\mathsf{OBJ}})$-time. A subproblem will terminate at producing a relation $T(\mathbf{A}_B)$ for some $B \in \mathcal{B}$ because the proof sequence will, by definition, reach a point where $\delta_{B|\emptyset} \geq w > 0$ for some $B \in \mathcal{B}$.

The worst case is obtained when the algorithm branches as far as possible only to have to restart at Case 4b with a slightly shorter proof sequence of length at most $D(3\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\mu}\|_1) - 1$. Thus, overall the exponent of OBJ (in $\mathrm{poly}(\mathsf{OBJ})$) will be $\leq \frac{1}{2}D^2(3\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\delta}\|_1 + \|\boldsymbol{\mu}\|_1)^2$. (Note that this constant is data-independent because the optimal dual solution $(\boldsymbol{\delta}, \boldsymbol{\sigma}, \boldsymbol{\mu})$ can be taken to be an extreme point of the dual polyhedron (24), whose constraints are only on the input query.) Since OBJ equals the polymatroid bound $\mathsf{LogSizeBound}_{\Gamma_n \cap \mathsf{HDC}}(P)$, which is bounded by the vertex bound $\log(N^n)$, and n is a constant in data complexity, we have $\mathsf{OBJ} = O(\log N)$, and the runtime is $\tilde{O}(\mathrm{poly}(\log N) \cdot 2^{\mathsf{LogSizeBound}_{\Gamma_n \cap \mathsf{HDC}}(P)})$, as desired.

The last bit we have to verify is that every subproblem has a proof sequence that can eventually reach $\delta_{B|\emptyset} \geq w$ for some $B \in \mathcal{B}$. Since we start with a proof sequence and in Cases 1, 2, 3, and 4a we continue with the remaining steps of the input proof sequence, the only case where we have to worry about proving the existence of a proof sequence reaching $\delta_{B|\emptyset} \geq w$ is Case 4b. In this case we restart with inequality $\langle \boldsymbol{\lambda}', \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}', \mathbf{h}\rangle$. Lemma 3.9 guarantees that $\lambda'_B > 0$ implies $\lambda_B > 0$. Hence, we will be done if we can show that $\|\boldsymbol{\lambda}'\|_1 > 0$. To this end, we show that the algorithm maintains another invariant, that for every subproblem it always holds that

$$\sum_{(X,Y)} n(\delta_{Y|X}) \leq \|\boldsymbol{\lambda}\|_1 \cdot \mathsf{OBJ}. \tag{62}$$

We call the quantity $\sum_{(X,Y)} n(\delta_{Y|X})$ the "potential" of the subproblem at that point in time. At the initial iteration, the potential of the input problem was exactly the budget. When Case 1 applies, the potential is unchanged because $\delta_{I|I\cap J}$ was reduced by w, $\delta_{I\cup J|J}$ was increased by w, and they have the same support. When Case 2 applies, the potential does not increase because we subtracted $w \cdot n_{Y|\emptyset}$ from the potential and added $w \cdot n_{X|\emptyset} \leq w \cdot n_{Y|\emptyset}$ instead. Similarly in Case 3, we subtracted $w \cdot n_{Y|\emptyset}$ from the potential, and added $w \cdot (n^{(j)}_{X|\emptyset} + n^{(j)}_{Y|X}) \leq w \cdot n_{Y|\emptyset}$ to the potential. In Case 4a, we subtracted $w \cdot (n_{X|\emptyset} + n_{W|Z})$ and added at most the same amount to the potential. In Case 4b, since $\boldsymbol{\delta}' \leq \boldsymbol{\delta}$ and $\delta'_{Y|\emptyset} = \delta_{Y|\emptyset} - w$, the potential is reduced by strictly more than $w \cdot \mathsf{OBJ}$ because we subtracted $w \cdot (n_{X|\emptyset} + n_{W|Z}) > w \cdot \mathsf{OBJ}$ from the potential. Since $\|\boldsymbol{\lambda}\|_1$ was reduced by at most w, invariant (62) still holds. This also proves that $\|\boldsymbol{\lambda}\|_1 > w$, because the residual potential, which is non-negative, is strictly smaller than $(\|\boldsymbol{\lambda}\|_1 - w)\mathsf{OBJ} \leq \|\boldsymbol{\lambda}'\|_1\mathsf{OBJ}$.

Finally, we get back to the assumption that $N \leq 2^{\mathsf{OBJ}}$ and how to enforce it in case it is not initially satisfied. Suppose there is an input relation $T(\mathbf{A}_Y)$ where $n_{Y|\emptyset} \stackrel{\text{def}}{=} \log_2|T| > \mathsf{OBJ}$. If $\delta_{Y|\emptyset} = 0$, then T does not contribute to the bound nor the proof sequence and can be ignored. If $\delta_{Y|\emptyset} > 0$, then we could have replaced the original inequality with a new one $\langle \boldsymbol{\lambda}', \mathbf{h}\rangle \leq \langle \boldsymbol{\delta}', \mathbf{h}\rangle$ satisfying the conditions stated in Lemma 3.9, in the same way we handled Case 4b above. However, since the new inequality has $\mathsf{OBJ}' < \mathsf{OBJ}$ and OBJ is already the optimal (i.e. lowest) bound possible, this scenario is impossible. □