D W Q

Foundations of **D**ata **W**arehouse **Q**uality

National Technical University of Athens (NTUA)
Informatik V & Lehr- und Forschungsgebiet Theoretische Informatik (RWTH)
Institute National de Recherche en Informatique et en Automatique (INRIA)
Deutsche Forschungszentrum für künstliche Intelligenz (DFKI)
University of Rome «La Sapienza» (Uniroma)
Istituto per la Ricerca Scientifica e Tecnologica (IRST)

W. Nutt, Y. Sagiv, S. Shurin

Deciding Equivalences among Aggregate Queries

Proc. 17th Symmposium on Principles of Database Systems (PODS)

Seattle, Washington, June 1998.

DWQ : ESPRIT Long Term Research Project, No 22469
Contact Person : Prof. Yannis Vassiliou, National Technical University of Athens,
15773 Zographou, GREECE Tel +30-1-772-2526 FAX: +30-1-772-2527, e-mail: yv@cs.ntua.gr

Deciding Equivalences among Aggregate Queries

Werner Nutt*
German Research Center for
Artificial Intelligence GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
Werner.Nutt@dfki.de

Yehoshua Sagiv
Institute for Computer Science
The Hebrew University
Jerusalem 91904, Israel
sagiv@cs.huji.ac.il

Sara Shurin
Institute for Computer Science
The Hebrew University
Jerusalem 91904, Israel
sarina@cs.huji.ac.il

Abstract

Equivalence of aggregate queries is investigated for the class of conjunctive queries with comparisons and the aggregate operators min, max, count, count-distinct, and sum. Essentially, this class contains all unnested SQL queries with the above aggregate operators, with a `WHERE` clause consisting of a conjunction of comparisons, and without a `HAVING` clause. The comparisons can be interpreted over either a dense order (e.g., over the rationals) or a discrete order (e.g., over the integers). Generally, however, different techniques and characterizations are needed in each of these two cases. For queries with either max or min, equivalence is characterized in terms of dominance mappings, which can be viewed as a generalization of containment mappings. For queries with the count-distinct operator, a sufficient condition for equivalence is given in terms of equivalence of conjunctive queries under set semantics. For some special cases, it is shown that this condition is also necessary. For conjunctive queries with comparisons but without aggregation, equivalence under bag-set semantics is characterized in terms of isomorphism. This characterization essentially remains the same also for queries with the count operator. Moreover, this characterization also applies to queries with the sum operator if the queries have either constants or comparisons, but not both. In the general case (i.e., both comparisons and constants), the characterization of the equivalence of queries with the sum operator is more elaborate. All the characterizations given in the paper are decidable with polynomial space. Finally, it is shown that all the characterizations for min-, max-, count-, and sum-queries yield polynomial-time algorithms for linear queries, i.e., queries with no repeated predicates in their bodies.

1 Introduction

The emergence of data warehouses and decision-support systems has highlighted the importance of efficiently processing aggregate queries. Towards this end, it is essential to develop algorithms for two major problems. One is optimizing aggregate queries. The other is using materialized views in the evaluation of those queries. Considerable work has recently been done on these topics (e.g., [CKPS95, GHQ95, DJLS96]), but a coherent understanding of the underlying principles is still lacking. Therefore, most algorithms and techniques are based on sufficient conditions, and complete characterizations have been found only for very restricted cases. A better understanding of these problems requires a complete characterization of equivalences among aggregate queries. In this paper, we provide for the first time algorithms for deciding equivalence that apply to a large class of aggregate queries.

The class of queries investigated in this paper consists of conjunctive queries with comparisons and aggregations. Essentially, this class contains all unnested SQL queries with aggregations, with a `WHERE` clause consisting of a conjunction of comparisons, and without a `HAVING` clause. The comparisons can be interpreted as either a dense order (e.g., over the rationals) or a discrete order (e.g., over the integers). Generally, however, different techniques and characterizations are needed in each of these two cases.

For the purpose of deciding equivalence, we show that a given query can be decomposed into several queries, such that each query has a single aggregation. Therefore, we deal separately with each aggregate operator, and we name types of queries according to that operator. Specifically, we consider max-queries, min-queries, sum-queries, count-queries, and cntd-queries (where "cntd" means "count-distinct"). In this paper, we do not consider queries with the average operator.

Traditionally, equivalence of conjunctive queries (without aggregations) has mostly been investigated under set semantics, which means that both the operands of a given query and the result are sets. Equivalence of this type is called *set equivalence*. However, for a general treatment of aggregate operators, it is necessary to consider bag-set semantics. Under that semantics, the operands of a query are sets, but the result is a bag. In [CV93], equivalence of conjunctive queries (with neither comparisons nor constants) under bag-set semantics, called *bag-set equivalence*, was characterized in terms of isomorphism. That result also applies to count-queries, since equivalence of count-queries is essentially the same as bag-set equivalence of non-aggregate queries. We provide a proof of this result that also allows for constants in the query. The proof is based on an analysis of counting functions that count how often a particular tuple is returned over different databases. Moreover, we extend the result also to conjunctive queries with comparisons, using a

*Currently at the Institute of Computer Science, The Hebrew University, Jerusalem 91904, Israel

different proof technique.

For max-queries and min-queries, we characterize equivalence in terms of dominance mappings, which can be viewed as a generalization of containment mappings. For queries with the count-distinct operator, we provide a sufficient condition for equivalence in terms of equivalence of conjunctive queries under set semantics (i.e., set equivalence). For some special cases, we show that this condition is also necessary.

Equivalence of count-queries is essentially the same as equivalence of conjunctive queries with bag-set semantics. For sum-queries, bag-set equivalence is a sufficient condition for equivalence. However, a complete characterization of the equivalence of sum-queries is more elaborate than bag-set equivalence, and we break it into several subcases. For sum-queries without constants (but with comparisons involving only variables), equivalence can be characterized in terms of bag-set equivalence. Furthermore, the proof technique for this case also applies in the presence of integrity constraints. For the general case of sum-queries (i.e., both constants and comparisons), a different proof technique and characterization are needed. As a special corollary, a simplified characterization, in terms of isomorphism, is obtained for sum-queries with constants, but without comparisons.

We also consider the special case of *linear* (conjunctive) queries, i.e., queries having in their body only a single occurrence of each predicate. For linear queries with neither comparisons nor aggregations, set equivalence is known to be decidable in polynomial time. We show that for linear queries, bag-set equivalence as well as equivalence of max-queries, min-queries, count-queries and sum-queries are all in polynomial time. (For count-distinct, a polynomial-time sufficient condition is presented.) Since linear queries are very common in practical applications, this result is quite important from a practical point of view.

2 Preliminaries

2.1 Conjunctive Queries

A *term* is either a variable or a constant. We denote variables as w, x, y and z, constants as d, e, and terms as s, t, u and v. We denote predicates as p, q and r. A *relational atom* has the form $p(s_1, \ldots, s_k)$, where p is a predicate of arity k. Sometimes we write $p(\bar{s})$, where $\bar{s}$ denotes the terms $s_1, \ldots, s_k$.

The *ordering predicates* are $<$, $\leq$, $>$, and $\geq$.[1] They are interpreted over numbers and strings. An *ordering atom* or *comparison* has the form $s_1 \ \rho \ s_2$, where ρ is an ordering predicate. The comparisons $s_1 < s_2$ and $s_1 > s_2$ are *strict,* while $s_1 \leq s_2$ and $s_1 \geq s_2$ are *non-strict.*

If C and C' are conjunctions of comparisons over an ordered type, we write $C \models C'$ if C' is a *consequence* of C. Note, that for different types, the consequences of C may be different. For instance, over the integers $x > 0 \models x \geq 1$, which is not true over the rationals.

An *atom* is either a relational atom or a comparison. Atoms are denoted as a, b, etc. A *conjunctive query* is an expression of the form

$$q(s_1, \ldots, s_k) \leftarrow a_1 \ \& \ \cdots \ \& \ a_n. \tag{1}$$

The atom $q(s_1, \ldots, s_k)$ is called the *head* of the query, and the atoms $a_1, \ldots, a_n$ appear in the *body* and can be relational or comparisons. The arguments can be variables as well as constants. If the body contains no comparisons, then the query is *relational.* We abbreviate a query as

$$q(\bar{s}) \leftarrow B(\bar{w}), \tag{2}$$

where $B(\bar{w})$ stands for the body of the query and $\bar{w}$ for the variables occurring in the body. Similarly, we may write a conjunctive query as $q(\bar{s}) \leftarrow R(\bar{w}) \ \& \ C(\bar{z})$, in case we want to distinguish between the relational atoms and the comparisons in the body. We may also write a conjunctive query as $q(\bar{s}) \leftarrow R \ \& \ C$ if there is no need to indicate explicitly the variables in the body. The terms in $\bar{s}$ are called *distinguished terms.* The variables appearing only in the body are called *nondistinguished variables.*

By abuse of notation, we will often refer to a query by its head $q(\bar{s})$ or simply by the predicate of its head q.

2.2 Semantics of Conjunctive Queries

An *assignment* γ is a mapping of the terms appearing in a given conjunctive query to constants, such that each constant is mapped to itself. Assignments are naturally extended to tuples and atoms.

Under *set semantics,* a conjunctive query $q(\bar{s}) \leftarrow B(\bar{w})$ defines a new relation $q^{\mathcal{D}}$, for a given database $\mathcal{D}$, as follows:

$$q^{\mathcal{D}} \ := \ \{\gamma(\bar{s}) \mid \gamma \text{ is an assignment that satisfies } B(\bar{w}) \text{ w.r.t. } \mathcal{D}\}.$$

Thus, $q^{\mathcal{D}}$ is obtained by restricting the assignments satisfying the body of the query to the tuple of variables appearing in the head.

Under *bag-set semantics,* a conjunctive query $q(\bar{s}) \leftarrow B(\bar{w})$ defines a multiset $\{\!\{q\}\!\}^{\mathcal{D}}$ of tuples for a given database $\mathcal{D}$. The bag $\{\!\{q\}\!\}^{\mathcal{D}}$ contains the same tuples as the relation $q^{\mathcal{D}}$, but each tuple $\gamma(\bar{s})$ occurs as many times as there are assignments γ' that satisfy $B(\bar{w})$ and agree with γ on $\bar{s}$. Letting $\{\!\{\cdot\}\!\}$ denote a multiset, we formally define $\{\!\{q\}\!\}^{\mathcal{D}}$ in analogy to $q^{\mathcal{D}}$ as:

$$\{\!\{q\}\!\}^{\mathcal{D}} \ := \ \{\!\{\gamma(\bar{s}) \mid \gamma \text{ satisfies } B(\bar{w})\}\!\}.$$

A query q is *contained* in a query q' under set-semantics if $q^{\mathcal{D}} \subseteq q'^{\mathcal{D}}$ for all databases $\mathcal{D}$.

Two queries q and q' are *equivalent* under set-semantics, or *set-equivalent*, if for every database, they return the same set as a of result. Obviously, two queries are set-equivalent if and only if they contain each other.

Similarly, q and q' are *equivalent* under bag-set-semantics, or *bag-set-equivalent*, if for every database, they return the same result with the same multiplicities, that is, $\{\!\{q\}\!\}^{\mathcal{D}} = \{\!\{q'\}\!\}^{\mathcal{D}}$ for all databases $\mathcal{D}$.

2.3 Homomorphisms between Conjunctive Queries

Let $q(\bar{s}) \leftarrow R \ \& \ C$ and $q'(\bar{s}') \leftarrow R' \ \& \ C'$ be conjunctive queries. A *homomorphism* from q' to q is a substitution θ of the variables of q' with terms of q, such that

1. $\theta\bar{s}' = \bar{s}$;
2. $p(\theta\bar{t}')$ is in R for every relational atom $p(\bar{t}')$ of R';
3. $C \models \theta(s') \ \rho \ \theta(t')$ for every comparison $s' \ \rho \ t'$ in C'.

[1] We will use the notation $s = t$ as abbreviation for the conjunction $s \leq t \ \& \ t \leq s$.

A homomorphism is an *isomorphism* if it maps variables to variables, is bijective, and its inverse is also a homomorphism. The queries q' and q are *isomorphic* if there is an isomorphism from q' to q.[2] Thus isomorphic queries are identical up to a renaming of the nondistinguished variables and up to the multiplicity of atoms.

A substitution θ is a *relational homomorphism* from q' to q if it is a *homomorphism* from $q'(\bar{s}') \leftarrow R'$ to $q(\bar{s}) \leftarrow R$, i.e., if it satisfies Conditions (1) and (2) in the definition of homomorphism. For relational queries, the existence of a homomorphism from q' to q is a necessary and sufficient condition for the containment of q in q'.

2.4 Linearizations and Linear Expansions

Generally, the comparisons in the body of a give query induce a partial order among the terms of the query. In order to deal with containment and equivalence of arbitrary queries, which may have comparisons, this partial order should be extended to a linear order. In this section, we describe how to create such linearizations.

Let $T = D \cup W$ be a set of terms, where D is a set of constants and W is a set of variables. A *linearization* of T is a set of comparisons L over the terms in T, such that for any s, $t \in T$, the set L implies exactly one of $s < t$, $s = t$, or $s > t$.

Thus, a linearization partitions the terms into equivalence classes, such that the terms in each class are equal and the classes are arranged in a strict linear order. In each class of L, there is at most one constant. Otherwise, L would be unsatisfiable and entail any consequence.

Given a conjunction of comparisons C and a set of terms T that contains all terms appearing in C, we only consider linearizations that are *compatible* with C, that is, $L \cup C$ is satisfiable.

Example 2.1 Let C be the following conjunction of comparisons:

$$0 < z_1 \ \& \ z_1 < z_2 \ \& \ z_1 < 3.$$

The set of terms T appearing in C is $\{0, z_1, z_2, 3\}$. There are three linearizations of T that are compatible with C:

$$\{0 < z_1 < 3 < z_2\}$$
$$\{0 < z_1 < 3 = z_2\}$$
$$\{0 < z_1 < z_2 < 3\}.$$

Let $q(\bar{s}) \leftarrow R \ \& \ C$ be a conjunctive query. Let D be a set of constants that contains the constants of q, and let W be the set of variables occurring in q. Moreover, let L be a linearization of $D \cup W$ that is compatible with C. The substitution ϕ is *canonical* for L if it maps all elements in an equivalence class of L to one term of that class, and if an equivalence class contains a constant, then it maps the class to that constant. The query

$$q_L(\phi(\bar{s})) \leftarrow \phi R \ \& \ \phi L$$

that is obtained from q by first replacing C with L and then eliminating all equalities, by applying a canonical substitution ϕ, is a *linearization* of q w.r.t. L. Note that ϕ is a homomorphism from q to q_L. With $\mathcal{L}_D(q)$ we denote the set of all linearizations of $D \cup W$ that are compatible with the comparisons of q.

[2]Note that our definition of isomorphism of queries does not depend on the multiplicity with which an atom occurs in a query.

A *linear expansion* of q over D is a family of queries $(q_L)_{L \in \mathcal{L}_D(q)}$, where each q_L is a linearization of q w.r.t. L. If q and D are clear from the context, we simply write $(q_L)_L$. Let $(q_L)_L$ and $(q'_M)_M$ be the linear expansions of q and q' over D, respectively. We say that $(q_L)_L$ and $(q'_M)_M$ are *isomorphic* if there is a bijection $\mu: \mathcal{L}_D(q) \to \mathcal{L}_D(q')$, such that q_L and $q'_{\mu(L)}$ are isomorphic for all $L \in \mathcal{L}_D(q)$.

For a given query q, there is no unique linear expansion over a set D, because the canonical substitutions that produce the linearizations q_L are in general not uniquely determined. However, it is easy to see that any two such linear expansions are isomorphic. Furthermore, the linear expansion of a query depends on whether the comparisons are interpreted over the integers or over the rationals. Generally, the linear expansion over the integers is *not* isomorphic to the linear expansion over the rationals.

For set semantics, a query q' contains a query q if and only if there is a homomorphism from q' to every q_L in $(q_L)_{L \in \mathcal{L}_D(q)}$. In this paper, we will show how this condition is generalized to bag-set semantics and to aggregate queries.

2.5 Reduced Queries

Our techniques for deciding equivalences between count-distinct and sum-queries apply to queries in a particular normal form. We say that a query $q(\bar{x}) \leftarrow R \ \& \ C$ is *reduced* if

- there are no variables x, y occurring in C such that $C \models x = y$;
- there is no variable x occurring in C such that $C \models x = d$ for a constant d.

Proposition 2.2 *For every conjunctive query one can compute in polynomial time an equivalent reduced conjunctive query.*

2.6 Virtual Constants

Consider Example 2.1. If the comparisons are interpreted over the integers, then in one of the three linearizations, namely $\{0 < z_1 < z_2 < 3\}$, the two variables are actually equal to constants that do not appear explicitly in C, that is, $z_1 = 1$ and $z_2 = 2$.

Formally, suppose that comparisons are interpreted over the integers. Let C be a set of comparisons, W be the set of variables of C, and D be a set of constants comprising the constants of C. We say that d is a *virtual constant* of C w.r.t. D if there is a linearization L of $D \cup W$ that is compatible with C, such that $L \models s = d$ for some term $s \in D \cup W$. We denote the set of virtual constants of C w.r.t. D as $vc_C(D)$. Note that $D \subseteq vc_C(D)$, but $D = vc_C(D)$ is not necessarily true.

The following Lemma shows that virtual constants are located between two elements d^-, d^+ of D if the space between d^- and d^+ can consistently be filled with a strict chain of variables of C.

Lemma 2.3 (Filling Up Spaces) *Let C be a set of comparisons over the integers and D be a set of constants comprising the constants of C. Then d is a virtual constant of C w.r.t. D if and only if $d \in D$, or there are d^-, $d^+ \in D$ and $k := d^+ - d^- + 1$ variables $w_1, \ldots, w_m \in W$ such that*

- *$d^- < d < d^+$, and*
- *$\{d^- < w_1, w_1 < w_2, \ldots, w_k < d^+\} \cup C$ is satisfiable.*

One may wonder, whether and when the process of adding virtual constants to a set D terminates. The following lemma shows that we just have to add them once, and after that no new virtual constants come into existence.

Lemma 2.4 (Adding Virtual Constants is a Closure Operation) *Let C be a set of comparisons over the integers and D be a set of constants comprising the constants of C. Then we have*

1. $D \subseteq vc_C(D)$
2. $vc_C(D) = vc_C(vc_C(D))$.

Computing the virtual constants of two sets of comparisons can be done by computing the virtual constants independently for each set.

Lemma 2.5 (Two Sets of Comparisons) *Let C, C' be two sets of comparisons over the integers and D be a set of constants comprising the constants of C and C'. Then we have*

1. $vc_C(vc_{C'}(D)) = vc_{C'}(vc_C(D)) = vc_C(D) \cup vc_{C'}(D)$
2. $vc_C(vc_C(D) \cup vc_{C'}(D)) = vc_{C'}(vc_C(D) \cup vc_{C'}(D)) = vc_C(D) \cup vc_{C'}(D)$.

2.7 Reduced Linear Expansions

Let $q(\bar{x}) \leftarrow R \,\&\, C$ be a conjunctive query. We say that a linear expansion $(q_L)_{L \in \mathcal{L}_D(q)}$ of q over D is *reduced* if every query q_L is reduced. A sufficient criterion for a linear expansion to be reduced is that it be taken over a set of constants D such that D contains all virtual constants of the comparisons of q w.r.t. D.

Lemma 2.6 (Reduced Linear Expansions) *Let $q(\bar{x})$ be a conjunctive query, D be a set of constants that contains the constants of q, and $(q_L)_{L \in \mathcal{L}_D(q)}$ be a linear expansion of q over D. If D contains all virtual constants of C w.r.t. D, that is, if $vc_C(D) = D$, then the linear expansion $(q_L)_{L \in \mathcal{L}_D(q)}$ is reduced.*

3 Aggregate Queries

We give an abstract account of aggregate queries that are definable in SQL without using the `having` construct.

3.1 Syntax of Aggregate Queries

We consider the *aggregation functions* *min*, *max*, *count*, *cntd*, *sum*. The function *cntd* is read as "count distinct." An *aggregate term* has one of the forms

$$min(y),\ max(y),\ cntd(y),\ count,\ sum(y).$$

Observe that *count* does not take an argument. Aggregate terms are denoted as $\alpha(y)$.

We consider aggregate queries with a single aggregate in the head. In the full version, we show that equivalence of queries with several aggregate functions in the head can be reduced to equivalence of queries with a single aggregation in the head. Formally, an *aggregate query* has the form

$$q(x_1, \ldots, x_k, \alpha(y)) \leftarrow B(\bar{s}), \tag{3}$$

where

- $x_1, \ldots, x_k$ are distinct variables, called the *grouping variables*;
- the variable y, called *aggregation variable*, is different from $x_1, \ldots, x_k$;
- each of the variables x_i and y occurs in the body $B(\bar{s})$ of the query.

Similarly to conjunctive queries, we distinguish between relational aggregate queries and arbitrary aggregate queries, which may contain comparisons in their body.

3.2 Semantics of Aggregate Queries

Over a database $\mathcal{D}$, an aggregate query $q(\bar{x}, \alpha(y)) \leftarrow B(\bar{s})$ yields a new relation $q^{\mathcal{D}}$. To define the relation $q^{\mathcal{D}}$, we proceed in three steps.

First, we partition the set of assignments satisfying the body $B(\bar{s})$ of the query into equivalence classes. Two assignments γ_1, γ_2 are equivalent, $\gamma_1 \sim_q \gamma_2$, if they agree on $\bar{x}$, that is, $\gamma_1(\bar{x}) = \gamma_2(\bar{x})$. We denote the equivalence class of γ under this relation as $[\gamma]_q$. Obviously, the class $[\gamma]_q$ is uniquely determined by the tuple $\bar{d} := \gamma(\bar{x})$. Therefore, we will refer to the class $[\gamma]_q$ also as the *group* of $\bar{d}$. If q is clear from the context, we drop the subscript.

Next, we define how to evaluate an aggregate term $\alpha(y)$ on a class of assignments $[\gamma]$, written $\alpha(y).[\gamma]$:

$$\begin{aligned}
max(y).[\gamma] &:= \max_{\gamma' \in [\gamma]} \gamma'(y) \\
min(y).[\gamma] &:= \min_{\gamma' \in [\gamma]} \gamma'(y) \\
cntd(y).[\gamma] &:= |\{\gamma'(y) \mid \gamma' \in [\gamma]\}| \\
count.[\gamma] &:= |[\gamma]| \\
sum(y).[\gamma] &:= \sum_{\gamma' \in [\gamma]} \gamma'(y).
\end{aligned}$$

Here, by $|S|$ we denote the cardinality of a set S. Note that the function $cntd(y)$ returns the number of distinct values to which y is bound.

In order for the above definitions to be sensible we assume that all assignments are suitably typed so that maxima and minima are taken over an ordered set of values and sum and average are taken over numbers. Obviously, well-typedness of assignments can be guaranteed if database relations and queries are well-typed. Since it is obvious how to define and test such well-typedness, we do not dwell on this issue in this paper, but simply take well-typedness for granted.

Finally, the query $q(\bar{x}, \alpha(y)) \leftarrow B(\bar{s})$ is evaluated on $\mathcal{D}$ by first partitioning the assignments satisfying the body into equivalence classes and then, for each class $[\gamma]_q$, concatenating the characteristic tuple of values $\gamma(\bar{x})$ with the evaluation of $\alpha(y)$ on $[\gamma]_q$. Formally,

$$q^{\mathcal{D}} := \left\{ (\gamma(\bar{x}),\ \alpha(y).[\gamma]_q) \;\middle|\; \gamma \text{ satisfies } B(\bar{s}) \right\},$$

where $(\gamma(\bar{x}),\ \alpha(y).[\gamma]_q)$ denotes the concatenation of the tuple $\gamma(\bar{x})$ and the number $\alpha(y).[\gamma]_q$.

3.3 Equivalence of Aggregate Queries

Two queries are *compatible* if they have identical heads. We will discuss equivalence of compatible aggregate queries of

the following form.

$$\begin{aligned} q(\bar{x}, \alpha(y)) &\leftarrow B(\bar{s}). \\ q'(\bar{x}, \alpha(y)) &\leftarrow B'(\bar{w}'). \end{aligned}$$

We say that q and q' are *equivalent* if for every database $\mathcal{D}$, the two queries define the same relation, i.e., $q^{\mathcal{D}} = q'^{\mathcal{D}}$.

There are cases where two queries are equivalent, even if they contain different aggregation functions. However, we do not discuss such pathological cases.

Our approach for studying the equivalence of compatible aggregate queries will be to associate to every simple aggregate query a conjunctive query and to reduce the equivalence of aggregate queries to related properties of the associated conjunctive queries.

If $q(\bar{x}, \alpha(y)) \leftarrow B(\bar{s})$ is an aggregate query with aggregate function *max*, *min*, *cntd*, or *sum*, then the *core* of q is the conjunctive query

$$breve{q}(\bar{x}, y) \leftarrow B(\bar{s}). \quad (4)$$

The core of a query $q(\bar{x}, count) \leftarrow B(\bar{s})$ is the conjunctive query

$$breve{q}(\bar{x}) \leftarrow B(\bar{s}). \quad (5)$$

Observe that for aggregation functions with an argument, the argument appears in the head of the core, and that for *count*, which does not have an argument, the head of the core contains only the grouping variables.

4 Max-Queries

In this section, we show that equivalence of max-queries of the form $q(\bar{x}, max(y)) \leftarrow B(\bar{w})$ can be reduced to related properties of their cores $breve{q}(\bar{x}, y) \leftarrow B(\bar{w})$. All the results for *max* can easily be translated into results for *min*.

Let $q(\bar{x}, y)$ and $q'(\bar{x}, y)$ be two conjunctive queries. We say that q *is dominated* by q' if for every database, whenever q returns a tuple $(\bar{d}, d)$, then q' returns a tuple $(\bar{d}, d')$ with $d' \geq d$. We say that q and q' *dominate each other* if q is dominated by q' and q' is dominated by q.

Proposition 4.1 (Equivalence and Dominance) *Two* max*-queries are equivalent if and only if their cores dominate each other.*

Obviously, if a conjunctive query q is contained in another conjunctive query q', then q is dominated by q'. If the queries are relational, then the converse also holds.

Proposition 4.2 (Dominance and Containment) *Let q and q' be relational conjunctive queries. Then q is dominated by q' if and only if q is contained in q'.*

It is not true, however, that dominance implies containment if queries may contain built-in predicates.

Example 4.3 Consider the queries

$$\begin{aligned} q(y) &\leftarrow p(y) \;\&\; p(z_1) \;\&\; p(z_2) \;\&\; z_1 < z_2 \\ q'(y) &\leftarrow p(y) \;\&\; p(z) \;\&\; z < y. \end{aligned}$$

Both queries return answers if there are at least two elements in p. If this is the case, then q returns all elements of p, while p returns all elements but the least. Thus, the two queries dominate each other. However, they are not set-equivalent, since q contains q', but q' does not contain q.

In analogy to containment, we check dominance by means of dominance mappings. Consider two conjunctive queries $q(\bar{x}, y) \leftarrow B$ and $q'(\bar{x}, y) \leftarrow B'$, which are the cores of compatible max-queries. Let D be the set of constants that appear in either q or q'. Consider a linearization $L \in \mathcal{L}_D(q)$, and let $q_L(\bar{s}, t) \leftarrow B_L$ be the linearization of q w.r.t. L. A *dominance mapping* from q' to q_L is a substitution θ of the variables of q' with terms of q, such that

1. $\theta\bar{x} = \bar{s}$;
2. for every relational atom $r(\bar{u}')$ in B', $r(\theta\bar{u}')$ is an atom of B_L;
3. for every comparison $u' \;\rho\; v'$ in B', the comparison $\theta u' \;\rho\; \theta v'$ is implied by L;
4. L implies $t \leq \theta y$.

Theorem 4.4 (Dominance with Comparisons) *Let q and q' be two conjunctive queries with comparisons, where q and q' are the cores of compatible max-queries. Let D be the set of constants that appear in either q or q'. Then q is dominated by q' if and only if for every linearization $L \in \mathcal{L}_D(q)$, there exists a dominance mapping from q' to the linearization q_L of q w.r.t. L.*

Now we are able to give a precise characterization of the complexity of the dominance problem, using earlier results for relational queries [CM77, ASU79] and for conjunctive queries with comparisons [vdM92].

Theorem 4.5 (Complexity of Dominance)

1. *Dominance of conjunctive queries is Π_2^{P}-complete.*
2. *Dominance of relational conjunctive queries is* NP*-complete.*

5 Count-Distinct-Queries

We show that for queries with the aggregation function *cntd*, equivalence of the cores under set-semantics is a sufficient condition and we give criteria for when it is also a necessary condition.

We remind the reader that the core of a *cntd*-query $q(\bar{x}, cntd(y)) \leftarrow B(\bar{w})$ is the query $breve{q}(\bar{x}, y) \leftarrow B(\bar{w})$. If the cores of two *cntd*-queries are equivalent under set-semantics, then they return the same values for corresponding groups. Thus, in particular, they return they same number of distinct values. This gives us a necessary condition for the equivalence of *cntd*-queries.

Proposition 5.1 (Sufficiency of Set-Equivalence) *Two cntd-queries are equivalent if their cores are equivalent under set-semantics.*

The following example shows that in general, we cannot expect the converse of Proposition 5.1 to be correct.

Example 5.2 Consider the two queries

$$\begin{aligned} q(cntd(y)) &\leftarrow p(y) \;\&\; y = 1 \;\&\; p(2) \\ q'(cntd(y)) &\leftarrow p(y) \;\&\; y = 2 \;\&\; p(1). \end{aligned}$$

Then both queries return a result if and only if the database contains the atoms $p(1)$, and $p(2)$. Moreover, both queries return the count 1, but the first query obtains it, because its

core returns the value 1, while the second obtains it because its core returns the value 2.

Obviously, the two queries are not reduced. Thus, the example shows that only for reduced *cntd*-queries we can expect set-equivalence of the cores to be a necessary condition.

In addition to reducedness, we need other conditions. If p is an n-ary predicate and $\bar{s} = (s_1, \ldots, s_n)$ is an n-tuple of terms, then a *position* in the atom $p(\bar{s})$ is a number $j \in 1..n$, and we say that s_j is the term in position j. Let s, t be terms occurring in the body of a query q. We say that s occurs in a *similar position* as t in q if there are atoms $p(\bar{s})$ and $p(\bar{t})$ in the body and a number j such that s is in position j in $p(\bar{s})$ and t is in position j in $p(\bar{t})$.

Theorem 5.3 (Reduction to Set-Semantics) *Let $q(\bar{x}, cntd(y))$ and $q'(\bar{x}, cntd(y))$ be two reduced cntd-queries where no variable in a similar position as y occurs in a strict comparison. Suppose that one of the following conditions holds:*

1. *q and q' range over the rationals;*
2. *q and q' do not contain constants;*
3. *in q and q', no variable that occurs in a similar position as y is involved in any comparison.*

Then q and q' are equivalent if and only if their cores are set-equivalent.

We can simplify the conditions in Theorem 5.3 if all the comparisons in the two queries are non-strict.

Corollary 5.4 (Cntd-Queries with Non-strict Comparisons) *Let q and q' be two cntd-queries that contain only non-strict comparisons. Then their cores are equivalent under set-semantics if q and q' are equivalent and one of the following conditions holds:*

1. *q and q' range over the rationals;*
2. *q and q' do not contain constants.*

Example 5.5 Equivalence under set-semantics fails to be a necessary condition when the requirements of Theorem 5.3 are not fulfilled. To see this, consider the queries

$$\begin{aligned} q(cntd(y)) &\leftarrow p(y)\ \&\ p(z)\ \&\ y < z \\ q'(cntd(y)) &\leftarrow p(y)\ \&\ p(z)\ \&\ z < y. \end{aligned}$$

Then both queries give a result if there are at least two elements in p. The core of q returns all elements of p but the greatest, while the core of q' returns all but the least. Thus, both cores return the same number of elements, but they are not set-equivalent. Theorem 5.3 does not apply because in both queries y is involved in a strict comparison.

6 Count-Queries and Bag-Set-Equivalence

For count-queries it is easy to show the following result.

Proposition 6.1 *Two count-queries are equivalent if and only if their cores are bag-set-equivalent.*

Thus, in the remainder of this section, we will investigate bag-set-equivalence of conjunctive queries. In [CV93], it was claimed that two relational queries without constants are bag-set equivalent if and only if they are isomorphic. We are able to show the same even when relational queries are allowed to have constants. Our proof is based on an analysis of counting functions. More precisely, we construct out of each query an infinite family of databases and count how often a particular tuple is returned over these databases. We show that the counting function are distinct if the queries are not isomorphic.

Theorem 6.2 (Bag-Set-Equivalence is Isomorphism) *Two relational conjunctive queries (possibly with constants) are bag-set-equivalent if and only if they are isomorphic.*

If queries have comparisons, then it is not true that bag-set-equivalence entails isomorphism.

Example 6.3 Consider the queries

$$\begin{aligned} q &\leftarrow p(x)\ \&\ p(y)\ \&\ p(z)\ \&\ x < y\ \&\ x < z \\ q' &\leftarrow p(x)\ \&\ p(y)\ \&\ p(z)\ \&\ x < z\ \&\ y < z. \end{aligned}$$

Clearly, they are not isomorphic. However, they are bag-set-equivalent.

Isomorphism of the linear expansions of two queries (but not isomorphism of the original queries) is a sufficient criterion for bag-set-equivalence of conjunctive queries with comparisons. However, the proof technique is different from the one used for relational queries.

Theorem 6.4 (Isomorphism Implies Bag-Set-Equivalence) *Let $q(\bar{x})$ and $q'(\bar{x})$ be two conjunctive queries. If q and q' have isomorphic linear expansions, then they are bag-set-equivalent.*

Due to space limitations, we don't give a proof. However, there is a proof sketch for the analogous theorem about *sum*-queries (7.7).

For the converse of the preceding theorem, we have to be careful with the set D over which we construct the linear expansion: it must comprise the constants in both queries.

Theorem 6.5 (Bag-Set-Equivalence Implies Isomorphism) *Let $q(\bar{x})$ and $q'(\bar{x})$ be conjunctive queries, and let D be the set of constants occurring in q or q'. If q and q' are bag-set-equivalent, then their linear expansions over D are isomorphic.*

Again, there is a proof sketch for the analogous theorem about *sum*-queries (7.9).

Our characterization of bag-set-equivalence gives us immediately an upper complexity bound.

Theorem 6.6 (Upper Complexity Bound) *Bag-set-equivalence of conjunctive queries with comparisons can be decided with polynomial space.*

Proof. Let q, q' be two conjunctive queries with comparisons. Let $(q_L)_L$, $(q'_M)_M$ be linear expansions of q and q', respectively, $\mathcal{Q}$ be the set of all queries occurring in $(q_L)_L$ and $\mathcal{Q}'$ be the set of those occurring in $(q'_M)_M$.

The two linear expansions are isomorphic iff for every $q_L \in \mathcal{Q}$ there are as many isomorphic queries in $\mathcal{Q}$ as there are in $\mathcal{Q}'$, and, similarly, for every $q'_M \in \mathcal{Q}'$, there are as many isomorphic queries in $\mathcal{Q}'$ as there are in $\mathcal{Q}$.

Each of the two conditions can be checked with polynomial space as follows. In an outer loop, we enumerate all elements of $\mathcal{Q}$. During the enumeration, for each $q_L \in \mathcal{Q}$, we enumerate in an inner loop all elements of $\mathcal{Q}$ and count how many are isomorphic to q_L. Then, in a subsequent inner loop, we enumerate all elements of $\mathcal{Q}'$ and count those that are isomorphic to q_L. In a second outer loop, we check the analogous condition for elements $q'_M \in \mathcal{Q}'$.

At each stage of the computation, there are at most two nested loops, each of which needs no more than polynomial space. Thus, the entire algorithm can be executed with polynomial space. □

7 Sum-Queries

For *sum*-queries, bag-set-equivalence of their cores is a sufficient condition for equivalence. As we shall see later, it is also a necessary condition if there are either no constants or no comparisons.

Proposition 7.1 (Bag-Set-Equivalence Implies Sum-Equivalence) *Two sum-queries are equivalent if their cores are equivalent under bag-set-semantics.*

However, from the fact that for two queries, the *sums* over the y-values in two corresponding groups, characterized by the same tuple of $\bar{x}$-values, are the same for all databases, we cannot always deduce that the y-values themselves are the same and occur with the same multiplicities.

Example 7.2 Consider the two queries

$$\begin{aligned} q(sum(y)) \quad &\leftarrow \quad p(1)\ \&\ p(2)\ \&\ p(3)\ \& \\ &\qquad\ p(y)\ \&\ 1 \leq y \leq 3 \\ q'(sum(y)) \quad &\leftarrow \quad p(1)\ \&\ p(2)\ \&\ p(3)\ \& \\ &\qquad\ p(y)\ \&\ 1 \leq y \leq 2\ \& \\ &\qquad\ p(z)\ \&\ 1 \leq z \leq 2. \end{aligned}$$

where all variables range over the integers. Both queries return a result if and only if the database contains the atoms $p(1)$, $p(2)$, and $p(3)$. Moreover, both queries return the number 6, but the first query obtains it as $6 = 1 + 2 + 3$, while the second obtains it as $6 = 1 + 2 + 1 + 2$. Thus, the two *sum*-queries are equivalent, but their cores are not—neither under set semantics nor under bag-set semantics.

To enforce in the first query that exactly the numbers 1, 2, and 3 are returned, we exploit the fact that there are no other integers y with $1 \leq z \leq 3$. In a similar way, we enforce that the second query outputs exactly the numbers 1 and 2, and that each of them is output exactly twice.

Over databases that contain rational numbers, which have a dense ordering, the two queries are not equivalent. For example, over the database

$$\{p(1),\ p(1.5),\ p(2),\ p(3)\},$$

the first query returns $q(7.5)$, while the second query returns $q'(13.5)$.

This example illustrates the difficulty in finding a characterization of equivalence for sum-queries. It also shows that sum-queries with comparisons ranging over the integers have to be treated differently from sum-queries with comparisons over a dense order. We provide two different proof techniques (and two different characterizations) for the equivalence of sum-queries. One proof technique applies to the special case of sum-queries without constants, and the second proof technique applies to the general case (i.e., sum-queries with constants).

7.1 Sum-Queries Without Constants

In the special case of sum-queries without constants (but with comparisons), we can use a proof technique that is different from the one for the general case. The technique is based on transforming queries and databases by strictly monotonic mappings. A mapping $\phi: \tau \to \tau$ on an ordered type τ is *strictly monotonic* if $\phi(u) < \phi(v)$ for all $u, v \in \tau$, such that $u < v$. If a is an atom containing variables and elements of τ as constants, then ϕa is the atom where each u is replaced with $\phi(u)$. Similarly, for a query q, database $\mathcal{D}$, and a multiset of atoms A, we define ϕq, $\phi\mathcal{D}$, and ϕA as the query, the database, and the multiset, respectively, that are obtained by replacing each occurrence of an atom a with ϕa.

Proposition 7.3 *Let q be a conjunctive query without constants and $\mathcal{D}$ be a database over an ordered type τ. Let ϕ be a strictly monotonic mapping on τ. Then*

$$\{\!\{q\}\!\}^{\phi\mathcal{D}} = \phi(\{\!\{q\}\!\}^{\mathcal{D}}).$$

Theorem 7.4 (Sum-Queries without Constants) *Let q and q' be sum-queries without constants. If q and q' are equivalent, then their cores $\breve{q}$ and $\breve{q}'$ are bag-set-equivalent.*

Proof. Assume that q and q' are equivalent. Let $\mathcal{D}$ be a database. We have to show that

$$\{\!\{\breve{q}\}\!\}^{\mathcal{D}} \quad = \quad \{\!\{\breve{q}'\}\!\}^{\mathcal{D}}. \tag{6}$$

This claim is difficult to show for an arbitrary database, because the same sum of y-values may be produced in different ways, as can be seen in Example 7.2. We therefore transform $\mathcal{D}$ into another database, where the multiplicities of each y-value in a group w.r.t. q and q' can be read off the sum over the group.

First, we determine an upper bound for the multiplicities of y-values, i.e., for the number of times that a tuple $(\bar{d}, d)$ can be produced by the queries $\breve{q}$ and $\breve{q}'$. Such an upper bound is the number of assignments satisfying the bodies of $\breve{q}$ and $\breve{q}'$. Let l be the maximum of the numbers of variables appearing in the bodies of $\breve{q}$ and $\breve{q}'$, and let n be the number of data elements appearing in $\mathcal{D}$. Then n^l is an upper bound for the number of assignments over $\mathcal{D}$ that satisfy the bodies of $\breve{q}$ or $\breve{q}'$.

Let $u_1 < u_2 < \ldots < u_n$ be the data elements appearing in $\mathcal{D}$, which are all elements of some numerical type. Let $M > n^l$, and let ϕ be a strictly monotonic mapping on τ such that $\phi(u_i) = M^i$.

To prove Equation (6), it suffices to show that

$$\{\!\{\breve{q}\}\!\}^{\phi\mathcal{D}} \quad = \quad \{\!\{\breve{q}'\}\!\}^{\phi\mathcal{D}}, \tag{7}$$

since by Proposition 7.3 this implies that $\phi(\{\!\{\breve{q}\}\!\}^{\mathcal{D}}) = \phi(\{\!\{\breve{q}'\}\!\}^{\mathcal{D}})$, which implies that $\{\!\{\breve{q}\}\!\}^{\mathcal{D}} = \{\!\{\breve{q}'\}\!\}^{\mathcal{D}}$, because ϕ is a bijection between the active domains of $\mathcal{D}$ and $\phi\mathcal{D}$.

Since the *sum*-queries q and q' are equivalent, they have corresponding groups over $\phi\mathcal{D}$, i.e., if $\breve{q}$ returns $(\bar{d}, d)$, then $\breve{q}'$ returns $(\bar{d}, d')$ for some d', and vice versa.

Now, consider a fixed $\bar{d}$. Let $m_1, \ldots, m_n$ be the multiplicities of the y-values $M^1, \ldots, M^n$ in the group of $\bar{d}$ w.r.t. q, and let $m'_1, \ldots, m'_n$ be their multiplicities in the group of $\bar{d}$ w.r.t. q'. Since q and q' are equivalent, the sums over the multisets of y-values in the groups of q and q' are the same, i.e.,

$$\sum_{i=1}^{n} m_i M^i \quad = \quad \sum_{i=1}^{n} m'_i M^i. \tag{8}$$

As over $\mathcal{D}$, the number of assignments over $\phi\mathcal{D}$ satisfying the bodies of q and q' is at most $n^l < M$. Thus, $m_i < M$ and $m'_i < M$ for all i. Hence, the sums in Equation (8) are M-adic representations of the same number. The coefficients in such a representation are always uniquely determined, so that $m_i = m'_i$ for all i. Hence, over $\phi\mathcal{D}$, each y-value has the same multiplicity in the groups of $\bar{d}$ w.r.t. q and q'.

Since $\bar{d}$ was chosen arbitrarily, this shows that $\{\!\{\breve{q}\}\!\}^{\phi\mathcal{D}} = \{\!\{\breve{q}'\}\!\}^{\phi\mathcal{D}}$, which implies our claim. □

The proof of the above theorem only relies on the fact that queries without constants are generic under monotonic mappings. This property also holds for databases that satisfy certain integrity constraints, like functional dependencies or referential integrity. Consequently, the characterization is true also in those cases.

7.2 Sum-Queries with Constants

We want to check whether two *sum*-queries of the form $q(\bar{x}, sum(y)) \leftarrow R \,\&\, C$ and $q'(\bar{x}, sum(y)) \leftarrow R' \,\&\, C'$, possibly containing constants, are equivalent. To this end, we consider tow reduced linear expansions $(\breve{q}_L)_{L \in \mathcal{L}_D(\breve{q})}$ and $(\breve{q}'_M)_{M \in \mathcal{L}_D(\breve{q}')}$ of the cores $\breve{q}$ and $\breve{q}'$ of q and q'.

In each linear expansion, we distinguish between those linearizations whose summation terms are variables and those whose summation terms are constants. We say that a query $\breve{q}_L(\bar{s}, s)$ in $(\breve{q}_L)_L$ is a *variable query* if the summation term s is a variable, and we say that it is a *constant query* if s is a constant.

Example 7.5 Consider again the queries q, q' introduced in Example 7.2. The query $\breve{q}$ has three linearizations

$$\breve{q}(d_i) \quad \leftarrow \quad p(1) \,\&\, p(2) \,\&\, p(3), \quad i = 1, 2, 3$$

where $d_i = i$. The query $\breve{q}'$ has four linearizations

$$\breve{q}'(d'_{ij}) \quad \leftarrow \quad p(1) \,\&\, p(2) \,\&\, p(3), \quad i = 1, 2,\ j = 1, 2$$

where $d'_{ij} = i$.

All linearizations are constant queries. If one of them returns a result over a database, then the others do so as well. However, the queries are not isomorphic, because they differ in the summation term.

We capture this relationship with the term of weak isomorphism. A substitution θ is a *weak homomorphism* from a query $p(\bar{s}, s) \leftarrow B$ to a query $p'(\bar{t}, t) \leftarrow B'$, if θ is a homomorphism from $\bar{p}(\bar{s}) \leftarrow B$ to $\bar{p}'(\bar{t}) \leftarrow B'$. Analogously, we define weak isomorphisms and weak isomorphism of queries. We write $p \sim p'$ if p and p' are weakly isomorphic. Intuitively, a weak homomorphism from $\breve{q}_L(\bar{s}, s)$ to $\breve{q}'_M(\bar{t}, t)$ is a homomorphism that does not pay attention to the summation terms s and t. We say that p and p' are *weakly set-equivalent* if $\bar{p}$ and $\bar{p}'$ are equivalent under set-semantics.

Isomorphism is an equivalence relation on the queries in $(\breve{q}_L)_L$ and in $(\breve{q}'_M)_M$. We denote the class of queries in $(\breve{q}_L)_L$ that are isomorphic to $\breve{q}_L$ as $[\breve{q}_L]$. Similarly, $[\breve{q}'_M]$ denotes the class of queries in $(\breve{q}'_M)_M$ that are isomorphic to $\breve{q}'_M$. Let $\mathcal{Q}$ be the set of all classes $[\breve{q}_L]$, and $\mathcal{Q}'$ be the set of all classes $[\breve{q}'_M]$. We say that $[\breve{q}_L]$ is weakly isomorphic to $[\breve{q}_{L'}]$, if $\breve{q}_L$ and $\breve{q}_{L'}$ are weakly isomorphic. We write in this case $[\breve{q}_L] \sim [\breve{q}_{L'}]$. We partition $\mathcal{Q}$ into the sets $\mathcal{Q}_\mathcal{V}$ and $\mathcal{Q}_\mathcal{C}$ that consist of the classes of variable queries and constant queries, respectively. There is a similar partition of $\mathcal{Q}'$.

Two classes of constant queries $Q_0 \in \mathcal{Q}_\mathcal{C}$ and $Q'_0 \in \mathcal{Q}'_\mathcal{C}$ are *associated* if they are weakly isomorphic. If for a class in $\mathcal{Q}_\mathcal{C}$ or in $\mathcal{Q}'_\mathcal{C}$ their is no associated class, then we say it is associated to the empty class. If $\breve{q}_L$ is a constant query, then we denote the summation constant as $\sigma(\breve{q}_L)$. We say that $(\breve{q}_L)_L$ and $(\breve{q}'_M)_M$ are *in balance* if for every pair Q_0, Q'_0 of associated classes of constant queries we have

$$\sum_{\substack{Q \in \mathcal{Q}_\mathcal{C} \\ Q \sim Q_0}} \sum_{\breve{q}_L \in Q} \sigma(\breve{q}_L) \quad = \quad \sum_{\substack{Q' \in \mathcal{Q}'_\mathcal{C} \\ Q' \sim Q'_0}} \sum_{\breve{q}'_M \in Q'} \sigma(\breve{q}'_M). \tag{9}$$

The definition can be rephrased as follows. For every constant query $\breve{q}_L$, collect all weakly isomorphic constant queries in $(\breve{q}_L)_L$ and sum up their summation constants. Do the same for all constant queries in $(\breve{q}'_M)_M$ that are weakly isomorphic to $\breve{q}_L$. The resulting sums must be the same.

Example 7.6 The cores of the queries q, q' introduced in Example 7.2 have linear expansions that are in balance. All their linearizations are given in Example 7.5. They are all constant queries and weakly isomorphic to each other. The sum for $\breve{q}$ is 6, and the sum for $\breve{q}'$ is also 6.

As another example, consider the queries

$$\begin{aligned} p(sum(y)) \quad &\leftarrow \quad r(y) \,\&\, r(z) \,\&\, r(w) \,\&\, \\ &\qquad 0 < y \,\&\, 0 \leq z \,\&\, 0 < w \\ p'(sum(y)) \quad &\leftarrow \quad r(y) \,\&\, r(z) \,\&\, r(w) \,\&\, \\ &\qquad 0 \leq y \,\&\, 0 \leq z \,\&\, 0 < w. \end{aligned}$$

The core of the first query does not have any linearization that is a constant query. The core of the second has four, namely

$$\begin{aligned} p'_1(0) \quad &\leftarrow \quad r(0) \,\&\, r(z) \,\&\, 0 < z \\ p'_2(0) \quad &\leftarrow \quad r(0) \,\&\, r(z) \,\&\, 0 < z \\ p'_3(0) \quad &\leftarrow \quad r(0) \,\&\, r(z) \,\&\, r(w) \,\&\, 0 < z \,\&\, z < w \\ p'_4(0) \quad &\leftarrow \quad r(0) \,\&\, r(z) \,\&\, r(w) \,\&\, 0 < w \,\&\, w < z. \end{aligned}$$

Query p'_1 is obtained from $L_1 = \{0 = y = w < z\}$, while p'_2 is obtained from $L_2 = \{0 = y < z = w\}$. They form two equivalence classes of isomorphic queries, $\{p'_1,\ p'_2\}$ and $\{p'_3,\ p'_4\}$. Both classes are associated to the empty set. Since the summation constant is always 0, we obtain in both cases 0 as the sum. This is also the result that we obtain when summing over the empty set. Thus, the expansions of $\breve{q}$ and $\breve{q}'$ are in balance.

Let $\mathcal{L}^v_D(\breve{q})$ consist of those linearizations that do not identify y with a constant. We say that two linear expansions $(\breve{q}_L)_L$, $(\breve{q}'_M)_M$ are *variable isomorphic* if there is a bijection $\mu\colon \mathcal{L}^v_D(\breve{q}) \to \mathcal{L}^v_D(\breve{q}')$ such that $\breve{q}_L$ and $\breve{q}'_{\mu(L)}$ are isomorphic.

Theorem 7.7 (Sufficient Condition for Equivalence) *Let q, q' be sum-queries. Then q and q' are equivalent if*

- *$\breve{q}$ and $\breve{q}'$ are weakly set-equivalent, and*
- *$\breve{q}$ and $\breve{q}'$ have linear expansions that are in balance and variable isomorphic.*

Proof. We only give an outline. Since $\breve{q}$ and $\breve{q}'$ are weakly set-equivalent, $\breve{q}$ returns a group for the tuple $\bar{d}$ if and only if $\breve{q}'$ does.

Let $(\breve{q}_L)_L$, $(\breve{q}'_M)_M$ be linear expansions as in the statement of the theorem. The fact that $(\breve{q}_L)_L$, $(\breve{q}'_M)_M$ are in balance, guarantees that those values in the groups of $\bar{d}$ that are equal to constants in the queries sum up to the same results in q and in q'. The isomorphism of $(\breve{q}_L)_{L \in \mathcal{L}^{\mathrm{v}}_D(\breve{q})}$ and $(\breve{q}'_M)_{M \in \mathcal{L}^{\mathrm{v}}_D(\breve{q}')}$ guarantees that values in the groups of $\bar{d}$ that are distinct from constants are returned with the same multiplicity by each core. Hence, both queries return the same sums. □

Example 7.8 Consider again the queries p and p' in Example 7.6. As shown before, the linear expansions of their cores are in balance. It is also easy to check that the linear expansions are variable isomorphic. Thus, by Theorem 7.7, p and p' are equivalent.

To formulate a necessary condition for the equivalence of *sum*-queries, we have to assume that, in the case of comparisons over the integers, linear expansions are taken w.r.t. to sets of constants that are closed under the addition of virtual constants (see Subsection 2.6). In the case of comparisons over dense orders, we define $vc_C(D) = D$.

Theorem 7.9 (Necessary Condition for Equivalence) *Let $q(\bar{x}, y) \leftarrow R \,\&\, C$, $q'(\bar{x}, y) \leftarrow R' \,\&\, C'$ be sum-queries, and let D be a set of constants such that $D = vc_C(D_0) \cup vc_{C'}(D_0)$, where D_0 comprises the constants of q and q'. If q and q' are equivalent, then*

- *$\breve{q}$ and $\breve{q}'$ are weakly set-equivalent,*
- *$\breve{q}$ and $\breve{q}'$ have linear expansions over D that are in balance and variable isomorphic.*

Proof. We only give an outline. Since q and q' are equivalent, q returns a sum for the tuple $\bar{d}$ if and only if q' does. Hence, $\breve{q}$ and $\breve{q}'$ are weakly set-equivalent.

Assume that q and q' are equivalent, and let $(\breve{q}_L)_L$ and $(\breve{q}'_M)_M$ be linear expansions of the cores of q and q' over D. Note that by the choice of D the linear expansions are reduced. Suppose that they are not in balance. Then there is a pair of associated classes of constant queries Q, Q' for which

$$\sum_{\substack{Q \in \mathcal{Q}_C \\ Q \sim Q_0}} \sum_{\breve{q}_L \in Q} \sigma(\breve{q}_L) \quad \neq \quad \sum_{\substack{Q' \in \mathcal{Q}'_C \\ Q' \sim Q'_0}} \sum_{\breve{q}'_M \in Q'} \sigma(\breve{q}'_M).$$

We can choose the pair such that the queries in Q and Q' have a minimal number of variables and relational atoms and construct a database $\mathcal{D}$ out of one of the queries. Over this database, Q and Q' return groups that sum up to distinct numbers. Because of the minimality of Q and Q', one can show that the difference cannot be compensated by classes of constant queries that are not associated with Q and Q'. Thus, the difference can only be compensated by variable queries, which means that the linearizations of q and q' are not variable isomorphic.

If the linearizations are not variable isomorphic, then there is a counter-example consisting of two isomorphic classes Q, Q' of variable queries. Again, we can choose such a counter-example in a way such that the number of variables and relational atoms is minimal. We construct a database $\mathcal{D}_0$ from one of the queries, say $\breve{q}_{L_0}$, using an assignment γ_0 that satisfies L_0. All the linearizations that return answers over this database are weakly isomorphic to $\breve{q}_{L_0}$. Since they are not necessarily isomorphic, but only weakly isomorphic, they may return different numbers to be summed up. Because of the minimality of the counter-example, however, variable queries that return the same values are isomorphic.

If q and q' return different sums over this database, then we are done. If not, we choose among all those values d the maximal one, say d_0, such that the number of queries $\breve{q}_L$ returning d_0, say l, differs from the number of queries $\breve{q}'_M$ returning d_0, say m. These queries contribute sums ld_0 and md_0 to the overall sums returned by q and q'.

The value d_0 is not contained in D, because it is returned by a variable query. Therefore, there is a variable z such that $\gamma_0(z) = d_0$. Since L is reduced, there is an assignment γ_1 satisfying L_0 such that $d_1 := \gamma_1(z) \neq d_0$. In particular, γ_1 can be chosen in such a way that $\gamma_1(z') = \gamma_0(z')$ for all z' such that $\gamma_0(z') < d_0$. We denote the database constructed from $\breve{q}_{L_0}$ with γ_1 as $\mathcal{D}_1$. Let S_0 be the sum returned by q and q' over $\mathcal{D}_0$, and let S_1, S'_1 be the sums returned by them over $\mathcal{D}_1$. Then $S_1 - S'_1 = S_0 + ld_1 - ld_0 - (S_0 + md_1 - md_0) = (l - m)(d_1 - d_0) \neq 0$. Hence, q and q' return different sums over $\mathcal{D}_1$, which contradicts our assumption that they are equivalent.

This proves the theorem. □

The two preceding theorems can be specialized for queries without comparisons.

Theorem 7.10 (Sum-Queries without Comparisons) *Let q and q' be sum-queries without comparisons and $\breve{q}$ and $\breve{q}'$ be their cores. Then the following are equivalent:*

1. *q and q' are equivalent;*
2. *$\breve{q}$ and $\breve{q}'$ are bag-set-equivalent;*
3. *$\breve{q}$ and $\breve{q}'$ are isomorphic.*

The conditions in Theorem 7.9 are necessary for the equivalence of *sum*-queries. By Theorem 7.7, they are also sufficient. This gives us an upper complexity bound for the problem of deciding equivalence of *sum*-queries.

Theorem 7.11 (Upper Complexity Bound) *Equivalence of sum-queries can be decided with polynomial space.*

Proof. Let q and q' be two *sum*-queries and D_0 be the set of constants occurring in q and q'.

We first observe that the set of virtual constants $D = vc_C(D_0) \cup vc_{C'}(D_0)$ is of polynomial size w.r.t. q and q' and can be computed with polynomial space.

Then, we note that weak set-equivalent is in Π^P_2, and therefore can be decided with polynomial space.

Finally, with algorithms similar to the one described in the proof of Theorem 6.6, one can check that the cores $\breve{q}$, $\breve{q}'$ of q, q', respectively, have linear expansions that are in balance and variable isomorphic. □

8 Linear Queries

A query is *linear* if the body does not have multiple occurrences of the same predicate. For conjunctive queries under set semantics it is known that equivalence is decidable in polynomial time. We can generalize this result to the classes of queries considered in this paper.

Theorem 8.1 (PTIME-Complexity) *For linear queries, the following problems can be decided in polynomial time:*

- *bag-set-equivalence of conjunctive queries;*
- *equivalence of max-queries;*
- *equivalence of count-queries;*
- *equivalence of sum-queries.*

9 Conclusion

Containment and equivalence under set semantics have been studied extensively for conjunctive queries [CM77, ASU79, JK83, SS92], for conjunctive queries with built-ins [vdM92, LS95], for queries with union and difference [SY81], and for conjunctive queries defined by Datalog programs [LS95, LMSS93]. Containment for conjunctive queries under bag-set semantics, which is the semantics of SQL, has been investigated in [CV93, IR95]. We have extended these results to equivalence of conjunctive queries with comparisons under bag-set semantics, and to equivalence of conjunctive queries with comparisons and aggregations. The equivalences discussed in this paper are decidable with polynomial space. but a tight lower bound is given only for min-queries and max-queries. In the case of linear queries, equivalences are decidable in polynomial time.

The work of [RSSS98] on aggregation constraints considers a different problem related to aggregate queries; that is, satisfiability of a conjunction of aggregation constraints. An interesting issue is how to combine our results with those of [RSSS98] in the investigation of equivalences among aggregation queries with a `HAVING` clause.

Other topics for future research include equivalence of queries with the average operator, and adaptation of our results to the view usability problem.

Acknowledgments

This research was supported in part by the Esprit Long Term Research Project 22469 "Foundations of Data Warehouse Quality" (DWQ), and by a grant from the Israeli Ministry of Science and the French Ministry of Research and Technology.

References

[ASU79] A.V. Aho, Y. Sagiv, and J.D. Ullman. Efficient optimization of a class of relational expressions. *ACM Transactions on Database Systems*, 4(4):435–454, 1979.

[CKPS95] S. Chaudhuri, S. Krishnamurthy, S. Potarnianos, and K. Shim. Optimizing queries with materialized views. In P.S.Yu and A.L.P. Chen, editors, *Proc. 11th International Conference on Data Engineering*, Taipei, March 1995. IEEE Computer Society.

[CM77] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proc. 9th Annual ACM Symposium on Theory of Computing*, 1977.

[CV93] S. Chaudhuri and M. Vardi. Optimization of real conjunctive queries. In *Proc. 12th Symposium on Principles of Database Systems*, Washington (D.C., USA), May 1993. ACM Press.

[DJLS96] Sh. Dar, H.V. Jagadish, A.Y. Levy, and D. Srivastava. Answering queries with aggregation using views. In *Proc. 22nd International Conference on Very Large Data Bases*, Bombay (India), September 1996. Morgan Kaufmann Publishers.

[GHQ95] A. Gupta, V. Harinarayan, and D. Quass. Aggregate query processing in data warehouses. In *Proc. 21st International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers, August 1995.

[IR95] Y.E. Ioannidis and R. Ramakrishnan. Beyond relations as sets. *ACM Transactions on Database Systems*, 20(3):288–324, 1995.

[JK83] D.S. Johnson and A. Klug. Optimizing conjunctive queries that contain untyped variables. *SIAM Journal on Computing*, 12(4):616–640, 1983.

[LMSS93] A.Y. Levy, I. Singh Mumick, Y. Sagiv, and O. Shmueli. Equivalence, query-reachability, and satisfiability in datalog extensions. In *Proc. 12th Symposium on Principles of Database Systems*, pages 109–122, Washington (D.C., USA), May 1993. ACM Press.

[LS95] A.Y. Levy and Y. Sagiv. Semantic query optimization in datalog programs. In *Proc. 14th Symposium on Principles of Database Systems*, pages 163–173, San Jose (California, USA), Proc. 14th Symposium on Principles of Database Systems 1995. ACM Press.

[RSSS98] K.A. Ross, D. Srivastava, P.J. Stuckey, and S. Sudarshan. Foundations of aggregation constraints. *Theoretical Computer Science*, 190, 1998. (to appear).

[SS92] Y. Sagiv and Y. Saraiya. Minimizing restricted-fanout queries. *Discrete Applied Mathematics*, 40:245–264, 1992.

[SY81] Y. Sagiv and M. Yannakakis. Equivalence among relational expressions with the union and difference operators. *J. ACM*, 27(4):633–655, 1981.

[vdM92] R. van der Meyden. The complxity of querying indefinite data about linearly ordered domains. In *Proc. 11th Symposium on Principles of Database Systems*, pages 331–345, San Diego (California, USA), May 1992. ACM Press.