# Tight Bounds of Circuits for Sum-Product Queries

AUSTEN Z. FAN, University of Wisconsin–Madison, USA
PARASCHOS KOUTRIS, University of Wisconsin–Madison, USA
HANGDONG ZHAO, University of Wisconsin–Madison, USA

In this paper, we ask the following question: given a Boolean Conjunctive Query (CQ), what is the smallest circuit that computes the provenance polynomial of the query over a given semiring? We answer this question by giving upper and lower bounds. Notably, it is shown that any circuit $F$ that computes a CQ over the tropical semiring must have size $\log |F| \geq (1-\epsilon) \cdot \mathsf{da\text{-}entw}$ for any $\epsilon > 0$, where da-entw is the degree-aware entropic width of the query. We show a circuit construction that matches this bound when the semiring is idempotent. The techniques we use combine several central notions in database theory: provenance polynomials, tree decompositions, and disjunctive Datalog programs. We extend our results to lower and upper bounds for formulas (i.e., circuits where each gate has outdegree one), and to bounds for non-Boolean CQs.

CCS Concepts: • **Theory of computation → Database theory**.

Additional Key Words and Phrases: conjunctive queries; circuit lower bound; disjunctive Datalog

**ACM Reference Format:**
Austen Z. Fan, Paraschos Koutris, and Hangdong Zhao. 2024. Tight Bounds of Circuits for Sum-Product Queries. *Proc. ACM Manag. Data* 2, 2 (PODS), Article 87 (May 2024), 20 pages. https://doi.org/10.1145/3651588

## 1 INTRODUCTION

In this paper, we study circuits that compute the *sum-product polynomial* for a Conjunctive Query (CQ). Formally, a sum-product polynomial is parameterized by a semiring $\mathbb{S}$, a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ associated with a full CQ $Q$, and an instance $I$:

$$p_I^{\mathcal{H}} := \bigoplus_{t \in Q(I)} \bigotimes_{e \in \mathcal{E}} x_{t[e]}^e \tag{1}$$

where $Q(I)$ is the output of running $Q$ over instance $I$ and $x_{t[e]}^e$ is a variable that captures the value of the input tuple $t[e]$ (which is the projection of $t$ on $e$) in the semiring domain $\mathbf{D}$. This work focuses on CQs as they capture a fundamental and widely used fragment of Relational Algebra.

A circuit $F$ over a semiring $\mathbb{S}$ is a Directed Acyclic Graph (DAG) with input nodes the variables $x_{t[e]}^e$ of fan-in 0 and every other node labelled by $\oplus$ or $\otimes$ of fan-in 2; these nodes are called $\oplus$-*gates* and $\otimes$-*gates* respectively [12]. A circuit $F$ that computes the sum-product polynomial has an output gate with fan-out 0 that, by sequentially going through all gates in a bottom-up fashion, evaluates the polynomial [1]. The size of the circuit $F$, denoted as $|F|$, is the number of gates in $F$. A circuit is a *formula* if the outdegree of each gate is one.

[1]Implicitly here the circuit computes a Boolean CQ whose head is empty. However, our framework can also capture CQs with projections.

Authors' addresses: Austen Z. Fan, University of Wisconsin–Madison, Madison, USA, afan@cs.wisc.edu; Paraschos Koutris, University of Wisconsin–Madison, Madison, USA, paris@cs.wisc.edu; Hangdong Zhao, University of Wisconsin–Madison, Madison, USA, hangdong@cs.wisc.edu.


ACM 2836-6573/2024/5-ART87
https://doi.org/10.1145/3651588

The use of circuits as a model of computation has been extensively studied in the realm of theoretical computer science [21, 26]. Particularly to the field of database theory, the study of circuits bears significant relevance for the following two reasons.

First, we can view circuits as a computational model that corresponds to algorithms that *solely* exploit the algebraic semiring structure. In other words, a circuit describes an algorithm that can interact with the semiring values of each tuple only by multiplying/adding two existing values, and storing the resulting output so that it can be reused. Such an algorithm cannot use branching, or compare two elements of the semiring. Moreover, the cost of algorithm is measured by the number of semiring operations. By limiting the computational model, it is now possible to obtain unconditional lower bounds. For example, a circuit over the Boolean semiring that computes the multiplication of two matrices will exclude the possibility of *fast matrix multiplication* [5] and therefore a $\Omega(n^3)$ lower bound in the circuit size is possible [27].

Second, circuits that compute the sum-product polynomial of a CQ can be viewed as a concise representation of the corresponding provenance polynomial interpreted over the given semiring. The provenance polynomial captures the *why*-provenance, or *lineage*, and represents the most detailed level of information for the occurrence of a tuple in the output [11]. In fact, there is a natural one-to-one correspondence between the monomials of the provenance polynomial and the tuples in the output. By tracing the *parse tree* of a monomial in the circuit, one may rediscover the lineage of the tuple associated to the monomial.

The central question we study in this paper is: *given a CQ, a semiring $\mathbb{S}$, and a set of constraints on an instance, what is the smallest semiring circuit we can construct?*

***Known Bounds.*** To unpack this question, consider an instance $I$ and a full CQ $Q$. One can check that it is always possible to construct a circuit with $O(|Q(I)|)$ gates by following exactly the structure of the polynomial (1): the circuit computes the sum of all monomials, where each monomial is a product of $|\mathcal{E}|$ input gates. Hence, if the constraints are cardinality constraints on the input relations, we obtain an upper bound that matches the AGM bound [3].

However, we can use the distributive property of a semiring to compress the size of the circuit. For instance, consider the query $Q(u, v, w) \leftarrow R(u, v) \wedge S(v, w)$ and suppose we are given the constraints $|R| \leq N, |S| \leq N$. The circuit construction above has an AGM bound of $N^2$. But the sum-product polynomial of $Q$ can be rewritten as:

$$\bigoplus_{(u,v)\in R,(v,w)\in S} x_{uv} \otimes y_{vw} = \bigoplus_{v} \left( \bigoplus_{(u,v)\in R} x_{uv} \right) \otimes \left( \bigoplus_{(v,w)\in S} y_{vw} \right)$$

Following this structure, we can now construct a semiring circuit with $O(N)$ size. Note that the saving in terms of size is a result of factorization in the circuit.

No general upper or lower bound is known for semiring circuits, but we do know bounds for restricted cases. If the query $Q$ is *hierarchical*, then it is known [23] that the provenance polynomial over the Boolean semiring can be expressed as a read-once formula, which in turn implies an optimal circuit of size $O(|I|)$ over the Boolean semiring. When the instance constraints consist only of a uniform active domain constraint (of size $n$) and $\mathcal{H}$ is a graph, Komarath et al. [20] showed a $\Theta(n^{\text{tw}+1})$ bound[2], where tw is the *treewidth* of the graph. The key idea of the lower bound is that a parse tree of a monomial in the circuit corresponds to a tree decomposition of the hypergraph. For the upper bound, the circuit is factorized using the tree decomposition that achieves the treewidth.

However, the above approach does not generalize when we consider hypergraphs and more complex constraints (such as cardinality constraints, functional dependencies, or degree constraints).

[2]The lower bound holds for the tropical semiring, while the upper bound holds for any semiring.

The reason is that the parse trees of different monomials in the circuit might correspond to different tree decompositions. Choosing among multiple tree decompositions means that we can obtain a better upper bound, but complicates the lower bound.

Related to semiring circuits are the concepts of $d$-representations and $f$-representations in factorized databases [24]. Indeed, a $d$-representation (resp. $f$-representation) is essentially a circuit (resp. formula) where *every* monomial corresponds to the *same* tree decomposition. This restriction makes it possible to obtain tight lower bounds in the case of uniform cardinality constraints: for example, if $N$ is the cardinality bound, a $d$-representation has size $O(N^{\mathsf{fhw}})$, where fhw is the fractional hypertree width of the hypergraph. However, semiring circuits have no structural restrictions and thus can take advantage of multiple tree decompositions to obtain more compact representations. The connection to factorized databases opens up many interesting problems of leveraging circuits for succinct representations of provenance polynomials, fast query evaluation, and in-database learning [13, 16, 17, 22, 28].

### 1.1 Our Contribution

***Lower Bounds.*** We prove that any circuit over the counting semiring, the tropical semiring or any multilinear circuit[3] over the Boolean semiring that computes a CQ with hypergraph $\mathcal{H}$ has size at least $2^{(1-\epsilon)\cdot \mathsf{da\text{-}entw}(\mathcal{H})}$ for any $\epsilon > 0$, where da-entw($\mathcal{H}$) is the *degree-aware entropic width* for $\mathcal{H}$ (Theorem 3.7, Corollary 3.11). The degree-aware entropic width is shown to be the *tightest* parameter for bounding the output size of a disjunctive Datalog rule [19]. To the best of our knowledge, this result is the first that gives an *unconditional* lower bound for *every* CQ in some model of computation.

***Upper Bounds.*** We construct a multilinear and homogeneous circuit $F$ of size $O(2^{\mathsf{da\text{-}entw}(\mathcal{H})})$ that produces the provenance polynomial for $\mathcal{H}$ over any idempotent semiring (Theorem 4.1). We emphasize that, combined with our lower bound results, we completely solve the problem of the smallest size of a multilinear circuit computing the provenance polynomial over the Boolean semiring (up to asymptotic tightness [19]). In fact, our result becomes stronger over the tropical semiring, i.e. the lower bound and the upper bound of the circuit size coincide, while we do not restrict the circuit to be multilinear.

The circuit construction may be seen as the *existence of an optimal algorithm*, or the *existence of an optimal sequence of operations*, in terms of the number of semiring $\oplus$ and $\otimes$ operations involved. Our circuit upper bound is *constructive*: although it remains open whether the problem of computing da-entw($\mathcal{H}$) is decidable [15], our method can construct a circuit of the desired size without computing da-entw($\mathcal{H}$). Although the time to construct such a circuit may exceed the circuit size itself, such a circuit can be *reused* to compute instances with different *annotations* [11].

***Extension to Formulas.*** We extend the lower and upper bounds to formulas, which are circuits where the outdegree of every gate is exactly one, by replacing da-entw($\mathcal{H}$) with the weaker width measure of ida-entw($\mathcal{H}$), i.e., *inflationary degree-aware entropic width* (Theorem 5.2).

***Extension to non-Boolean CQs.*** Finally, we show upper and lower bounds for circuits that simultaneously compute the polynomials of multiple output tuples of a non-Boolean CQ (Theorem 6.1).

[3]That is, a circuit whose $\otimes$-gates always multiplies a disjoint set of variables.

Table 1. Summary of results.

| Semiring | Upper Bound | Lower Bound |
|---|---|---|
| Boolean | da-entw [Thm. 4.1] | ? |
| Tropical | da-entw [Thm. 4.1] | da-entw [Thm. 3.7] |
| **Semiring Class** | **Upper Bound** | **Lower Bound** |
| Idempotent | da-entw [Thm. 4.1] | da-entw [Thm. 3.7] |
| All | da-efhtw [Thm. 4.3] | da-entw [Thm. 3.7] |

## 2 PRELIMINARIES

***Conjunctive Queries.*** A *Conjunctive Query* (CQ) $Q$ is an expression associated to a hypergraph $\mathcal{H} = ([n], \mathcal{E})$ where $[n] = \{1, \ldots, n\}$ and some $U \subseteq [n]$:

$$Q(\mathbf{x}_U) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e)$$

where each $R_e$ is a relation of arity $|e|$, the variables $x_1, x_2, \ldots, x_n$ take values in some discrete domain, and $\mathbf{x}_e := (x_i)_{i \in e}$. We say that $Q$ is *Boolean* if $U = \emptyset$ and *full* if $U = [n]$.

***Semirings.*** A (commutative) *semiring* is an algebraic structure $\mathbb{S} := (\mathbf{D}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\oplus$ and $\otimes$ are the *addition* and *multiplication* in $\mathbb{S}$ such that: (1) $(\mathbf{D}, \oplus, \mathbf{0})$ and $(\mathbf{D}, \otimes, \mathbf{1})$ are commutative monoids, (2) $\otimes$ is distributive over $\oplus$, (3) $\mathbf{0}$ is an annihilator of $\otimes$ in $\mathbf{D}$. We say that $\mathbb{S}$ is *idempotent* if $x \oplus x = x$ for every $x \in \mathbf{D}$.

Green, Karvounarakis and Tannen [11] made the beautiful observation that the semantics of CQ are in natural correspondence to the underlying semirings. For example, set semantics and bag semantics naturally correspond to computing a query over the *Boolean semiring* $\mathbb{B} := (\{0, 1\}, \vee, \wedge, \text{FALSE}, \text{TRUE})$ and the *Counting semiring* $\mathbb{C} := (\mathbb{N}, +, \cdot, 0, 1)$. The optimization setting often relates to the *Tropical semiring* $\mathbb{T} := (\mathbb{Z}, \min, +, +\infty, 0)$.

***Sum-Product Polynomials.*** Following the notation in [11], the *sum-product polynomial* for a full CQ $Q$ is parameterized by an underlying semiring $\mathbb{S}$, a hypergraph $\mathcal{H}$, and an instance $I$:

$$p_I^{\mathcal{H}} := \bigoplus_{t \in Q(I)} \bigotimes_{e \in \mathcal{E}} x^e_{t[e]} \tag{2}$$

where $x^e_{t[e]}$ is a variable that captures the value of the tuple $t[e] \in R_e$ in the semiring domain $\mathbf{D}$. Here, $t[e]$ denotes the projection of $t$ on $e$. We omit the dependency of $\mathbb{S}$ in the notation whenever it will be clear from the context.

When we work over the counting semiring, the sum-product polynomial becomes a polynomial:

$$p_I^{\mathcal{H}} := \sum_{t \in Q(I)} \prod_{e \in \mathcal{E}} x^e_{t[e]}$$

Such polynomials are *multilinear* (the exponent of a variable in a monomial is 1) and *homogeneous* (the sum of the exponents in every monomial is the same). Multilinearity holds because we only deal with *self-join-free* queries, where a relational name appears only once in the query body. Over the Boolean semiring, the sum-product polynomial captures the Boolean provenance of the query.

***Tree Decompositions.*** We recall the notion of *tree decomposition* in Robertson and Seymour's theory [25].

*Definition 2.1 (Tree Decomposition).* A *tree decomposition* of a hypergraph $\mathcal{H}$ is a pair $(\mathcal{T}, \chi)$, where $\mathcal{T}$ is a tree and $\chi$ maps each node $t$ of the tree to a subset $\chi(t)$ of $V(\mathcal{H})$, called a bag, such that:

(1) every hyperedge $e \in E(\mathcal{H})$ is a subset of $\chi(t)$ for some $t \in V(\mathcal{T})$; and
(2) for every vertex $v \in V(\mathcal{H})$, the set $\{t | v \in \chi(t)\}$ is a non-empty connected subtree of $\mathcal{T}$.

The second requirement is also called the *running intersection property*. We say that a tree decomposition is *non-redundant* if no bag is a subset of another; otherwise it is *redundant*. We let $\mathrm{TD}(\mathcal{H})$ be the set of all non-redundant tree decompositions of $\mathcal{H}$.

***Circuits over Semirings.*** A *circuit* $F$ over a semiring $\mathbb{S}$ is a Directed Acyclic Graph (DAG) with input nodes (with fan-in 0) variables in a set $S_x$ containing $x^e_{t[e]}$'s in RHS of Equation 2 and the constants $\mathbf{0}, \mathbf{1}$. Every other node is labelled by $\oplus$ or $\otimes$ and has fan-in 2; these nodes are called $\oplus$-*gates* and $\otimes$-*gates*, respectively. An input gate of $F$ is any gate with fan-in 0 and an output gate of $F$ is any gate with fan-out 0. The size of the circuit $F$, denoted as $|F|$, is the number of gates in $F$. The depth of $F$ is the length of the longest path from an input to an output node.

A circuit $F$ can be viewed as a concise representation of a polynomial $\hat{F}$ over the variable set $S_x$. A circuit $F$ is said to *compute* a polynomial $p$ if $\hat{F}$ and $p$ coincide as functions (interpreted over the semiring $\mathbb{S}$), and is said to *produce* a polynomial $p$ if $\hat{F}$ and $p$ have exactly the same terms, i.e. monomials with their coefficients, syntactically [4].

A circuit $F$ is *homogeneous* if every polynomial produced in each gate in $F$ is homogeneous, and is *multilinear* if for every $\otimes$-gate $u = v \otimes w$, the polynomials produced at gates $v$ and $w$ have a disjoint set of variables. Clearly, if $F$ produces $p$, then $F$ must compute $p$, but not necessarily the other way around. However, in some cases the converse is also true.

Lemma 2.2 ([12]). *A circuit $F$ that computes a multilinear polynomial $p$ over $\mathbb{C}$ or $\mathbb{T}$ must be multilinear; moreover, $F$ produces $p$.*

Note that a circuit $F$ is homogeneous if and only if the polynomial $p$ produced by it is homogeneous. Let $\mathbb{S}(p)$ and $\mathbb{S}[p]$ be the minimum size of a circuit over semiring $\mathbb{S}$ *computing* and *producing* $p$, respectively. Let $\mathbb{S}_{\mathsf{lin}}(p)$ be the minimum size of a *multilinear* circuit over semiring $\mathbb{S}$ computing $p$. Clearly, we have $\mathbb{S}[p] \geq \mathbb{S}(p)$ and $\mathbb{S}_{\mathsf{lin}}(p) \geq \mathbb{S}(p)$. The following lemma allows one to transfer the circuit size lower bound from one semiring to another.

Lemma 2.3 ([12]). *If $p$ a is multilinear and homogeneous polynomial, then $\mathbb{B}_{\mathsf{lin}}(p) = \mathbb{T}(p) = \mathbb{T}[p] = \mathbb{C}(p) = \mathbb{C}[p]$.*

***Entropic Functions.*** Let $n$ be a positive integer. A function $h : 2^{[n]} \to \mathbb{R}_+$ is called a *set function* on $[n] = \{1, 2, \ldots, n\}$. A set function is an *entropic function* of order $n$ if there exist random variables $A_1, \ldots, A_n$ such that $h(S) = H((A_i)_{i \in S})$ for any $S \subseteq [n]$, where $H$ is the joint entropy of a set of variables. We denote by $\Gamma^*_n$ the set of all entropic functions of order $n$, and $\overline{\Gamma}^*_n$ the topological closure of $\Gamma^*_n$.

In the following sections, all missing proofs can be found in the appendix.

## 3 LOWER BOUNDS

In this section, we prove a general lower bound on the size of semiring circuits. The key ingredient in this bound is relating the size of a semiring circuit to the output size of a disjunctive Datalog rule. In particular, we construct a set of disjunctive Datalog rules such that an output bound on any rule implies a size bound on the circuit. The connection to disjunctive Datalog comes from the observation that the parse tree of any monomial in the circuit corresponds to a tree decomposition. Different monomials may correspond to different tree decompositions; however, we can always guarantee that we can map such a decomposition to one of the targets in the head of the disjunctive

[4]The polynomials $p^{\mathcal{H}}_I$ we consider in this paper have the property that all monomials have coefficient $\mathbf{1} \in \mathbb{S}$.

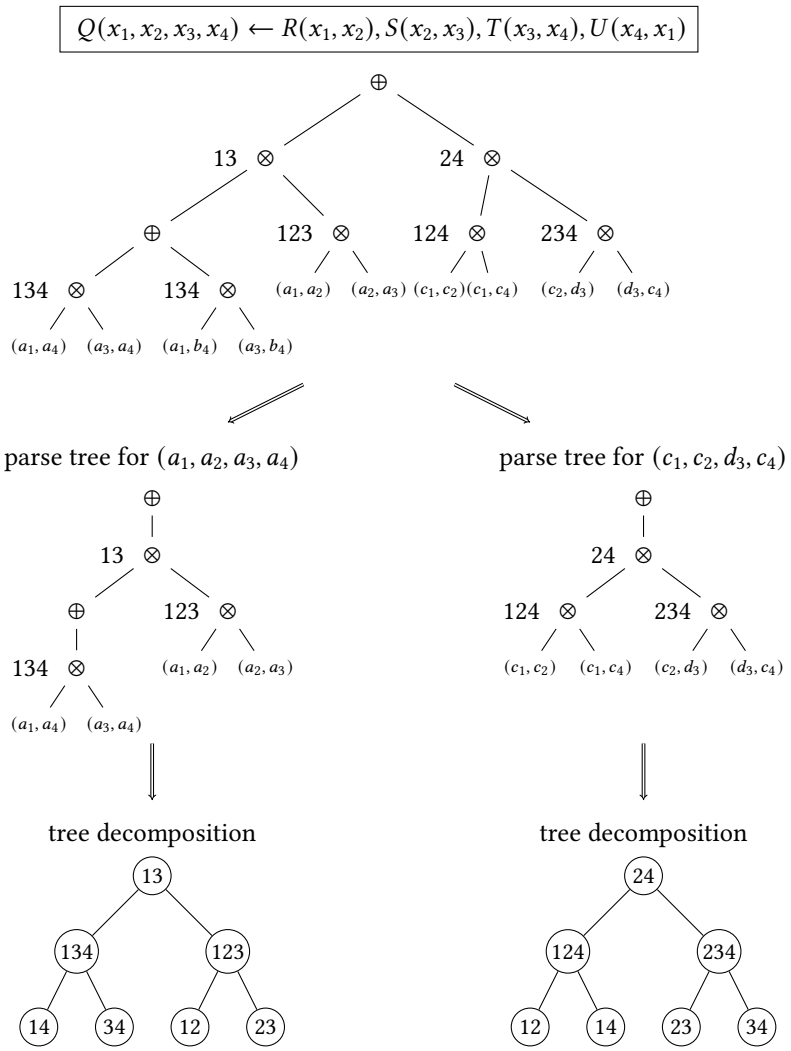


Fig. 1. A circuit for the 4-cycle query, where the numbers to the left of a ⊗-gate are the assigned type.

Datalog rule. This mapping in turn implies a mapping from the ⊗-gates of the circuit to the targets of the rules. By using existing constructions for worst-case instances for disjunctive Datalog, we obtain the desired bound on the circuit size.

### 3.1 Properties of Circuits

For the purposes of this subsection, we fix a circuit $F$ that computes $p_I^{\mathcal{H}}$ over the counting semiring $\mathbb{C}$. We will use the circuit of Figure 1 as our running example. By Lemma 2.2, $F$ produces $p_I^{\mathcal{H}}$ and thus there is a one-to-one correspondence between the monomials in $F$ and the tuples in $Q(I)$; therefore, we will use the terms monomial and output tuple of $Q$ interchangeably.

***Parse Trees.*** Informally, a parse tree captures how a monomial is produced in the circuit.

*Definition 3.1 (Parse tree).* A *parse tree* pt is a rooted tree in a circuit $F$ defined inductively as follows:

(1) The root of pt is an output gate.
(2) If a $\otimes$-gate is in pt, include all of its children in $F$ as its children in pt.
(3) If a $\oplus$-gate is in pt, include exactly one of its children in $F$ as its children in pt.

The subtree of a parse tree pt that includes a $\otimes$-gate $g$ and all of its descendants is denoted as $\mathrm{pt}(g)$. The tree obtained by removing $\mathrm{pt}(g)$ from pt is called the *tree outside* $\mathrm{pt}(g)$.

By a *parse tree of a monomial*, we mean a parse tree that tracks how the monomial is computed in the circuit. By Lemma 2.2, for a circuit that computes a multilinear polynomial over $\mathbb{C}$, in particular $p_I^{\mathcal{H}}$, there exists exactly one parse tree for any monomial. Figure 1 shows two different parse trees for two output tuples, $(a_1, a_2, a_3, a_4)$ and $(c_1, c_2, d_3, c_4)$.

***Parse Trees as Tree Decompositions.*** We first generalize an observation made by Komarath et al. [20] that a tree decomposition for $\mathcal{H}$ can be constructed from the parse tree of a monomial. Our construction works for any hypergraph and any instance $I$, while the construction in [20] is only for graphs and instances that are cartesian products over a fixed domain. Furthermore, our construction has the nice property that there exists a one-to-one correspondence between the input gates and $\otimes$-gates in the parse tree and the nodes in the tree decomposition.

For a monomial $q$ in $p_I^{\mathcal{H}}$, we define a structure $\mathcal{T}_q = (\mathcal{T}, \chi)$ inductively in a bottom-up fashion from its parse tree:

(1) For an input gate $g$ of the variable $x^e_{t[e]}$, add a node $v_g$ in $\mathcal{T}$ with $\chi(v_g) = e$. The input gate $g$ is said to be *associated to* the node $v_g$.
(2) For a $\oplus$-gate $g$, associate $g$ to the node that is associated to $g$'s single child in the parse tree.
(3) For a $\otimes$-gate $g$, let $g_1$ and $g_2$ be its children. Let $q_1, q_2$ be the monomials computed at $g_1, g_2$ respectively; and $B_g$ be the set of vertices $v \in \mathcal{V}(\mathcal{H})$ such that all hyperedges incident to $v$ appear either exclusively in $q_1$ *or* exclusively in $q_2$. We add a node $v_g$ with $\chi(v_g) = (\chi(v_{g_1}) \cup \chi(v_{g_2})) \setminus B_g$ as the parent of $v_{g_1}$ and $v_{g_2}$ in $\mathcal{T}$. We associate $g$ with the new node $v_g$.

Note that the nodes in $\mathcal{T}$ are in one-to-one correspondence to the input gates and $\otimes$-gates in the parse tree $\mathrm{pt}_q$. Moreover:

Lemma 3.2. *For any monomial $q$ in $p_I^{\mathcal{H}}$, the structure $\mathcal{T}_q = (\mathcal{T}, \chi)$ is a tree decomposition of $\mathcal{H}$. Moreover:*

*(1) If $v \in V(\mathcal{T})$ is a leaf then $\chi(v) = e$ for some hyperedge $e \in \mathcal{E}(\mathcal{H})$;*
*(2) For every hyperedge $e \in \mathcal{E}(\mathcal{H})$ there exists a unique leaf $v \in V(\mathcal{T})$ such that $\chi(v) = e$.*
*(3) Every internal node in $\mathcal{T}$ has exactly two children.*

Proof. The constructed graph $\mathcal{T}$ is clearly a tree since $\mathrm{pt}_q$ is a tree. Every hyperedge $e \in \mathcal{E}(\mathcal{H})$ appears in some bag of $\mathcal{T}$ (in particular a leaf of $\mathcal{T}$), since the monomial $q$ must include a variable of the form $x^e_{t[e]}$. Finally, note that a vertex $v$ from a hyperedge stops being included in the bags only after all of its hyperedges have been considered in the subtree; hence, the running intersection property is satisfied. The three properties of the tree decomposition are straightforward. □

Figure 1 shows the two tree decompositions resulting from the two parse trees of the circuit. Let $\mathsf{TD}_{\mathsf{C}}(\mathcal{H})$ be the set of all tree decompositions that can be constructed from a monomial of a circuit that computes $p_I^{\mathcal{H}}$ for some instance $I$ as above. We will next characterize the structure of $\mathsf{TD}_{\mathsf{C}}(\mathcal{H})$. The set may contain redundant tree decompositions, but we claim that its size depends only on $\mathcal{H}$.

Lemma 3.3. $|\mathsf{TD}_{\mathsf{C}}(\mathcal{H})| \leq 2^{|\mathcal{V}|^{(2|\mathcal{E}|-1)}}$.

Proof. From Lemma 3.2, the total number of nodes for any decomposition in $\mathrm{TD}_{\mathrm{C}}(\mathcal{H})$ is at most $(2|\mathcal{E}| - 1)$. Indeed, such a tree decomposition has exactly $|\mathcal{E}|$ leaves, and $|\mathcal{E} - 1|$ internal nodes, since every internal node has exactly two children. Since any bag can take at most $2^{|\mathcal{V}|}$ possible values, we obtain the desired lower bound. □

***Properties of Tree Decompositions.*** We observe that if two tree decompositions share a ⊗-gate $g$, then the corresponding bags are identical in the trees.

Lemma 3.4. *Let $q_1, q_2$ be two monomials in $p_I^{\mathcal{H}}$ and $\mathcal{T}_{q_1} = (\mathcal{T}_1, \chi_1), \mathcal{T}_{q_2} = (\mathcal{T}_2, \chi_2)$ be their corresponding tree decompositions. If the parse trees of $q_1, q_2$ share a common ⊗-gate $g$, then $\chi_1(v_g) = \chi_2(v_g)$.*

Proof. Suppose for the sake of contradiction that $\chi_1(v_g) \neq \chi_2(v_g)$. Without loss of generality, assume $v \in \chi_1(v_g)$ and $v \notin \chi_2(v_g)$. Let $E_v = \{e \in \mathcal{E} : v \in e\}$. Let $E_v^{\ell,2} \subseteq E_v$ (resp. $E_v^{r,2}$) be the set of hyperedges such that some input gate $x_*^e$, $e \in E_v$, occurs in the left (resp. right) subtree of $\mathrm{pt}_{q_2}(g)$. Since $v \notin \chi_2(v_g)$, one of the sets $E_v^{\ell,2}, E_v^{r,2}$ must be empty and the other set must be equal to $E_v$. Indeed, if both $E_v^{\ell,2}$ and $E_v^{r,2}$ are non-empty, then $v \in \chi_2(v_g)$ by the running intersection property. Therefore if $v \notin \chi_2(v_g)$ and if, say, $E_v^{\ell,2}$ is non-empty, then $E_v = E_v^{\ell,2}$.

Similarly we define $E_v^{\ell,1}$ and $E_v^{r,1}$ for the other parse tree. Since $v \in \chi_1(v_g)$, for one of the two sets (w.l.o.g. the left) we must have $\emptyset \subsetneq E_v^{\ell,1} \subsetneq E_v$. Indeed, it cannot be the case that $E_v^{\ell,1} = E_v^{r,1} = \emptyset$, since that implies $v \notin \chi_1(v_g)$. W.l.o.g. assume $\emptyset \subsetneq E_v^{\ell,1}$, then $E_v^{\ell,1} \subsetneq E_v$, since otherwise $v$ should have been forgotten at $g$, i.e. $v \notin \chi_1(v_g)$. Thus, we have $E_v^{\ell,1} \subsetneq E_v^{\ell,2}$ or $E_v^{\ell,2} \subsetneq E_v^{\ell,1}$ .

W.l.o.g. assume $E_v^{\ell,1} \subsetneq E_v^{\ell,2}$. Now, we take the parse tree $\mathrm{pt}_{q_2}$ and replace the left subtree of $\mathrm{pt}_{q_2}(g)$ with the left subtree of $\mathrm{pt}_{q_1}(g)$. This obtains a new parse tree in $F$, by Definition 3.1, where some $e \in E_v$ does not have any input gate of the form $x_*^e$. The monomial of that parse tree does not correspond to any monomial of $p_I^{\mathcal{H}}$, a contradiction. □

In light of Lemma 3.4, it is possible to assign a *type* $\mathrm{tp}(g)$ to each ⊗-gate $g$ as $\chi(v_g)$ for some decomposition $\mathcal{T}_q = (\mathcal{T}, \chi)$ of a monomial $q$. In other words, the circuit $F$ yields a *globally consistent type assignment* to each ⊗-gate in $F$. Figure 1 shows the associated types to each gate, where the input instance consists of the leafs and $\{1, 3, 4\}$, say, is a shorthand for $\{x_1, x_3, x_4\}$. Moreover, we have a similar statement as in [20]:

Lemma 3.5. *Let $t_1, t_2 \in Q(I)$ such that their parse trees in $F$ share a common gate $g$. Then, $t_1[\mathrm{tp}(g)] = t_2[\mathrm{tp}(g)]$.*

### 3.2 Disjunctive Datalog Rules

In this section, we explore a fundamental connection between disjunctive Datalog rules and circuits. A *disjunctive Datalog rule* [19] is an expression

$$P : \bigvee_{B \in \mathcal{B}} T_B(\mathbf{x}_B) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e)$$

where $B \subseteq [n]$ and $\mathcal{B} \subseteq 2^{[n]}$, where $2^{[n]}$ is the power set of $[n]$. The body is similar to that of a CQ, while the head is a disjunction of output relations $T_B$ which are called *targets*. Given an instance $I$, a *model* of a disjunctive Datalog rule is a tuple $\mathbf{T} = (T_B)_{B \in \mathcal{B}}$ of relations, denoted as $\mathbf{T} \models P$, such that for any tuple $t$, if $t[e] \in R_e$ for all $e \in \mathcal{E}$, then there exists a target $T_B \in \mathbf{T}$ such that $t[B] \in T_B$. The size of a model is defined to be $\max_B |T_B|$ and the output size of a disjunctive Datalog rule over instance $I$ is the minimum size over all models, i.e., $|P(I)| := \min_{\mathbf{T}:\mathbf{T} \models P} \max_{B \in \mathcal{B}} |\mathbf{T}_B|$.

One of the main insights in this paper is that one can transfer lower bounds of output size for a disjunctive Datalog rule associated with a CQ into lower bounds on the number of ⊗-gates in

its corresponding circuit. More formally, we need to introduce the following definitions. Let $\mathcal{F}$ be a set of tree decompositions of $\mathcal{H}$. Let $\mathcal{M}_{\mathcal{F}}$ be the set of all maps $\beta : \mathcal{F} \to 2^{\mathcal{V}}$, where $2^{\mathcal{V}}$ is the power set of $\mathcal{V}$, such that $\beta(\mathcal{T}, \chi) = \chi(t)$ for some $t \in V(\mathcal{T})$. Such $\beta$ is called a *bag selector* that chooses a bag from each tree decomposition. Let $\mathbf{B}_{\mathcal{F}}$ be the collection of images of all $\beta \in \mathcal{M}_{\mathcal{F}}$, i.e. $\mathbf{B}_{\mathcal{F}} = \{\mathcal{B} \mid \mathcal{B} = \mathsf{image}(\beta) \text{ for some } \beta \in \mathcal{M}_{\mathcal{F}}\}$.

The following is our key lemma for circuit lower bounds.

Lemma 3.6. *Let $\beta \in \mathcal{M}_{\mathsf{TD_C}}$ be any bag selector for the set of tree decompositions $\mathsf{TD_C}$ and $\mathcal{B} = \mathsf{image}(\beta)$. Consider the disjunctive Datalog rule*

$$P : \bigvee_{B \in \mathcal{B}} T_B(\mathbf{x}_B) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e). \tag{3}$$

*Let $F$ be a circuit that computes $p_I^{\mathcal{H}}$ over the counting semiring $\mathbb{C}$. Then, $|F| \geq |P(I)|$.*

Proof. Fix the globally consistent type assignment tp to each $\otimes$-gate in $F$ given by Lemma 3.4. Let $S_B$ be the set of $\otimes$-gates in $F$ that has type $B \subseteq [n]$. By Lemma 3.5, each gate $g \in S_B$ can be uniquely associated with a value $w(g) := t[\mathsf{tp}(g)]$ for any tuple $t$ in the output of the RHS of $P$ such that $g$ is in $t$'s parse tree. Let $T_B := \{w(g) \mid g \in S_B\}$; we claim that $\{T_B\}_{B \in \mathcal{B}}$ is a model for $P$. Indeed, take any output tuple $t$; its corresponding monomial $q$ must be computed by $F$. Take the parse tree $\mathsf{pt}_q$, construct its tree decomposition $\mathcal{T}_q$, and consider the gate $g$ corresponding to the bag $B = \beta(\mathcal{T}_q)$ (such a bag exists because $\mathcal{T}_q \in \mathsf{TD_C}$). By Lemma 3.5, $t[B] = w(g) \in T_B$.

Take $B \in \mathcal{B}$ that maximizes $|S_B|$. Therefore, we have $\log |S_B| \geq \log |P(I)|$. Since $|F| \geq |S_B|$, this concludes the proof. □

### 3.3 The Lower Bound

Let DC be a set of triples $(X, Y, N_{Y|X})$ for some $X \subset Y \subseteq [n]$ and $N_{Y|X} \in \mathbb{N}$ that encodes a set of *degree constraints*. An instance $I$ satisfies the constraints if for every relation $R_e$ in $I$ with $X \subseteq Y \subseteq e$ and every tuple $t_X$ we have $|\pi_Y(R_e \ltimes t_X)| \leq N_{Y|X}$. A constraint of the form $(\emptyset, e, N_e)$ is simply a cardinality constraint. The degree constraints on an instance can be translated as constraints on entropic functions as follows:

$$\mathsf{HDC} := \left\{ h : 2^{[n]} \to \mathbb{R}_+ \mid \bigwedge_{(X,Y,N_{Y|X}) \in \mathsf{DC}} h(Y|X) \leq \log N_{Y|X} \right\}$$

where $h(Y|X) := h(Y) - h(X)$. For a given set of degree constraints HDC, define the "scaled-up" degree constraints $\mathsf{HDC} \times k$ as the same set of constraints but with all the degree bounds multiplied by $k$. We also define

$$\mathsf{ED} := \{h : 2^{[n]} \to \mathbb{R}_+ \mid h(e) \leq 1, \forall e \in \mathcal{E}\}$$
$$\mathsf{VD} := \{h : 2^{[n]} \to \mathbb{R}_+ \mid h(\{v\}) \leq 1, \forall v \in [n]\}$$

We now define the *entropic width* and *degree-aware entropic width* as follows, respectively:

$$\mathsf{entw}(\mathcal{H}, \mathsf{ED}) := \max_{h \in \overline{\Gamma}_n^* \cap \mathsf{ED}} \min_{(\mathcal{T}, \chi) \in \mathsf{TD}} \max_{v \in V(\mathcal{T})} h(\chi(v))$$

$$\text{da-entw}(\mathcal{H}, \mathsf{HDC}) := \max_{h \in \overline{\Gamma}_n^* \cap \mathsf{HDC}} \min_{(\mathcal{T}, \chi) \in \mathsf{TD}} \max_{v \in V(\mathcal{T})} h(\chi(v))$$

In fact, these two quantities are defined in [19]: da-entw$(\mathcal{H}, \mathsf{HDC})$ is called "entropic degree-aware submodular width" to emphasize the connection to their notion of "degree-aware submodular width." Since in our work we do not involve submodular functions, and moreover this concept can

be defined from entropic functions *per se*, we simplify the names instead of attempting to change the names in any way. We sometimes will abbreviate da-entw($\mathcal{H}$, HDC) (*resp.* entw($\mathcal{H}$, ED)) as da-entw($\mathcal{H}$) (*resp.* entw($\mathcal{H}$)) when the set of degree constraints is clear from the context.

We are now ready to state our main theorem.

Theorem 3.7. *For any $\epsilon > 0$ and any hypergraph $\mathcal{H}$, there exists an instance $I$ and $k > 0$ that satisfies the constraints* HDC $\times k$ *such that any circuit $F$ that computes the polynomial $p_I^{\mathcal{H}}$ over the counting semiring $\mathbb{C}$ has size*

$$\log |F| \geq (1-\epsilon) \cdot \textit{da-entw}(\mathcal{H}, \mathsf{HDC} \times k).$$

To prove the theorem, we will need to recall the results of Khamis, Ngo and Suciu [19], who showed that the entropic bound is asymptotically tight under degree constraints. Their construction transforms the *group characterizable entropy function* from Chan and Yeung [8] into a database instance inspired by Gogacz and Toruńczyk [10].

Lemma 3.8 ([19]). *Let* HDC *be the degree constraints for any disjunctive Datalog rule $P$ where $\max_{h \in \overline{\Gamma}_n^* \cap \mathsf{HDC}} \min_{B \in \mathcal{B}} h(B) \geq 1$. For any $\epsilon > 0$, there exists a scale factor $k$ and a database instance $I$ satisfying* HDC $\times k$ *for which the following holds:*

$$\log |P(I)| \geq (1-\epsilon) \cdot \max_{h \in \overline{\Gamma}_n^* \cap \mathsf{HDC} \times k} \min_{B \in \mathcal{B}} h(B).$$

We will also need two technical lemmas. The first lemma is a slight generalization of a result in [19] (see the proof of Corollary 7.13).

Lemma 3.9. *For a family of set functions $H$ and a set of tree decompositions $\mathcal{F}$ of hypergraph $\mathcal{H}$,*

$$\max_{h \in H} \min_{(\mathcal{T},\chi) \in \mathcal{F}} \max_{v \in V(\mathcal{T})} h(\chi(v)) = \max_{\mathcal{B} \in \mathbf{B}_{\mathcal{F}}} \max_{h \in H} \min_{B \in \mathcal{B}} h(B).$$

For the second lemma, recall that $\mathsf{TD}(\mathcal{H})$ is the set of all non-redundant tree decompositions of the hypergraph $\mathcal{H}$. A set function $h$ is *monotone* if $h(X) \leq h(Y)$ whenever $X \subseteq Y$.

Lemma 3.10. *For every monotone set function $h$,*

$$\min_{(\mathcal{T},\chi) \in \mathsf{TD}(\mathcal{H})} \max_{v \in V(\mathcal{T})} h(\chi(v)) \leq \min_{(\mathcal{T},\chi) \in \mathsf{TD}_{\mathsf{C}}(\mathcal{H})} \max_{v \in V(\mathcal{T})} h(\chi(v))$$

We now have all the ingredients to prove the main theorem.

Proof of Theorem 3.7. For any scale factor $k > 0$:

$$\begin{aligned} \textsf{da-entw}(\mathcal{H}, \mathsf{HDC} \times k) &\leq \max_{h \in \overline{\Gamma}_n^* \cap \mathsf{HDC} \times k} \min_{(\mathcal{T},\chi) \in \mathsf{TD}_{\mathsf{C}}} \max_{v \in V(\mathcal{T})} h(\chi(v)) \\ &= \max_{\mathcal{B} \in \mathbf{B}_{\mathsf{TD}_{\mathsf{C}}}} \max_{h \in \overline{\Gamma}_n^* \cap \mathsf{HDC} \times k} \min_{B \in \mathcal{B}} h(B) \end{aligned} \tag{4}$$

The first inequality is implied by Lemma 3.10 and the monotonicity of all entropic functions. The second equality is implied by Lemma 3.9. For a bag selector $\beta$ and $\mathcal{B} = \mathsf{image}(\beta)$, consider

$$P_\beta : \bigvee_{B \in \mathcal{B}} T_B(\mathbf{x}_B) \leftarrow \bigwedge_{e \in \mathcal{E}} R_e(\mathbf{x}_e).$$

By Lemma 3.8, there exists an integer $k_\beta > 0$ and an instance $I_\beta$ such that

$$\log |P(I_\beta)| \geq (1-\epsilon) \cdot \max_{h \in \overline{\Gamma}_n^* \cap \mathsf{HDC} \times k_\beta} \min_{B \in \mathcal{B}} h(B).$$

Let $k_0 = \max_\beta k_\beta$, and consider the bag selector $\beta_0$ that maximizes the RHS of Equation 4 for $k = k_0$, and let $I_0 = I_{\beta_0}$. Then, $|P(I_0)| \geq (1-\epsilon) \cdot \text{da-entw}(\mathcal{H}, \mathsf{HDC} \times k_0)$. Finally, let $F$ be a circuit that computes $p^{\mathcal{H}}_{I_0}$ over the counting semiring $\mathbb{C}$. By Lemma 3.6 we have $|F| \geq |P(I_0))|$. □

Combined with Lemma 2.3, we immediately obtain the following.

Corollary 3.11. *For any $\epsilon > 0$, there exists an instance $I$ and integer $k > 0$ that satisfies the constraints $k \times \mathsf{HDC}$ such that any circuit $F$ that computes $p^{\mathcal{H}}_I$ over a semiring in $\{\mathbb{T}, \mathbb{B}_{\mathsf{lin}}\}$ has size $\log |F| \geq (1-\epsilon) \cdot$ da-entw$(\mathcal{H}, \mathsf{HDC} \times k)$.*

As another illustration of the power of Theorem 3.7, consider the case where the degree constraints HDC are ED (which corresponds to a uniform cardinality constraint on each relation) or VD (which corresponds to a uniform constraint on the active domain size). Given a set of constraints $C$, write $C \vdash I$ if an instance $I$ satisfies the constraints. Now, let us define:

$$\mathbb{S}(\mathcal{H}, C) := \max_{C \vdash I} \{\mathbb{S}(p^{\mathcal{H}}_I)\}$$

Then combined with Proposition 7.3 in [19], the following lower bounds follow immediately:

Theorem 3.12. *For $\mathbb{S} \in \{\mathbb{C}, \mathbb{T}, \mathbb{B}_{\mathsf{lin}}\}$, we have:*

$$\mathbb{S}(\mathcal{H}, \mathsf{ED} \times \log N) = \Omega(N^{entw(\mathcal{H})})$$
$$\mathbb{S}(\mathcal{H}, \mathsf{VD} \times \log n) = \Omega(n^{tw(\mathcal{H})+1})$$

*where $tw(\mathcal{H})$ is the treewidth, defined as $\min_{(\mathcal{T},\chi)\in \mathsf{TD}} \max_{v \in V(\mathcal{T})} |\chi(v)|$.*

Observe that the second lower bound generalizes the result in [20] to any hypergraph.

## 4 UPPER BOUNDS

In this section, we show our upper bound results by constructing circuits that have matching size to our lower bound for certain classes of semirings.

Theorem 4.1. *Let $I$ be any instance that satisfies the degree constraint* DC*. There exists a multilinear and homogeneous circuit $F$ of size $O(2^w)$ that produces the polynomial $p^{\mathcal{H}}_I$ over any idempotent semiring, where $w =$ da-entw$(\mathcal{H})$.*

To prove Theorem 4.1, we introduce the following lemma.

Lemma 4.2. *Let $I$ be any instance that satisfies the degree constraint* DC*. Let* TD *be the set of all non-redundant tree decompositions of $\mathcal{H}$. For every $(\mathcal{T}, \chi) \in \mathsf{TD}$, let $Q_{(\mathcal{T},\chi)}$ be the CQ*

$$Q_{(\mathcal{T},\chi)}(\mathbf{x}_\mathcal{V}) \leftarrow \bigwedge_{v \in V(\mathcal{T})} S_{\chi(v)}(\mathbf{x}_{\chi(v)}). \tag{5}$$

*Then, there exists an instance $I_\mathcal{T} = (S_{\chi(v)})_{v \in V(\mathcal{T})}$ such that*

*(1) $p^{\mathcal{H}}_I = \bigoplus_{(\mathcal{T},\chi)\in\mathsf{TD}} p^{\mathcal{T}}_{I_\mathcal{T}}$ over any idempotent semiring, where $p^{\mathcal{T}}_{I_\mathcal{T}}$ is the polynomial of the CQ $Q_{(\mathcal{T},\chi)}$ over the instance $I_\mathcal{T}$.*
*(2) the size of the instance $|I_\mathcal{T}|$ is $O(2^w)$, where $w =$ da-entw$(\mathcal{H})$.*

The key takeaway of Lemma 4.2 is: we can construct a circuit of size $O(2^w)$ that produces $p^{\mathcal{H}}_I$ by building a subcircuit of size $O(2^w)$ for every $p^{\mathcal{T}}_{I_\mathcal{T}}$ and summing their output gates.

Proof of Theorem 4.1. By Lemma 4.2, it suffices to build a subcircuit (with output gate $f_T$) that produces $p^{\mathcal{T}}_{I_{\mathcal{T}}}$ for each tree decomposition $(\mathcal{T}, \chi) \in \mathsf{TD}$ . To produce the final circuit, we simply build a circuit $F$ of the form

$$(f_{\mathcal{T}_1} \oplus (f_{\mathcal{T}_2} \oplus \cdots (f_{\mathcal{T}_m} \oplus \mathbf{0}))),$$

where $\mathcal{T}_1, \ldots, \mathcal{T}_m$ are all the tree decompositions in TD.

We build the subcircuit for every tree decomposition $(\mathcal{T}, \chi) \in \mathsf{TD}$ by following the structure of the tree decomposition from the leaves to the root bag (root the tree arbitrarily; say at node $v_0$). We augment the tree decomposition with a hyperedge assignment mapping $\mu : \mathcal{E} \to V(\mathcal{T})$, which maps each hyperedge $e$ to a node $v$ in $V(\mathcal{T})$ such that $e \subseteq \chi(v)$. That is, $\mu(\cdot)$ partitions the hyperedges disjointly across the bags of $(\mathcal{T}, \chi)$ (this avoids over-multiplying of $x^e_*$ in the circuit construction). For any node $v \in V(\mathcal{T})$ in the decomposition, let $\mu^{-1}(v)$ denote the set of hyperedges of $\mathcal{H}$ mapped to $v$. Moreover, let $\mathsf{anc}_v$ be the variables of $\chi(v)$ that occur in its (unique) parent bag. For the root node $v_0$, we define $\mathsf{anc}_{v_0} = \emptyset$. Trivially, $\mathsf{anc}_v \subseteq \chi(v)$.

(1) *(join within the bag $\chi(v)$)* First, for each node $v \in V(\mathcal{T})$ and $\mathbf{i}_{\chi(v)} \in S_{\chi(v)}(\mathbf{x}_{\chi(v)})$, we associate a $\otimes$-gate denoted by $g_{\chi(v),\mathbf{i}_{\chi(v)}}$. Let $\mu^{-1}(v) = \{e_1, \ldots, e_\ell\} \subseteq \mathcal{E}$. For every $\mathbf{i}_{\chi(v)} \in S_{\chi(v)}(\mathbf{x}_{\chi(v)})$, we introduce a subcircuit

$$(x^{e_1}_{\mathbf{i}_{e_1}} \otimes (x^{e_2}_{\mathbf{i}_{e_2}} \otimes \cdots (x^{e_\ell}_{\mathbf{i}_{e_\ell}} \otimes \mathbf{1}))).$$

If $\mu^{-1}(v) = \emptyset$, the subcircuit is $\mathbf{1} \otimes \mathbf{1}$. As such, we define $g_{\chi(v),\mathbf{i}_{\chi(v)}}$ as the top $\otimes$-gate of this subcircuit.

(2) *(join child bags with the parent bag $\chi(v)$)* Next, for each node $v \in V(\mathcal{T})$ and $\mathbf{i}_{\chi(v)} \in S_{\chi(v)}(\mathbf{x}_{\chi(v)})$, we will associate another $\otimes$-gate denoted by $G_{\chi(v),\mathbf{i}_{\chi(v)}}$ constructed as follows: (suppose that the children of $v$ are the nodes $u_1, u_2, \ldots, u_\ell$; if $v$ is a leaf node, $\ell = 0$):

(2.1) *(project child bags)* We construct $\oplus$-gates for each child node $u_i$ that project to the variables in $\mathsf{anc}_{u_i}$. Specifically, consider a child node $u$. For each $\mathbf{i}_{\mathsf{anc}_u} \in \pi_{\mathsf{anc}_u}(S_{\chi(u)}(\mathbf{x}_{\chi(u)}))$, we introduce a subcircuit as follows. Let $\{\mathbf{j}_{\chi(u)} \in S_{\chi(u)}(\mathbf{x}_{\chi(u)}) \mid \mathbf{j}_{\mathsf{anc}_u} = \mathbf{i}_{\mathsf{anc}_u}\} = \{\mathbf{j}_1, \ldots, \mathbf{j}_m\}$. Then, the subcircuit takes the form

$$(G_{u,\mathbf{j}_1} \oplus (G_{u,\mathbf{j}_2} \oplus \cdots (G_{u,\mathbf{j}_m} \oplus \mathbf{0}))).$$

Here, $m$ is the degree of $\mathbf{i}_{\mathsf{anc}_u}$ in $S_{\chi(u)}(\mathbf{x}_{\chi(u)})$, i.e. the number of occurrences of $\mathbf{i}_{\mathsf{anc}_u}$ in $S_{\chi(u)}(\mathbf{x}_{\chi(u)})$. Thus, $m \geq 1$. Let $s_{u,\mathbf{i}_{\mathsf{anc}_u}}$ be the top $\oplus$-gate of this subcircuit.

(2.2) *(join across bags)* We construct a subcircuit for each $\mathbf{i}_{\chi(v)} \in S_{\chi(v)}(\mathbf{x}_{\chi(v)})$, which will be of the form:

$$(s_{u_1,\mathbf{i}_{\mathsf{anc}_{u_1}}} \otimes (s_{u_2,\mathbf{i}_{\mathsf{anc}_{u_2}}} \otimes \cdots (s_{u_\ell,\mathbf{i}_{\mathsf{anc}_{u_\ell}}} \otimes g_{\chi(v),\mathbf{i}_{\chi(v)}}))).$$

Note that this is well-defined, since for every child node $u_i$ of $v$, $\mathsf{anc}_{u_i} \subseteq \chi(v)$. Then, for $\ell > 0$, $G_{\chi(v),\mathbf{i}_{\chi(v)}}$ is the top $\otimes$-gate of this subcircuit. If $\ell = 0$, $G_{\chi(v),\mathbf{i}_{\chi(v)}}$ is the same gate as $g_{\chi(v),\mathbf{i}_{\chi(v)}}$.

(3) *(top $\oplus$-gate)* Finally, we construct the output gate $f_T$. For the root node $v_0$, we have $\mathsf{anc}_{v_0} = \emptyset$, and we perform the construction of the first step for the internal node case (viewing $v_0$ as the child node of an imaginary node that corresponds to an empty bag). This will create one subcircuit; we take its top $\oplus$-gate $s_{v_0,\mathbf{i}_\emptyset}$ to be the output gate of the circuit (i.e. $f_{\mathcal{T}} = s_{v_0,\mathbf{i}_\emptyset}$).

We now reason the number of gates in the resulting subcircuit for every tree decomposition.

($\otimes$-gates) For every node $v$, let $\ell$ be the number of child nodes of $v$. We introduce $(\ell + |\mu^{-1}(v)| + 1) \cdot |S_{\chi(v)}|$ $\otimes$-gates, where $(\ell + |\mu^{-1}(v)| + 1)$ is independent of the instance $I$. By Lemma 4.2, we know $|S_{\chi(v)}| = O(2^w)$. Hence, the total number of $\otimes$-gates introduced across all bags is $\sum_{v \in V(\mathcal{T})} O(2^w) = O(2^w)$.

($\oplus$-gates) Take any node $u \in V(\mathcal{T})$. Then, for every $\mathbf{i}_{\mathsf{anc}_u} \in \pi_{\mathsf{anc}_u} S_{\chi(u)}$, we construct a subcircuit at step (2.1) using $m$ $\oplus$-gates, where $m$ is the degree of $\mathbf{i}_{\mathsf{anc}_u}$ in the table $S_{\chi(u)}$, i.e. $m = |S_{\chi(u)} \ltimes \mathbf{i}_{\mathsf{anc}_u}| = O(2^w)$. Hence, the total number of $\oplus$-gates introduced across each node is $|S_{\chi(u)}| = O(2^w)$. The

number of $\oplus$-gates used to add up all tree decompositions is $O(1)$ (i.e. $(f_{\mathcal{T}_1} \oplus (f_{\mathcal{T}_2} \oplus \cdots (f_{\mathcal{T}_m} \oplus \mathbf{0})))$), since there are only a constant number of tree decompositions (independent of the instance $I$). □

***Notes on the Circuit Construction.*** Even though da-entw($\mathcal{H}$) is not known to be computable [15], we construct a circuit (as shown in the proof of Theorem 4.1) of size as predicated by da-entw($\mathcal{H}$), without computing the exact value of da-entw($\mathcal{H}$). In particular, we can always construct the circuit in time $O(|Q(I)|)$ for an instance $I$.

Additionally, the depth of the circuit in the above construction can be polynomial in the database instance because it depends on the degree $m$ whenever we do summation. However, it is easy to see that we can implement the summation of $m$ elements through a circuit of depth $O(\log(m)) = O(w)$, by using a binary tree of $\oplus$-gates, e.g. $(G_{u,\mathbf{j}_1} \oplus G_{u,\mathbf{j}_2}) \oplus (G_{u,\mathbf{j}_3} \oplus \mathbf{0})$ instead of the left-linear $(G_{u,\mathbf{j}_1} \oplus (G_{u,\mathbf{j}_2} \oplus (G_{u,\mathbf{j}_3} \oplus \mathbf{0})))$. Hence, the above semiring circuit can be constructed to have depth *logarithmic* in the instance $I$.

***Non-Idempotent Semirings.*** Idempotency is necessary to the correctness of Lemma 4.2 and Theorem 4.1. Indeed, in our current construction the same monomial $q$ can be produced via more than one tree decompositions, in which case a non-idempotent summation would lead to an incorrect polynomial. If we drop the idempotency requirement, we show the following weaker upper bound:

Theorem 4.3. *There is a multilinear and homogeneous circuit of size $O(2^w)$ that produces $p_I^{\mathcal{H}}$ over any semiring, where*

$$w = \textit{da-efhtw}(\mathcal{H}) := \min_{(\mathcal{T},\chi)\in \mathsf{TD}} \max_{v\in V(\mathcal{T})} \max_{h\in \overline{\Gamma}_n^* \cap \mathsf{HDC}} h(\chi(v)).$$

## 5 BOUNDS FOR FORMULAS

We define a *formula* over a semiring $\mathbb{S}$ to be a circuit over $\mathbb{S}$ such that the outdegree of every gate is exactly one (with the exception of the output gate, which has outdegree 0). In other words, the difference between a formula and a circuit is that any gate output cannot be reused across different gates. As an example, the circuit depicted in Figure 1 is actually a formula.

To capture lower bounds and upper bounds of formulas, we need to introduce a new type of tree decomposition.

*Definition 5.1.* A tree decomposition $(\mathcal{T}, \chi)$ of a hypergraph $\mathcal{H}$ is *inflationary* if $\mathcal{T}$ can be rooted such that whenever $t \in V(\mathcal{T})$ is an ancestor of $t' \in V(\mathcal{T})$, then $\chi(t) \subset \chi(t')$.

In other words, if we take any root-to-leaf path in $\mathcal{T}$, we will see bags with a strictly increasing set of variables. Both tree decompositions in Figure 1 are inflationary. Let $\mathsf{TD}_{\mathsf{F}}(\mathcal{H})$ be the set of all inflationary tree decompositions of $\mathcal{H}$. Note that $|\mathsf{TD}_{\mathsf{F}}(\mathcal{H})|$ is a quantity dependent on $\mathcal{H}$ and independent of an instance $I$.

***Discussion.*** The notion of an inflationary tree decomposition is in direct correspondence to that of an elimination tree [20] of a graph, and of an $f$-tree [24] of a hypergraph. However, it will be more useful for our width measure definitions to work with tree decompositions.

### 5.1 Upper Bounds

Define the following width measures:

$$\textsf{ida-entw}(\mathcal{H}) := \max_{h\in \overline{\Gamma}_n^* \cap \mathsf{HDC}} \min_{(\mathcal{T},\chi)\in \mathsf{TD}_{\mathsf{F}}} \max_{v\in V(\mathcal{T})} h(\chi(v))$$

$$\textsf{i-entw}(\mathcal{H}) := \max_{h\in \overline{\Gamma}_n^* \cap \mathsf{ED}} \min_{(\mathcal{T},\chi)\in \mathsf{TD}_{\mathsf{F}}} \max_{v\in V(\mathcal{T})} h(\chi(v))$$

The only difference with the definition of da-entw is that we consider a different set of tree decompositions. It is easy to check that ida-entw$(\mathcal{H}) \geq$ da-entw$(\mathcal{H})$ and i-entw$(\mathcal{H}) \geq$ entw$(\mathcal{H})$.

Theorem 5.2. *There exists a formula $F$ of size $O(2^w)$ that produces $p_I^{\mathcal{H}}$ over any idempotent semiring, where $w =$ ida-entw$(\mathcal{H})$.*

*Example 5.3.* Consider the 4-cycle query

$$Q(x_1, x_2, x_3, x_4) \leftarrow R(x_1, x_2), S(x_2, x_3), T(x_3, x_4), U(x_4, x_1)$$

with uniform cardinality constraints $|R|, |S|, |T|, |U| \leq N$. Observe that an $f$-representation for this query would have size $\Theta(N^2)$. However, we can do better by considering multiple tree decompositions. Indeed, consider the two inflationary tree decompositions as shown in Figure 1. These tree decompositions create 4 disjunctive rules, where their targets have size only $O(N^{3/2})$. This implies that we can create a formula for any idempotent semiring with size only $O(N^{3/2})$.

### 5.2 Lower Bounds

Let $(\mathcal{T}, \chi)$ be a rooted tree decomposition. For some $t \in V(\mathcal{T})$, we define $\overline{\chi}(t)$ to be the union of the bags that occur in the unique path from $t$ to the root. Note that $\overline{\chi}(t) \supseteq \chi(t)$.

Lemma 5.4. *Let $q_1, q_2$ be two monomials in $p_I^{\mathcal{H}}$ and $\mathcal{T}_{q_i} = (\mathcal{T}_i, \chi_i)$, $i = \{1, 2\}$, be their tree decompositions in a formula $F$. If the parse trees of $q_1, q_2$ in $F$ share a common gate $g$, then $\overline{\chi_1}(v_g) = \overline{\chi_2}(v_g)$.*

Proof. Since $F$ is a formula, the parse trees of $q_1, q_2$ share not only $g$, but also all the other $\otimes$-gates from $g$ to the root. Let $g_1, g_2, \ldots, g_\ell (= g)$ be these gates. From Lemma 3.4, we must have $\chi_1(v_{g_i}) = \chi_2(v_{g_i})$ for every $i = 1, \ldots, \ell$. Hence, $\overline{\chi_1}(v_g) = \bigcup_i \chi_1(v_{g_i}) = \bigcup_i \chi_2(v_{g_i}) = \overline{\chi_2}(v_g)$. □

In light of Lemma 3.4, we can assign a *type* $\overline{\mathsf{tp}}(g)$ to each $\otimes$-gate $g$ as $\overline{\chi}(v_g)$ for some tree decomposition $\mathcal{T}_q = (\mathcal{T}, \chi)$ of a monomial $q$. Moreover:

Lemma 5.5. *Let $t_1, t_2 \in Q^{\mathcal{H}}(I)$ such that their parse trees in a formula $F$ share a common gate $g$. Then, $t_1[\overline{\mathsf{tp}}(g)] = t_2[\overline{\mathsf{tp}}(g)]$.*

Proof. Since $F$ is a formula, the parse trees of $q_1, q_2$ share not only $g$, but also all the other $\otimes$-gates from $g$ to the root. Let $g_1, g_2, \ldots, g_\ell (= g)$ be these gates. From Lemma 3.5, we must have $t_1[\mathsf{tp}(g_i)] = t_2[\mathsf{tp}(g_i)]$ for every $i = 1, \ldots, \ell$. Hence, we have the desired equality. □

We can now show the main lower bound for formulas.

Theorem 5.6. *For any $\epsilon > 0$, there exists an instance $I$ and $k > 0$ that satisfies the constraints $k \times$ HDC such that any formula $F$ that computes $p_I^{\mathcal{H}}$ over the counting semiring $\mathbb{C}$ has size*

$$\log |F| \geq (1 - \epsilon) \cdot \textit{ida-entw}(\mathcal{H}).$$

Similar to Theorem 3.12, when the degree constraints are the uniform cardinality constraints $\mathsf{ED} \cdot \log N$, we obtain a lower bound of $\Omega(N^{\text{i-entw}(\mathcal{H})})$ for the size of any formula over the counting or tropical semiring.

## 6 BEYOND SUM-PRODUCT QUERIES

In this section, we extend our results to the case where we are given a full CQ $Q$ with hypergraph $\mathcal{H} = ([n], \mathcal{E})$ along with projection variables $U \subseteq [n]$. The task is to construct a circuit that computes the polynomial for every output tuple. Formally, given an instance $I$ and a tuple $t_U \in \pi_U(Q(I))$, we want to compute the polynomial

$$p_I^{\mathcal{H}}[t_U] := \sum_{s \in Q(I) : s[U] = t_U} \prod_{e \in \mathcal{E}} x^e_{s[e]}.$$

We seek to find the size of the smallest circuit that contains one output gate each computing a polynomial above. We write $\mathbb{S}(f_1, \ldots, f_m)$ to denote the size of the smallest circuit over semiring $\mathbb{S}$ that computes all the polynomials $f_1, \ldots, f_m$. Then, we can define

$$\mathbb{S}(\mathcal{H} \mid U, \mathsf{HDC}) \coloneqq \max_{\mathsf{HDC} \vdash I} \mathbb{S}(\{p_I^{\mathcal{H}}[t_U] \mid t_U \in \pi_U(Q(I))\}).$$

Theorem 6.1. *For a subset $U \subseteq [n]$, define $\mathcal{H}_U = ([n], \mathcal{E} \cup \{U\})$. Then, for a semiring $\mathbb{S} \in \{\mathbb{T}, \mathbb{C}, \mathbb{B}_{\mathsf{lin}}\}$[5], we have:*

$$\frac{1}{3} \cdot \mathbb{S}(\mathcal{H}_U, \mathsf{HDC}) \leq \mathbb{S}(\mathcal{H} \mid U, \mathsf{HDC}) \leq 4 \cdot \mathbb{S}(\mathcal{H}_U, \mathsf{HDC}).$$

The theorem implies that, instead of looking at a set of polynomials, it suffices to consider a single polynomial that corresponds to a CQ with the same set of constraints HDC and the same hypergraph with the exception of an additional hyperedge that contains the projection variables $U$. Hence, all lower and upper bounds from the previous sections can be used to characterize this case as well.

Proof. We start with the first inequality. Consider the instance $I_U$ that maximizes $\mathbb{S}(\mathcal{H}_U, \mathsf{HDC})$. Let $I_U = I \cup \{R_U\}$, where $I$ is the instance for the hypergraph $\mathcal{H}$. We will show that

$$\mathbb{S}(p_{I_U}^{\mathcal{H}_U}) \leq 3 \cdot (\{p_I^{\mathcal{H}}[t_U] \mid t_U \in \pi_U(Q(I))\}).$$

This completes the first part, because $\mathbb{S}(\mathcal{H}_U, \mathsf{HDC}) = \mathbb{S}(p_{I_U}^{\mathcal{H}_U})$ and $\mathbb{S}(\{p_I^{\mathcal{H}}[t_U] \mid t_U \in \pi_U(Q(I))\}) \leq \mathbb{S}(\mathcal{H} \mid U, \mathsf{HDC})$.

Suppose that we have a circuit $F$ that computes all polynomials $\{p_I^{\mathcal{H}}[t_U] \mid t_U \in \pi_U(Q(I))\}$ using $|F|$ gates. Observe that:

$$p_{I_U}^{\mathcal{H}_U} = \sum_{t_U \in \pi_U(Q(I)) \cap R_U} \left( x_{t_U}^U \cdot p_I^{\mathcal{H}}[t_U] \right).$$

The above equation leads to a direct construction for $p_{I_U}^{\mathcal{H}_U}$. For every output gate of $F$ corresponding to $t_U \in \pi_U(Q(I)) \cap R_U$, we introduce a new $\otimes$ gate that multiplies it with the input gate $x_{t_U}^U$. This needs at most $|F|$ more gates. Then, we perform a summation over all these gates, which costs at most $|F|$ gates. In total, the resulting circuit has size at most $3|F|$.

We next show the second inequality. Consider the instance $I$ that maximizes $\mathbb{S}(\mathcal{H} \mid U, \mathsf{HDC})$. Let $I_U = I \cup \{R_U\}$, where $R_U = \pi_U(Q(I))$. Note that this is a valid instance for $\mathcal{H}_U$ (with corresponding CQ $Q_U$), since the hyperedge $U$ has no constraints. We can write the polynomial as:

$$p_{I_U}^{\mathcal{H}_U} = \sum_{t \in Q_U(I_U)} \left( x_{t[U]}^U \cdot \prod_{e \in \mathcal{E}} x_{t[e]}^e \right) = \sum_{t \in Q(I)} \left( x_{t[U]}^U \cdot \prod_{e \in \mathcal{E}} x_{t[e]}^e \right).$$

We now consider the partial derivatives of the above polynomial. The *partial derivative* of a multilinear polynomial $f$ w.r.t. a variable $x$, denoted as $\partial f / \partial x$, is obtained by removing all monomials not containing $x$, and removing $x$ from the remaining monomials. For every variable $x_{t_U}^U$, where $t_U \in \pi_U(Q(I))$, we have:

$$\frac{\partial p_{I_U}^{\mathcal{H}_U}}{\partial x_{t_U}^U} = p_I^{\mathcal{H}}[t_U].$$

We now apply the following beautiful result from [4]: for every polynomial $f$ and the set of all variables $S$ of this polynomial, $\mathbb{S}(f, \{\partial f / \partial x\}_{x \in S}) \leq 4 \cdot \mathbb{S}(f)$. Moreover, since we can always ignore

[5]We slightly abuse the notation here; strictly speaking, $\mathbb{B}_{\mathsf{lin}}$ is not a semiring but a requirement on the circuit computing over $\mathbb{B}$.

output gates that we do not need to use in the circuit, for any subset of variables $S'$ in $f$, we also have $\mathbb{S}(\{\partial f/\partial x\}_{x\in S'}) \le 4\cdot\mathbb{S}(f)$. Finally, we can write:

$$\begin{aligned}4\cdot\mathbb{S}(\mathcal{H}_U,\mathsf{HDC}) &\ge 4\cdot\mathbb{S}(p_{I_U}^{\mathcal{H}_U}) \ge \mathbb{S}(\{\partial p_{I_U}^{\mathcal{H}_U}/\partial x_{t_U}^{U} \mid t_U\in\pi_U(Q(I))\})\\ &= \mathbb{S}(\{p_I^{\mathcal{H}}[t_U] \mid t_U\in\pi_U(Q(I))\}) = \mathbb{S}(\mathcal{H}\mid U,\mathsf{HDC}).\end{aligned}$$

This completes the proof. □

*Example 6.2.* Consider the following CQ:

$$Q(x,y,z) \leftarrow R(x,y)\wedge S(y,z)\wedge T(z,x)$$

with $U=\{x,y\}$ and uniform cardinality constraints $|R|,|S|,|T|\le N$. Theorem 6.1 tells us that to find a (tropical) circuit that computes the polynomials for every output tuple, we need to consider the hypergraph with hyperedges $\{x,z\},\{x,y\},\{y,z\}$ and the same cardinality constraints. The entropic width for this case is $3/2$, so the circuit size for this example is $\Theta(N^{3/2})$.

When a CQ is full and we want to compute the polynomial on all possible outputs, Theorem 6.1 forces us to consider only one possible tree decomposition that puts all variables in a single bag (since $U=[n]$). Hence, the circuit size is equal to the worst-case output size of the query.

## 7 RELATED WORK

***Circuits for Datalog.*** Deutch et al. [9] initiated the study of computing provenance polynomial by circuits. They focused on the provenance polynomials for Datalog. Specifically, they constructed Boolean circuits for Datalog provenance and explored for which semirings a *faithful representation* can be achieved. Amarilli et al. [1] considered provenance polynomials for union of conjunctive queries (UCQ) over the *universal provenance semiring* $\mathbb{N}[X]$. However, in their work the database instances were restricted to be tree-like, i.e., of bounded treewidth. Lastly, factorised databases, and in particular $d$-representations and $f$-representations [24] can be viewed as special cases of circuits governed by exactly one tree decomposition.

***Circuits for Polynomials.*** The question of finding lower bounds on the size of a circuit that represents a polynomial has been considered for different types of circuits. Jukna [12] studied this question for different types of semiring circuits, while most work has looked at Boolean (monotone or non-monotone) circuits [27].

***Query Evaluation by Circuits.*** Wang and Yi [29] constructed circuits for CQs under degree constraints by instantiating PANDA [19]. Their circuits have polylogarithmic depth and their sizes match the polymatroid bound up to polylogarithmic factors. However, their (Boolean) circuits make essential use of $\neg$-gates and do not generalize to arbitrary semirings. In contrast, our circuits only use the operations $\oplus$ and $\otimes$ and admit unconditional lower bounds.

## 8 CONCLUSION

In this paper, we study circuits for sum-product queries, and prove strong lower bounds and upper bounds of the circuits size.

One immediate open problem is to close the gap between the upper bound and lower bound for the circuit size over the Boolean semiring. Another interesting future direction is to explore the CQs with self-join from the lens of (non-)multi-linearity of the provenance polynomial [2, 6, 7, 14].

## REFERENCES

[1] Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. 2015. Provenance Circuits for Trees and Treelike Instances. In *ICALP (2) (Lecture Notes in Computer Science, Vol. 9135)*. Springer, 56–68.

[2] Antoine Amarilli and Benny Kimelfeld. 2022. Uniform Reliability of Self-Join-Free Conjunctive Queries. *Log. Methods Comput. Sci.* 18, 4 (2022).

[3] Albert Atserias, Martin Grohe, and Dániel Marx. 2008. Size Bounds and Query Plans for Relational Joins. In *FOCS*. IEEE Computer Society, 739–748.

[4] Walter Baur and Volker Strassen. 1983. The Complexity of Partial Derivatives. *Theor. Comput. Sci.* 22 (1983), 317–330.

[5] Markus Bläser. 2013. Fast Matrix Multiplication. *Theory Comput.* 5 (2013), 1–60.

[6] Karl Bringmann, Nofar Carmeli, and Stefan Mengel. 2022. Tight Fine-Grained Bounds for Direct Access on Join Queries. In *PODS*. ACM, 427–436.

[7] Nofar Carmeli and Luc Segoufin. 2023. Conjunctive Queries With Self-Joins, Towards a Fine-Grained Enumeration Complexity Analysis. In *PODS*. ACM, 277–289.

[8] Terence H. Chan and Raymond W. Yeung. 2002. On a relation between information inequalities and group theory. *IEEE Trans. Inf. Theory* 48, 7 (2002), 1992–1995.

[9] Daniel Deutch, Tova Milo, Sudeepa Roy, and Val Tannen. 2014. Circuits for Datalog Provenance. In *ICDT*. OpenProceedings.org, 201–212.

[10] Tomasz Gogacz and Szymon Torunczyk. 2017. Entropy Bounds for Conjunctive Queries with Functional Dependencies. In *ICDT (LIPIcs, Vol. 68)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 15:1–15:17.

[11] Todd J. Green, Gregory Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *PODS*. ACM, 31–40.

[12] Stasys Jukna. 2015. Lower Bounds for Tropical Circuits and Dynamic Programs. *Theory Comput. Syst.* 57, 1 (2015), 160–194.

[13] Ahmet Kara, Hung Q. Ngo, Milos Nikolic, Dan Olteanu, and Haozhe Zhang. 2020. Maintaining Triangle Queries under Updates. *ACM Trans. Database Syst.* 45, 3 (2020), 11:1–11:46.

[14] Aziz Amezian El Khalfioui and Jef Wijsen. 2023. Consistent Query Answering for Primary Keys and Conjunctive Queries with Counting. In *ICDT (LIPIcs, Vol. 255)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 23:1–23:19.

[15] Mahmoud Abo Khamis, Phokion G. Kolaitis, Hung Q. Ngo, and Dan Suciu. 2021. Bag Query Containment and Information Theory. *ACM Trans. Database Syst.* 46, 3 (2021), 12:1–12:39.

[16] Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2018. In-Database Learning with Sparse Tensors. In *PODS*. ACM, 325–340.

[17] Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2020. Learning Models over Relational Data Using Sparse Tensors and Functional Dependencies. *ACM Trans. Database Syst.* 45, 2 (2020), 7:1–7:66.

[18] Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. 2016. FAQ: Questions Asked Frequently. In *PODS*. ACM, 13–28.

[19] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2023. What do Shannon-type Inequalities, Submodular Width, and Disjunctive Datalog have to do with one another? arXiv:1612.02503

[20] Balagopal Komarath, Anurag Pandey, and Chengot Sankaramenon Rahul. 2022. Monotone Arithmetic Complexity of Graph Homomorphism Polynomials. In *ICALP (LIPIcs, Vol. 229)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 83:1–83:20.

[21] Edward A. Lee and Alberto L. Sangiovanni-Vincentelli. 1998. A framework for comparing models of computation. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 17, 12 (1998), 1217–1229.

[22] Dan Olteanu. 2020. The Relational Data Borg is Learning. *Proc. VLDB Endow.* 13, 12 (2020), 3502–3515.

[23] Dan Olteanu and Jiewen Huang. 2008. Using OBDDs for Efficient Query Evaluation on Probabilistic Databases. In *SUM (Lecture Notes in Computer Science, Vol. 5291)*. Springer, 326–340.

[24] Dan Olteanu and Jakub Závodný. 2015. Size Bounds for Factorised Representations of Query Results. *ACM Trans. Database Syst.* 40, 1 (2015), 2:1–2:44.

[25] Neil Robertson and Paul D. Seymour. 1984. Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B* 36, 1 (1984), 49–64.

[26] John E. Savage. 1998. *Models of computation - exploring the power of computing*. Addison-Wesley.

[27] Claus-Peter Schnorr. 1976. A Lower Bound on the Number of Additions in Monotone Computations. *Theor. Comput. Sci.* 2, 3 (1976), 305–315.

[28] Amir Shaikhha, Maximilian Schleich, and Dan Olteanu. 2021. An Intermediate Representation for Hybrid Database and Machine Learning Workloads. *Proc. VLDB Endow.* 14, 12 (2021), 2831–2834.

[29] Yilei Wang and Ke Yi. 2022. Query Evaluation by Circuits. In *PODS*. ACM, 67–78.

## A MISSING PROOFS

This section contains the missing proofs from the main body of the paper.

Proof of Lemma 3.5. A similar statement was proved in [20]. Assume for contradiction that for some $v \in \mathrm{tp}(g)$, $i_1 = t_1[v] \neq t_2[v] = i_2$. There are two cases:

(1) $v$ is going to be forgotten in the parent of $B_g$ in $T_{q_1}$: By the construction of $T_{q_1}$, both of the children of $g$ in $F$ must compute $i_1$ for $v$. If the left subtree of $\mathrm{pt}_{q_2}(g)$ computed $i_2$ for $v$, then replace the right subtree of $\mathrm{pt}_{q_2}(g)$ with the right subtree of $\mathrm{pt}_{q_1}(g)$. Otherwise $i_2$ for $v$ is computed in the right subtree of or outside $\mathrm{pt}_{q_2}(g)$, in that case replace the left subtree of $\mathrm{pt}_{q_2}(g)$ with the left subtree of $\mathrm{pt}_{q_1}(g)$.
(2) Otherwise, by the construction of $T_{q_1}$, $i_1$ is computed in at least one of the children of $g$ in $F$ and outside the $\mathrm{pt}_{q_1}(g)$. Then, if $i_2$ is computed in $\mathrm{pt}_{q_2}(g)$, then replace $\mathrm{pt}_{q_1}(g)$ with $\mathrm{pt}_{q_2}(g)$ in $\mathrm{pt}_{q_1}$. Otherwise $i_2$ must be computed outside $\mathrm{pt}_{q_2}(g)$ in $\mathrm{pt}_{q_2}$, then replace $\mathrm{pt}_{q_2}(g)$ with $\mathrm{pt}_{q_1}(g)$ in $\mathrm{pt}_{q_2}$.

The resulting subtree in $F$ is a parse tree for some monomial that has $i_1$ *and* $i_2$ for $v$, which leads to a contradiction since any monomial in $p_I^{\mathcal{H}}$ must has consistent values for all variables. □

Proof of Lemma 3.9. We can write:

$$\begin{aligned}\max_{h\in H}\min_{(T,\chi)\in\mathcal{F}}\max_{v\in V(\mathcal{T})} h(\chi(v)) &= \max_{h\in H}\max_{\beta\in\mathcal{M}_{\mathcal{F}}}\min_{(T,\chi)\in\mathcal{F}} h(\beta((T,\chi)))\\ &= \max_{\beta\in\mathcal{M}_{\mathcal{F}}}\max_{h\in H}\min_{B\in\mathsf{image}(\beta)} h(B)\\ &= \max_{\mathcal{B}\in\mathbf{B}_{\mathcal{F}}}\max_{h\in H}\min_{B\in\mathcal{B}} h(B)\end{aligned}$$

where the first equality is implied by Lemma 7.12 in [19]. □

Proof of Lemma 3.10. Fix some $h$, and let $\mathcal{T} \in \mathrm{TD}_{\mathrm{C}}$ be the decomposition that minimizes the RHS term. We say that a tree decomposition $\mathcal{T}_1$ is *dominated* by another tree decomposition $\mathcal{T}_2$ if every bag of $\mathcal{T}_1$ is a subset of some bag of $\mathcal{T}_2$. We can always construct a non-redundant decomposition $\mathcal{T}' = (T', \chi')$ that is dominated by $\mathcal{T}$ by removing the redundant bags (see for example [18]). Clearly, we have $\mathcal{T}' \in \mathrm{TD}$. Then, because of the monotonicity of $h$:

$$\max_{v\in V(\mathcal{T})} h(\chi(v)) \geq \max_{v\in V(T')} h(\chi'(v)) \geq \min_{(T,\chi)\in\mathsf{TD}}\max_{v\in V(\mathcal{T})} h(\chi(v)).$$

This concludes the proof. □

Proof of Lemma 4.2. Recall that TD is the set of all non-redundant tree decompositions. As Section 3.3, we define $\mathbf{B}_{\mathrm{TD}}$ and for each $\mathcal{B} \in \mathbf{B}_{\mathrm{TD}}$, the disjunctive Datalog rule $P$ as in (3). We construct a model $(T_B)_{B\in\mathcal{B}}$ (i.e. a set of tables $T_B(\mathbf{x}_B)$, one for each $B \in \mathcal{B}$) using the constructive proof of Lemma 4.1 in [19]. By Lemma 3.9, the tables constructed for each disjunctive Datalog rule are of size $O(2^w)$, where $w = \mathsf{da\text{-}entw}(\mathcal{H})$.

Next, at each bag $\chi(v)$ in every tree decomposition $(\mathcal{T}, \chi) \in \mathrm{TD}$, we construct a table $S_{\chi(v)}(\mathbf{x}_{\chi(v)})$ ($\mathbf{x}_{\chi(v)}$ is its schema) as follows: (1) take the union over all output tables $T_{\chi(v)}(\mathbf{x}_{\chi(v)})$ from every disjunctive Datalog rule defined by $\mathcal{B} = \mathsf{image}(\beta) \in \mathbf{B}_{\mathrm{TD}}$, if the bag selector $\beta$ selects this bag $\chi(v)$ from $(\mathcal{T}, \chi)$, and (2) semijoin-reduce the union by every input relation $R_e(\mathbf{x}_e)$, to prune off tuples that do not contribute to any monomials in $p_I^{\mathcal{H}}$. In other words (using $\ltimes$ for semijoins),

$$S_{\chi(v)}(\mathbf{x}_{\chi(v)}) = \bigcup_{\mathsf{image}(\beta)\in\mathbf{B}_{\mathrm{TD}},\ \beta(\mathcal{T},\chi)=\chi(v)} T_{\chi(v)} \ltimes_{e\in\mathcal{E}} R_e(\mathbf{x}_e).$$

As a result, it is easy to check that $S_{\chi(v)}(\mathbf{x}_{\chi(v)})$ is of size $O(2^w)$. In addition, we augment each tree decomposition with a hyperedge assignment mapping $\mu : \mathcal{E} \to V(\mathcal{T})$, which maps each hyperedge $e$ to a bag $t$ in $(\mathcal{T}, \chi)$. Effectively, $\mu(\cdot)$ partitions the hyperedges across the bags of $(\mathcal{T}, \chi)$. By Corollary 7.13 in [19], the query results can be written as the following union of CQs (one per tree decomposition):

$$Q(I) = \bigcup_{(\mathcal{T},\chi)\in \mathsf{TD}} Q_{(\mathcal{T},\chi)}(I_{\mathcal{T}}).$$

Recall that the query $Q_{(\mathcal{T},\chi)}(\mathbf{x}_{\mathcal{V}}) \leftarrow \bigwedge_{v\in V(\mathcal{T})} S_{\chi(v)}(\mathbf{x}_{\chi(v)})$ for each $(\mathcal{T}, \chi) \in \mathsf{TD}$ is evaluated over the instance $I_{\mathcal{T}} = (S_{\chi(v)}(\mathbf{x}_{\chi(v)}))_{v\in V(\mathcal{T})}$. We use $p^{\mathcal{T}}_{I_{\mathcal{T}}}$ to denote the sum-product polynomial of $Q_{(\mathcal{T},\chi)}$ over the instance $I_{\mathcal{T}}$. Therefore, we have

$$\begin{aligned}
p^{\mathcal{H}}_I &= \bigoplus_{t\in Q(I)} \bigotimes_{e\in\mathcal{E}} x^e_{t[e]} \\
&= \bigoplus_{t\in Q(I)} \bigotimes_{v\in V(\mathcal{T})} \bigotimes_{e\in\mathcal{E}:\mu(e)=v} x^e_{t[e]} && \text{(for any } (\mathcal{T},\chi)\in\mathsf{TD}) \\
&= \bigoplus_{t\in Q(I)} \bigoplus_{(\mathcal{T},\chi)\in\mathsf{TD}} \bigotimes_{v\in V(\mathcal{T})} \bigotimes_{e\in\mathcal{E}:\mu(e)=v} x^e_{t[e]} && \text{(by idempotence of } \oplus) \\
&= \bigoplus_{(\mathcal{T},\chi)\in\mathsf{TD}} \bigoplus_{t\in Q(I)} \bigotimes_{v\in V(\mathcal{T})} \bigotimes_{e\in\mathcal{E}:\mu(e)=v} x^e_{t[e]} && \text{(by commutativity of } \oplus) \\
&= \bigoplus_{(\mathcal{T},\chi)\in\mathsf{TD}} \bigoplus_{t\in Q_{(\mathcal{T},\chi)}(I_{\mathcal{T}})} \bigotimes_{v\in V(\mathcal{T})} \bigotimes_{e\in\mathcal{E}:\mu(e)=v} x^e_{t[e]} && \text{(by Corollary 7.13 in [19])} \\
&= \bigoplus_{(\mathcal{T},\chi)\in\mathsf{TD}} p^{\mathcal{T}}_{I_{\mathcal{T}}} && \text{(by definition of } p^{\mathcal{T}}_{I_{\mathcal{T}}})
\end{aligned}$$

□

Similar to Theorem 4.1, we show the following lemma for non-idempotent semirings.

Lemma A.1. *Let $I$ be any instance that satisfies the degree constraint* DC. *Let $(\mathcal{T}, \chi)$ be a tree decomposition of $\mathcal{H}$ that attains the minimum in the definition of da-efhtw($\mathcal{H}$). Then, there exists an instance $I_{\mathcal{T}} = (S_{\chi(v)})_{v\in V(\mathcal{T})}$ such that*

*(1) $p^{\mathcal{H}}_I = p^{\mathcal{T}}_{I_{\mathcal{T}}}$ over any semiring, where $p^{\mathcal{T}}_{I_{\mathcal{T}}}$ is the polynomial of $Q_{(\mathcal{T},\chi)}(5)$ over the instance $I_{\mathcal{T}}$, and*
*(2) the size of the instance $|I_{\mathcal{T}}|$ is $O(2^w)$, where $w$ = da-efhtw($\mathcal{H}$).*

Proof. We start by choosing a tree decomposition $(\mathcal{T}, \chi)$ that attains the minimum in the definition of da-efhtw($\mathcal{H}$). Following the proof of Lemma 4.2, we construct a table $S_{\chi(v)}(\mathbf{x}_{\chi(v)})$ for each bag $\chi(v)$ in $(\mathcal{T}, \chi)$, and obviously, $p^{\mathcal{H}}_I = p^{\mathcal{T}}_{I_{\mathcal{T}}}$.

By Lemma 4.1 in [19], the table $S_{\chi(v)}(\mathbf{x}_{\chi(v)})$ is of size $O(2^s)$, where $s = \max_{h\in\overline{\Gamma}^*_n\cap\mathsf{HDC}} h(\chi(v))$ (as only one tree decomposition is considered). Thus, the instance $I_{\mathcal{T}} = (S_{\chi(v)}(\mathbf{x}_{\chi(v)}))_{v\in V(\mathcal{T})}$ of $Q_{(\mathcal{T},\chi)}$ is of size $O(2^w)$, where $w = \max_{v\in V(\mathcal{T})} \max_{h\in\overline{\Gamma}^*_n\cap\mathsf{HDC}} h(\chi(v)) =$ da-efhtw($\mathcal{H}$). □

Using Lemma A.1, Theorem 4.1 is now immediate. This is because one can simply apply the circuit construction for $p^{\mathcal{T}}_{I_{\mathcal{T}}}$ in the proof of Theorem 4.1, as the final circuit for $p^{\mathcal{H}}_I$.

Proof of Theorem 5.6. Let $\mathcal{M}_{\mathsf{TD}_\mathsf{F}}$ be the set of all maps $\overline{\beta} : \mathsf{TD}_\mathsf{F}(\mathcal{H}) \to 2^{\mathcal{V}}$ such that $\overline{\beta}(\mathcal{T}, \chi) = \chi(t)$ for some $t \in V(\mathcal{T})$. Let $\mathbf{B}_{\mathsf{TD}_\mathsf{F}}$ be the collection of images of all $\overline{\beta} \in \mathcal{M}$, i.e. $\mathbf{B}_{\mathsf{TD}_\mathsf{F}} = \{\mathcal{B} \mid \mathcal{B} =$

$\mathsf{image}(\overline{\beta})$ for some $\overline{\beta} \in \mathcal{M}_{\mathsf{TD_F}}\}$. We claim, similarly to Lemma 3.9:

$$\text{ida-entw}(\mathcal{H}) \le \max_{\mathcal{B}\in\mathbf{B}_{\mathsf{TD_F}}} \max_{h\in\overline{\Gamma}_n^*\cap\mathsf{HDC}} \min_{B\in\mathcal{B}} h(B).$$

Fix the bag selector $\overline{\beta}$ that maximizes the RHS above and let $\mathcal{B} = \mathsf{image}(\overline{\beta})$. Consider the disjunctive Datalog rule $P : \bigvee_{B\in\mathcal{B}} T_B(\mathbf{x}_B) \leftarrow \bigwedge_{e\in\mathcal{E}} R_e(\mathbf{x}_e)$. Note that the LHS of $P$ is finite. Let $\mathbf{D}$ be the worst-case database instance as constructed in Lemma 3.8.

Consider the type assignment $\overline{\mathsf{tp}}$ to each $\otimes$-gate in $F$ given by Lemma 5.4. Let $S_B \subseteq S$ be the set of $\otimes$-gates that has type $B \subseteq [n]$. By Lemma 5.5, each gate $g \in S_B$ can be uniquely associated with a value $w(g)$. Let $T_B := \{w(g) \mid g \in S_B\}$; we claim that $\{T_B\}_{B\in\mathcal{B}}$ is a model for $P$.

Indeed, take any output tuple $t$; its corresponding monomial $q$ must be computed by $F$. Take the parse tree $\mathsf{pt}_q$, and construct its tree decomposition $T_q$. Now, note that $T_q = (\mathcal{T}, \chi)$ may not belong to $\mathsf{TD_F}$. Thus, we construct a new tree decomposition $T_q' = (T, \overline{\chi})$. By construction, $T_q'$ is inflationary and hence $T_q' \in \mathsf{TD_F}$. Now, consider the gate $g$ corresponding to the bag $B = \overline{\beta}(T_q')$. By Lemma 5.5, $t[B] = w(g) \in T_B$. Finally, take $B \in \mathcal{B}$ that maximizes $S_B$. By Lemma 3.9 and Lemma 3.8, we have $\log|S_B| \ge \log|P(\mathbf{D})| \ge (1-\epsilon)\cdot\text{ida-entw}(\mathcal{H})$. Since $|F| \ge |S_B|$, this concludes the proof. □

Proof of Theorem 5.2. The proof initially follows the proof of Theorem 4.1, with the only difference being replacing $\mathsf{TD}$ with $\mathsf{TD}_F$. As in the proof of Theorem 4.1, it suffices to build a subformula (with output gate $f_T$) for each tree decomposition $(\mathcal{T}, \chi) \in \mathsf{TD}_F$ that produces $p_{I_\mathcal{T}}^{\mathcal{T}}$. To produce the final formula for $p_I^{\mathcal{H}}$ we build a formula $F = (f_{T_1} \oplus (f_{T_2} \oplus \cdots (f_{T_m} \oplus \mathbf{0})))$, where $T_1, \ldots, T_m$ are all the tree decompositions in $\mathsf{TD_F}$.

We build the subformula for every $(\mathcal{T}, \chi) \in \mathsf{TD_F}$ by following inductively the structure of the tree decomposition from the root to the leaves. Since any internal bag is contained in a leaf bag in an inflationary decomposition, we will take $\mu(\cdot)$ to be an assignment of hyperedges to the *leaves* of $T$. For any leaf node $v \in V(\mathcal{T})$, let $\mu^{-1}(v)$ denote the set of hyperedges of $\mathcal{H}$ mapped to $v$. Moreover, let $\mathsf{anc}_v$ be the variables of $\chi(v)$ that occur in its (unique) parent bag. For the root node $v_0$, we define $\mathsf{anc}_{v_0} = \emptyset$. Trivially, we have $\mathsf{anc}_v \subseteq \chi(v)$. We inductively define $\mathsf{F}_v[\mathbf{i}_{\mathsf{anc}_v}]$ as follows.

If $v$ is an internal node with children $u_1, \ldots, u_\ell$, then

$$\mathsf{F}_v[\mathbf{i}_{\mathsf{anc}_v}] := \bigoplus_{\mathbf{j}_{\chi(v)}\in S_{\chi(v)}:\mathbf{j}_{\mathsf{anc}_v}=\mathbf{i}_{\mathsf{anc}_v}} \mathsf{F}_{u_1}[\mathbf{j}_{\chi(v)}] \otimes \cdots \otimes \mathsf{F}_{u_\ell}[\mathbf{j}_{\chi(v)}]$$

If $v$ is a leaf node with $\mu^{-1}(v) = \{e_1, \ldots, e_\ell\} \subseteq E(\mathcal{H})$, then

$$\mathsf{F}_v[\mathbf{i}_{\mathsf{anc}_v}] := \bigoplus_{\mathbf{j}_{\chi(v)}\in S_{\chi(v)}:\mathbf{j}_{\mathsf{anc}_v}=\mathbf{i}_{\mathsf{anc}_v}} x_{\mathbf{j}_{e_1}}^{e_1} \otimes \cdots \otimes x_{\mathbf{j}_{e_\ell}}^{e_\ell}$$

Finally, the output gate $f_T$ corresponds to the top gate of the formula $\mathsf{F}_{v_0}[\emptyset]$.

Observe that the corresponding circuit is a formula because of the inflationary property of all tree decompositions. Indeed, a subcircuit $\mathsf{F}_v[\mathbf{i}_{\mathsf{anc}_v}]$ can only be used in a unique path to the output gate. It remains to reason about the formula size. Since it is a formula, to obtain an upper bound it suffices to count the number of subformulas of the form $x_{\mathbf{j}_{e_1}}^{e_1} \otimes \cdots \otimes x_{\mathbf{j}_{e_\ell}}^{e_\ell}$ produced by the above construction. There are $|S_{\chi(v)}(\mathbf{x}_{\chi(v)})| = O(2^w)$ many subformulas that correspond to any given leaf bag $v$, hence the total number is also $O(2^w)$. The number of $\oplus$-gates used to add up all tree decompositions is $O(1)$, since there are only constant many tree decompositions (independent of the instance $I$). Hence, the total number of gates in the formula is $O(2^w)$. □

Received December 2023; revised February 2024; accepted March 2024