The Containment Problem for *Real* Conjunctive Queries with Inequalities

T.S. Jayram
IBM Almaden
jayram@almaden.ibm.com

Phokion G. Kolaitis*
IBM Almaden
kolaitis@almaden.ibm.com

Erik Vee
IBM Almaden
vee@almaden.ibm.com

ABSTRACT

Query containment is a fundamental algorithmic problem in database query processing and optimization. Under set semantics, the query-containment problem for conjunctive queries has long been known to be NP-complete. In real database systems, however, queries are usually evaluated under bag semantics, not set semantics. In particular, SQL queries are evaluated under bag semantics and return multisets as answers, since duplicates are not eliminated unless explicitly requested. The exact complexity of the query-containment problem for conjunctive queries under bag semantics has been an open problem for more than a decade; in fact, it is not even known whether this problem is decidable.

Here, we investigate, under bag semantics, the query-containment problem for conjunctive queries with inequalities. It has been previously shown that, under set semantics, this problem is complete for the second level of the polynomial hierarchy. Our main result asserts that, under bag semantics, the query-containment problem for conjunctive queries with inequalities is undecidable. Actually, we establish the stronger result that this problem is undecidable even if the following two restrictions hold at the same time: (1) the queries use just a single binary relation; and (2) the total number of inequalities is bounded by a certain fixed value. Moreover, the same undecidability results hold under bag-set semantics.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*query processing, relational databases*; H.2.3 [**Database Management**]: Languages—*query languages*; F.2.2 [**Analysis of Algorithms & Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*

General Terms

Algorithms, Management, Languages, Theory

Keywords

conjunctive queries, query containment, inequalities, bag semantics, bag-set semantics, undecidability

*On leave from UC Santa Cruz

PODS'06, June 26–28, 2006, Chicago, Illinois, USA.

1. Introduction and Summary of Results

Query containment is regarded as a fundamental algorithmic problem in database query processing and optimization. This problem asks: given two queries Q and Q', is it true that $Q(D) \subseteq Q'(D)$, for every database D? Over the years, researchers have investigated in depth the query-containment problem for several different classes of frequently asked database queries. The class of conjunctive queries (that is, select-project-join queries) is arguably the most prominent among these classes. Chandra and Merlin [3] showed that the query-containment problem for conjunctive queries is NP-complete. After this, researchers investigated the worst-case complexity of the query-containment problem for broader classes of database queries that naturally subsume conjunctive queries. Specifically, Sagiv and Yannakakis [13] showed that the query-containment problem for unions of conjunctive queries is Π_2^p-complete, where Π_2^p is the second level of the polynomial hierarchy (NP is the first level of the polynomial hierarchy - see [11]). Klug [7] studied conjunctive queries with comparison predicates $\neq$, $<$, and $\leq$. He showed that the query-containment problem for conjunctive queries with comparison predicates is in Π_2^p, and conjectured that this upper bound is tight. Klug's conjecture was subsequently confirmed by van der Meyden [14], who proved that the query-containment problem for conjunctive queries with comparison predicates is Π_2^p-complete. As a matter of fact, van der Meyden showed that even the query-containment problem for conjunctive queries with just inequalities ($\neq$) as the only comparison predicate is Π_2^p-complete; this result was further refined in [8].

All aforementioned complexity-theoretic results were obtained under the assumption that queries are evaluated under *set* semantics. This means that the database relations given as inputs to queries are sets (i.e., no duplicate tuples are allowed) and that queries return sets as answers. In real database systems, however, queries are usually evaluated under *bag* semantics, not set semantics: input database relations may be bags (multisets), and queries may return bags as answers. In particular, SQL queries are evaluated under bag semantics, since duplicate tuples are not eliminated unless explicitly specified in the syntax using the `SELECT DISTINCT` construct. In addition to faster response, the reason for not eliminating duplicate tuples in SQL is that the values of aggregate operators, such as `AVG` and `COUNT`, depend on the multiplicities of the tuples in the database relations. In a paper titled "Optimization of *Real* Conjunctive Queries" [4], Chaudhuri and Vardi drew attention to this discrepancy between database theory and practice, and raised the question of whether the known complexity results about conjunctive queries carry over from set semantics to bag semantics.

Chaudhuri and Vardi [4] discovered that, under bag semantics, the query-containment problem for conjunctive queries is Π_2^p-hard, which implies that, in all likelihood, the change from set semantics to bag semantics is accompanied by a jump in complexity. Moreover, they found that the same hardness result holds under *bag-set* semantics, the variant of bag semantics in which the input database relations are sets, but the queries return bags as answers. Chaudhuri and Vardi, however, were not able to pinpoint the exact complexity of this problem. As a matter of fact, even though more than a decade has passed since the publication of [4], it is still not known whether, under bag semantics, the query-containment problem for conjunctive queries is decidable. On the other hand, Ioannidis and Ramakrishnan [6] showed that, under bag semantics, the query-containment problem for unions of conjunctive queries is undecidable.

In this paper, we study, under bag semantics, the query-containment problem for conjunctive queries with inequalities ($\neq$). Our study is motivated by two considerations: first, this problem is of interest *in its own right*, as conjunctive queries with inequalities form the most natural extension of conjunctive queries with comparison predicates; second, tools developed in the course of this study may turn out to be of use in attacking, under bag semantics, the containment problem for conjunctive queries. Our main result is that, under bag semantics, the query-containment problem for conjunctive queries with inequalities is *undecidable*. Note that, in general, the inputs to this problem are conjunctive queries over arbitrary relational schemas and with an arbitrary number of inequalities. Thus, it is natural to ask whether solvability results can be obtained by considering restrictions in which the relations in the schemas have bounded arities or the total number of inequalities in the queries is bounded. We establish the stronger result that, under bag semantics, the query-containment problem for conjunctive queries with inequalities is undecidable, even if the following two restrictions hold at the same time: (1) the queries use just a single binary relation; and (2) the total number of inequalities is bounded by a certain fixed value.

To prove these results, we first show that, under bag semantics, the query-containment problem for conjunctive queries with inequalities is polynomial-time equivalent to the same problem under bag-set semantics; moreover, we exhibit polynomial-time reductions that do not increase the total number of inequalities in the queries. We also show that, under either semantics, the problem is polynomial-time equivalent to the restriction in which the queries use just a single binary relation. After this, the crucial undecidability result is established by showing that Hilbert's Tenth Problem has a recursive reduction to the query-containment problem for conjunctive queries with inequalities and under bag-set semantics. This reduction is carried out in two stages. In the first stage, we identify a class of special databases, which we call *polynomial encoders*, and construct a family of conjunctive queries (without inequalities) so that the evaluation of polynomials can be simulated by evaluating these queries on polynomial encoders under bag-set semantics. This makes it possible to recursively reduce Hilbert's Tenth Problem to the query-containment problem for conjunctive queries (without inequalities) and under bag-set semantics, provided the input databases are restricted to be polynomial encoders. In the second stage, we amend appropriately the queries used in the first stage (in particular, we add inequalities), so that the containment holds over arbitrary databases, not just polynomial encoders.

It should be pointed out that Hilbert's Tenth Problem was also used by Ioannidis and Ramakrishman [6] in proving that, under bag semantics, the containment problem for unions of conjunctive queries is undecidable. Their reduction, however, is much simpler than ours, as the union operation can be used to easily simulate the addition operation on monomials. In the absence of the union operation, we have to develop rather elaborate combinatorial machinery to simulate the evaluation of polynomials; moreover, our simulation makes an essential use of the presence of inequalities in the queries. It should also be pointed out that in a series of papers, including [1, 2, 12], a group of researchers studied, under bag semantics, the query-containment problems for conjunctive queries with and without comparison predicates, but they did not settle the decidability question for these problems. The main technical assertion in these papers is a necessary and sufficient condition for query containment under bag semantics, a condition that involves a family of *canonical* databases. Unfortunately, there is a subtle counting error in the proof of the claimed necessary and sufficient condition. In fact, a counterexample to this condition is reported in [16].

Finally, it is interesting to note the similarities and differences between query containment and query equivalence, another fundamental problem in database query optimization. Clearly, query equivalence is always reducible to query containment. Consider the case of conjunctive queries (without inequalities). Under set semantics, both query equivalence and query containment are NP-complete [3]. However, the situation is, in all likelihood, different under bag semantics, since query equivalence has the same complexity as GRAPH ISOMORPHISM (hence, is in NP), while query containment is Π_2^P-hard [4]. In the case of conjunctive queries with inequalities, Nutt, Sagiv, and Shurin [10] showed that query equivalence under bag semantics is in PSPACE (see also [5, 15]). Thus, our undecidability result for query containment under bag semantics shows a provable dramatic difference in complexity between the two problems.

2. Basic Concepts and Notation

A *bag* or *multiset* is a collection of objects each of which occurs one or more times in the collection. A *relation* R of arity k is a bag of k-tuples whose elements belong to some underlying fixed domain. If R is a k-ary relation and $(A_1, \ldots, A_k)$ is a k-tuple, then we write $|R(A_1, \ldots, A_k)|$ to denote the *multiplicity* (number of occurrences) of $(A_1, \ldots, A_k)$ in the bag R. In particular, $|R(A_1, \ldots, A_k)| = 0$ means that $(A_1, \ldots, A_k)$ does not occur in the bag R.

A *database schema* (or, simply, *schema*) is a set $\mathcal{S}$ of distinct relation symbols $\{R_1, \ldots, R_t\}$ of fixed arities $k_1, \ldots, k_t$. A *database instance* (or, simply, a *database*) D for the schema $\mathcal{S}$ is a set of relations, also denoted by $R_1, \ldots, R_t$, of arities $k_1, \ldots, k_t$. If R_i is one of the relations of a database D and $(A_1, \ldots, A_{k_i})$ is a k_i-tuple, we will write $|R(A_1, \ldots, A_{k_i})|_D$, to emphasize that R_i is a relation in the database D, unless the database is understood from the context in which case we will write $|R(A_1, \ldots, A_{k_i})|$. If $|R(A_1, \ldots, A_{k_i})|_D > 0$, we say that $R(A_1, \ldots, A_{k_i})$ is a *fact* of D.

In what follows, we will upper-case letters to denote elements of the domain, and lower-case letters to denote variables. As usual, $\neq$ denotes the *built-in* inequality relation with the standard interpretation over any domain.

DEFINITION 1 (CONJUNCTIVE QUERY WITH INEQUALITIES). Let $\mathcal{S} = \{R_1, \ldots, R_t\}$ be a schema and n a non-negative integer. A n-ary *conjunctive query with inequalities* over $\mathcal{S}$ is a rule of the form:

$$Q(x_1, \ldots, x_n) :- T_1(\vec{z_1}), \ldots, T_p(\vec{z_p}), \tag{1}$$

where T_i is one of the relation symbols in $\mathcal{S}$ or the $\neq$ symbol, and each $\vec{z_i}$ is a tuple of variables in set $Z = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$.

Each expression $T_i(\vec{z_i})$ is a *subgoal* of Q; the list of subgoals is the *body* of Q, while $Q(x_1, \ldots, x_n)$ is the head of Q. Note that the same subgoal may occur more than once in the body of Q. We only consider *safe* queries, that is, each variable x_j in the head must appear in at least one of the subgoals of Q that involve a relation symbol in $\mathcal{S}$. The variables $x_1, \ldots, x_n$ are called *free* and the variables $y_1, \ldots, y_m$ are called *bound*. When $n = 0$ (i.e., Q has no free variables), then we say that Q is a *bound* (or, *Boolean*) query. □

An *assignment mapping* (or, simply, an *assignment*) is a mapping τ from the set of free and bound variables of a query Q to the underlying domain of elements. If $\vec{z} = (z_1, \ldots, z_s)$ is a tuple of variables, we will write $\tau(\vec{z})$ to denote the tuple $(\tau(z_1), \ldots, \tau(z_s))$. Assignments are used to define the semantics of queries, that is, to define what it means to evaluate a query on a database. Under *set semantics*, relations are assumed to be sets, and evaluating a query essentially amounts to determining whether there is an assignment that satisfies every subgoal of the query. Here, we consider two different kinds of semantics, namely *bag* semantics and *bag-set* semantics. The difference from set semantics is that, instead of asking whether a satisfying assignment exists, we now want to know how many different satisfying assignment are there. Informally, in both bag semantics and bag-set semantics, the query is evaluated over a database algebraically by treating the conjunction as a product and then "marginalizing out" the bound variables y_j by summing over all choices for the y_j's. The difference between bag and bag-set semantics is whether the multiplicities of the tuples in the database relations are taken into account while evaluating the product.

Formally, let D be a database over a schema $\mathcal{S} = \{R_1, \ldots, R_t\}$ and let Q be a conjunctive query with inequalities as in (1). The *result of evaluating Q on D under bag semantics* is the n-ary relation $\text{EVAL}_{\mathsf{B}}(Q, D)$ defined as follows: for every n-tuple $(A_1, \ldots, A_n)$ of elements from the domain,

$$|\text{EVAL}_{\mathsf{B}}(Q, D)(A_1, \ldots, A_n)| = \sum_{\tau} \prod_{i=1}^{p} |T_i(\tau(\vec{z_i}))|_D,$$

where τ ranges over all assignments such that $\tau(x_i) = A_i$, $1 \le i \le n$. For a bound query, the result $\text{EVAL}(Q, D)$ is just a single non-negative integer. For bag-set semantics, we remove all duplicates of facts in the instance D so that each relation is a set. Let $\widetilde{D}$ denote the database obtained this way. The *result of evaluating Q under bag-set semantics* is the n-ary relation $\text{EVAL}_{\mathsf{BS}}(Q, D) = \text{EVAL}_{\mathsf{B}}(Q, \widetilde{D})$. When the subscript is omitted, $\text{EVAL}(Q, D)$ simply refers to query evaluation under bag semantics.

DEFINITION 2 (QUERY CONTAINMENT). Let Q and Q' be two n-ary conjunctive queries with inequalities over some schema $\mathcal{S}$. We say that Q *is contained in Q' under bag semantics*, denoted by $Q \subseteq_{\mathsf{B}} Q'$ if for every database D over $\mathcal{S}$ and for every n-tuple $(A_1, \ldots, A_n)$ of elements from the domain, we have that

$$|\text{EVAL}_{\mathsf{B}}(Q, D)(A_1, \ldots, A_n)| \le |\text{EVAL}_{\mathsf{B}}(Q', D)(A_1, \ldots, A_n)|.$$

In other words, the multiplicities of facts of tuples corresponding to Q are no more than those corresponding to Q'. The concept *Q is contained in Q' under bag-set semantics*, denoted by $Q \subseteq_{\mathsf{BS}} Q'$, is defined in an analogous way using $\text{EVAL}_{\mathsf{BS}}(Q, D)$ and $\text{EVAL}_{\mathsf{BS}}(Q', D)$.

The main problem that we consider in this paper is query containment under bag semantics and under bag-set semantics. In fact, we parameterize this problem into a family of sub-problems using the characteristics of the database schema and the number of inequalities in the queries as parameters.

DEFINITION 3 (QUERY CONTAINMENT PROBLEM). Let k be a non-negative integer and m, d be two positive integers. We write $\mathsf{ConQC}_{\mathsf{B}}(k, m, d)$ to denote the following decision problem: given a schema $\mathcal{S}$ having at most m relations, each of arity at most d, and given two conjunctive queries Q and Q' each of which has at most k inequalities, is $Q \subseteq_{\mathsf{B}} Q'$?

We write $\mathsf{ConQC}_{\mathsf{BS}}(k, m, d)$ to denote the same problem under bag-set semantics.

We also allow (some of) these parameters to be unbounded; we denote this by setting the appropriate parameter(s) to ∞. For example, $\mathsf{ConQC}_{\mathsf{B}}(\infty, \infty, \infty)$ denotes the conjunctive query problem under bag semantics where there are no restrictions on any one of the three parameters. □

To simplify our presentation, we will make systematic use of the concept of a *view*. Informally, a view V is defined by a rule in which other previously defined views can be part of the body of V. We do not allow views to be defined in terms of themselves, that is, recursive views are prohibited. In effect, views are queries defined by a non-recursive Datalog program with inequalities. A formal self-contained inductive definition of views over a schema $\mathcal{S}$ is as follows. For the base case, any relation symbol in $\mathcal{S} \cup \{\neq\}$ with variables as arguments is a view. Inductively, suppose $V_1, \ldots V_k$ are views that are already defined. Then, a view V can be defined by any rule of the

$$V(x_1, \ldots, x_n) :- T_1(\vec{z_1}), \ldots, T_p(\vec{z_p}),$$

where each $\vec{z_i}$ is a tuple of variables from the set $Z = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$, and each $T_i(\vec{z_i})$ is obtained from one of the views V_j, $1 \le j \le k$, by replacing the free variables of V_j by corresponding variables from $\vec{z_j}$.

It is clear that every view can be *unfolded* to a conjunctive query with inequalities. This is done by tracing the inductive definition of views and replacing bound variables in such a way that the subgoals of the views do not have common ("local" to the subgoal) bound variables. For example, consider the views

$$U(u_1, u_2) :- A(u_1, u_2, w),$$

$$W(u_1, u_2) :- A(u_1, u_2, w), A(u_1, u_2, w), B(u_1, w), u_1 \neq w$$

$$V(x_1, x_2) :- U(x_1, y), U(x_1, y), W(x_2, y)$$

on the schema $\{A, B\}$. Unfolding V, yields the following conjunctive query with inequalities:

$$Q(x_1, x_2) :- A(x_1, y, w_1), A(x_1, y, w_2), A(x_2, y, w_3), \\ A(x_2, y, w_3), B(x_2, w_3), x_2 \neq w_3.$$

Given view V and database D, we can define $\text{EVAL}_{\mathsf{B}}(V, D)$, the result of evaluating V on D under bag semantics, by first unfolding V to a query Q and then evaluating Q on D under bag semantics. In a similar way, we can define the result $\text{EVAL}_{\mathsf{BS}}(V, D)$ of evaluating V on D under bag-set semantics.

By induction on the definition of views, it is not hard to show that if we have a view

$$V(x_1, \ldots, x_n) :- T_1(\vec{z_1}), \ldots, T_p(\vec{z_p}),$$

then for every database D and every n-tuple $(A_1, \ldots, A_n)$ of elements from the domain, we have that

$$|\text{EVAL}_{\mathsf{B}}(V, D)(A_1, \ldots, A_n)| = \sum_{\tau} \prod_{i=1}^{p} |\text{EVAL}_{\mathsf{B}}(T_i, D)(\tau(\vec{z_i}))|,$$

where τ ranges over all assignments such that $\tau(x_i) = A_i$, for $1 \le i \le n$. Moreover, $|\text{EVAL}_{\mathsf{BS}}(V, D)(A_1, \ldots, A_n)|$ may be evaluated similarly. In the sequel, we will use these two facts repeatedly.

We conclude this section with two more concepts. Let Q be a conjunctive query with inequalities over a schema $\mathcal{S}$ defined by the rule $Q(x_1, \ldots, x_n) :- T_1(\vec{z_1}), \ldots, T_p(\vec{z_p})$, where each $T_i \in \mathcal{S} \cup \{\neq\}$. For an assignment τ that satisfies the inequalities in Q, we define $\tau(Q)$ to be the database over $\mathcal{S}$ whose relations are the bags of facts $T_i(\tau(\vec{z_i}))$, for every $T_i \in \mathcal{S}$. Finally, we say that τ is a *homomorphism* from Q to a database D, if every fact in $\tau(Q)$ appears in D.

3. Bag vs. bag-set semantics

Depending on whether we use bag semantics or bag-set semantics, evaluating queries can give different answers. So it is natural to wonder whether the complexity of the conjunctive query containment problem (with or without inequalities) under bag semantics is different than its complexity under bag-set semantics.

The following theorem shows that in fact, the two problems are polytime reducible to each other, so their complexities are essentially the same. In fact, the theorem shows something stronger: The conjunctive query containment problem (with or without inequalities), under both bag-set and bag semantics, in which we have no restrictions on the number of relations or their arity, is polytime reducible to the conjunctive query containment problem (with or without inequalities, respectively) in which queries are restricted to use a single binary relation. Previously, Chadhuri and Vardi [4] noted that $\mathsf{ConQC_B}(0,\infty,\infty)$ is polytime reducible to $\mathsf{ConQC_{BS}}(0,\infty,\infty)$. Our claim is stronger, not just because it applies to $\neq$-constraints, but because it also provides a reduction that reduces the queries to have a single binary relation.

THEOREM 1. *For any $k \geq 0, m \geq 1$, and $d \geq 1$, we have the following:*

$$\begin{aligned} \mathsf{ConQC_{BS}}(k,m,d) &\leq_{\mathsf{P}} \mathsf{ConQC_B}(2k,m,d) \\ \mathsf{ConQC_B}(k,\infty,\infty) &\leq_{\mathsf{P}} \mathsf{ConQC_{BS}}(k,1,2) \end{aligned}$$

The reductions hold even if we allow any combination of k, m, d to be ∞. (If $k = \infty$, we interpret $2k = \infty$ as well.)

PROOF SKETCH. We first give the reduction for the first item. Suppose we are given an instance of $\mathsf{ConQC_{BS}}(k,r,m)$ in the form of queries Q_1, Q_2, which may have up to k $\neq$-constraints each. (In the case that $k = \infty$, we allow them to have arbitrarily many $\neq$-constraints.) Since we are considering bag-set semantics, we may assume without loss of generality that every subgoal of Q_1 and Q_2 appears with multiplicity one. Let Q_1' be query Q_1 modified so that each of its subgoals appears with multiplicity exactly two. Define

$$\mathsf{Q}_1^{\mathsf{B}} :- \mathsf{Q}_1 \wedge \mathsf{Q}_1 \quad \text{and} \quad \mathsf{Q}_2^{\mathsf{B}} :- \mathsf{Q}_2 \wedge \mathsf{Q}_1'$$

Note that queries $\mathsf{Q}_1^{\mathsf{B}}$ and $\mathsf{Q}_2^{\mathsf{B}}$ both have at most $2k$ $\neq$-constraints. We claim that $\textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_1, D) \leq \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_2, D)$ for all D if and only if $\textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1^{\mathsf{B}}, D') \leq \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2^{\mathsf{B}}, D')$ for all D'.

The key to this reduction is that if all of the facts in database D have multiplicity one, then $\textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1, D) = \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1', D)$. On the other hand, if D has facts with multiplicity greater than one, then query $\mathsf{Q}_2^{\mathsf{B}}$ will benefit more than $\mathsf{Q}_1^{\mathsf{B}}$. Specifically, suppose that

$$\textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_1, D) > \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_2, D)$$

for some database D. Of course, this implies by definition that

$$\textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1, \widetilde{D}) > \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2, \widetilde{D})$$

(Recall that $\widetilde{D}$ is the database obtained from D by setting the multiplicity of all facts in D to one.) Hence,

$$\begin{aligned} \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1^{\mathsf{B}}, \widetilde{D}) &= \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1, \widetilde{D}) \cdot \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1, \widetilde{D}) \\ &> \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2, \widetilde{D}) \cdot \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1', \widetilde{D}) \\ &= \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2^{\mathsf{B}}, \widetilde{D}) \end{aligned}$$

On the other hand, suppose

$$\textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_1, D) \leq \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_2, D)$$

for all D. For query Q given by $\mathsf{Q} : -\mathsf{A}_1(\vec{z}_1), \ldots, \mathsf{A}_p(\vec{z}_p)$, define the *weight of homomorphism* ϕ mapping from Q to D to be the value $\prod_{i\in[p]} |\mathsf{A}_i(\phi(\vec{z}_i))|_D$. Then for any $w > 0$, let k_w be the number of homomorphisms from Q_1 to D that have weight w. Clearly,

$$\begin{aligned} \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_1, D) &= \sum_w k_w \\ \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1, D) &= \sum_w w k_w \\ \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1', D) &= \sum_w w^2 k_w \end{aligned}$$

Noting that $\textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2, D) \geq \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_2, D) \geq \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_1, D)$, we see

$$\begin{aligned} &\textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2 \wedge \mathsf{Q}_1', D) \\ &\quad = \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2, D) \cdot \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1', D) \\ &\quad \geq \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_1, D) \cdot \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1', D) \\ &\quad = \left(\sum_w k_w\right)\left(\sum_w w^2 k_w\right) \\ &\quad \geq \left(\sum_w w \cdot k_w\right)^2 \text{ by Cauchy-Schwartz} \\ &\quad = \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1 \wedge \mathsf{Q}_1, D) \end{aligned}$$

That is, $\textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1^{\mathsf{B}}, D) \leq \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2^{\mathsf{B}}, D)$. So the reduction works as claimed. This completes the proof for the first item.

We now give the reduction for the second item. Suppose we are given Q_1 and Q_2, each with at most k inequalities. Note that Q_1 and Q_2 may have subgoals with multiplicity greater than one. Further, suppose that $\mathsf{Q}_1, \mathsf{Q}_2$ are defined over r relations, $\mathsf{R}_1, \ldots, \mathsf{R}_r$ with arities $k_1, \ldots, k_r$ respectively, and that the queries use variables $\mathcal{V} = \{v_1, \ldots, v_n\}$. Without loss of generality, we assume $k_i \geq 2$ for all $i \in [r]$.

Given such a query, Q, suppose we may write Q as

$$\mathsf{Q} :- \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{k_i} \mathsf{R}_i(\vec{z}_{ij})$$

where each $\vec{z}_{ij}$ belongs to $\mathcal{V}$. Again, note that the above expression may have repeated subgoals. We define a corresponding query, denoted Q^{BS}, which uses a single binary relation R, in terms of views $\mathsf{View}_1, \ldots, \mathsf{View}_r$:

$$\mathsf{Q}^{\mathsf{BS}} :- \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{k_i} \mathsf{View}_i(\vec{z}_{ij})$$

where for each $i \in [r]$, we define View_i using binary relation R as follows.

$$\begin{aligned} &\mathsf{Path}_\ell(s,t) \\ &\quad :- \quad \mathsf{R}(s, v_1), \mathsf{R}(v_1, v_2), \ldots, \mathsf{R}(v_{\ell-1}, t) \\ &\mathsf{View}_i(u_1, \ldots, u_{k_i}) \\ &\quad :- \quad \mathsf{Path}_{r+2}(s_1, u_1), \ldots, \mathsf{Path}_{r+2}(s_{k_i}, u_{k_i}) \\ &\qquad\quad\; \mathsf{Path}_{i+1}(s_1, t_1), \ldots, \mathsf{Path}_{i+1}(s_{k_i}, t_{k_i}) \\ &\qquad\quad\; \mathsf{R}(s_2, t_1), \ldots, \mathsf{R}(s_{k_i}, t_{k_i-1}) \end{aligned}$$

Notice that Q^{BS} has no repeated subgoals. We claim that

$$\textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1, D) \leq \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2, D)$$

for all D if and only if

$$\textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_1^{\mathsf{BS}}, D') \leq \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_2^{\mathsf{BS}}, D')$$

for all D'. The details of the proof of this claim appear in the full version of this paper. Here, we only give a rough intuition.

First of all, observe that each subgoal of the original queries, e.g. $R_i(\vec{z}_{ij})$, is replaced by an analogous view, e.g. $\mathsf{View}_i(\vec{z}_{ij})$. Given a database D, we can perform a similar operation: For each fact in D, say $R_i(Z_1, \ldots, Z_{k_i})$, we replace it with a set of facts corresponding to the *canonical database* associated with $\mathsf{View}_i(Z_1, \ldots, Z_{k_i})$. (Specifically, for each subgoal $R(x, y)$ in the unfolding of View_i, add fact $R(X, Y)$ to the database, where X and Y are constants. Furthermore, we require that each bound variable be mapped to its own unique constant, and that each free variable u_ℓ is mapped to the constant Z_ℓ.) Call this new database D^{BS}. It is not hard to see that every homomorphism from Q to D has a corresponding homomorphism from Q^{BS} to D^{BS}. With some more work, it is possible to show that the converse is true as well: for every homomorphism from Q^{BS} to D^{BS}, there is a corresponding homomorphism from Q to D. (The exact structure of View_i was chosen carefully to ensure this.) That is,

$$\textsc{Eval}_{\mathsf{B}}(\mathsf{Q}, D) = \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}^{\mathsf{BS}}, D^{\mathsf{BS}}) \tag{2}$$

On the other hand, suppose we are given a database D, and we wish to construct a database, which we denote D^{B}, with the property that

$$\textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}^{\mathsf{BS}}, D) = \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}, D^{\mathsf{B}}) \tag{3}$$

In this case, for each tuple $X_1, \ldots, X_{k_i}$ consisting of constants from the domain of D, we add fact $\mathsf{R}_i(X_1, \ldots, X_{k_i})$ to database D^{B} with multiplicity $\textsc{Eval}_{\mathsf{BS}}(\mathsf{View}_i(X_1, \ldots, X_{k_i}), D)$. (Strictly speaking, we do not add the fact if the evaluation has value 0.) It is not difficult to see that produces a database D^{B} satisfying equation (3).

So, if there is a D such that $\textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1, D) > \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2, D)$, then $\textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_1^{\mathsf{BS}}, D^{\mathsf{BS}}) > \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_2^{\mathsf{BS}}, D^{\mathsf{BS}})$, by equation (2). Conversely, if $\textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_1^{\mathsf{BS}}, D) > \textsc{Eval}_{\mathsf{BS}}(\mathsf{Q}_2^{\mathsf{BS}}, D)$ for some D, then $\textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_1, D^{\mathsf{B}}) > \textsc{Eval}_{\mathsf{B}}(\mathsf{Q}_2, D^{\mathsf{B}})$, by equation (3). The theorem thus follows.

□

COROLLARY 2. *For all* $k \geq 0$,

$$\begin{aligned}
\mathsf{ConQC}_{\mathsf{B}}(k, \infty, \infty) &\leq_{\mathsf{P}} \mathsf{ConQC}_{\mathsf{BS}}(k, 1, 2) \\
&\leq_{\mathsf{P}} \mathsf{ConQC}_{\mathsf{B}}(2k, 1, 2) \\
\mathsf{ConQC}_{\mathsf{BS}}(k, \infty, \infty) &\leq_{\mathsf{P}} \mathsf{ConQC}_{\mathsf{BS}}(2k, 1, 2) \\
&\leq_{\mathsf{P}} \mathsf{ConQC}_{\mathsf{B}}(4k, 1, 2)
\end{aligned}$$

In the next section, we will construct queries using many relations, in order to show that $\mathsf{ConQC}_{\mathsf{BS}}(k, \infty, \infty)$ is undecidable for some bounded k. Corollary 2 implies that both $\mathsf{ConQC}_{\mathsf{BS}}(2k, 1, 2)$ and $\mathsf{ConQC}_{\mathsf{BS}}(4k, 1, 2)$ are undecidable as well.

4. Undecidability of conjunctive query containment

In this section, we show that the conjunctive query containment problem with inequalities is undecidable under bag-set semantics by exhibiting a reduction from Hilbert's Tenth Problem. Our reduction will use homogeneous polynomials of degree d (i.e., each term is a product of d variables, not necessarily distinct) with non-negative coefficients. We will use the following version of Hilbert's Tenth Problem in our reduction, which can be obtained by combining the results of Matiayasevich [9] with some additional arguments (details will appear in the full version of this paper).

THEOREM 3. *Let $P_1(x_1, \ldots, x_n)$, and $P_2(x_1, \ldots, x_n)$ be homogeneous polynomials of degree d each having the same set of terms with positive integer coefficients. Further, assume that x_1 divides both $P_1(\vec{x})$ and $P_2(\vec{x})$. Then it is undecidable to determine whether there are non-negative integers $x_1, \ldots, x_n$ such that*

$$P_1(x_1, \ldots, x_n) > x_1^d \cdot P_2(x_1, \ldots, x_n)$$

This problem is undecidable even if we restrict ourselves to homogeneous polynomials of degree $d = 5$ and with $n = 59$ variables. □

Throughout the remainder of the paper, both $P_1(x_1, \ldots, x_n)$ and $P_2(x_1, \ldots, x_n)$ will refer to degree d homogeneous polynomials each having the same set of m terms with positive integer coefficients. For $j \in [m]$, we associate the j-th term with an ordered d-tuple $\mathcal{T}_j$ in $[n]^d$; since x_1 divides both $P_1(\vec{x})$ and $P_2(\vec{x})$, we further assume that the first entry of $\mathcal{T}_j$ is 1. Abusing notation slightly, we will also think of $\mathcal{T}_j$ as a multiset. For example, if $\mathcal{T}_j = (1, 1, 3, 7)$, then $\prod_{i \in \mathcal{T}_j} x_i = x_1^2 x_3 x_7$. We write

$$P_1(\vec{x}) = \sum_{j=1}^{m} \alpha_j \prod_{i \in \mathcal{T}_j} x_i \quad \text{and} \quad P_2(\vec{x}) = \sum_{j=1}^{m} \beta_j \prod_{i \in \mathcal{T}_j} x_i,$$

where $\alpha_j, \beta_j \geq 0$ for all j. For the rest of the paper, we will fix the m-dimensional vectors α and β corresponding to P_1 and P_2, respectively.

4.1 Proof Overview

Our goal is to show that the undecidable problem on polynomials described in Theorem 3 can be reduced to the problem of query containment under bag semantics. Since the proof of the reduction is fairly involved, we will break it into several steps. We begin in Section 4.2 by describing two queries Poly_1 and Poly_2 that do not have any inequalities. Further, we will restrict the evaluations of these queries over a very specific class of database instances, called *polynomial encoders*. For each $\xi \in \mathbb{N}_0^n$, we will construct a polynomial encoder, denoted D_ξ, and show in Theorem 5 that $\textsc{Eval}(\mathsf{Poly}_1, D_\xi) = P_1(\xi)$ and $\textsc{Eval}(\mathsf{Poly}_2, D_\xi) = \xi_1^d P_2(\xi)$. By Theorem 3, there is no algorithm for the query containment problem if the evaluation is restricted to the class of polynomial encoders.

For the full problem where all instances are considered, we have to work considerably harder, and this is where we use the power of inequalities. In Section 4.3, we will extend the queries Poly_1 and Poly_2 to produce queries Q_1 and Q_2, respectively, with inequalities. For technical reasons, we will work with *augmented polynomial encoders*, which are database instances that consist of polynomial encoders augmented by a small set of facts. Now, when we consider databases that are augmented polynomial encoders, the arguments we laid out in Section 4.2 go through as before. But what if the database does not have the desired structure? Then the left-hand query, Q_1, could potentially *cheat* by mapping to the database in ways that we did not anticipate. Although we cannot stop this, we can guarantee that for every map of Q_1 to the database that cheats, there is a corresponding map from Q_2 to the database. We do this by defining a view $\mathsf{CounterCheating}$ that will be part of Q_2. This view contains many copies of Poly_1 along with appropriate inequality constraints. For every map from Q_1 to the database that cheats, there is a corresponding map from one of the copies of Poly_1 in $\mathsf{CounterCheating}$ that maps in an identical fashion. This ensures that cheating helps Q_2 as much as it helps Q_1. The definition of $\mathsf{CounterCheating}$ and the notion of cheating, together with the proofs are described in Section 4.3.

4.2 Conjunctive Queries over Polynomial Encoders

The polynomial encoder for $\xi \in \mathbb{N}_0^n$, denoted D_ξ, will use domain elements $X_1, \ldots, X_n$, corresponding to variables of the polynomial, domain elements $T_1, \ldots, T_m$, corresponding to terms of the polynomial, as well as auxiliary elements T_0 and $U_{k,1}, \ldots, U_{k,\xi_k}$ for each $k \in [m]$. We will describe its structure later in this section.

Recall that α and β denote the vector of coefficients for $P_1(\vec{x})$ and $P_2(\vec{x})$, respectively. Both Poly_1 and Poly_2 will be described in terms of the following views: $\mathsf{Term}(u_1, \ldots, u_d, z_0), \mathsf{Value}(v)$, $\mathsf{Coeff}_\alpha(z_0)$, and $\mathsf{Coeff}_\beta(z_0)$. The following lemma states the several key properties of these views when evaluated on a polynomial encoder. We defer its proof to the next subsection.

LEMMA 4. *Let D_ξ be a polynomial encoder. Then*

1. $\textsc{Eval}(\mathsf{Term}, D_\xi) = \{(X_{i_1}, \ldots, X_{i_d}, T_j) | (i_1, \ldots, i_d) = \mathcal{T}_j\}$, *with each element having multiplicity one.*

2. $\textsc{Eval}(\mathsf{Value}, D_\xi) = \{X_1, \ldots, X_n\}$, *with element X_k having multiplicity ξ_k, for all $k \in [n]$.*

3. $\textsc{Eval}(\mathsf{Coeff}_\alpha, D_\xi) = \{T_0, \ldots, T_j\}$, *where T_j has multiplicity α_j for all $j \in [m]$, and T_0 has multiplicity one.*

4. *Likewise,* $\textsc{Eval}(\mathsf{Coeff}_\beta, D_\xi) = \{T_0, \ldots, T_j\}$, *where T_j has multiplicity β_j for all $j \in [m]$, and T_0 has multiplicity one.* □

We now define Poly_1 and Poly_2 in terms of the above views.

$$\begin{aligned}\mathsf{Poly}_1 \coloneq\ & \mathsf{Term}(u_1, \ldots, u_d, z_0), \mathsf{Coeff}_\alpha(z_0),\\ & \mathsf{Value}(u_1), \ldots, \mathsf{Value}(u_d)\end{aligned}$$

and

$$\begin{aligned}\mathsf{Poly}_2 \coloneq\ & \mathsf{Term}(u_1, \ldots, u_d, z_0), \mathsf{Coeff}_\beta(z_0),\\ & \underbrace{\mathsf{Value}(u_1), \ldots, \mathsf{Value}(u_1)}_{d \text{ times}},\\ & \mathsf{Value}(u_1), \ldots, \mathsf{Value}(u_d)\end{aligned}$$

Given Lemma 4, we can prove the key theorem of this section that the evaluation of Poly_1 and Poly_2 over polynomial encoders indeed results in the evaluations of polynomials $P_1(\vec{x})$ and $P_2(\vec{x})$, respectively.

THEOREM 5. *Let $\xi \in \mathbb{N}_0^n$, and let D_ξ be a polynomial encoder. Then* $\textsc{Eval}(\mathsf{Poly}_1, D_\xi) = P_1(\xi)$ *and* $\textsc{Eval}(\mathsf{Poly}_2, D_\xi) = \xi_1^d P_2(\xi)$.

PROOF. Fix a $\xi \in \mathbb{N}_0^n$, and let $D = D_\xi$. First, consider the value of $\textsc{Eval}(\mathsf{Poly}_1, D)$:

$$\begin{aligned}\sum_{\substack{i_1,\ldots,i_d,T_j:\\(i_1,\ldots,i_d)=\mathcal{T}_j}} & |\mathsf{Term}(X_{i_1}, \ldots, X_{i_d}, T_j)| \cdot |\mathsf{Coeff}_\alpha(T_j)|\\ & \cdot |\mathsf{Value}(X_{i_1})| \ldots |\mathsf{Value}(X_{i_d})|\end{aligned}$$

By Lemma 4, the above expression evaluates to $\sum_j \alpha_j \prod_{i \in \mathcal{T}_j} \xi_i = P_1(\xi)$, as we claimed. Applying Lemma 4, a similar proof shows that $\textsc{Eval}(\mathsf{Poly}_2, D) = \sum_j \beta_j \xi_1^d \prod_{i \in \mathcal{T}_j} \xi_i = \xi_1^d P_2(\xi)$, as desired. □

The rest of this subsection is devoted to the proof of Lemma 4. We first give the definitions of the views Term, Value, Coeff_α, and Coeff_β and the polynomial encoders.

Let $\mathsf{R}, \mathsf{R}_1, \ldots, \mathsf{R}_d, \mathsf{S}_0, \ldots, \mathsf{S}_m$ be binary relation names, and define the views as follows. Here, we let $N_k \triangleq \sum_{j=1}^k \alpha_j$ for all $k \in [m]$.

$$\begin{aligned}\mathsf{Term}(\vec{u}, z_0) \coloneq\ & \mathsf{R}_1(u_1, z_0), \ldots, \mathsf{R}_d(u_d, z_0)\\ \mathsf{Value}(v) \coloneq\ & \mathsf{R}(v, v')\\ \mathsf{Coeff}_\alpha(z_0) \coloneq\ & \mathsf{S}_1(z_0, z_1), \ldots, \mathsf{S}_1(z_{N_1-1}, z_{N_1}),\\ & \mathsf{S}_2(z_{N_1}, z_{N_1+1}), \ldots, \mathsf{S}_2(z_{N_2-1}, z_{N_2}),\\ & \vdots\\ & \mathsf{S}_m(z_{N_{m-1}}, z_{N_{m-1}+1}), \ldots\\ & \quad \ldots, \mathsf{S}_m(z_{N_m-1}, z_{N_m}),\\ & S_0(z_{N_m}, z_{N_m})\end{aligned}$$

We define $\mathsf{Coeff}_\beta(z_0)$ in analogy with $\mathsf{Coeff}_\alpha(z_0)$, where we replace each N_k with $M_k \triangleq \sum_{j=1}^k \beta_j$ for all $k \in [m]$.

We are now ready to describe the polynomial encoder D_ξ as the union of two databases D^* and $\overline{D}_\xi$ with disjoint relations. D^* is independent of ξ and is defined below:

$$\begin{aligned}\mathsf{R}_k =& \{(X_i, T_j) \mid i \text{ is the } k\text{-th entry of } \mathcal{T}_j\}, \quad \forall k \in [d]\\ \mathsf{S}_j =& \{(T_j, T_j), (T_{j+1}, T_{j+1}), \ldots, (T_m, T_m)\}\\ & \cup \{(T_j, T_0)\} \cup \{(T_0, T_0)\}, \qquad \forall j \in [m]\\ \mathsf{S}_0 =& \{(T_0, T_0)\}\end{aligned}$$

For each $\xi \in \mathbb{N}_0^n$, we define $\overline{D}_\xi$ as follows:

$$\begin{aligned}\mathsf{R} =& \{(X_1, U_{1,1}), \ldots, (X_1, U_{1,\xi_1}),\\ & \vdots\\ & (X_n, U_{n,1}), \ldots, (X_n, U_{n,\xi_n})\}\end{aligned}$$

Observe that each fact in the polynomial encoder D_ξ has multiplicity 1. Moreover, $D_\xi = D^*$, when $\xi = 0$.

PROOF OF LEMMA 4. For part (1), notice that $R_k(X_i, T_j)$ is a fact of D_ξ if and only if i is the kth entry of $\mathcal{T}_j$, for each $i \in [n]$, $j \in [m]$, $k \in [d]$. Hence, $\textsc{Eval}(\mathsf{Term}, D_\xi)$ returns the tuple $(X_{i_1}, \ldots, X_{i_d}, T_j)$ if and only if $(i_1, \ldots, i_d) = \mathcal{T}_j$. Since all variables in this view are free, each tuple has multiplicity one, as we claimed.

For part (2), since $\mathsf{Value}(v) \coloneq \mathsf{R}(v, v')$, v must map to X_k for some $k \in [n]$. Then, v' can map to exactly one of the ξ_k elements $U_{k,1}, \ldots, U_{k,\xi_k}$. Therefore, the query returns the set $\{X_1, \ldots, X_n\}$ with element X_k having multiplicity precisely ξ_k.

The proof for parts (3) and (4), are similar so we prove only part (3). For every $j \in [m]$, the query Coeff_α has a subgoal $S_j(z_{i-1}, z_i)$ for every i such that $N_{j-1} < i \leq N_j$. Since only facts in D_ξ where T_0 is the first component are $S_j(T_0, T_0)$, for all $j \in [m]$, we have by induction that (*) if z_i is mapped to T_0, then z_k must be mapped to T_0 for all $k > i$. Similarly, the only facts in D_ξ where T_j is the second component are $S_{j'}(T_j, T_j)$, for all $j' \leq j$. Therefore, (**) if z_i is mapped to T_j for some j, then $i \leq N_j$, and z_k is mapped to T_j for all $k < i$.

In D_ξ, the only choices for z_0 are $T_0, T_1, \ldots T_m$. Consider the case when z_0 maps to T_0. By (*), every z_k maps to T_0, and this satisfies all the subgoals so the multiplicity of T_0 is 1.

Next, consider the case when z_0 is mapped to T_j for some $j \in [m]$. To satisfy the subgoal S_0, z_N must map to T_0. In the database $\mathsf{S}_j(T_j, T_0)$ is the only fact in which T_j is paired with any other element. Together with (*) and (**), it follows that the only maps that satisfy the query are such that for some i where $N_{j-1} < i \leq N_j$, z_k is mapped to T_j for all $k < i$, and z_k is mapped to T_j for all $k \geq i$. Each of these $N_j - N_{j-1} = \alpha_j$ choices of i result in z_0 being mapped to T_j, so the multiplicity of T_j is α_j, as desired. □

4.3 The Full Construction

We start by formally defining an *augmented polynomial encoder*. Let D^{sink} be the database on universe $\{C_0, C_1\}$ containing

$$\begin{aligned} \mathsf{R} &= \{(C_0, C_0)\} \\ \mathsf{R}_i &= \{(C_0, C_0), (C_1, C_0)\}, \qquad \forall i \in [d] \\ \mathsf{S}_j &= \{(C_0, C_0), (C_0, C_1)\}, \qquad \forall j \in [m] \\ \mathsf{S}_0 &= \{(C_0, C_0)\} \end{aligned}$$

The class of augmented polynomial encoders are simply $D_\xi^{\text{aug}} = D_\xi \cup D^{\text{sink}}$, for each $\xi \in \mathbb{N}_0^n$.

LEMMA 6. *Let* $\mathsf{Poly}_1, \mathsf{Poly}_2$ *be defined as in the last subsection. For all* $\xi \in \mathbb{N}_0^n$*, we have*

$$\begin{aligned} \text{EVAL}(\mathsf{Poly}_1, D_\xi^{\text{aug}}) &= 1 + P_1(\xi) \\ \text{EVAL}(\mathsf{Poly}_2, D_\xi^{\text{aug}}) &= 1 + \xi_1^d P_2(\xi) \end{aligned}$$

PROOF. First of all, consider $\text{EVAL}(\mathsf{Poly}_1, D^{\text{sink}})$. For the subgoal $\mathsf{Value}(u_i)$, the only fact in D^{sink} involving R is $\mathsf{R}(C_0, C_0)$, therefore each u_i must map to C_0. Now, examining the subgoal $\mathsf{Term}(\vec{u}, z_0)$, it is clear that with each u_i mapping to C_0, z_0 must map to C_0 as well. Further, the subgoal $S_0(z_{N_m}, z_{N_m})$ can only be satisfied by mapping z_{N_m} to C_0. So z_ℓ must map to C_0 as well, for all $\ell \in [N_m]$. Hence, $\text{EVAL}(\mathsf{Poly}_1, D^{\text{sink}})$ returns the tuple $(C_0, \ldots, C_0)$ with multiplicity 1.

To finish the proof, observe that the constants appearing in D_ξ and D^{sink} are disjoint, and the constraints are such that no single map of the variables can use the constants in both databases at the same time. Since $D_\xi^{\text{aug}} = D_\xi \cup D^{\text{sink}}$, we have

$$\begin{aligned} &\text{EVAL}(\mathsf{Poly}_1, D_\xi^{\text{aug}}) \\ &= \text{EVAL}(\mathsf{Poly}_1, D^{\text{sink}}) + \text{EVAL}(\mathsf{Poly}_1, D_\xi) = 1 + P_1(\xi) \end{aligned}$$

The proof showing $\text{EVAL}(\mathsf{Poly}_2, D_\xi^{\text{aug}}) = 1 + \xi_1^d P_2(\xi)$ follows similarly. $\square$

We would like to guarantee that the databases we consider contain a copy of the augmented polynomial encoder $D_0^{\text{aug}} = D^* \cup D^{\text{sink}}$, where D^* is defined in Section 4.2. Recall that the *canonical query* Q_D corresponding to a database D is one where there is a distinct variable x corresponding to a domain element X in the database; the query is a conjunction of subgoals $R(x_1, \ldots, x_k)$ such that $R(X_1, \ldots, X_k)$ is a fact in the database. All variables are free in Q_D. The crucial property is that given any database D', any map for evaluating Q_D on D' induces a copy of D in D'. We define AugDB as the conjunction of two sub-queries (1) the canonical query associated with D_0^{aug}, in which we identify variable x_i with constant X_i for each $i \in [n]$, variable t_j with constant T_j for each $j \in [m]$, and variables c_0, c_1 with constants C_0, C_1 respectively, and (2) inequalities constraints on every pair of distinct variables. All the variables are free in AugDB.

Recall the definitions of a homomorphism τ and the database $\tau(Q)$ for a query Q, which were given at the end of Section 2. Notice that the inequalities of (2) guarantee that if ψ is a homomorphism from AugDB to D, then $\psi(\mathsf{AugDB})$ is isomorphic to D_0^{aug}, since distinct variables are mapped to distinct constants. In particular, this means that if $\text{EVAL}(\mathsf{AugDB}, D) > 0$, then D must contain an isomorphic copy of D_0^{aug}. We restate this in the following lemma

LEMMA 7. *Let D be a database, and let ψ be a homomorphism from* AugDB *to D. Then* $\psi(\mathsf{AugDB})$ *is isomorphic to* D_0^{aug}*. Therefore, if* $\text{EVAL}(\mathsf{AugDB}, D) > 0$*, then D must contain an isomorphic copy of* D_0^{aug}.

Next, we define our queries

$$\begin{aligned} \mathsf{Q}_1 \quad &\text{:-} \quad \mathsf{Poly}_1, \mathsf{AugDB}(\vec{x}, \vec{t}, \vec{c}) \\ \mathsf{Q}_2 \quad &\text{:-} \quad \mathsf{Poly}_2, \mathsf{CounterCheating}(\vec{x}, \vec{t}, \vec{c}) \end{aligned}$$

We are now ready to define cheating more formally.

Let D be a database, and suppose $D' \subseteq D$ (i.e. all facts in D' belong to D) is a database isomorphic to D_0^{aug}. Let ϕ be a homomorphism from Poly_1 to D. Roughly speaking, we think of ϕ as cheating with respect to D' if any of the subgoals—ignoring those subgoals involving Value—are mapped to facts not in D'. More formally, define Q :– $\mathsf{Term}(\vec{u}, z_0), \mathsf{Coeff}_\alpha(z_0)$. Notice that Q is essentially the query Poly_1 with all subgoals involving Value removed. Further, let ϕ' be the restriction of ϕ to the variables in $\mathsf{vars}(\mathsf{Q}) = \{u_1, \ldots, u_d, z_0, \ldots, z_{N_m}\}$. Notice that ϕ' is a homomorphism from Q to D. We say that *ϕ cheats with respect to D'* if ϕ' is not a homomorphism from Q to D'.

Now, for each homomorphism, ψ, from AugDB to D, the database $\psi(\mathsf{AugDB})$ is isomorphic to D_0^{aug}, by Lemma 7. The following key technical lemma guarantees that for each of these $\psi(\mathsf{AugDB})$, the value of $\mathsf{CounterCheating}$ is large enough to counter the number of homomorphisms from Poly_1 to D that cheat with respect to $\psi(\mathsf{AugDB})$. We defer the proof of this, as well as the definition of $\mathsf{CounterCheating}$, to the next subsection.

LEMMA 8. *Let D be a database, and let* Poly_1 *be defined as in the last subsection.*

- *If D is an augmented polynomial encoder, then*

  $$\text{EVAL}(\mathsf{AugDB}, D) = \text{EVAL}(\mathsf{CounterCheating}, D)$$

  Both return only the tuple $(\vec{X}, \vec{T}, \vec{C})$, with multiplicity one.

- *For general D, let ψ be a homomorphism from* AugDB *to D, if one exists. If there are γ homomorphisms from* Poly_1 *to D that cheat with respect to* $\psi(\mathsf{AugDB})$*, then*

  $$|\mathsf{CounterCheating}(\psi(\vec{x}, \vec{t}, \vec{c}))|_D \geq 1 + \gamma$$

With Lemma 6 and Lemma 8 in hand, we are ready for the main result of this paper. Once $\mathsf{CounterCheating}$ is defined, it will be easy to see that the queries Q_1 and Q_2 use at most n^{2d} inequality constraints. Remember here that we only need $n = 59$ and $d = 5$.

THEOREM 9. *For some k bounded by n^{2d}, where $n \leq 59$ and $d \leq 5$, the problem* $\mathsf{ConQC}_{\mathsf{BS}}(k, \infty, \infty)$ *is undecidable.*

PROOF. We will show that $\mathsf{Q}_1 \subseteq_{\mathsf{BS}} \mathsf{Q}_2$ if and only if $P_1(\xi) \leq (\xi_1)^d P_2(\xi)$ for all $\xi \in \mathbb{N}_0^n$. It will thus follow by Theorem 3 that $\mathsf{ConQC}_{\mathsf{BS}}(k, \infty, \infty)$ is undecidable.

First, suppose that there is a $\xi \in \mathbb{N}_0^n$ such that $P_1(\xi) > \xi_1^d P_2(\xi)$. Lemma 6 shows that $\text{EVAL}(\mathsf{Poly}_1, D_\xi^{\text{aug}}) = 1 + P_1(\xi) > 1 + \xi_i^d P_2(\xi) = \text{EVAL}(\mathsf{Poly}_2, D_\xi^{\text{aug}})$. Further, Lemma 8 guarantees that $\text{EVAL}(\mathsf{AugDB}, D_\xi^{\text{aug}}) = \text{EVAL}(\mathsf{CounterCheating}, D_\xi^{\text{aug}})$. Hence, $\text{EVAL}(\mathsf{Q}_1, D_\xi^{\text{aug}}) > \text{EVAL}(\mathsf{Q}_2, D_\xi^{\text{aug}})$, as we wanted.

Now, suppose that $P_1(\xi) \leq \xi_i^d P_2(\xi)$ for all $\xi \in \mathbb{N}_0^n$. Let D be a database, and let Ψ be the set of homomorphisms from AugDB to D. For each $\psi \in \Psi$, let N_ψ be the number of homomorphisms from Poly_1 to D that cheat with respect to ψ. Further, we need to count the number of homomorphisms from Poly_1 to D that do not cheat with respect to ψ.

Recall that in our definition of cheating, we intentionally ignored the view Value, which has the effect of ignoring relation R. But homomorphisms from Poly_1 to D must respect R. So we need a way to "add R back."

To this end, define D^ψ iteratively as follows:

1. Start with D^ψ equal to $\psi(\mathsf{AugDB})$.

2. For each fact of D, check if it is of the form $\mathsf{R}(\psi(x_i), C)$ for some i and some constant C. If it is, then add that fact to D^ψ (with multiplicity one)

It is not hard to see that D^ψ is isomorphic to $D^{\text{aug}}_{\xi_\psi}$ for some ξ_ψ. (In particular, the value of its ith entry, $(\xi_\psi)_i = |\mathsf{Value}(\psi(x_i))|_D$.) It is also not hard to see that if ϕ does not cheat with respect to $\psi(\mathsf{AugDB})$, then ϕ is a homomorphism from Poly_1 to D^ψ. Thus, the number of homomorphisms from Poly_1 to D that do not cheat is

$$\text{EVAL}(\mathsf{Poly}_1, D^\psi) = 1 + P_1(\xi_\psi)$$

by Lemma 6. Hence, we see

$$\text{EVAL}(\mathsf{Q}_1, D) \quad = \quad \sum_{\psi \in \Psi} (1 + P_1(\xi_\psi) + N_\psi)$$

Denoting the first coordinate of ξ_ψ by $\xi_{\psi,1}$, we also see that

$$\begin{aligned}\text{EVAL}(\mathsf{Poly}_2, D^\psi) \quad &\geq \quad 1 + (\xi_{\psi,1})^d P_2(\xi_\psi) \\ &\geq \quad 1 + P_1(\xi_\psi)\end{aligned}$$

Putting this together with Lemma 8 completes the proof:

$$\begin{aligned}&\text{EVAL}(\mathsf{Q}_2, D) \\ &\quad \geq \quad \sum_{\psi \in \Psi} (1 + P_1(\xi_\psi)) \cdot (1 + N_\psi) \\ &\quad \geq \quad \sum_{\psi \in \Psi} (1 + P_1(\xi_\psi) + N_\psi) \\ &\quad = \quad \text{EVAL}(\mathsf{Q}_1, D)\end{aligned}$$

□

Thus, applying Corollary 2 (and counting the number of inequalities in Q_1 and Q_2 a little more carefully), we have the following corollary.

COROLLARY 10. *For some k bounded by n^{2d}, where $n \leq 59$ and $d \leq 5$, we have that both*

$\mathsf{ConQC_B}(k, 1, 2)$ *is undecidable, and*

$\mathsf{ConQC_{BS}}(k, 1, 2)$ *is undecidable.*

4.4 Proof Sketch of Lemma 8

We now sketch the proof for the main technical lemma. We begin by defining $\mathsf{CounterCheating}$. We break this into 8 main components. One of the components is simply AugDB. Each of the other 7 is to counter a different type of cheating that could occur. We outline these 7 ways of cheating below.

Recall that Poly_1 was a bound query. In defining our anti-cheating views, it will be necessary to use inequalities involving the variables of Poly_1. Hence, we define Poly'_1 analogously to Poly_1, but in such a way that many of its variables become free. Specifically, let $\mathsf{Coeff}'_\alpha(\vec{z})$ be defined as the query with the same body as Coeff_α, but with free variables $z_0, \ldots, z_{N_m}$. Then define

$$\begin{aligned}&\mathsf{Poly}'_1(\vec{u}, \vec{z}) \\ &\quad :- \quad \mathsf{Term}(\vec{u}, z_0), \mathsf{Coeff}'_\alpha(\vec{z}), \mathsf{Value}(u_1), \ldots, \mathsf{Value}(u_d)\end{aligned}$$

For all $k \in [d]$ and all $j \in [m]$, we define the following views.

$$\begin{aligned}
&\mathsf{Counter}_1(t_1, \ldots, t_m) :- \\
&\qquad \mathsf{Poly}'_1(\vec{u}, \vec{z}), \wedge_{\ell \in [m]} z_0 \neq t_\ell \\
&\mathsf{Counter}_2(t_0) :- \\
&\qquad \mathsf{Poly}'_1(\vec{u}, \vec{z}), z_{N_m} \neq t_0 \\
&\mathsf{Counter}^k_3(\vec{x}) :- \\
&\qquad \mathsf{Poly}'_1(\vec{u}, \vec{z}), \wedge_{i \in [n]} u_k \neq x_i \\
&\mathsf{Counter}^k_4 :- \\
&\qquad \mathsf{Poly}'_1(\vec{u}, \vec{z}), R_k(w, z_0), w \neq u_k \\
&\mathsf{Counter}^j_5(t_0) :- \\
&\qquad \mathsf{Poly}'_1(\vec{u}, \vec{z}), S_j(z_0, w), w \neq z_0, w \neq t_0 \\
&\mathsf{Counter}^j_6(t_j) :- \\
&\qquad \mathsf{Poly}'_1(\vec{u}, \vec{z}), S_j(z_0, w), w \neq z_0, z_0 \neq t_j \\
&\mathsf{Counter}^j_7(t_0) :- \\
&\qquad \mathsf{Poly}'_1(\vec{u}, \vec{z}), S_j(z_{N_m}, w), w \neq t_0
\end{aligned}$$

Notice that each of the above views contains Poly'_1 as a subgoal. So if it were not for the $\neq$-constraints, each view would return a multiplicity at least as large as Poly_1 since Poly_1 is just Poly'_1 with all variables bound. The $\neq$-constraints guarantee that there is only one homomorphism, unless there are maps from Poly_1 to the database that cheat.

Define $\mathsf{CounterCheating}(\vec{x}, \vec{t}, \vec{c})$ to be the conjunction of the above views and $\mathsf{AugDB}(\vec{x}, \vec{t}, \vec{c})$. Note that the free variables $(\vec{x}, \vec{t}, \vec{c})$ are the same for each view, but the variables $(\vec{u}, \vec{z})$ are local to each view.

We first sketch the proof that if D is an augmented polynomial encoder, then

$$\text{EVAL}(\mathsf{AugDB}, D) = \text{EVAL}(\mathsf{CounterCheating}, D)$$

with both expressions returning only the tuple $(\vec{X}, \vec{T}, \vec{C})$, with multiplicity one. It is not difficult to check that $\text{EVAL}(\mathsf{AugDB}, D)$ returns only the tuple $(\vec{X}, \vec{T}, \vec{C})$ with multiplicity one. Further, since $\mathsf{CounterCheating}$ contains AugDB as a subgoal, we see that evaluating $\mathsf{CounterCheating}$ on D returns only the tuple $(\vec{X}, \vec{T}, \vec{C})$. However, we still need to verify that its multiplicity is one.

We do this by showing that there is exactly one homomorphism from each view $\mathsf{Counter}_i$ to D that maps $(\vec{x}, \vec{t}, \vec{c})$ to $(\vec{X}, \vec{T}, \vec{C})$. Consider the view $\mathsf{Counter}^j_6$ for some j. The proof for each of the other views $\mathsf{Counter}_i$ is similar. Since $\mathsf{Counter}^j_6$ contains $\mathsf{Poly}'_1(\vec{u}, \vec{z})$ as a subgoal, there is precisely one homomorphism that maps $(\vec{u}, \vec{z})$ to $(C_0, \ldots, C_0)$ and w to C_1. In analogy with the proof of Lemma 4, we see that any other homomorphism ϕ must map z_0 to $T_{j'}$ for some j'. Now, if $j' \neq j$, then $\phi(w) = T_{j'}$, contradicting the $\neq$-constraint $w \neq z_0$. If $j' = j$, then $\phi(z_0) = T_j$, contradicting $z_0 \neq t_j$.

We now sketch the proof of the second item. Let D be any database, and let ψ be a homomorphism from AugDB to D. Further, suppose that there are γ homomorphisms from Poly_1 to D that cheat with respect to $\psi(\mathsf{AugDB})$. We will show

$$|\mathsf{CounterCheating}(\psi(\vec{u}, \vec{t}, \vec{c}))|_D \geq 1 + \gamma$$

For convenience and readability, relabel the constants in D so that $\psi(x_i) = X_i$ for all $i \in [n]$, $\psi(t_j) = T_j$ for all $j \in [m]$, and $\psi(c_\ell) = C_\ell$ for $\ell = 0, 1$. Under the relabeling of the constants, $\psi(\mathsf{AugDB})$ is the canonical database D^{aug}_0.

We partition cheating homomorphisms into seven classes. Notice that each class N_i corresponds to the respective $\mathsf{Counter}_i$. Let ϕ_0 be the homomorphism mapping every variable in Q'_1 to C_0, and let Φ

be the set of homomorphisms from Q_1' to D excluding ϕ_0. For each $j \in [m], k \in [d]$, define

1. $N_1 = \{\phi \in \Phi |$ for all $j \in [m]$, $\phi(z_0) \neq T_j\}$.
2. $N_2 = \{\phi \in \Phi | \ \phi(z_{N_m}) \neq T_0\}$.
3. $N_3^k = \{\phi \in \Phi |$ for all $i \in [n]$, $\phi(u_k) \neq X_i\}$.
4. $N_4^k = \{\phi \in \Phi | \ \phi(u_1, \ldots, u_d, t) = (X_{i_1}, \ldots, X_{i_d}, T_{j'})$ for some $j' \in [m]$, but i_k is not the kth entry of $\mathcal{T}_{j'}\}$.
5. $N_5^j = \{\phi \in \Phi \mid \phi(z_{N_m}) = T_0, \exists_\ell$ such that $\mathsf{S}_j(z_\ell, z_{\ell+1})$ is a subgoal of Poly_1, $\exists_{j' \in [m]} \phi(z_0) = \phi(z_\ell) = T_{j'}$, but $\phi(z_{\ell+1})$ is neither $T_{j'}$ nor T_0 $\}$.
6. $N_6^j = \{\phi \in \Phi \mid \phi(z_{N_m}) = T_0, \exists_\ell$ such that $\mathsf{S}_j(z_\ell, z_{\ell+1})$ is a subgoal of Poly_1, $\exists_{j' \in [m]} \phi(z_0) = \phi(z_\ell) = T_{j'}$ with $j' \neq j$, but $\phi(z_{\ell+1}) = T_0$ $\}$.
7. $N_7^j = \{\phi \in \Phi \mid \phi(z_{N_m}) = T_0, \exists_\ell$ such that $\mathsf{S}_j(z_\ell, z_{\ell+1})$ is a subgoal of Poly_1, $\phi(z_\ell) = T_0$, but $\phi(z_{\ell+1}) \neq T_0$ $\}$.

CLAIM 11. *If ϕ is a homomorphism from Poly_1 to D that cheats with respect to $\psi(\mathsf{AugDB})$, then ϕ belongs to $N_1 \cup N_2 \bigcup_{k \in [d]} (N_3^k \cup N_4^k) \bigcup_{j \in [m]} (N_5^j \cup N_6^j \cup N_7^j)$.*

PROOF. Let ϕ be a homomorphism that cheats with respect to $\psi(\mathsf{AugDB})$. Throughout this proof, we assume that $\phi(z_{N_m}) = T_0$, for otherwise $\phi \in N_1$. Likewise, we assume $\phi(z_0) = T_j$ for some j, for otherwise $\phi \in N_2$. Finally, we assume for each k, there is an ℓ such that $\phi(u_k) = X_\ell$, for otherwise $\phi \in N_3^k$.

If ϕ cheats, then some subgoal of Poly_1, not involving relation R, is mapped to a fact that is not in $\psi(\mathsf{AugDB})$. We say such a subgoal is *mapped incorrectly*.

First, suppose subgoal $\mathsf{R}_k(a, b)$ is mapped incorrectly by ϕ. Note that $a = u_k$ and $b = z_0$. By our assumption, $\phi(u_k) = X_\ell$ for some $\ell \in [n]$ and $\phi(z_0) = T_j$ for some j. Hence, $\mathsf{R}_k(X_\ell, T_j)$ is not a fact for $\psi(\mathsf{AugDB})$. So ℓ is not the kth entry in $\mathcal{T}_j$. That is, $\phi \in N_4^k$.

Now, suppose a subgoal involving relation $\mathsf{S}_{k'}$ is mapped incorrectly by ϕ, for some k'. Let ℓ be the smallest index such that, for some $j \in [m]$, $S_j(z_\ell, z_{\ell+1})$ is a subgoal of Poly_1 that is mapped incorrectly by ϕ. Note that $\phi(z_\ell)$ is either T_0 or $T_{j'}$ for some $j' \in [m]$. If it is T_0, then $\phi(z_{\ell+1}) \neq T_0$ since $S_j(z_\ell, z_{\ell+1})$ is mapped incorrectly. Hence, $\phi \in N_7^j$.

If $\phi(z_\ell) = T_{j'}$, then $\phi(z_0) = T_{j'}$ by the minimality of ℓ. Further, either (1) $\phi(z_{\ell+1}) = T_0$, (2) $\phi(z_{\ell+1}) \neq T_0, T_{j'}$, or (3) $\phi(z_{\ell+1}) = T_{j'}$. In the first case, we see $j' \neq j$ since $S_j(z_\ell, z_{\ell+1})$ is mapped incorrectly. Hence, $\phi \in N_6^j$. In the second case, we see $\phi \in N_5^j$. In the third case, let ℓ' be the smallest index such that $\phi(z_{\ell'+1}) \neq T_{j'}$, and let k' be such that $S_{k'}(z_{\ell'}, z_{\ell'+1})$ is a subgoal of Poly_1. Notice that $\phi(z_0) = \phi(z_{\ell'}) = T_{j'}$. Further, notice that $j > j'$ since $S_j(T_{j'}, T_{j'})$ is not a fact for D_ψ, hence $k' \neq j'$. So if $\phi(z_{\ell'+1}) = T_0$, we see $\phi \in N_6^{k'}$. On the other hand, if $\phi(z_{\ell'+1}) \neq T_0$, then $\phi \in N_5^{k'}$. $\square$

Claim 11 thus gives us that $\gamma \leq |N_1| + |N_2| + \sum_{k \in [d]} (|N_3^k| + |N_4^k|) + \sum_{j \in [m]} (|N_5^j| + |N_6^j| + |N_7^j|)$.

Our next claim simply says that each $\mathsf{Counter}$ does its job.

CLAIM 12. *For all $k \in [d], j \in [m]$,*

$$\begin{aligned}
|\mathsf{Counter}_1(T_1, \ldots, T_m)|_D &\geq 1 + |N_1| \\
|\mathsf{Counter}_2(T_0)|_D &\geq 1 + |N_2| \\
|\mathsf{Counter}_3^k(\vec{X})|_D &\geq 1 + |N_3^k| \\
|\mathsf{Counter}_4^k|_D &\geq 1 + |N_4^k| \\
|\mathsf{Counter}_5^j(T_0)|_D &\geq 1 + |N_5^j| \\
|\mathsf{Counter}_6^j(T_j)|_D &\geq 1 + |N_6^j| \\
|\mathsf{Counter}_7^j(T_0)|_D &\geq 1 + |N_7^j|
\end{aligned}$$

PROOF SKETCH. Consider a homomorphism ϕ mapping Poly_1 to D. Then each local variable u_i in Poly_1 gets mapped to $\phi(u_i)$, and likewise, each local variable z_i in Q_1' gets mapped to $\phi(z_i)$. Suppose, e.g., $\phi \in N_1$. Then we may specify a corresponding homomorphism $\hat{\phi}$ from $\mathsf{Counter}_1$ to D such that $\hat{\phi}$ maps each local variable u_i of $\mathsf{Counter}_1$ to $\phi(u_i)$, and likewise, $\hat{\phi}$ maps each local variable z_i of $\mathsf{Counter}_1$ to $\phi(z_i)$. Finally, we set $\hat{\phi}(t_j) = T_j$ for each j. It is not hard to see that $\hat{\phi}$ is indeed a homomorphism, and further, that it respects all the $\neq$-constraints of $\mathsf{Counter}_1$. Hence, for every cheating homomorphism in N_1, there is a corresponding homomorphism from $\mathsf{Counter}_1$ to D. Furthermore, there is a homomorphism from $\mathsf{Counter}_1$ to D in which every variable is mapped to C_0. It thus follows that

$$|\mathsf{Counter}_1(T_1, \ldots, T_m)|_D \geq 1 + |N_1|$$

An analogous proof holds for each of the other cases. We include a proof sketch for the last four inequalities, since in each of those cases, we must also specify where the local variable w is mapped.

4. Let $k \in [d]$, let $\phi \in N_4^k$, and consider where the bound variables of Poly_1 are mapped by ϕ. In particular, $\phi(u_k) = X_i$ for some $i \in [n]$ and $\phi(z_0) = T_j$ for some $j \in [m]$, but i is not the kth entry of $\mathcal{T}_j$. Hence, we may map the corresponding bound variables of $\mathsf{Counter}_4^k$ in precisely the same way without violating any $\neq$-constraints. As for the variable w, we may map it to $X_{i'}$, where i' is the k entry of $\mathcal{T}_j$. Since $\phi(u_k) \neq \phi(w)$, no $\neq$-constraint is violated.

   Furthermore, there is an additional homomorphism mapping from $\mathsf{Counter}_4^k$ to D in which we map w to C_1 and all the other bound variables to C_0, giving an extra homomorphism. Hence,

   $$|\mathsf{Counter}_4^k|_D \geq 1 + |N_4^k|$$

5. Let $j \in [m]$, let $\phi \in N_5^j$, and consider where the bound variables of Poly_1 are mapped by ϕ. In particular, there are variables $z_\ell, z_{\ell+1}$ such that $\mathsf{S}_j(z_\ell, z_{\ell+1})$ is a subgoal of Poly_1, $\phi(z_\ell) = \phi(z_0) = T_{j'}$ for some $j' \in [m]$, but $\phi(z_{\ell+1})$ is mapped to neither T_0 nor $T_{j'}$. We may map the corresponding bound variables of $\mathsf{Counter}_5^j$ in precisely the same way without violating any $\neq$-constraints. As for the variable w, we may map it to $\phi(z_{\ell+1})$. Since $\phi(w) \neq T_0$ and $\phi(w) \neq \phi(z_0)$, no $\neq$-constraint is violated.

   Furthermore, there is an additional homomorphism mapping from $\mathsf{Counter}_5^j$ to D in which we map w to C_1 and all the other bound variables to C_0, giving an extra homomorphism. Hence,

   $$|\mathsf{Counter}_5^j(T_0)|_D \geq 1 + |N_5^j|$$

6. Let $j \in [m]$, let $\phi \in N_6^j$, and consider where the bound variables of Poly_1 are mapped by ϕ. In particular, there are variables $z_\ell, z_{\ell+1}$ such that $\mathsf{S}_j(z_\ell, z_{\ell+1})$ is a subgoal of Poly_1, $\phi(z_\ell) = \phi(z_0) = T_{j'}$ for some $j' \in [m] with j' \neq j$, but $\phi(z_{\ell+1}) = T_0$. We may map the corresponding bound variables of $\mathsf{Counter}_6^j$ in precisely the same way without violating any $\neq$-constraints. As for the variable w, we may map it to T_0, since $S_j(T_{j'}, T_0)$ must be a fact of D. Since $\phi(z_0) \neq T_j$ and $\phi(w) \neq \phi(z_0)$, no $\neq$-constraint is violated.

   Furthermore, there is an additional homomorphism mapping from $\mathsf{Counter}_6^j$ to D in which we map w to C_1 and all the other bound variables to C_0, giving an extra homomorphism. Hence,

   $$|\mathsf{Counter}_6^j(T_j)|_D \geq 1 + |N_6^j|$$

7. Let $j \in [m]$, let $\phi \in N_7^j$, and consider where the bound variables of Poly_1 are mapped by ϕ. In particular, there are variables $z_\ell, z_{\ell+1}$ such that $\mathsf{S}_j(z_\ell, z_{\ell+1})$ is a subgoal of Poly_1, $\phi(z_\ell) = T_0$, but $\phi(z_{\ell+1}) \neq T_0$. We may map the corresponding bound variables of $\mathsf{Counter}_7^j$ in precisely the same way without violating any $\neq$-constraints. As for the variable w, we may map it to $\phi(z_{\ell+1})$. Since $\phi(w) \neq T_0$, no $\neq$-constraint is violated.

   Furthermore, there is an additional homomorphism mapping from $\mathsf{Counter}_7^j$ to D in which we map w to C_1 and all the other bound variables to C_0, giving an extra homomorphism. Hence,

$$|\mathsf{Counter}_7^j(T_0)|_D \geq 1 + |N_7^j|$$

□

To complete the proof:

$$\begin{aligned}
&\textsc{Eval}(\mathsf{CounterCheating}(\psi(\vec{x}, \vec{t}, \vec{c})), D) \\
&\geq \quad (1 + |N_1|)(1 + |N_2|) \prod_{k=1}^{d} [(1 + |N_3^k|)(1 + |N_4^k|)] \\
&\qquad \cdot \prod_{j=1}^{m} [(1 + |N_5^j|)(1 + |N_6^j|)(1 + |N_7^j|)] \\
&\geq \quad 1 + |N_1| + |N_2| + \sum_{k=1}^{d} (|N_3^k| + |N_4^k|) \\
&\qquad + \sum_{j=1}^{m} (|N_5^j| + |N_6^j| + |N_7^j|) \\
&\geq \quad 1 + \gamma
\end{aligned}$$

5. Concluding Remarks

We showed that, under bag semantics, the query containment problem for conjunctive queries with inequalities is undecidable. Actually, even drastic restrictions of this problem are undecidable; specifically, undecidability persists even if the following two restrictions hold at the same time: (1) the queries involve a single binary relation symbol; and (2) the total number of inequalities is bounded by a certain fixed (albeit large) value. Furthermore, the same undecidability results hold under bag-set semantics.

These strong undecidability results reveal that there is no hope of using query containment as a tool to optimize conjunctive queries with inequalities in real database systems. At the same time, these results motivate several different lines of investigation. From a database-theoretic point of view, it would be interesting to identify syntactic or structural conditions that may give rise to classes of conjunctive queries with inequalities for which the containment problem under bag semantics is decidable (or, even better, has low complexity). From a graph-theoretic point of view, note that we made heavy use of directed graphs in the reduction from Hilbert's 10th Problem. Thus, it is natural to ask whether, under bag semantics, the containment problem for conjunctive queries with inequalities is decidable when the queries involve a single binary relation symbol that is interpreted by an undirected graph.

Our work was originally motivated from the decidability question for the conjunctive-query containment problem (without inequalities) under bag semantics. While this question remains unanswered, we hope that the combinatorial tools developed here may turn out to be of use in resolving this long-standing question.

6. References

[1] N. Brisaboa and H. Hernández. Testing bag-containment of conjunctive queries. *Acta Informatica*, 34(7):557–578, 1997.

[2] N. Brisaboa, H. Hernández, J. Paramá, and M. Penabad. Conjunctive queries with built-in predicates with variables and constants over any ordered domain. In *ABDIS Research Communications*, pages 46–57. 1998.

[3] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proc. 9th ACM Symp. on Theory of Computing*, pages 77–90, 1977.

[4] S. Chaudhuri and M. Vardi. Optimization of *real* conjunctive queries. In *Proceedings of the 12th ACM Symposium on the Principles of Database Systems*, pages 59–70, 1993.

[5] S. Cohen, W. Nutt, and A. Serebrenik. Rewriting aggregate queries using views. In *Proc. 18th ACM Symp. on Principles of Database Systems*, pages 155–166, 1999.

[6] Y. Ioannidis and R. Ramakrishnan. Containtment of conjunctive queries: beyond relations as sets. *ACM Transactions on Database Systems*, 20(3):288–324, September 1995.

[7] A. Klug. On conjunctive queries containing inequalities. *Journal of the Association for Computing Machinery*, 35(1):146–160, January 1988.

[8] P. Kolaitis, D. Martin, and M. Thakur. On the complexity of the containment problem for conjunctive queries with built-in predicates. In *Proceedings of the 17th ACM Symposium on the Principles of Database Systems*, pages 197–204, 1998.

[9] Y. Matiyasevich. *Hilbert's 10th Problem*. MIT Press, 1993.

[10] W. Nutt, Y. Sagiv, and S. Shurin. Deciding equivalences among aggregate queries. In *Proc. 17th ACM Symp. on Principles of Database Systems*, pages 214–223, 1998.

[11] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[12] M. Penabad, N. Brisaboa, H. Hernández, and J. Paramá. A general procedure to check conjunctive query containment. *Acta Informatica*, 38(7):489–529, 2002.

[13] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *Journal of the Association for Computing Machinery*, 27(4):633–655, October 1980.

[14] R. van der Meyden. The complexity of querying infinite data about linearly ordered domains. *Journal of Computer and System Sciences*, 54(1):113–135, Feb. 1997.

[15] E. Vee. The equivalence problem for *real* conjunctive queries with inequalities. Submitted, 2006.

[16] E. Vee. A note on "Testing bag-containment of conjunctive queries". Submitted to *Acta Informatica*, 2006.