HEWLETT
PACKARD

# Optimization of Real Conjunctive Queries

Surajit Chaudhuri, Moshe Y. Vardi*
Software Technology Laboratory

HPL-93-26
May, 1993

optimization, logic,
SQL, complexity

The optimization problem for conjunctive queries has been studied extensively. Unfortunately, this research almost invariably assumes set-theoretic semantics (i.e., duplicates are eliminated). In contrast, SQL queries have bag-theoretic semantics (i.e., in general duplicates are not eliminated). In this paper we study the optimization problems for conjunctive queries under bag-theoretic semantics. We show that optimization techniques from the set-theoretic setting do not carry over to the bag-theoretic setting.

* IBM Almaden Research Center, San Jose, CA

Also published as IBM Research Report RJ9335
To be published in the proceedings of the *Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Washington, D.C., May 25-27, 1993

# 1 Introduction

Techniques to optimize relational queries are based on transforming a given query into a query that is semantically equivalent but less expensive to evaluate. Thus, deciding *equivalence* of queries is one of the most fundamental questions in query optimization. In general, equivalence of relational queries is undecidable, so research has focused on certain classes of relational queries. A class that has attracted a significant amount of attention is the class of *conjunctive queries*, since a large number of queries that arise in practice fall into this class.

The optimization problem for conjunctive queries has been studied extensively [ASU79A, ASU79B, CM77, DBS90]; see [U89](Chapter 14). The focus of this research is the minimization of the number of conjuncts, which corresponds to minimizing the number of joins. As a result of this research, the optimization problem for conjunctive queries is very well understood. We know how to minimize conjunctive queries and we know what the computational complexity of such minimization is.

There is a difficulty, however, in applying the results of the research on optimization of conjunctive queries to query optimization in real-life database management systems. The above mentioned research almost invariably assumes set-theoretic semantics; relations, either database relations or results of queries are *sets*, that is, they do not contain duplicate tuples. In contrast, SQL, the query language used in commercial databases, has bag-theoretic semantics; relations are *bags* (another term is *multisets*), that is, duplicate tuples are not eliminated unless elimination is explicitly requested. This is done for two reasons. First, duplicate elimination might be computationally expensive. Second, aggregate functions (such as COUNT) are sensitive to the multiplicity of tuples.

The change in the underlying semantics makes it necessary to re-examine the optimization problem for conjunctive queries in the bag-theoretic setting. Do the known results about optimization of conjunctive queries carry over from the set-theoretic setting to the bag-theoretic setting? As we shall know, the results do not carry over.

Equivalence of conjunctive queries is typically approached via the notion of *containment*. In the set-theoretic setting, we say that a query is contained in another if the answer to the former is always a subset of the answer to the latter. Two queries are equivalent if they are contained in each other. In the bag-theoretic setting, we say that a query is contained in another if the answer to the former is always a subbag of the answer to the latter, and again equivalence can be expressed in terms of containment. We thus start our investigation by considering conjunctive query containment under bag-theoretic semantics.

Containment of conjunctive queries in the bag-theoretic setting seems to be harder than containment in the set-theoretic setting. The latter problem is known to be NP-complete [ASU79A, ASU79B, CM77]. In contrast, while we do have some sufficient and some necessary conditions for containment, we do not even know whether containment under bag-theoretic semantics is decidable. We prove that the problem is $\Pi_2^p$-hard. $\Pi_2^p$ is at the second level of the polynomial hierarchy, while NP$(= \Sigma_1^p)$ is at the first level of the polynomial-time hierarchy [St77]. Since the polynomial-time hierarchy is believed to be strict, this indicates that the change of the underlying semantics does increase the computational complexity of the problem.

Since key concept for query optimization is equivalence (rather than containment), we then examine the equivalence problem in the bag-theoretic semantics. Here surprisingly, the problem seems to be easier. While equivalence under set-theoretic semantics is still NP-complete [ASU79A, ASU79B, CM77], equivalence under bag-theoretic semantics has the same complexity as the graph-isomorphism problem, which is in NP but is not known to be NP-hard [GJ79]. In fact, we show that two conjunctive queries are equivalent under bag-theoretic semantics precisely when the queries are isomorphic. An important consequence of this result is that under bag-theoretic semantics no optimization of conjunctive queries by removal of conjuncts is possible; any optimization must be restricted to reordering the conjuncts in the query.

The above result seems to paint a fairly bleak picture for practical optimization of conjunctive queries. We present, however, two results that indicate that perhaps not all is lost. First, we consider unions of conjunctive queries. Under set-theoretic semantics, containment and equivalence of union of conjunctive queries reduces to containment and equivalence, respectively, of conjunctive queries [SY81]. We show that this result does not hold under bag-theoretic semantics. This leaves open the possibility of meaningful optimization for unions of conjunctive queries. We also consider the situation where the database relations are known to be sets (i.e., no duplicates are allowed). Intuitively, this situation represents the middle ground between set-theoretic and bag-theoretic semantics. We show that in this case some optimization is possible.

## 2 Bag-Theoretic Semantics

A *bag* is a set of *annotated elements*; the annotation of an element, also called the *multiplicity* of the element, is a positive integer. Intuitively, a bag may contain duplicate occurrences of an element; the multiplicity of an element indicates the number of duplicates for the element in the set. The multiplicity of an element $a$ in a bag $B$ will be denoted by $|a|_B$ (or simply $|a|$, when $B$ is clear from the context). A *relation* is a bag of tuples of some fixed arity. A *database* is an assignment of relations to relation names.

**Example 2.1:** Let the relation PART consists of the following annotated tuples:

$$\{(engine, Seattle; [2]), (flap, Seattle; [1]), (wing, Portland; [1])\}$$

where ";" marks the end of the tuple and the multiplicity is indicated in square brackets. This means that PART contains two copies of the tuple $(engine, Seattle)$, but only one copy of each of the other tuples. That is, $|(engine, Seattle)| = 2$, $|(flap, Seattle)| = 1$, and $|(wing, Portland)| = 1$. ▮

We say that a bag $B$ is a *subbag* of $B'$ if each element of $B$ is contained also in $B'$ with a greater than or equal multiplicity. We will define the *subbag* relationship by $\subseteq_b$. The subset relationship will be denoted by $\subseteq_s$. The equality relationship between two bags (resp., sets) will be denoted by the $=_b$ $(=_s)$.

**Example 2.2:** In this example, $B \subseteq_b B'$.

$$\begin{aligned} B &= \{(engine, Seattle; [1])\} \\ B' &= \{(engine, Seattle; [2]), (flap, Seattle; [1]), (wing, Portland; [1])\} \end{aligned}$$

▮

We will also define the *bag union* operator. The bag union of two bags is obtained by adding the multiplicity factors for each tuple in either of the bags. The bag union will be denoted by $\sqcup$.

**Example 2.3:** Let us consider the bags $B$ and $B'$ in Example 2.2. The $B \sqcup B'$ is the bag:

$$\{(engine, Seattle; [3]), (flap, Seattle; [1]), (wing, Portland; [1])\}$$

∎

# 3 Conjunctive Queries

In this section we describe the bag-theoretic semantics of conjunctive queries. Since our motivation is the optimization of "real-life" queries, we start with the description of SQL conjunctive queries.

## 3.1 SQL Conjunctive Queries

An *SQL conjunctive query* is an SQL query of the following form:

```
SELECT  columnlist
FROM    rellist
WHERE   equalitylist
```

where `columnlist` is the list of relation attributes to be selected, `rellist` is the list of relation (called *table* in SQL) names (possibly with tuple variables), and `equalitylist` is a conjunction of equalities among relation attributes. (For the complete syntax of SQL, we refer the reader to [D87].)

**Example 3.1:** The following is an example of a conjunctive query.

```
SELECT  SUPPLIER.ID, PART.ID
FROM    SUPPLIER, PART
WHERE   SUPPLIER.CITY = PART.CITY
```

The operational semantics of SQL conjunctive queries is defined as follows (see [D87] for the operational semantics of SQL). First, the cross product of the relations in `rellist` is taken. Next, we apply each of the selection conditions in `equalitylist` to each tuple obtained in the cross product. Finally, the qualifying tuples are projected on the attributes that are among `columnlist`. The details will be described in the full paper.

**Example 3.2:** Let us consider the conjunctive SQL query in Example 3.1. Let us assume that the relation for SUPPLIER consists of the tuple

$$\{(Boeing, Seattle; [1])\}$$

and the relation for PART consists of the tuples:

$$\{(engine, Seattle; [2]), (wing, Portland; [1]), (flap, Seattle; [1])\}$$

Therefore, the cross product of the relations results in the relation:

$$\{(Boeing, Seattle, engine, Seattle; [2]), (Boeing, Seattle, wing, Portland; [1]), (Boeing, Seattle, flap, Seattle; [1])\}$$

After application of the condition in the **equalitylist** only the following tuples qualify:

$$\{(Boeing, Seattle, engine, Seattle; [2]), (Boeing, Seattle, flap, Seattle; [1]),$$

Finally, the application of the selection list results in the following relation as answer to the query in Example 3.1.

$$\{(Boeing, engine; [2]), (Boeing, flap; [1])\}$$

▮

## 3.2 Logical Conjunctive Queries

In this subsection, we describe the logical syntax of conjunctive queries and their denotational semantics. The two approaches (of this section and the previous section) are then shown to be equivalent (for a detailed semantical account of SQL, see [NPS91]; see also [MPR90]).

A *logical conjunctive query* is a rule of the form:

$$Query(\mathbf{X}){:}\text{-}C_1(\mathbf{X}_1), \ldots, C_n(\mathbf{X}_n)$$

where $\mathbf{X}$ and the $\mathbf{X}_i$'s are tuples of variables, and the $C_j$'s are relation names. $Query(\mathbf{X})$ is the *head* of the query and $C_1(\mathbf{X}_1), \ldots, C_n(\mathbf{X}_n)$ is the *body* of the query (we will sometimes look upon a query itself as the bag or set of literals in the body). Note that there are no explicit equalities in this representation; rather, equalities are captured by multiple occurrences of variables. The head variables $\mathbf{X}$ are the selected variables, and will be called *distinguished variables.*

**Example 3.3:** The query in Example 3.2 can be expressed as:

$$Query(s\_id, p\_id) : -Supplier(s\_id, c\_id), Part(p\_id, c\_id)$$

▮

The denotational semantics of logical conjunctive queries is defined in terms of *assignment mappings*. An assignment mapping of a conjunctive query $Q$ as above *into* a database $D$ is an assignment of data values in $D$ to the variables of $Q$ such that every conjunct in the body of $Q$ is mapped to a tuple in $D$. Let $\theta$ be an assignment mapping of $Q$ into database $D$, and let $X$ be a variable in $Q$. We denote by $\theta(X)$ the data value to which $\theta$ maps $X$, and we denote by $\theta(C_i(\mathbf{X}_i))$ the tuple to which $C_i(\mathbf{X}_i)$ is mapped.

**Example 3.4:** Let us consider the query in Example 3.3 and database in Example 3.2. The mapping where *s_id* is mapped to *Boeing*, *c_id* to *Seattle* and *p_id* to *engine*, is an assignment mapping. ▮

We can now define the tuple derived by a query due to an assignment mapping $\theta$. Let $m_i = |\theta(C_i(\mathbf{X}_i))|$, $i = 1, \ldots, n$. The *result due to $\theta$ of $Q$ over $D$* is the tuple $(\theta(\mathbf{X}); [m])$ with the multiplicity $m = m_1 m_2 \ldots m_n$.

**Example 3.5:** Consider Example 3.4. The result due to the assignment mapping there is $\{(Boeing, engine; [2])\}$ ▮

The *result of* a query $Q$ *over* a database $D$, denoted $Q(D)$, is given by $\sqcup_\theta r_\theta$, where $\theta$ is any assignment mapping of $Q$ into $D$ and $r_\theta$ is the result due to $\theta$. That is, $Q(D)$ is obtained by taking the bag union over all assignment mappings of the results due to these assignment mappings over $D$.

**Example 3.6:** Let us consider the query in Example 3.3 and database in Example 3.2. There are two assignment mappings that are possible: (a) *s_id* mapped to *Boeing*, *c_id* to *Seattle* and *p_id* to *engine*; (b) Same as (1), except that *p_id* is mapped to *wing*. The result of assignment (a) is given in Example 3.5. The result of assignment (b) is $\{(Boeing, wing; [1])\}$. Therefore, the evaluation results in the bag:

$$\{(Boeing, engine; [2]), (Boeing, wing; [1])\}$$

▮

### 3.2.1 Logical vs. SQL Conjunctive Queries

In this section, we briefly sketch a transformation from the SQL syntax to logical syntax of conjunctive queries.

We will use the canonical form of SQL conjunctive query as introduced in the beginning of Section 3. For simplicity, we assume that no relation name is repeated more than once in `rellist`. The following sequence of steps generate the logical syntax.

1. For every attribute of every relation in `rellist`, introduce a distinct variable.
2. For every relation in `rellist` introduce a corresponding conjunct with the variables in the same order as the attributes in the schema information.

3. Since every attribute corresponds to a distinct variable, for each equality predicate in `equalitylist`, there is a corresponding equality among variables such as $X = Y$. These equalities induces an equivalence relation on the variables. We select a representative from each equivalence class, and replace each variable by its representative.

4. The distinguished variables in the head are the representatives of the variables that correspond to the attributes in the `columnlist`.

**Example 3.7:** Consider the SQL query in Example 3.1. Assume that the schema has `ID` and `CITY` as the first and the second attributes of the `SUPPLIER` and the `PART` relations. In the first step, we introduce the variables *s_id*, *c_id* for attributes of `SUPPLIER` and variables *p_id* and *c1_id* for the attributes of `Part`. In the second step of the transformation, we create the body

$$Supplier(s\_id, c\_id), Part(p\_id, c1\_id).$$

In the third step, we apply the equality $c\_id = c1\_id$ to obtain the body

$$Supplier(s\_id, c\_id), Part(p\_id, c\_id).$$

We add the head $Query(s\_id, p\_id)$ in the fourth step. The result is:

$$Query(s\_id, p\_id) : -Supplier(s\_id, c\_id), Part(p\_id, c\_id)$$

∎

Let us assume that $Transform(Q)$ denotes the conjunctive query obtained from an SQL query $Q$ by the above transformation.

**Theorem 3.8:** *Let $Q$ be an SQL conjunctive query. Then the results of applying $Q$ and $Transform(Q)$ to any database $D$ are equal.*

It will be easier to state and prove our results about the containment and equivalence problems in terms of the logical syntax and semantics. Therefore, for the rest of this paper, we use the latter.

# 4 Containment of Conjunctive Queries

## 4.1 Basic Definitions and Results

A query $Q$ is *bag contained* in another query $Q'$, denoted by $Q \leq_b Q'$, if $Q(D) \subseteq_b Q'(D)$ over any database $D$. In contrast, *set containment* of $Q$ in $Q'$ will be denoted by $Q \leq_s Q'$.

An important observation is that bag containment for conjunctive queries is a *strictly stronger* relationship than set containment for conjunctive queries.

**Proposition 4.1:**

(a) *For any two conjunctive queries $Q$ and $Q'$, if $Q \leq_b Q'$ holds, then also $Q \leq_s Q'$.*

(b) *There exist conjunctive queries $Q$ and $Q'$, such that $Q \leq_s Q'$ holds, but $Q \leq_b Q'$ does not hold.*

We illustrate Proposition 4.1(b) by the following example.

**Example 4.2:** Let us consider the following queries:

$$\begin{array}{rcl} Q(X) & \text{:-} & p(X), q(X) \\ Q'(X) & \text{:-} & p(X) \end{array}$$

Clearly, $Q \subseteq_s Q'$. Let us consider, however, a database with the tuples $\{(p(a); [1]), (q(a); [2])\}$. Then, the (bag) result of $Q$ has two tuples, whereas the (bag) result of $Q'$ has exactly one tuple. Therefore, $Q \not\leq_b Q'$. ▮

## 4.2 Conditions for Containment

We have seen in the previous subsection that bag containment is strictly stronger than set containment. Thus, one would expect a characterization of bag containment would be obtained by strengthening the known characterization of set containment in terms of *containment mappings*.

A containment mapping from a query $Q'$ to a query $Q$ is a mapping $\sigma$ of variables of $Q'$ to variables of $Q$ such that $\sigma$ maps each conjunct in the body of $Q'$ into a conjunct in the body of $Q$. The mapping is required to be an identity mapping for the distinguished variables. It is known that $Q \leq_s Q'$ precisely when there is a containment mapping from $Q'$ to $Q$ (see [CM77]).

How can we strengthen this characterization to get a characterization of bag containment? The next proposition provides a clue.

**Proposition 4.3:** *Let $Q$ and $Q'$ be conjunctive queries.*

(a) *If $Q \leq_b Q'$, then for any relation name $p$, the number of $p$-conjuncts in the query $Q'$ is no less than the corresponding number in the query $Q$.*

(b) *If $Q \leq_b Q'$, then for every conjunct $l$ in $Q$, there is a containment mapping $\sigma$ from $Q'$ to $Q$ such that $l \in \sigma(Q')$.*

The above proposition brings out a basic difference between set containment and bag containment. If $Q \leq_s Q'$, then the body of $Q'$ is less constraining than the body of $Q$. Thus, the fewer conjuncts there are in $Q'$, the "easier" it is for $Q$ to be contained in $Q'$. Fewer conjuncts in $Q'$, however, would means that there would be fewer assignment mappings, and consequently $Q'$ would result in tuples of lower multiplicity. To get bag containment, we need to ensure that $Q'$ yields tuples with high enough multiplicity, and that means that $Q'$ should have enough conjuncts. As Proposition 4.3 shows, that requires “coverage” of the conjuncts in $Q$ by the conjuncts in $Q$. We now show that a strong enough notion of “coverage” yields a sufficient condition for bag containment.

A containment mapping $\sigma$ from $Q'$ to $Q$ is *onto* if $Q \subseteq_b \sigma(Q')$.

**Proposition 4.4:** *Let $Q$ and $Q'$ be conjunctive queries. If there is a containment mapping from $Q'$ onto $Q$, then $Q \leq_b Q'$.*

**Example 4.5:** Consider the following three queries:

$$\begin{array}{rcl} Q(X,Y) & \text{:-} & s(X,Z), t(W,Y), t(Z,W') \\ Q'(X,Y) & \text{:-} & s(X,Z), t(Z,Y) \\ Q''(X,Y) & \text{:-} & s(X,Z), t(Z,Y), u(Y,W) \end{array}$$

Observe that there is a containment mapping from $Q$ onto $Q'$ and a containment mapping from $Q$ to $Q''$. Thus, we have that $Q' \leq_b Q$ and $Q'' \leq_s Q$. On the other hand, there is no containment mapping from $Q$ onto $Q''$. Indeed, it also turns out that $Q'' \not\leq_b Q$. ▮

The condition of Proposition 4.4 turns out to be necessary and sufficient for a large class of conjunctive queries.

**Theorem 4.6:** *Let $Q$ and $Q'$ be conjunctive queries. If $Q$ has no two conjuncts with the same relation name, then $Q \leq_b Q'$ iff there is a containment mapping from $Q'$ onto $Q$.*

Proposition 4.4 and Theorem 4.6 were discovered independently in Ioannidis and Ramakrishnan [IR92].

In general, however, the existence of an onto containment mapping is not a necessary condition for bag containment. In the following example, $Q \leq_b Q'$, but there is no onto containment mapping from $Q'$ to $Q$.

**Example 4.7:** Let us consider the following two queries.

$$\begin{array}{rcl} Q'(X,Z) & \equiv & p(X), q(U,Y), q(V,Y), r(Z) \\ Q(X,Z) & \equiv & p(X), q(U,X), q(V,Z), r(Z) \end{array}$$

It is easy to observe that there is no onto containment mapping from $Q'$ to $Q$. We can show, however, that $Q \leq_b Q'$.

▮

The definition of containment involves quantification over *all* databases. In the full paper, we will show that this quantification can sometimes be replaced by quantification over a *finite* number of databases, where the multiplicities are given *symbolically.* While this does not yield a decision procedure for bag containment, it does provide a method to test bag containment in certain cases, such as the queries in Example 4.7.

### 4.3 Complexity of Containment

The complexity of set containment is known to be NP-complete [CM77]. Since the condition of Proposition 4.4 and Theorem 4.6 are closely related to the characterization of set containment, it is not surprising that this condition has a similar complexity.

**Theorem 4.8:** *The problem of determining whether there is a conjunctive mapping from a conjunctive query $Q'$ onto a conjunctive query $Q$ is NP-complete.*

The suggestion of intractability of set containment given by the NP-completeness result if [CM77] is somewhat misleading, since the complexity is in terms of the size of the queries, which is typically much smaller than the size of the database. To test if $Q \leq_s Q'$, we simply have to apply $Q'$ to the body of $Q$ and see whether this yields the goal tuple of $Q$. For many queries that arise in practice, this algorithm is quite practical. In the full paper we will describe a similar algorithm for testing existence of onto containment mappings.

Recall that the existence of onto containment mapping is in general sufficient but not necessary, so Theorem 4.8 tells us nothing about the complexity of bag containment in general. We now describe a lower bound for this problem in terms of the polynomial-time hierarchy. The polynomial-time hierarchy is defined in terms of *oracle* Turing machines; the reader is referred to [GJ79, St77] for details. The class $\Pi_2^p$ is in the second level of the hierarchy. It is believed that NP is strictly contained in $\Pi_2^p$.

We can now state the lower bound for bag containment.

**Theorem 4.9:** *The bag containment problem is $\Pi_2^p$-hard.*

Theorem 4.9 suggests that bag containment is indeed harder than set containment. The precise complexity of bag containment is an open problem. We do not even know if the problem is decidable.

## 5 Equivalence of Conjunctive Queries

The focus on containment stems from the fact that equivalence is reducible to containment. In the set-theoretic setting both equivalence and containment of conjunctive queries are NP-complete [CM77]. We saw, however, in Section 4 that the containment problem in the bag-theoretic semantics is quite difficult. This motivates studying equivalence directly.

A query $Q$ is *bag equivalent* to another query $Q'$ iff over any database $D$, we have that $Q(D) =_b Q'(D)$. If $Q$ and $Q'$ are bag equivalent, we will denote it by $Q \equiv_b Q'$. In contrast, the *set equivalence* of $Q$ and $Q'$ will be denoted by $Q \equiv_s Q'$. Clearly, $Q \equiv_b Q'$ holds precisely when both $Q \leq_b Q'$ and $Q' \leq_b Q$ hold.

It is straightforward to prove results, similar to Lemma 4.1, showing that the equivalence of conjunctive queries under bag-theoretic semantics is a strictly stronger property than the equivalence of conjunctive queries under set-theoretical semantics.

**Example 5.1:** Consider the queries $Q$ and $Q'$.

$$\begin{aligned} Q(X) &\ :- \ p(X) \\ Q'(X) &\ :- \ p(X), p(X) \end{aligned}$$

Clearly, $Q \equiv_s Q'$. Let us consider the database consisting of $\{(p(a); [2])\}$. Over this database, $Q$ returns two duplicates but $Q'$ returns four duplicates. Therefore, $Q \not\equiv_b Q'$. ∎

We saw in Section 4 that bag containment of a conjunctive query $Q$ in a conjunctive query $Q'$ requires "coverage" of $Q$ by $Q'$. We had necessary conditions for such coverage (Proposition 4.3) and a sufficient condition for such coverage (Proposition 4.4). We now show that for bag equivalence a simple coverage condition is necessary and sufficient.

We say that two conjunctive queries $Q$ and $Q'$ are *isomorphic* iff there are one-to-one containment mappings from $Q'$ onto $Q$ and vice versa.

**Theorem 5.2:** *Let $Q$ and $Q'$ be conjunctive queries. $Q \equiv_b Q'$ iff $Q$ and $Q'$ are isomorphic.*

**Corollary 5.3:** *Bag equivalence of conjunctive queries is polynomially equivalent to graph isomorphism.*

Corollary 5.3 tells us that bag equivalence of conjunctive queries is perhaps easier than set equivalence of conjunctive queries. While the latter problem is NP-complete [CM77], graph isomorphism is known to be in NP, but it is not known to be NP-complete [GJ79]. Focusing on the complexity here, misses, however, the point. The crux of the matter in optimizing conjunctive queries in the set-theoretic setting is the replacement of a conjunctive query by an equivalent conjunctive query with a smaller number of conjuncts: it is known that for every conjunctive query $Q$ there is a *minimally equivalent* conjunctive query $Q'$, i.e., $Q'$ is equivalent to $Q$ and no other conjunctive query equivalent to $Q$ has fewer conjuncts than $Q'$ [CM77]. According to Theorem 5.2, however, two conjunctive queries are bag-equivalent precisely when they are identical up to renaming and reordering. Thus, the basic optimization technique for conjunctive queries in the set-theoretic setting is simply not applicable in the bag-theoretic setting. An interesting question is whether Theorem 5.2 carry over to the larger class of conjunctive queries *with inequalities* [K88].

# 6 Union of Conjunctive Queries

## 6.1 SQL Syntax and Semantics

SQL provides the ability to take union of the bags obtained by evaluating individual queries. The SQL statement for bag union is given by:

```
A UNION ALL B
```

where A and B are SQL statements and are union-compatible, i.e., contains the same number of attributes (the corresponding attributes are also required to be type compatible). For our purpose, A and B are SQL conjunctive queries. Let us assume that $T_A$ and $T_B$ are relations obtained by evaluating A and B over a database. Then, the relations for A UNION ALL B is obtained by taking the bag union of $T_A$ and $T_B$.

**Example 6.1:** Consider the SQL query given below. The schema of the database consists of three relations PART(ID, CITY), SUPPLIER(ID, CITY), CAPITAL(CITY, COUNTRY).

```
SELECT PART.ID
FROM  PART, SUPPLIER
WHERE PART.CITY = SUPPLIER.CITY

ALL UNION

SELECT PART.ID
FROM PART, COUNTRY
WHERE PART.CITY = COUNTRY.CAPITAL
```

Let us assume that the relation for SUPPLIER has the tuple

$$\{(Boeing, Seattle; [1])\}$$

and the relation for PART has the following tuples:

$$\{(engine, Seattle; [1]), (flap, Portland; [1]), (engine, Seattle; [1]), (brake, Pittsburgh; [1])\}$$

Then, the first SQL query yields the relations

$$\{(engine; [2])\}.$$

The second SQL query yields the relation

$$\{(flap; [1])\}.$$

Therefore, the combined query results in

$$\{(engine; [2]), (flap; [1])\}.$$

∎

## 6.2 Logical Syntax and Semantics

A union $U$ of conjunctive expressions is an expression of the form

$$Q_1(\mathbf{X}) \sqcup \ldots \sqcup Q_n(\mathbf{X}),$$

where each $Q_i$ is a conjunctive query and all the $Q_i$'s has the same arity and the same set of distinguished variables. The result of $U$ over a database $D$ is $\sqcup_{1 \leq i \leq n} Q_i(D)$. The equivalence between the SQL approach and the logical approach for conjunctive queries can be easily extended to equivalence for union of conjunctive queries.

**Example 6.2:** We can represent the SQL query given in Example 6.1 by $Q_1(I) \sqcup Q_2(I)$, where

$$Q_1(I)\text{:-}part(I, C), supplier(I, C)$$
$$Q_2(I)\text{:-}part(I, C), capital(C, N)$$

▮

### 6.3 Equivalence and Containment

Sagiv and Yannakakis [SY81] have shown that for conjunctive queries, if $Q \leq_s \bigcup_i Q_i$, then there must exist some $Q_j$ in the union such that $Q \leq_s Q_j$. This suggests the following approach to optimizing a union $\bigcup_i Q_i$ of conjunctive queries in the set-theoretic setting:

(a) eliminate redundant conjunctive queries, i.e., eliminate $Q_i$ if $Q_i \leq_s Q_j$ for some $j \neq i$, and then

(b) replace each remaining conjunctive query by a minimally equivalent conjunctive query.

Assume hypothetically that the result of Sagiv and Yannakakis did carry over to the bag-theoretic setting. Could we then carry over the optimization technique to the bag-theoretic setting? The answers seems to be negative. First, step (a) above is not applicable, since, even if $Q_i \leq_b Q_j$, both $Q_i$ and $Q_j$ contributes to the multiplicity of the tuples in the answer. Second, step (b) above is not applicable, since each conjunctive query in the union is by itself not minimizable according to Theorem 5.2.

It so happens, however, that the result of Sagiv and Yannakakis does not carry over to the bag-theoretic semantics.

**Proposition 6.3:** *There are conjunctive queries $Q$, $Q'$ and $Q''$ such that $Q \leq_b Q' \sqcup Q''$, but $Q \not\leq_b Q'$ and $Q \not\leq_b Q''$.*

**Proof:** The following queries satisfy the claims of the proposition.

$$Q(X, Z) : -p(X), q(U, X), q(V, Z), r(Z)$$
$$Q'(X, Z) : -p(X), q(U, X), q(V, X), r(Z)$$
$$Q''(X, Z) : -p(X), q(U, Z), q(V, Z), r(Z)$$

▮

The failure of Sagiv-Yannakakis' Theorem leaves open the possibility of meaningful optimization for unions of conjunctive queries. A first step in that direction would be to obtain a characterization of bag equivalence for unions of conjunctive queries.

# 7 Set-Valued Databases

We will use the term *set-valued relation* to refer to a relation that is a set, i.e., a relation with no duplicates. *Set-valued* databases are defined analogously. An important special case of containment and equivalence arises when the relations in the database are set-valued. This case arise often in practice.

**Example 7.1:** Consider the following query, which returns the age of all students. We can assume that the `STUDENT` relation contains no duplicates and consists of the attributes ID and Age respectively. However, duplicates may be generated due to projection. Observe that the duplicates that are generated can be processed later by an aggregate function such as `COUNT` or `AVERAGE`.

$$Q'(age) : -Student(id, age)$$

▮

## 7.1 Containment

A query $Q$ is *bag-set contained* in another query $Q'$, denoted by $Q \leq_{bs} Q'$, if $Q(D) \subseteq_b Q'(D)$ over any set-valued database $D$. It turns out that bag-set containment is an intermediate relationship between bag containment and set containment.

**Example 7.2:** Consider a variant of the query in Example 7.1. The query, given below, considers only those students who are also employed.

$$Q(age) : -Student(id, age), Emp(id, jobtitle)$$

It is easy to see that $Q \leq_s Q'$, but $Q \not\leq_{bs} Q'$, since a student may have multiple jobs. ▮

**Example 7.3:** Consider the following queries.

$$\begin{aligned} Q'(id) &: - \quad Student(id, age) \\ Q(id) &: - \quad Student(id, age), Female(id) \end{aligned}$$

It is not hard to verify that $Q \leq_{bs} Q'$, but $Q \not\leq_b Q'$. ▮

When the database relations are set-valued, the condition of Proposition 4.3 has to be weakened.

**Proposition 7.4:** *Let $Q$ and $Q'$ be conjunctive queries. If $Q \leq_{bs} Q'$, then for every variable $v$ in $Q$, there is a containment mapping $\sigma$ from $Q'$ to $Q$ such that $v \in \sigma(Q')$.*

Example 7.3 provides us with a clue to provide a sufficient condition for bag-set containment. The sufficient condition in Proposition 4.4 says that for if the containment mapping is onto then $Q \leq_b Q'$. Certainly, this is also a sufficient condition for the restricted case. However, we can weaken the condition for the restricted case. A containment mapping $\sigma$ from $Q'$ to $Q$ is *variable-onto* if $V \subseteq_s \sigma(V')$ where $V$ And $V'$ are the set of variables in the query $Q$ and $Q'$ respectively.

**Proposition 7.5:** *Let $Q$ and $Q'$ be conjunctive queries. If there is a containment mapping from $Q'$ variable-onto $Q$, then $Q \leq_{bs} Q'$.*

**Example 7.6:** Consider Example 7.3. Observe that the only containment mapping from $Q'$ to $Q$ is variable-onto. Therefore, it follows from Proposition 7.5 that $Q \leq_{bs} Q'$. ▮

We note that it follows from Example 4.7 that Proposition 7.5 is not a necessary condition for bag-set containment.

### 7.1.1 Complexity of Containment

The complexity of determining whether there is a variable-onto mapping from one query to another is similar in nature to the problem of determining whether there is an onto containment mapping. Thus, the following result is not surprising.

**Theorem 7.7:** *The problem of determining whether there is a conjunctive mapping from a conjunctive query $Q'$ variable-onto a conjunctive query $Q$ is NP-complete.*

We observe that the condition of Theorem 7.7 is a sufficient condition and does not tell us about the complexity of bag-set containment. The following Proposition establishes a connection between bag containment and bag-set containment.

**Proposition 7.8:** *There is a polynomial reduction of bag containment to bag-set containment.*

From Proposition 7.8 and Theorem 4.9 the following result follows.

**Theorem 7.9:** *The bag-set containment problem is $\Pi_2^p$-hard.*

As in the unrestricted case, the decision problem for bag-set containment remains open.

## 7.2 Equivalence

A query $Q$ is *bag-set equivalent* to another query $Q'$, denoted $Q \equiv_{bs} Q'$, if we have that $Q(D) =_b Q'(D)$ for any set-valued database $D$. As with containment, bag-set equivalence is an intermediate relationship between bag equivalence and set equivalence.

**Example 7.10:** Observe that the following two queries are set equivalent but not bag-set equivalent.

$$\begin{aligned} Q(X,Y) &:- \; p(X,Z), p(X,Y) \\ Q(X,Y) &:- \; p(X,Y) \end{aligned}$$

▮

A key difference between bag equivalence and bag-set equivalence is that duplicate literals are redundant under bag-set equivalence, since each database tuple has multiplicity one. We will say that $Q'$ is a *canonical representation* of a query $Q$ if all duplicate literals are removed from $Q$.

**Theorem 7.11:** *Let $Q_1$ and $Q_2$ be conjunctive queries. $Q_1 \equiv_{bs} Q_2$ iff $Q'_1 \equiv_b Q'_2$ where $Q'_1$ and $Q'_2$ are canonical representations of $Q_1$ and $Q_2$ respectively.*

**Corollary 7.12:** *Bag-set equivalence of conjunctive queries is polynomially equivalent to graph isomorphism.*

Theorem 7.11 shows that only very limited optimization, namely that of removing duplicate literals, is possible in the case where relations are set-valued. In the full paper, we discuss optimization of conjunctive queries over databases where only some of the relations are known to be set-valued.

# 8 Related Work

Bag containment and equivalence for conjunctive queries were first addressed by Dayal et al. [DGK82]. These notions were also addressed by Klausner [K86] [1] in the context of an extended relational algebra with additional control over duplicate elimination than SQL. As a result, the query containment and equivalence problems are harder than in our model, and the results obtained by Klausner are weaker than our results. Recently, Ioannidis and Ramakrishnan [IR92] independently addressed the problem of containment in the bag-theoretic setting and found some sufficient conditions similar to ours. They do not consider the equivalence problem, and nor do they study the aspect of computational complexity.

# 9 Concluding Remarks

In this paper we studied the optimization problems for conjunctive queries under bag-theoretic semantics. We showed that optimization techniques from the set-theoretic setting do not carry over to the bag-theoretic setting. We found that bag containment of conjunctive queries seems to be computationally harder than set containment of conjunctive queries. We found further that in the bag-theoretic setting two conjunctive queries are equivalent precisely when they are isomorphic. As a consequence, unlike conjunctive queries in the set-theoretic setting, it is not possible to minimize conjunctive queries in the bag-theoretic setting by removal of conjuncts. This is an a posteriori justification of the current emphasis on join ordering rather than on join elimination in commercial database management systems (See [S*79, JK84]).

We have shown that Sagiv-Yannakakis' Theorem [SY81] does not extend to the bag-theoretic setting, leaving open the possibility of optimization for unions of conjunctive queries. Finally, we discussed the case of containment and equivalence of queries in the bag-theoretic setting where the database relations have no duplicates.

[1] Klausner also corrected the earlier results by Dayal et al..

**Acknowledgement** This work was inspired by a discussion with Waqar Hasan who brought to our attention the need to address the practical problem of optimizing queries with bag semantics. Umesh Dayal helped us by pointing to past work in this area. Joseph Albert has given useful comments on an earlier version of the draft.

# 10 References

[ASU79A] Aho A. V., Sagiv Y, Ullman J. D., "Equivalence of Relational Expressions," *SIAM Journal of Computing 8:2*, pp. 218-246.

[ASU79B] Aho A. V., Sagiv Y, Ullman J. D., "Efficient Optimization of a Class of Relational Queries," *ACM Transactions on Database Systems*, 4:4, pp. 435-454.

[CM77] Chandra A.K., Merlin P.M., "Optimal Implementation of conjunctive queries in relational databases," *Proc. 9th ACM Symp. on Theory of Computing*, New York, 1977, pp. 77-90.

[D87] Date C. J., *A Guide to the SQL Standard*, Addison Wesley, 1987.

[DGK82] Dayal U., Goodman N., Katz R. H., "An Extended Relational Algebra with Control over Duplicate Elimination," *Proc. of the First ACM Symposium on Principles of Database Systems*, pp. 117-123, 1982.

[DBS90] Dublish P., Biskup J., Sagiv Y., "Optimization of a subclass of conjunctive queries," *Proc. 3rd Int'l Conf. on Database Theory*, 1990, pp. 455–469.

[GJ79] Garey M., Johnson D.S., *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. Freeman and Co., San Francisco, 1979.

[IR92] Ioannidis Y. E., Ramakrishnan R., "Generalized Containment of Conjunctive Queries," Computer Science Technical Report, University of Wisconsin-Madison, 1992.

[JK84] Jarke M., Koch J., "Query Optimization in Database Systems," *ACM Computing Surveys 16:2*, pp. 111-152.

[K86] Klausner A., "Multirelations in Relational Databases," Ph.D thesis, Harvard University, 1986.

[K88] Klug A., "On Conjunctive Queries containing Inequalities," *J.ACM 35:1*, pp. 146-160.

[MPR90] Mumick I. S., Pirahesh H., Ramakrishnan R., "The Magic of Duplicates and Aggregates," *Proc. of the 16th VLDB Conference*, Brisbane, 1990, pp. 264-277.

[NPS91] Negri M., Pelagatti G., Sbattella L., "Formal Semantics of SQL Queries," *ACM Transactions on Database Systems, 16:3*, 1991, pp. 513-534.

[S*79] Selinger P. G. et.al.: Access Path Selection in a Relational Database Management. *Proc. of the ACM SIGMOD Conference on Management of Data*, June 79, pp.23-34.

[St77] Stockmeyer L.J., "The polynomial-time hierarchy", *Theoretical Computer Science*, Vol 3, 1977, pp. 1–22.

[SY81] Sagiv Y., Yannakakis M., "Equivalences among Relational expressions with the union and difference operators," *JACM 27*,1980, pp. 633-655.

[U89] Ullman J. D., *Principles of Database and Knowledge-base Systems*, Vol 2, Computer Science Press, 1989.