Containment of Conjunctive Queries: Beyond Relations as Sets

YANNIS E. IOANNIDIS and RAGHU RAMAKRISHNAN
University of Wisconsin

Conjunctive queries are queries over a relational database and are at the core of relational query languages such as SQL. Testing for containment (and equivalence) of such queries arises as part of many advanced features of query optimization, for example, using materialized views, processing correlated nested queries, semantic query optimization, and global query optimization. Earlier formal work on the topic has examined conjunctive queries over *sets* of tuples, where each query can be viewed as a function from sets to sets. Containment (and equivalence) of conjunctive queries has been naturally defined based on set inclusion and has been shown to be an NP-complete problem.

Even in SQL, however, queries over *multisets* of tuples may be posed. In fact, relations are treated as multisets by default, with duplicates being eliminated only after explicit requests Thus, in order to reason about containment/equivalence of a large class of SQL queries, it is necessary to consider a generalization of conjunctive queries, in which relations are interpreted as multisets of tuples: The view of a relation as a *set* of tuples must be generalized.

In this paper we study conjunctive queries over databases in which each tuple has an associated *label*. This generalized notion of a database allows us to consider relations that are *multisets* and relations that are *fuzzy sets*. As a special case, we can also model traditional set-relations by making the label associated with a tuple be either "true" (meaning that the tuple is in the relation) or "false" (meaning that the tuple is not in the relation). In order to keep our results general, we consider a variety of *label systems*, where each label system is essentially a set of conditions on the labels that can be associated with tuples. Once a result is established for a label system, it holds for all interpretations of relations that satisfy these conditions. For example, we present a necessary and sufficient condition for containment of conjunctive queries for label systems of a type that abstracts both the traditional set-relations and fuzzy sets. We also present a different necessary and sufficient condition for containment of a restricted class of conjunctive queries for a label system that abstracts relations as multisets. Finally, we show that containment of unions of conjunctive queries is decidable for label systems of the first type and undecidable for label systems of the second type This result underscores the fundamental difference between viewing relations as sets and as multisets, and motivates a closer examination of relations as multisets, given their importance in SQL.

The work of Y. E. Ioannidis was partially supported by the National Science Foundation under Presidential Young Investigator Grant IRI-9157368 and by grants from DEC, IBM, HP, AT & T, and Informix. The work of R Ramakrishnan was partially supported by a David and Lucile Packard Foundation Fellowship in Science and Engineering, by the National Science Foundation under a Presidential Young Investigator Award and under Grant IRI-9011563, and by grants from DEC, Tandem, and Xerox.

Authors' address Computer Sciences Department, University of Wisconsin, Madison, WI 53706.

Categories and Subject Descriptors: F.m [**Theory of Computation**]: Miscellaneous; H.1.1 [**Models and Principles**]: Systems and Information Theory; H.2.3 [**Database Management**]: Languages—*query languages*; H.2.4 [**Database Management**]: Systems—*query processing*

General Terms: Algorithms, Languages, Theory

Additional Key Words and Phrases: Conjunctive queries, multisets, query containment and equivalence, query optimization

1. INTRODUCTION

The problem of syntactically characterizing containment and equivalence of conjunctive queries was addressed in the late 1970s by the work of Chandra and Merlin [1977] and by the tableau work of Aho et al. [1979]. In these pioneering papers, conjunctive queries were seen as functions from sets (of tuples) to sets, and containment was naturally defined based on set inclusion. Even in SQL, however, queries over *multisets* of tuples may be posed. In fact, relations are treated as multisets by default, with duplicates being eliminated only after explicit requests. Thus, in order to reason about equivalence of a large class of SQL queries, it is necessary to consider a generalization of conjunctive queries, in which relations are interpreted as multisets of tuples: Our view of a relation as a *set* of types must be generalized.

In this paper we study conjunctive queries over databases in which each tuples has an associated *label* [Ioannidis and Wong 1991]. This generalized notion of a database allows us to consider relations that are *fuzzy sets* or *multisets*, in addition to the case of relations as sets. We develop results on the decidability of conjunctive query containment for different *label systems*, that is, different conditions on the labels that can be associated with tuples.

In the rest of this section, we motivate our work, and describe our results and their significance.

1.1 Motivation for Generalized Conjunctive Queries

The unit of optimization in current relational optimizers is an SQL block, which corresponds closely to a conjunctive query. The basic idea behind most query optimizers is to examine a collection of equivalent evaluation plans for a given query and to use the plan with the least estimated cost. In addition, many advanced optimization techniques require an earlier, rewriting, step, where declarative queries are rewritten into equivalent ones, thus expanding the opportunities for optimization at the lower, algebraic (plan-based), level. The rewriting is often based on comparing the original query with (possibly pieces of) other queries with respect to equivalence or containment.

As a first example of such advanced query optimization techniques, consider the use of materialized views. In this case, a given query is compared against multiple view definitions (which may be seen as queries as well) to identify which of them may be combined to answer the query. This comparison is essentially testing for query containment. Supporting materialized

views to process and optimize queries is the topic of much recent work [Blakeley et al. 1986; Chaudhuri et al. 1995; Tsatalos et al. 1994], and designers are considering offering this option in their future systems; for example, Microsoft is contemplating the use of GMAPs [Tsatalos et al. 1994] in their system [Graefe 1995]. As a second example, consider the related problem of processing correlated nested SQL queries, where one of the alternatives is to process a more general nested query whose result can then be used to process the outer query. Figuring out what the appropriate more general nested query is involves query containment. With the release of the TPC-D benchmark, in the last year or so much research has started on such transformations in the context of processing and optimizing aggregate queries. As a third example, consider semantic query optimization. In this case, the system tries to figure out if integrity constraints (which may be seen as queries) can be used to modify a given query into a semantically equivalent one that may be more efficiently executed [King 1981; Shenoy and Ozsoyoglu 1987]. This is only possible if the antecedent of the integrity constraint, seen as a query, is contained in the given query. As a final example, in multiple/global query optimization, several queries are compared pairwise to identify equivalent common subqueries (subexpressions). This gives us the option of evaluating such subexpressions only once, hopefully resulting in an overall more efficient combination of plans [Grant and Minker 1981; Sellis 1986]. Again, these comparisons involve query containment tests, as the common subexpressions contain each of the full queries. From all of these examples, it should become clear that the questions of containment and equivalence of declarative queries are central to the design and implementation of many advanced query optimizers.

Currently, the formal results on conjunctive query equivalence and containment apply only when we regard a relation as a set of tuples [Chandra and Merlin 1977; Sagiv and Yannakakis 1980]. As mentioned, however, the *default* in SQL (unless **distinct** is used) is that duplicates are not eliminated, and the cardinalities of the tuples in the answer are significant. Hence, to reason rigorously about containment/equivalence in the context of current SQL query optimizers, we must consider conjunctive queries over relations that are *multisets* of tuples. This is a major motivation for the results presented in this paper, which provide a theoretical foundation for conjunctive query equivalence and containment over relations that are multisets. Moreover, these results are cast in a more general framework, and in fact deal with relations as fuzzy sets and with the traditional case of relations as sets as well. Our results demonstrate that the containment problems for sets and multisets are quite different in nature. Thus, we believe that they represent a necessary complement to the earlier results on conjunctive query containment [Chandra and Merlin 1977; Sagiv and Yannakakis 1980], which are widely recognized as foundational from a theoretical standpoint.

Next, we present two examples, one from view materialization and one from multiple query optimization, illustrating the query containment/equivalence problem and the significance of addressing it when relations are treated as multisets.

Example 1.1. Consider the following SQL view definitions:

```
create view EMPHRS1(ename, hours) as
  select distinct e.ename, e.hours
  from EMP e,
create view EMPHRS2(ename, hours) as
  select e.ename, e.hours
  from EMP e.
```

The database relation EMP has columns "ename," "dname," and "hours"; intuitively, each employee works in one or more departments for a certain number of hours. Users write queries that refer to the views EMPHRS1 and EMPHRS2, and we wish to evaluate these queries efficiently. If queries on EMPHRS1 are common, the system might choose to materialize this view. Now, given a query on EMPHRS2, when can we use the materialized version of EMPHRS1 to answer it? Such an optimization becomes especially important in an environment containing large numbers of complex, related view definitions, as is the case for instance in decision support applications.

To answer this question, we must determine when the two view definitions are *equivalent*. If EMPHRS1 and EMPHRS2 are regarded as sets of tuples, they are equivalent. However, if we take into account the number of copies of each tuple, they are different. For example, an employee could work five hours each in the Toy Department and the Sales Department, leading to two identical tuples in EMPHRS2, but only one tuple in EMPHRS1.

Consider the following query:

```
select ename, max(hours)
from EMPHRS2.
```

The difference between EMPHRS1 and EMPHRS2 is not important for this query, and we can use a materialized version of EMPHRS1, if it is available, in place of the reference to EMPHRS2. On the other hand, if the aggregate *max(hours)* is replaced by *sum(hours)*, the difference becomes crucial, and we cannot substitute EMPHRS1 for EMPHRS2. To make this distinction correctly, the SQL optimizer must recognize that the two view definitions are different under multiset semantics.

The next example illustrates a more complex optimization scenario in which, again, equivalence of queries over relations as multisets plays a central role.

Example 1.2. Consider the relational schema of a large corporation's database, which includes the following three relations:

MANUF(*mno*, location,...)
TRANSP(*tno*, location,...)
WAREHS(*wno*, location,...).

Attributes in italic are primary keys, "location" is the city where the manufacturing plant, the warehouse, or the transportation unit's headquarters are located, and other attributes are omitted for simplicity. Assume that, in a

downsizing effort of the company, to decide relocations of various operations, the following three SQL queries are submitted, and the system tries to optimize and then execute them together:

```
Q1: select m.mno, count(m.location)
    from MANUF m, TRANSP t
    where m.location = t.location
    group-by m.mno,
Q2: select w.wno, count(w.location)
    from WAREHS w, TRANSP t
    where w.location = t.location
    group-by w.wno,
Q3: select m.mno, w.wno, count(*)
    from MANUF m, WAREHS w, TRANSP t
    where m.location = t.location
      and w.location = t.location
    group-by m.mno, w.wno.
```

Q1 requests the number of transportation units based on the city of each manufacturing plant. Q2 requests the number of transportation units based on the city of each warehouse. Finally, Q3 requests the number of transportation units based on the city of each pair of colocated manufacturing plant and warehouse.

Most relational systems process aggregate queries by first processing aggregate-free subqueries to generate temporary relations containing the necessary data and then by performing the appropriate aggregations. The subqueries respectively corresponding to Q1, Q2, and Q3 are as follows:

```
SQ1: select m.mno, m.locations
       from MANUF m, TRANSP t
       where m.location = t.location
SQ2: select w.wno, w.location
       from WAREHS w, TRANSP t
       where w.location = t.location
SQ3: select m.mno, w.wno
       from MANUF m, WAREHS w, TRANSP t
       where m.location = t.location
       and w.location = t.location.
```

In the above SQL queries, it is natural to consider the relations named in the **from** clause to be sets of tuples. However, since the keyword **distinct** is not used in their formulation the resulting relations are *multisets*. For example, if there are three transportation units in San Francisco, which is the city of warehouse #34, there are three copies of the tuple (#34, San Francisco) in the result of SQ2. Retention of duplicates is critical in this case, since it enables us to count these tuples on a per-warehouse basis and to generate the information requested by Q2.

Instead of directly processing SQ3, a seemingly plausible alternative for a multiple query optimizer is to join the results of SQ1 and SQ2 on "location," thus reducing the number of required join operations. The query correspond-

ing to this approach is the following:

```
SQ4: select m.mno, w.wno
       from MANUF m, WAREHS w, TRANSP t, TRANSP t'
       where m.location = t.location
         and w.location = t'.location
         and m.location = w.location.
```

It is clear that SQ3 and SQ4 generate the exact same *sets* of (mno, wno) pairs, that is, pairs of colocated manufacturing plants and warehouses with at least a transportation unit based in the same city. (This may also be formally proved by applying the results mentioned earlier [Chandra and Merlin 1977].) Nevertheless, executing SQ4 instead of SQ3 is wrong in this case, because the subsequent aggregations require that the duplicates must be retained, and SQ3 and SQ4 generate different numbers of duplicates of each (mno, wno) pair. Specifically, SQ3 generates each (mno, wno) pair as many times as there are transportation units based on the city of the pair, whereas SQ4 generates each (mno, wno) pair as many times as there are *pairs* of transportation units based on the city of the pair, a rather meaningless result. Viewed as conjunctive queries, SQ3 is *contained* in SQ4 when relations are treated as multisets, but not vice versa, whereas they are *equivalent* when relations are treated as sets. Thus, these notions are different depending on how relations are treated. If this distinction is not made, systems may end up producing erroneous results.

Example 1.2 illustrates the main motivation for this work and the importance of the presented results. As we have seen, correctness of many realistic query optimization scenarios depends on the existence of techniques that compare queries with respect to containment or equivalence when relations are treated as multisets. Similar needs also arise for a variety of other interpretations of relations, for example, fuzzy sets, which are also included in this study.

In developing our results, we have chosen to model generalized relations as sets of tuples with associated *labels* [Ioannidis and Wong 1991]. As an example, a positive integer multiplicity (or number of copies, intuitively) is associated with every tuple in a multiset. As noted, current relational query languages such as SQL support a multiset semantics, and this has been found to be extremely useful for many common queries. Another example is fuzzy sets, where an arbitrary number in $[0, 1]$ (or certainty factor, intuitively) is associated with every tuple in a fuzzy set.

The conditions under which our results are applicable are stated in terms of the algebraic properties of the set of labels and associated label operations. Although it is true that the results for, say, multisets could be obtained without the abstraction of label systems, the framework has several important advantages. First, our results have more generality; for example, a single label system, type A, defined later, covers both fuzzy sets and traditional sets, and thus, our results about type A systems apply to both. Without the abstraction of labels and label systems, the same results would have to be

established independently for relations as sets and for relations as fuzzy sets, although the essence of the proofs is the same. Second, by abstracting the essential properties upon which our proofs depend, the framework of label systems allows us to understand more fully the basic differences between queries over different kinds of relations. Finally, it is possible that our results are applicable to other kinds of relational extensions with properties similar to the label systems that we study; the formulation of our results in terms of label systems makes it easy to extend our results, since it is only necessary to show that the new kind of relations satisfies the (simple) conditions for a given label system.

1.2 Results

Our main results are the following: First, we show that containment of both individual conjunctive queries and unions of them is decidable but NP-complete for a label system that generalizes relations as sets and relations as fuzzy sets. For the specific case of relations as sets, these results had been shown earlier as well [Chandra and Merlin 1977; Sagiv and Yannakakis 1980]. Second, we show that containment of unions of conjunctive queries is undecidable for a label system that captures relations as multisets. For individual conjunctive queries within that label system, we present sufficient conditions for containment, which are necessary and sufficient when the queries are in a restricted (but very common) form. Decidability of containment in the general case, however, remains an open problem.

Recently, Chaudhuri and Vardi [1993] studied multisets and independently discovered some of our results. (We discuss this further in Section 8.)

1.3 Outline of the Paper

The paper is organized as follows: In Section 2 we present some background material and also develop our generalized notion of a database, which we consider to be one of our important contributions. We introduce label systems and identify two types of label systems (types A and B), for which results are presented in this paper. In Section 3 we establish a general necessary condition for conjunctive query containment that is satisfied by a large class of label systems, including type A and type B systems. In Section 4 we show that the necessary condition identified in Section 3 is also sufficient for conjunctive query containment over databases with label systems of type A. This result strictly generalizes the condition of Chandra and Merlin [1977] and is also applicable to databases that deal with fuzzy sets. In Section 5 we present a sufficient condition for containment over databases with label systems of type B. For restricted classes of conjunctive queries in which there are no repeated predicates, we prove that this condition is necessary as well as sufficient. These results are applicable to databases that deal with multisets. In Section 6 we study unions of conjunctive queries. For label systems of type A, we present a necessary and sufficient condition for containment of unions of conjunctive queries, which generalizes a similar result for sets [Sagiv and Yannakakis 1980]. For label systems of type B, we prove that the

problem is, in general, undecidable. In Section 7 we establish some results concerning a class of label systems that captures the form of fuzzy-set reasoning used in expert systems like MYCIN, as well as label systems that are motivated by graph-theory problems. We discuss related work in Section 8. In Section 9 we outline several interesting directions for future work and summarize our results.

2. BACKGROUND AND BASIC DEFINITIONS

In this section we review some standard concepts and develop our model of databases and conjunctive queries. In particular, the definitions of label systems, databases, and conjunctive query containment are generalizations of the usual definitions.

2.1 Label Systems

We first introduce *label systems*, which are at the heart of our generalization of conjunctive queries. Intuitively, the ideal is to extend the relational model by attaching a *label* to each tuple. Labels can be used to capture certainty factors or multiplicity, for example. In our framework, labels are drawn from a special domain, and their usage must conform to certain rules that enable us to interpret labels using an appropriate semantics. It is possible to place different sets of rules upon labels, leading to different semantics and to differences in the difficulty of deciding issues like containment of conjunctive queries. In this paper we study two types of label systems, each characterized by a small collection of algebraic rules.

Definition 2.1. A *label system* $\mathscr{L}$ is a quintuple $\mathscr{L} = \langle L, *, +, 0, \leq \rangle$ such that

- L is a domain of at least two labels equipped with a partial order $\leq$;
- $*$ is a binary operation (called *multiplication*) on L that is associative and commutative;
- $+$ is a binary operation (called *addition*) on L that is associative and commutative; and
- 0 is the additive identity in L and is also an annihilator with respect to multiplication and the least element with respect to the partial order $\leq$, that is, $\forall a \in L$, $a + 0 = a$, $a * 0 = 0$, and $0 \leq a$.

When $a \leq b$ and $a \neq b$, then we use the notation $a < b$. The first label system studied in this paper is presented here:

Definition 2.2. A label system $\mathscr{L} = \langle L, *, +, 0, \leq \rangle$ is of *type A* if it satisfies the following:

(A1) $\forall a, b \in L - \{0\}$, $0 < a * b \leq a$.
(A2) $\forall a \in L$, $a * a = a$.
(A3) $\forall a, a', b, b' \in L$, $(a \leq a' \text{ and } b \leq b') \Rightarrow a + b \leq a' + b'$.
(A4) $\forall a, b \in L$, $a + b \leq a$ or $a + b \leq b$.

Note that condition (A2) states that multiplication is idempotent. Also, conditions (A3) and (A4) are equivalent to stating that $\leq$ is a total order and that $+$ is equal to *max* with respect to that total order. We chose the given presentation to make the analogy with condition (B2) of label systems of type B clearer.

Example 2.1. Binary truth values as well as certainty factors are two important examples of label systems of type A. For the case of truth values, $L = \{\text{"false"}, \text{"true"}\}$. The operation $*$ is logical *and*, the operation $+$ is logical *or*. The label "false" corresponds to 0 and serves as the additive identity of the annihilator for multiplication. Finally, $\leq$ is such that "false" $\leq$"true." It is easy to verify that all of the conditions for a type A label system are satisfied.

For certainty factors, L is equal to the set of real numbers between 0 and 1. The operation $*$ is *min*, and the operation $+$ is *max*. (This is the case, e.g., in the system proposed by van Emden [1986].) The number 0 serves as the additive identity and the multiplicative annihilator. Finally, $\leq$ is the regular total order on numbers. Again, all conditions for a type A label system are satisfied.

Definition 2.3. A label system $\mathscr{L} = \langle L, *, +, 0, \leq \rangle$ is of *type B* if it satisfies the following. The second label system studied in this paper is presented next:

(B1) $\forall a, b \in L - \{0\}$, $a \leq a * b$.
(B2) $\forall a, a', b, b' \in L$, $(a \leq a' \text{ and } b \leq b') \Rightarrow a + b \leq a' + b'$.
(B3) $\forall a \in L$, $\exists a' \in L$, $a < a'$.

Example 2.2. Arithmetic is the most important label system of type B. Specifically, L is equal to the set of nonnegative integers. The operations $*$ and $+$ are the usual multiplication and addition of numbers, respectively. The number 0 serves as the additive identity and the multiplicative annihilator. Finally, $\leq$ is the usual total order on numbers. Again, it is easy to verify that all of the conditions for a type B label system are satisfied.

2.2 Relation Instances with Respect to a Label System

In order to deal with multiple interpretations of relations, our treatment requires that a relation instance is defined in a new, generalized way. Before presenting the formal definition, we illustrate it with an example from a traditional SQL-based context to provide intuition. After the definition, additional examples focus on aspects of the definition itself.

For the purposes of this work, a relation instance is defined as a function from all possible tuples that could be formed from the domains of the relation's attributes to the labels of a system according to which the relation is interpreted. For example, consider the MANUF relation of Example 1.2 interpreted as a multiset. Assuming that there are three manufacturing plants #1, #7, and #9 located, respectively, in Boston, San Francisco, and

Boston, the corresponding MANUF instance would be formally specified as the following function:

$$\begin{aligned}(\#1, \text{Boston}) &\rightarrow 1,\\ (\#1, \text{San Francisco}) &\rightarrow 0,\\ (\#7, \text{Boston}) &\rightarrow 0,\\ (\#7, \text{San Francisco}) &\rightarrow 1,\\ (\#9, \text{Boston}) &\rightarrow 1,\\ (\#9, \text{San Francisco}) &\rightarrow 0.\end{aligned}$$

Similarly, if Boston has 3 transportation units and San Francisco has 5 transportation units, the result of query SQ1 would be formally specified as follows:

$$\begin{aligned}(\#1, \text{Boston}) &\rightarrow 3,\\ (\#1, \text{San Francisco}) &\rightarrow 0,\\ (\#7, \text{Boston}) &\rightarrow 0,\\ (\#7, \text{San Francisco}) &\rightarrow 5,\\ (\#9, \text{Boston}) &\rightarrow 3,\\ (\#9, \text{San Francisco}) &\rightarrow 0.\end{aligned}$$

That is, each possible tuple is associated with a nonnegative integer multiplicity, indicating the number of times it appears in the relation. This is formalized in the following:

Definition 2.4. Let Q be an n-ary predicate symbol, and let $D_1, \ldots, D_n$ be the domains of values of the arguments of Q. Predicate symbols over the same cross product of domains are called *compatible*. Also, let $\mathscr{L}$ be a label system with domain L. A *relation instance* for Q with respect to $\mathscr{L}$ is a total function[1] $\mathbf{Q}$: $D_1 \times \cdots \times D_n \rightarrow L$. Relation instances for compatible predicate symbols are called *compatible*. A *database instance* with respect to a label system $\mathscr{L}$ is a set of relation instances. Any element of the domain $D_1 \times \cdots \times D_n$ is called a *tuple* and is denoted by $\langle d_1, \ldots, d_n \rangle$, for $d_i \in D_i$, $1 \leq i \leq n$.

Example 2.3. The traditional view of relations as sets is captured through the first label system of type A given in Example 2.1. Usually, a relation instance is compactly represented as the subset of its domains containing the tuples that map to "true." Fuzzy sets may be captured through the second label system of type A given in Example 2.1, where each possible tuple is mapped to a certainty factor between 0 and 1. Finally, multisets may be captured through the label system of type B given in Example 2.2. In this case, as illustrated earlier, each tuple is mapped to a nonnegative integer, which indicates the number of copies of the given tuple in the multiset.

[1]Functions that denote relation instances appear in boldface.

In the sequel, whenever we refer to a database instance it is understood that it is with respect to a given label system. One could argue that an equivalent, and perhaps more natural, formulation of different interpretations of relations is obtained by adding an explicit column (field) that stores the label assigned to each tuple. The resulting extended relations would then be treated as regular sets. For example, instead of using atomic formulas of the form $Q(t_1, t_2, \ldots, t_n)$, one could use $Q(t_1, t_2, \ldots, t_n, l)$, where the domains of t_i remain unchanged and the domain of l is L. In fact, this alternative formulation would most likely be used as the basis for storing relation instances based on nontraditional label systems. However, as will be made clear shortly, explicit representation of the additional label column in conjunctive queries is not enough. The label column must be given a special status and must be treated in a special way that respects the semantics associated with it. Otherwise, wrong results will be obtained in several cases. The formalism of label systems, introduced in this section, makes the special semantics of labels explicit, and is therefore adopted for the rest of the paper.

2.3 Conjunctive Queries

Conjunctive queries are formally defined in this subsection. We have already mentioned that SQL blocks, that is, *flat* SQL queries, correspond closely to them. Specifically, conjunctive queries are essentially logic-based specifications of such SQL queries, whose syntax is more natural to deal with in formal work, as earlier studies of containment and equivalence have demonstrated. For example, consider the SQ1 query, which is expressed in flat SQL. The equivalent logic-based syntax for this query is

$$\text{MANUF}(\text{mno}, \text{loc}) \wedge \text{TRANSP}(\text{tno}, \text{loc}) \rightarrow \text{SQ1result}(\text{mno}, \text{loc}).$$

The join on location is captured by the use of the common variable loc in the MANUF and TRANSP predicates, whereas the appearance of mno and loc on the right-hand side of $\rightarrow$ indicates the query projections. Conjunctive queries require a predicate name for the result, which was arbitrarily chosen here to be SQ1result. The formal syntax of conjunctive queries is defined as follows:

Definition 2.5. A *conjunctive query* is a first-order formula of the form $A_1 \wedge A_2 \wedge \cdots \wedge A_m \xrightarrow{f} C$. All of the variables appearing in the formula are (implicitly) universally quantified. The formula to the left of $\rightarrow$ is called the *antecedent*, and that to the right of $\rightarrow$ is called the *consequent*. Each one of $C, A_1, A_2, \ldots, A_m$ is an *atomic formula* of the form $Q(t_1, t_2, \ldots, t_n)$, where Q is a relation (predicate) symbol and t_i, $1 \leq i \leq n$, is a variable. The predicate symbol in the consequent does not appear in the antecedent (i.e., recursion is not allowed). Variables that appear in the consequent are called *distinguished* and must appear in the antecedent as well, while all others are called *nondistinguished*. Finally a conjunctive query has an associated unary function f: $M \rightarrow M$, for some set M equipped with a partial order $\leq$ such that $\exists 0 \in M$, $\forall a \in M - \{0\}$, $0 < a$, and $0 < f(a)$.

Based on Definition 2.5, a conjunctive query is a purely syntactic construct. It obtains semantics only when it is interpreted based on a particular label system and processed based on that. Several distinct semantics can be given to the same conjunctive query by interpreting it based on different label systems. The only requirements for using a label system $\mathscr{L}$ to interpret a conjunctive query associated with a function f: $M \rightarrow M$ is that $L \subseteq M$ and $\forall a \in L$, $f(a) \in L$. The role of the function f will be clear when we discuss *valuations* of conjunctive queries.

Whenever we present a conjunctive query without an associated function f, then it is assumed that f is the identity function. Also, we use the convention that the atomic formula in the consequent of a conjunctive query always has the distinguished predicate symbol P, possibly subscripted with an indicator of the specific conjunctive query. Finally, unless otherwise noted, the phrases *atomic formulas of a conjunctive query* and *predicates of a conjunctive query* are used to refer to those in the antecedent.

2.4 The Result of a Conjunctive Query

To define the result of applying a conjunctive query to a database instance, we first have to capture the essence of inferring a single tuple in that result. Such a single-tuple inference, or derivation, is realized by instantiating each atomic formula in the antecedent of a conjunctive query with a single tuple.

For the purposes of this paper, such a derivation corresponds to the traditional combination of tuples (one from each participating relation) enhanced with a multiplication of their labels, which results in the label assigned to the derived tuple. For instance, consider query SQ1 of Example 1.2 with its relations interpreted as multisets and instantiated as mentioned earlier. Combining the (#1, Boston) tuple of MANUF, which is labeled 1, with, say, the (#24, Boston) tuple of TRANSP, which is labeled 1, generates the result tuple (#1, Boston) with label 1, capturing the existence of some transportation unit in the city of plant #1. Similarly, combining the (#7, Boston) tuple of MANUF, which is labeled 0, with the above (#24, Boston) tuple of TRANSP, generates the result tuple (#7, Boston) with label 0, since plant #7 is not even in Boston.

Valuations of conjunctive queries are used as the formal mechanism for single-tuple derivations in the query result and are defined next. After the definition, additional examples focus on aspects of the definition itself.

Definition 2.6. Consider a conjunctive query α of the form $A_1 \wedge A_2 \wedge \cdots \wedge A_m \overset{f}{\rightarrow} C$. A *valuation* θ of α is a pair of functions $\langle \theta_v, \theta_l \rangle$. Function θ_v is from the variables of α to some set of constants, and function θ_l is from the atomic formulas of α to labels such that

$$\theta_l(C) = f\left(\prod_{i=1}^{m} \theta_l(A_i) \right).$$

Applying θ on α gives an *instance* of α.

Example 2.4. To illustrate Definition 2.6 and the fact that it captures the intuition when dealing with familiar label systems, we discuss the cases of

Table I. Two Valuations θ and θ' in a Label System that Interprets Relations as Sets

Conjunctive query item	θ	θ'
x	$\theta_v(x) = \mathrm{K}$	$\theta'_v(x) = \mathrm{K}$
z	$\theta_v(z) = \mathrm{L}$	$\theta'_v(z) = \mathrm{L}$
y	$\theta_v(y) = \mathrm{N}$	$\theta'_v(y) = \mathrm{N}$
$Q(x, z)$	$\theta_l(Q(x, z)) =$ "true"	$\theta'_l(Q(x, z)) =$ "false"
$R(z, y)$	$\theta_l(R(z, y)) =$ "true"	$\theta'_l(R(z, y)) =$ "true"

relations as sets and relations as multisets. Consider the following conjunctive query:

$$Q(x, z) \wedge R(z, y) \rightarrow P(x, y).$$

Assume that the domain of all argument positions of all relations is $D = \{K, L, M, N\}$.

Consider the label system that interprets relations as sets (first label system in Example 2.1), and let the two valuations θ and θ' be as shown in Table I. Valuation θ represents the case where tuple (K, L) is in Q and tuple (L, N) is in R. Then, tuple (K, N) should be in the result. Definition 2.6 captures this intuition since, based on the definition of multiplication in the set label system, θ_l applied on the consequent of the conjunctive query yields

$$\theta_l(P(x, y)) = \theta_l(Q(x, z)) \text{ and } \theta_l(R(z, y)) = \text{"true"}.$$

On the other hand, valuation θ' represents the case where tuple (K, L) is not in Q and tuple (L, N) is in R, and therefore, tuple (K, N) should not be in the result. Again, Definition 2.6 captures this intuition since θ_l applied on the consequent of the conjunctive query yields

$$\theta_l(P(x, y)) = \theta_l(Q(x, z)) \text{ and } \theta_l(R(z, y)) = \text{"false"}.$$

We now turn our attention to the label system that interprets relations as multisets. Consider the two valuations θ and θ' shown in Table II. Valuation θ represents the case where tuple (K, L) appears in Q twice and tuple (L, N) appears in R five times. Then, tuple (K, N) should appear in the result ten times. Definition 2.6 captures this intuition since, based on the definition of multiplication in the multiset label system, θ_l applied on the consequent of the conjunctive query yields

$$\theta_l(P(x, y)) = \theta_l(Q(x, z)) * \theta_l(R(z, y)) = 2 * 5 = 10.$$

On the other hand, valuation θ' represents the case where tuple (K, L) does not appear in Q at all, and therefore, independent of the number of times tuple (L, N) appears in R, tuple (K, N) should not be in the result. Again, Definition 2.6 captures this intuition since θ_l applied on the consequent of the conjunctive query yields

$$\theta_l(P(x, y)) = \theta_l(Q(x, z)) * \theta_l(R(z, y)) = 0 * 5 = 0.$$

Table II. Two Valuations θ and θ' in a Label System that Interprets Relations as Multisets

Conjunctive query item	θ	θ'
x	$\theta_v(x) = \mathrm{K}$	$\theta'_v(x) = \mathrm{K}$
z	$\theta_v(z) = \mathrm{L}$	$\theta'_v(z) = \mathrm{L}$
y	$\theta_v(y) = \mathrm{N}$	$\theta'_v(y) = \mathrm{N}$
$Q(x, z)$	$\theta_l(Q(x, z)) = 2$	$\theta'_l(Q(x, z)) = 0$
$R(z, y)$	$\theta_l(R(z, y)) = 5$	$\theta'_l(R(z, y)) = 5$

Definition 2.7. Consider two conjunctive queries α and β with compatible consequents whose distinguished variables in the ith argument position, $1 \leq i \leq n$, are a_i and b_i, respectively. Let θ^α and θ^β be valuations of α and β, respectively. If, for all $1 \leq i \leq n$, $\theta_v^\alpha(a_i) = \theta_v^\beta(b_i)$, then θ^α and θ^β are *compatible*.

Note that in Definition 2.7 α and β do not have to be distinct. For valuations of the same conjunctive query, it is easy to show that compatibility is an equivalence relation over valuations. Compatible valuations are important because they capture derivations of the same tuple. This is a key concept needed in defining the result of applying a conjunctive query to a database instance, because the labels assigned to a tuple by compatible valuations need to be combined to determine the label assigned to it in the overall result.

Definition 2.8. A valuation θ of a conjunctive query α is *true* with respect to a database instance if, for every atomic formula $Q(x_1, \ldots, x_n)$ in α, $\mathbf{Q}(\langle \theta_v(x_1), \ldots, \theta_v(x_n) \rangle) = \theta_l(Q(x_1, \ldots, x_n))$.

We finally move to the definition of the result of applying a conjunctive query to a database instance. This is essentially done by combining the individual tuple derivations defined earlier. In particular, separate derivations of the same tuple are combined by adding the labels assigned to the tuple in each derivation. The outcome of the addition is the final label assigned to the tuple in the overall result. For example, consider query SQ1 of Example 1.2 with its relations interpreted as multisets and instantiated as mentioned earlier. The (#1, Boston) tuple of its result is derived three times, once for each of the TRANSP tuples with location = Boston combined with the (#1, Boston) tuple of MANUF. Each derivation assigns the label 1 to the resulting tuple, whose final label is therefore equal to 3, capturing the fact that three transportation units are based in the city of plant #1, as expected. The result of a query is formally defined in Definition 2.9. After the definition, additional examples focus on aspects of the definition itself.

Definition 2.9. Consider a conjunctive query α of the form $A_1 \wedge A_2 \wedge \cdots \wedge A_m \xrightarrow{f} C$ and a database instance I. Let Θ be the set of all valuations of α that are true with respect to I. Partition Θ based on the equivalence relation of valuation compatibility, and let Θ_i denote the partition that

Table III Instances of Q and R in a Label System that Interprets Relations as Sets

Domain member d	$\mathbf{Q}(d)$	$\mathbf{R}(d)$
(K, L)	"true"	"false"
(K, M)	"true"	"false"
(L, N)	"false"	"true"
(M, N)	"false"	"true"
All others	"false"	"false"

Table IV. Four Valuations that are Compatible in Generating Tuple (K, N) in the Consequent of the Conjunctive Query

Item	θ	θ'	θ''	θ'''
x	$\theta_v(x) = \mathrm{K}$	$\theta'_v(x) = \mathrm{K}$	$\theta''_v(x) = \mathrm{K}$	$\theta'''_v(x) = \mathrm{K}$
z	$\theta_v(z) = \mathrm{K}$	$\theta'_v(z) = \mathrm{L}$	$\theta''_v(z) = \mathrm{M}$	$\theta'''_v(z) = \mathrm{N}$
y	$\theta_v(y) = \mathrm{N}$	$\theta'_v(y) = \mathrm{N}$	$\theta''_v(y) = \mathrm{N}$	$\theta'''_v(y) = \mathrm{N}$
$Q(x, z)$	$\theta_t(Q(x, z)) =$ "false"	$\theta'_t(Q(x, z)) =$ "true"	$\theta''_t(Q(x, z)) =$ "true"	$\theta'''_t(Q(x, z)) =$ "false"
$R(z, y)$	$\theta_t(R(z, y)) =$ "false"	$\theta'_t(R(z, y)) =$ "true"	$\theta''_t(R(z, y)) =$ "true"	$\theta'''_t(R(z, y)) =$ "false"

generates tuple t in the distinguished variables of α. The *result* of applying α to I (denoted by $\alpha(I)$) is a relation instance $\mathbf{P}$, that is, a function, such that

$$\mathbf{P}(t) = \sum_{\theta \in \Theta_t} \theta_I(C) = \sum_{\theta \in \Theta_t} f\left(\prod_{i=1}^{m} \theta_I(A_i) \right).$$

The intuition behind Definition 2.9 is that all derivations of a single tuple based on the conjunctive query are grouped together. Each derivation assigns a different label to it, and these labels are combined (through addition) to find the label of the tuple in the result.

Example 2.5. To illustrate Definition 2.9, we discuss the cases of relations as sets and relations as multisets. We use the same conjunctive query as in Example 2.4,

$$Q(x, z) \wedge R(z, y) \rightarrow P(x, y),$$

with the same domain for all argument positions of all relations, $D = \{K, L, M, N\}$.

Consider the label system that interprets relations as sets, and let the instances of Q and R be as shown in Table III. There are 81 valuations that are true with respect to Table III (three variables, each assigned to one of four possible elements in its domain). Four of these valuations are compatible in generating tuple (K, N) in the consequent of the conjunctive query, and are given in Table IV. Valuations θ and θ''' generate the label "false" for tuple (K, N), while valuations θ' and θ'' generate the label "true." By adding, that is, by taking the logical *or* of these four labels, we obtain the label "true" for tuple (K, N). This was to be intuitively expected, because tuples (K, L) and (L, N) are in $\mathbf{Q}$ and $\mathbf{R}$, respectively, and therefore their composition (K, N) should be in the result. Similarly, tuple (K, N) is placed in the result as the

Table V. Four Valuations Compatible in Generating Tuple (K, M) in the Consequent of the Conjunctive Query

Item	θ	θ'	θ''	θ'''
x	$\theta_v(x) = \mathrm{K}$	$\theta'_v(x) = \mathrm{K}$	$\theta''_v(x) = \mathrm{K}$	$\theta'''_v(x) = \mathrm{K}$
z	$\theta_v(z) = \mathrm{K}$	$\theta'_v(z) = \mathrm{L}$	$\theta''_v(z) = \mathrm{M}$	$\theta'''_v(z) = \mathrm{N}$
y	$\theta_v(y) = \mathrm{M}$	$\theta'_v(y) = \mathrm{M}$	$\theta''_v(y) = \mathrm{M}$	$\theta'''_v(y) = \mathrm{M}$
$Q(x, z)$	$\theta_l(Q(x, z)) =$ "false"	$\theta'_l(Q(x, z)) =$ "true"	$\theta''_l(Q(x, z)) =$ "true"	$\theta'''_l(Q(x, z)) =$ "false"
$R(z, y)$	$\theta_l(R(z, y)) =$ "false"	$\theta'_l(R(z, y)) =$ "false"	$\theta''_l(R(z, y)) =$ "false"	$\theta'''_l(R(z, y)) =$ "false"

Table VI. Instances of Q and R in a Lebel System that Interprets Relations as Multisets

Domain member d	$\mathbf{Q}(d)$	$\mathbf{R}(d)$
(K, L)	2	0
(K, M)	3	0
(L, N)	0	5
(M, N)	0	2
All others	0	0

Table VII. Four Compatible Valuations that Generate Tuple (K, N) in the Consequent of the Conjunctive Query

Item	θ	θ'	θ''	θ'''
x	$\theta_v(x) = \mathrm{K}$	$\theta'_v(x) = \mathrm{K}$	$\theta''_v(x) = \mathrm{K}$	$\theta'''_v(x) = \mathrm{K}$
z	$\theta_v(z) = \mathrm{K}$	$\theta'_v(z) = \mathrm{L}$	$\theta''_v(z) = \mathrm{M}$	$\theta'''_v(z) = \mathrm{N}$
y	$\theta_v(y) = \mathrm{N}$	$\theta'_v(y) = \mathrm{N}$	$\theta''_v(y) = \mathrm{N}$	$\theta'''_v(y) = \mathrm{N}$
$Q(x, z)$	$\theta_l(Q(x, z)) = 0$	$\theta'_l(Q(x, z)) = 2$	$\theta''_l(Q(x, z)) = 3$	$\theta'''_l(Q(x, z)) = 0$
$R(z, y)$	$\theta_l(R(z, y)) = 0$	$\theta'_l(R(z, y)) = 5$	$\theta''_l(R(z, y)) = 2$	$\theta'''_l(R(z, y)) = 0$

composition of tuples (K, M) and (M, N), but multiple derivations have no effect since relations are interpreted as sets.

Now consider the four compatible valuations that are true with respect to the given database instance and that generate tuple (K, M) in the consequent of the conjunctive query, as given in Table V. All of these generate the label "false" for tuple (K, M), and therefore, its final label is "false" as well. Again, this was to be intuitively expected, because there are no tuples in **Q** and **R** whose composition could generate (K, M).

We now turn our attention to the label system that interprets relations as multisets (Example 2.2). Let the instances of Q and R be as shown in Table VI. The four compatible valuations that are true with the instance given in Table VI and that generate tuple (K, N) in the consequent of the conjunctive query are given in Table VII. Based on the multiplication semantics of $*$ and Definition 2.6, the four valuations generate the labels 0, 10, 6, and 0 for tuple (K, N). By adding these four labels, we obtain the label 16 for the tuple, which is exactly the number of copies one would intuitively expect for it. The composition of tuples (K, L) and (L, N) of **Q** and **R**, respectively, generates 10 of these copies, and the composition of (K, M) and (M, N) generates 6 more.

Table VIII. Four Compatible Valuations that Generate Tuple (K, M) in the Consequent of the Conjunctive Query

Item	θ	θ'	θ''	θ'''
x	$\theta_v(x) = \mathrm{K}$	$\theta'_v(x) = \mathrm{K}$	$\theta''_v(x) = \mathrm{K}$	$\theta'''_v(x) = \mathrm{K}$
z	$\theta_v(z) = \mathrm{K}$	$\theta'_v(z) = \mathrm{L}$	$\theta''_v(z) = \mathrm{M}$	$\theta'''_v(z) = \mathrm{N}$
y	$\theta_v(y) = \mathrm{M}$	$\theta'_v(y) = \mathrm{M}$	$\theta''_v(y) = \mathrm{M}$	$\theta'''_v(y) = \mathrm{M}$
$Q(x, z)$	$\theta_l(Q(x, z)) = 0$	$\theta'_l(Q(x, z)) = 2$	$\theta''_l(Q(x, z)) = 3$	$\theta'''_l(Q(x, z)) = 0$
$R(z, y)$	$\theta_l(R(z, y)) = 0$	$\theta'_l(R(z, y)) = 0$	$\theta''_l(R(z, y)) = 0$	$\theta'''_l(R(z, y)) = 0$

Since we are dealing with multisets, each separate copy counts, and therefore, 10 and 6 must be added. Indeed, this is precisely the number of copies of (K, N) generated by evaluating the above query in SQL.[2]

To conclude the example, consider the four compatible valuations that are true with respect to the given database instance and generate tuple (K, M) in the consequent of the conjunctive query, as shown in Table VIII. All of these valuations generate the label 0 for tuple (K, M), since $R(z, y)$ is always mapped to 0. Thus, the final label of tuple (K, M) is 0, which again is what intuition requires.

At this point, let us consider why explicit representation of labels as additional relation columns is not enough. Consider the conjunctive query used in the previous examples, and add a label column in the participating relations:

$$Q(x, z, l_Q) \wedge R(z, y, l_R) \wedge l_P = g(l_Q, l_R) \rightarrow P(x, y, l_P).$$

We claim that, independent of our choice of function g, the above cannot capture arbitrary interpretations of relations. The main problem is that the final label assigned to a tuple depends on multiple derivations, whereas g can only capture relationships of labels associated with a single tuple derivation. Thus, for example, if g is equal to $*$, then g captures the treatment of labels for a single valuation, but cannot capture the combination of multiple compatible valuations. For conjunctive queries where all variables are distinguished, this suffices. When variables are projected out of the result, however, additional machinery needs to be invoked, which cannot be represented as part of a single valuation. The formulation of the problem as presented in the previous definitions achieves what is desired. Moreover, in our opinion, it is also aesthetically pleasing because it separates syntax from semantics. There is a single syntactic definition of conjunctive queries, on top of which one may impose arbitrary semantics captured via label systems.

We now present some simple queries that illustrate the power of the generalizations that we have proposed. These complement the examples in the Introduction and are given in light of the formal definitions that we have established earlier. Our first example demonstrates the importance of multi-

[2] In SQL, multiset semantics is the *default*, and duplicate elimination must be explicitly requested by using the keyword **distinct**.

sets in languages like SQL. As we have already argued, this is the most compelling reason to study generalized conjunctive queries, since queries such as the ones discussed here are already widely used.

Example 2.6. The following query can be considered a view definition in SQL:

$$Department(dname, floor) \wedge Employee(ename, dname, sal) \rightarrow Deptsals(dname, sal).$$

This query is interpreted based on the label system $\mathscr{L}$ defined in Example 2.2, where each tuple in a relation has an associated cardinality. In this example, it is natural to consider the relations in the antecedents to be sets of tuples. However, unless the keyword **distinct** is used in the formulation of the SQL query, the *Deptsals* relation is a *multiset* of tuples in which a salary value appears as often as there are employees in the department earning that salary. This enables us to query the view, for example, to compute the sum of all salaries on a per-department basis, that is, the budget of each department, by using features such as GROUP-BY and the *sum* aggregate operation. If the view relation was considered to be a set of tuples, we could not compute the budget of a department by examining the view.

One could argue that explicit construction of multiset relations is not necessary to compute budgets, since it is possible to write an SQL query over the *Department* and *Employee* relations that directly groups by department and sums the (multiset of) salaries in each department. Although this is true, it does not diminish the importance of understanding conjunctive queries over relations as multisets, for two reasons. First, even in this single query formulation, it is important to recognize that there is an implicit projection of the salary field that generates a multiset of salaries, and thus, we need to account rigorously for the cardinality of each salary in the projection. Second, for a variety of reasons, it may not be desirable to give users access to the *Department* and *Employee* relations directly. For example, a user who is authorized to see the *Deptsals* view may nonetheless not be authorized to see what each individual earns. In such a situation, the explicit creation of a multiset relation is the most natural solution.

The following example illustrates the generalization to relations as fuzzy sets:

Example 2.7. The following queries attempt to find promising candidates in a football draft. The first query identifies linebackers who are big and have experience, and the second identifies receivers with speed and skill. The data have a degree of uncertainty, in that each fact has an associated "certainty factor" between 0 and 1. These factors are to be interpreted in terms of fuzzy logic, rather than as probabilities. That is, a fact *Big*(john) with an associated label of 0.7 is to be read as "John is quite big," rather than "It is quite likely that John is big." Furthermore, our confidence in the criteria expressed in the two rules is also limited, and thus, each rule has an associated "rule certainty factor." The certainty of a deduced fact is given by the minimum value of the

certainties associated with the facts used in the antecedent of the valuation multiplied (in the arithmetic sense) by the rule certainty factor. If a fact is deduced using several different valuations, the associated certainty factor is given by the maximum of the deduced certainty factors for it. (General rules about certainty factors and why they are sometimes more appropriate than probability-based reasoning are discussed elsewhere [Parsaye and Chignell 1988].)

$$Linebacker(x) \wedge Big(x) \wedge Experienced(x) \stackrel{0.7}{\rightarrow} Candidate(x),$$

$$Receiver(x) \wedge Fast(x) \wedge Skilled(x) \stackrel{0.6}{\rightarrow} Candidate(x).$$

The above desired semantics are captured by interpreting this query based on the second label system (fuzzy sets) defined in Example 2.1. The two conjunctive queries are respectively associated with the functions $f_1(a) = 0.7 * a$ and $f_2(a) = 0.6 * a$, having the real numbers between 0 and 1 as their domain. This gives the rule certainty factor semantics mentioned above.

The label system of type A used in Example 2.7 essentially captures the quantitative deduction framework proposed by van Emden [1986]; ignoring rule certainty factors, it is essentially Zadeh's [1965] treatment of intersection and union of fuzzy sets. Although we do not consider recursive queries, it is interesting to note that van Emden showed a correspondence between his framework and alpha–beta search trees for two-person games.

2.5 Conjunctive Query Containment

We are now ready to define conjunctive query containment in a general setting, where relations are interpreted not necessarily as sets but with respect to some label system. Intuitively, containment is based on comparing the labels assigned to the same tuple in the results of two queries based on the required partial order of the label system. For instance, as we saw in Example 1.2 of the Introduction, SQ3 and SQ4 generate exactly the same set of tuples, but SQ4 contains potentially more and certainty never fewer duplicates of each tuple than SQ3. Thus, based on a comparison of the multiplicities of each tuple, SQ3 is contained in SQ4, but not vice versa.

The above illustration is formalized for a general label system in Definition 2.10. After the definition, a more formal example is also provided.

Definition 2.10. Consider two relation instances $\mathbf{R}_1$ and $\mathbf{R}_2$ for a predicate R with domain $D_1 \times \cdots \times D_n$ with respect to a label system $\mathscr{L}$. $\mathbf{R}_1$ is *contained* in $\mathbf{R}_2$, denoted by $\mathbf{R}_1 \leq_r \mathbf{R}_2$, if, for each tuple $t \in D_1 \times \cdots \times D_n$, $\mathbf{R}_1(t) \leq \mathbf{R}_2(t)$. Clearly, $\leq_r$ is a partial order.

Example 2.8. We again use the label systems corresponding to sets and multisets to illustrate the above definition of containment. If $\mathbf{R}_1 \leq_r \mathbf{R}_2$, then, based on the above definition, there is no tuple t in the domain of $\mathbf{R}_1$ and $\mathbf{R}_2$ such that $\mathbf{R}_1(t) > \mathbf{R}_2(t)$.

For the set case, this is equivalent to the absence of a tuple t for which $\mathbf{R}_1(t) =$ "true" and $\mathbf{R}_2(t) =$ "false," that is, which is in $\mathbf{R}_1$ but not in $\mathbf{R}_2$. This captures precisely the traditional notion of set inclusion, $\mathbf{R}_1 \subseteq \mathbf{R}_2$.

For the multiset case, this is equivalent to the absence of a tuple t for which t has strictly more copies in $\mathbf{R}_1$ than in $\mathbf{R}_2$. Again, this is the traditional notion of inclusion between multisets based on number of copies.

Definition 2.11. For two conjunctive queries α and β, α is *more restrictive* than β, denoted $\alpha \leq_r \beta$, if, for any database instance I, $\alpha(I) \leq_r \beta(I)$.

Note that the symbol $\leq_r$ is overloaded in that it signifies containment of relations as well as containment of conjunctive queries. This is natural, since the latter is defined in terms of the former. The ordering $\leq_r$ denotes a partial order over both the set of compatible relation instances and the set of conjunctive queries.

2.6 Homomorphisms

We close this section with the definition of homomorphisms, which are useful tools in characterizing conjunctive query containment.

Definition 2.12. Consider two conjunctive queries α and β with compatible consequents. A *homomorphism* h: $\beta \rightarrow \alpha$ is a total function from the variables of β into those of α, such that

(1) if x, y are distinguished variables appearing in the same argument position in the consequent of β and α, respectively, then $h(x) = y$; and
(2) if $Q(x_1, \ldots, x_n)$ appears in β, then $Q(h(x_1), \ldots, h(x_n))$ appears in α.

Note that a homomorphism h: $\beta \rightarrow \alpha$ induces a total mapping from the atomic formulas of β to the atomic formulas of α. Occasionally, when no confusion arises, we use h to denote that induced mapping as well. If α contains repeated formulas, this mapping is not a function, but we can readily extend the definition of a homomorphism to require that it is a function. With this extension, we can define an *onto homomorphism* to be a homomorphism in which this function is onto, with respect to the set of atomic formulas in α. We also define a *variable-onto homomorphism* to be a homomorphism that is onto with respect to the set of variables in α. Note that every onto homomorphism is also variable-onto, but that the converse is not true.

Definition 2.13. For two functions f_1 and f_2 such that the range of f_2 is a subset of the domain of f_1, their *composition* is denoted by $f_1 \circ f_2$ and is defined as $f_1 \circ f_2(x) = f_1(f_2(x))$ for any member x in the domain of f_2.

3. TWO GENERAL RESULTS

In this section we establish two results that are applicable to almost all label systems and that are used extensively in the rest of the paper.

LEMMA 3.1. *Consider a label system $\mathscr{L}$ such that $\forall a, b \in L - \{0\}$, $a * b \neq 0$. For two conjunctive queries α and β, the inequality $\alpha \leq_r \beta$ holds with respect to $\mathscr{L}$ only if there exists a homomorphism h: $\beta \rightarrow \alpha$.*

PROOF. Let a_i (resp., b_i), $1 \leq i \leq n$, be the distinguished variable in the ith argument position of α (resp., β). Assume that $\alpha \leq_r \beta$. Consider a

valuation θ of α such that θ_v is one-to-one from the variables in α onto some set of constants C and θ_l maps all atomic formulas in α to nonzero elements of L. Consider a database instance such that for any relation $\mathbf{Q}$ the following is satisfied:

$$\mathbf{Q}(t) = \begin{cases} \theta_l(Q(x_1, \ldots, x_m)), & \text{if } t = \langle \theta_v(x_1), \ldots, \theta_v(x_m) \rangle \text{ for some } Q(x_1, \ldots, x_m) \text{ in the antecedent of } \alpha; \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathbf{P}_\alpha$ (resp., $\mathbf{P}_\beta$) be the result of applying α (resp., β) on that instance. Then, based on the requirements on $\mathcal{F}$ in the premise of the lemma and the requirements on f in the definition of conjunctive queries, the following holds: $\mathbf{P}_\alpha(\langle \theta_v(a_1), \ldots, \theta_v(a_n) \rangle) \neq 0$. Since $\alpha \leq_r \beta$ and because 0 is the least element of L with respect to $\leq$, it must be the case that $\mathbf{P}_\beta(\langle \theta_v(a_1), \ldots, \theta_v(a_n) \rangle) \neq 0$ as well. Thus, a valuation θ' of β exists that is true with respect to the given database instance that is compatible with θ and such that for any atomic formula $Q(y_1, \ldots, y_m)$ in β, $\mathbf{Q}(\langle \theta'_v(y_1), \ldots, \theta'_v(y_m) \rangle) \neq 0$. By the construction of the database instance, the above implies that θ'_v maps the variables of β into the set of constants C. Valuation θ_v is one-to-one and onto, so its inverse θ_v^{-1} is defined. Taking the composition $h = \theta_v^{-1} \circ \theta'_v$, it is easy to verify that it is a homomorphism from the variables of β to the variables of α. □

LEMMA 3.2. *Consider two conjunctive queries α and β, and assume that there exists a homomorphism h: $\beta \rightarrow \alpha$. Consider a database instance I and the set Θ_α (resp., Θ_β) of all valuations of α (resp., β) that are true with respect to I. Let F be a total function on Θ_α ranging over the set of valuations of β and defined as $F(\theta) = \theta \circ h$. Function F has the following property:*

$$\textit{For all} \quad \theta \in \Theta_\alpha, \; F(\theta) \in \Theta_\beta \; \textit{and} \; F(\theta) \; \textit{is compatible with} \; \theta.$$

PROOF. The proof of the lemma is based on the definition of homomorphisms. By property (2) of Definition 2.12, for each atomic formula $Q(y_1, \ldots, y_k)$ in β, $Q(h(y_1), \ldots, h(y_k))$ appears in α as well. This implies the following:

$$\begin{aligned} l \circ h(Q(y_1, \ldots, y_m)) &= \theta(h(Q(y_1, \ldots, y_m))) \\ &= \theta_l(Q(h(y_1), \ldots, h(y_m))) \\ &= \mathbf{Q}(\langle \theta_v(h(y_1)), \ldots, \theta_v(h(y_m)) \rangle) \\ &\qquad \text{since } \theta \text{ is true with respect to } I \\ &= \mathbf{Q}(\langle \theta_v \circ h(y_1), \ldots, \theta_v \circ h(y_n) \rangle) \end{aligned}$$

Hence, by Definition 2.8 valuation $F(\theta) = \theta \circ h$ of β is true with respect to I, that is, it is in Θ_β.

Let a_i (resp., b_i), $1 \leq i \leq n$, be the distinguished variable in the ith argument position of α (resp., β). By property (1) of Definition 2.12, $\theta_v(a_i) = \theta_v \circ h(b_i)$, $1 \leq i \leq n$, and therefore, θ and $F(\theta) = \theta \circ h$ are compatible. □

4. LABEL SYSTEMS OF TYPE A

The following theorem identifies a necessary and sufficient condition for conjunctive query containment over databases with label systems of type A:

THEOREM 4.1. *Consider two conjunctive queries* α: $A_1 \wedge \cdots \wedge A_{m1} \xrightarrow{f_\alpha} c_\alpha$ *and* β: $B_1 \wedge \cdots \wedge B_{m2} \xrightarrow{f_\beta} c_\beta$. *Assume that* $\forall a, b \in L$, $a \leq b \Rightarrow f_\alpha(a) \leq f_\beta(b)$. *Then, the inequality* $\alpha \leq_r \beta$ *holds with respect to a label system* $\mathcal{L}$ *of type A iff there exists a homomorphism* h: $\beta \to \alpha$.

PROOF. Let a_i (resp., b_i), $1 \leq i \leq n$, be the distinguished variable in the ith argument position of α (resp., β). Assume that $\alpha \leq_r \beta$. By property (A1), it follows that $\forall a, b \in L - \{0\}$, $a * b \neq 0$. Therefore, by applying Lemma 3.1, the "only-if" direction is proved.

For the "if" direction, assume that there exists a homomorphism h: $\beta \to \alpha$. Consider a database instance I and the set Θ_α (resp., Θ_β) of all valuations of α (resp., β) that are true with respect to I. Let F be defined as in Lemma 3.2. Then, F has the following additional property:

(a) For all $\theta \in \Theta_\alpha$, $\theta_l(c_\alpha) \leq (F(\theta_l))(c_\beta)$; that is, $\theta_l(c_\alpha) \leq \theta_l \circ h(c_\beta)$.

The proof of property (a) is based on specific characteristics of label systems of type A. Let $A = \{A_i: 1 \leq i \leq m1\}$ and $B = \{B_i: 1 \leq i \leq m2\}$ represent the set of atomic formulas in α and β, respectively. Consider the subset A_B of A consisting of the atomic formulas in α that are images of atomic formulas in β under h. Without loss of generality, assume that $A_B = \{A_i: 1 \leq i \leq k\}$ for some $k \geq 1$. Then, since $\theta_l \circ h \in \Theta_\beta$ (Lemma 3.2), the following holds:

$$\theta_l \circ h(c_\beta) = f_\beta\left(\prod_{i=1}^{m2} \theta_l \circ h(B_i)\right) = f_\beta\left(\prod_{i=1}^{k} \theta_l(A_i)\right). \tag{1}$$

The last equality is due to property (A2), which implies that, even if $h(B_i) = h(B_j)$ for some $i \neq j$, the product is not affected. On the other hand, the following is also true:

$$\theta_l(c_\alpha) = f_\alpha\left(\prod_{i=1}^{m1} \theta_l(A_i)\right) = f_\alpha\left(\prod_{i=1}^{k} \theta_l(A_i) * \prod_{i=k+1}^{m1} \theta_l(A_i)\right).$$

From the above, we conclude that $\theta_l(c_\alpha) \leq \theta_l \circ h(c_\beta)$, using property (A1) of multiplication and the conditions on f_α and f_β in the theorem. Thus, property (a) of F holds.

We proceed with the proof of the theorem. Partition Θ_α and Θ_β based on the equivalence relation of valuation compatibility, and let $\Theta_{t\alpha}$ and $\Theta_{t\beta}$ be the corresponding partitions that generate tuple t in the distinguished variables of α or β. Clearly, there is a one-to-one and onto correspondence between the partitions obtained for α and those obtained for β (since for both of them there is a single partition for each tuple in the domain of the consequent relation). Let V_α and V_β be multisets defined as follows: $V_\alpha =$

$\{\theta_l(c_\alpha)\colon\ \theta \in \Theta_{t\alpha}\}$ and $V_\beta = \{\theta'_l(c_\beta)\colon\ \theta' \in \Theta_{t\beta}\}$. By property (A4) of addition, there is some element $v_0 \in V_\alpha$ such that

$$\sum_{v \in V_\alpha} v \leq v_0. \tag{2}$$

Suppose that $v_0 = \theta_l(c_\alpha)$ and $v'_0 = F(\theta_l)(c_\beta)$, for some $\theta \in \Theta_{t\alpha}$. From property (a), $v_0 \leq v'_0$, and from Lemma 3.2, $v'_0 \in V_\beta$. By property (A3) of addition, the following holds:

$$v_0 = v_0 + 0 \leq v'_0 + \sum_{v \in V_\beta - \{v'_0\}} v = \sum_{v \in V_\beta} v. \tag{3}$$

Combining (2) and (3) yields $\Sigma_{v \in V_\alpha} v \leq \Sigma_{v \in V_\beta} v$. If $\alpha(I)$ and $\beta(I)$ are equal to the relations $\mathbf{P}_\alpha$ and $\mathbf{P}_\beta$, respectively, by Definition 2.9, the above implies that, for all tuples t in the result of α or β, $\mathbf{P}_\alpha(t) \leq \mathbf{P}_\beta(t)$. Therefore, for an arbitrary database instance I, $\alpha(I) \leq_r \beta(I)$, which also implies that $\alpha \leq_r \beta$. □

As we have mentioned earlier, conjunctive queries over traditional relational databases, which interpret relations as sets, are a special case of type A systems. Thus, Theorem 4.1 generalizes the result of Chandra and Merlin [1977], so that it is now applicable for alternative semantics as well, for example, for interpreting relations as fuzzy sets [Zadeh 1965]. A straightforward corollary is that testing for conjunctive query containment with respect to label systems of type A is identical to the same problem for the traditional case of sets. Hence, we have the following [Chandra and Merlin 1977]:

COROLLARY 4.1. *Testing for conjunctive query containment with respect to label systems of type A is NP-complete.*

5. LABEL SYSTEMS OF TYPE B

The following theorem identifies a sufficient condition for conjunctive query containment over databases with label systems of type B. Unfortunately, as Example 5.1 illustrates, it is not necessary, in general:

THEOREM 5.1. *Consider two conjunctive queries* $\alpha\colon\ A_1 \wedge \cdots \wedge A_{m1} \overset{f_\alpha}{\rightarrow} c_\alpha$ *and* $\beta\colon\ B_1 \wedge \cdots \wedge B_{m2} \overset{f_\beta}{\rightarrow} c_\beta$. *Assume that* $\forall a, b \in L,\ a \leq b \Rightarrow f_\alpha(a) \leq f_\beta(b)$. *If there exists an onto homomorphism* $h\colon\ \beta \rightarrow \alpha$, *then the inequality* $\alpha \leq_r \beta$ *holds with respect to a label system* $\mathscr{L}$ *of type B.*

PROOF. Let a_i (resp., b_i), $1 \leq i \leq n$, be the distinguished variable in the ith argument position of α (resp., β). Assume that there exists an onto homomorphism $h\colon\ \beta \rightarrow \alpha$. Consider a database instance I and the set Θ_α (resp., Θ_β) of all valuations of α (resp., β) that are true with respect to I. Let F be defined as in Lemma 3.2. Then, F has the following additional properties:

(a) For all $\theta \in \Theta_\alpha$, $\theta_l(c_\alpha) \leq (F(\theta_l))(c_\beta)$; that is, $\theta_l(c_\alpha) \leq \theta_l \circ h(c_\beta)$.
(b) F is one-to-one from Θ_α to Θ_β.

The proof of property (a) is based on the specific characteristics of label systems of type B. Let $A = \{A_i\colon\ 1 \le i \le m1\}$ and $B = \{B_i\colon\ 1 \le i \le m2\}$ represent the set of atomic formulas in α and β, respectively. Because h is onto, the set B can be partitioned into two subsets, say, B_A and B_Q, such that $h\colon B_A \to A$ is a bijection. Without loss of generality, assume that the two subsets are $B_A = \{B_i\colon\ 1 \le i \le m1\}$ and $B_Q = \{B_i\colon\ m1 + 1 \le i \le m2\}$. Thus, the following holds:

$$\theta_l \circ h(c_\beta) = f_\beta\left(\prod_{i=1}^{m2} \theta_l \circ h(B_i)\right) = f_\beta\left(\prod_{i=1}^{m1} \theta_l \circ h(B_i) * \prod_{i=m1+1}^{m2} \theta_l \circ h(B_i)\right)$$
$$= f_\beta\left(\prod_{i=1}^{m1} \theta_l(A_i) * \prod_{i=m1+1}^{m2} \theta_l \circ h(B_i)\right).$$

Also,

$$\theta_l(c_\alpha) = f_\alpha\left(\prod_{i=m1+1}^{m2} \theta_l \circ h(B_i)\right).$$

From the above, because of property (B1) of multiplication and the conditions on f_α and f_β in the theorem, we conclude that $\theta_l(c_\alpha) \le \theta_l \circ h(c_\beta)$ and therefore that property (a) of F holds.

The proof of property (b) consists of showing that, given two valuations $\theta, \theta' \in \Theta_\alpha$, if $\theta(\alpha) \ne \theta'(\alpha)$, then $\theta \circ h(\beta) \ne \theta' \circ h(\beta)$. Because $\theta(\alpha) \ne \theta'(\alpha)$, there must be at least one variable x in α such that

$$\theta_v(x) \ne \theta'_v(x). \tag{4}$$

Because h is onto, it is also variable-onto, and so every variable of α is an h-image of some variable of β. Assume that $x = h(y)$, for some variable y of β. The above combined with (4) implies that

$$\theta_v \circ h(y) \ne \theta'_v \circ h(y). \tag{5}$$

Therefore, $\theta_v \circ h(\beta) \ne \theta'_v \circ h(\beta)$, which implies that property (b) of F holds.

We proceed with the proof of the theorem. Partition Θ_α and Θ_β based on the equivalence relation of valuation compatibility, and let $\Theta_{t\alpha}$ and $\Theta_{t\beta}$ be the corresponding partitions that generate tuple t in the distinguished variables of α or β. As in the proof of Theorem 4.1, there is a one-to-one and onto correspondence between the partitions obtained for α and those obtained for β. Let V_α and V_β be multisets defined as follows: $V_\alpha = \{\theta_l(c_\alpha)\colon\ \theta \in \Theta_{t\alpha}\}$ and $V_\beta = \{\theta'_l(c_\beta)\colon\ \theta' \in \Theta_{t\beta}\}$. Let V_β^α be defined as follows: $V_\beta^\alpha = \{\theta'_l(c_\beta)\colon\ \theta' \in \Theta_{t\beta}$ and $\theta' = \theta \circ h$ for some $\theta \in \Theta_\alpha\}$. The properties of F imply that for every element $v \in V_\alpha$ there is an element $v' \in V_\beta^\alpha$ that corresponds to v (Lemma 3.2) such that $v \le v'$ (property (a)), which corresponds to no other element of V_α (property (b)). By property (B2) of addition, the above imply the following:

$$\sum_{v \in V_\alpha} v \le \sum_{v' \in V_\beta^\alpha} v' \le \sum_{v' \in V_\beta} v'. \tag{6}$$

If $\alpha(I)$ and $\beta(I)$ are equal to the relations $\mathbf{P}_\alpha$ and $\mathbf{P}_\beta$, respectively, by Definition 2.9, (6) implies that, for all tuples t in the result of α or β, $\mathbf{P}_\alpha(t) \leq \mathbf{P}_\beta(t)$. Therefore, for an arbitrary database instance I, $\alpha(I) \leq_r \beta(I)$, which also implies that $\alpha \leq_r \beta$. □

The following proposition provides a straightforward necessary condition for conjunctive query containment with respect to label systems of type B:

PROPOSITION 5.1. *For two conjunctive queries α and β, the inequality $\alpha \leq_r \beta$ holds with respect to a label system $\mathscr{L}$ of type B only if there exists a homomorphism h: $\beta \rightarrow \alpha$.*

PROOF. Assume that $\alpha \leq_r \beta$. By property (B1), it follows that $\forall a, b \in L - \{0\}$, $a * b \neq 0$. Therefore, by applying Lemma 3.1, the proposition is proved. □

By restricting the form of conjunctive queries, the following theorem shows that the condition of Theorem 5.1 is both necessary and sufficient:

THEOREM 5.2. *Consider two conjunctive queries α: $A_1 \wedge \cdots \wedge A_{m1} \overset{f_\alpha}{\rightarrow} c_\alpha$ and β: $B_1 \wedge \cdots \wedge B_{m2} \overset{f_\beta}{\rightarrow} c_\beta$. Assume that $\forall a, b \in L$, $a \leq b \Rightarrow f_\alpha(a) \leq f_\beta(b)$. If α does not contain repeated predicates, the inequality $\alpha \leq_r \beta$ holds with respect to a label system $\mathscr{L}$ of type B iff there exists an onto homomorphism h: $\beta \rightarrow \alpha$.*

PROOF. The "if" direction is an immediate consequence of Theorem 5.1. Suppose that $\alpha \leq_r \beta$. By Proposition 5.1, we know that there must be some homomorphism h: $\beta \rightarrow \alpha$. We first show that, in this case, there is a unique such homomorphism. Since there are no repeated predicates in α, for each atomic formula of β, there is a unique atomic formula in α that can be its image under any homomorphism. Therefore, for each variable in β its image is uniquely determined; that is, there is a unique homomorphism h: $\beta \rightarrow \alpha$.

It remains to be shown that this unique homomorphism is onto. Assume to the contrary that h is not onto. Then, there is an atomic formula in α that is not the image of any atomic formula in β. Let Q be the predicate in that atomic formula of α. Clearly, Q cannot appear in β. Without loss of generality, assume that all arguments of all predicates in the conjunctive queries have the same domain. Consider a constant c in that domain and a database instance such that each relation $\mathbf{R}$ satisfies the following:

$$\mathbf{R}(t) \neq 0, \quad \text{if } t \text{ has } c \text{ in all its arguments;}$$
$$\mathbf{R}(t) = 0, \quad \text{otherwise.}$$

Let $\mathbf{P}_\alpha$ (resp., $\mathbf{P}_\beta$) be the result of applying α (resp., β) on that instance. Let s be the tuple in these results that has c in all of its arguments. Let $\mathbf{P}_\beta(s) = l$, where, by the construction of the database instance, $l \in L - \{0\}$. By property (B3), there is a label $m \in L$ such that $m > l$. Suppose that t' is the tuple in $\mathbf{Q}$ that has c in all of its arguments, and choose $\mathbf{Q}(t') = m$. Then, by property

(B1), $\mathbf{P}_\alpha(t) \geq m > l = \mathbf{P}_\beta(t)$, which implies that $\alpha \not\leq_r \beta$, which is a contradiction. Hence, h must be onto. □

Recently, Chaudhuri and Vardi [1993] independently discovered Theorem 5.1, Proposition 5.1, and Theorem 5.2 for the case of multisets. It is natural to ask whether Theorem 5.2 can be strengthened to cover the case that β does not contain repeated predicates (while α possibly does). Unfortunately, the following example shows that this is not possible:

Example 5.1. Consider the following two conjunctive queries:

$$\alpha: \quad Q(x) \wedge Q(x) \rightarrow P(x),$$
$$\beta: \quad Q(x) \rightarrow P(x).$$

Let $\mathscr{L}$ be defined as follows: $L = N$ (the set of natural numbers including 0), $*$ is max, $+$ is the usual addition, and $\leq$ is the usual total order over the natural numbers. Clearly, $\alpha \leq_r \beta$, although there is no onto homomorphism from β to α.

By limiting the definition of type B label systems so that examples like the above are excluded, we can prove a stronger version of Theorem 5.2.

Definition 5.1. A label system $\mathscr{L} = \langle L, *, +, 0, \leq \rangle$ is of *type* B′ if it satisfies the following:

(B′1) $\forall a, b \in L - \{0\}$, $a \leq a * b$, and $\exists a \in L$, $\forall k \geq 1$, $a^k < a^{k+1}$.
(B′2) $\forall a, a', b, b' \in L$, $(a \leq a' \text{ and } b \leq b') \Rightarrow a + b \leq a' + b'$.
(B′3) $\forall a \in L$, $\exists a' \in L$, $a < a'$.

Observe that the only difference between label systems of type B and of type B′ is that there is an element in L whose product with itself is strictly larger than the element itself (property (B′1)). For these label systems, we have the following result:

THEOREM 5.3. *Consider two conjunctive queries* α: $A_1 \wedge \cdots \wedge A_{m1} \xrightarrow{f_\alpha} c_\alpha$ *and* β: $B_1 \wedge \cdots \wedge B_{m2} \xrightarrow{f_\beta} c_\beta$. *Assume that* $\forall a, b \in L$, $a \leq b \Rightarrow f_\alpha(a) \leq f_\beta(b)$. *If either* α *or* β *does not contain repeated predicates, the inequality* $\alpha \leq_r \beta$ *holds with respect to a label system* $\mathscr{L}$ *of type* B′ *iff there exists an onto homomorphism* h: $\beta \rightarrow \alpha$.

PROOF. The "if" direction as well as the "only-if" direction for the case where there are no repetitions in α are immediate consequences of Theorems 5.1 and 5.2, respectively. Suppose that β does not contain repeated predicates and that $\alpha \leq_r \beta$. By Proposition 5.1, we know that there must be some homomorphism h: $\beta \rightarrow \alpha$. Clearly, every such homomorphism must be one-to-one with respect to atomic formulas, since there is no repetition or predicates in β. Assume that h is not onto. If m (resp., n) is the number of atomic formulas in α (resp., β), then clearly $m > n$.

Let l be an element of L such that $\forall k \geq 1$, $l^k < l^{k+1}$. Property (B′1) ensures the existence of such a label. Without loss of generality, assume that

all arguments of all predicates in the conjunctive queries have the same domain. Consider a constant c in that domain and a database instance such that each relation $\mathbf{R}$ satisfies the following:

$$\mathbf{R}(t) = l, \quad \text{if } t \text{ has } c \text{ in all its arguments,}$$
$$\mathbf{R}(t) = 0, \quad \text{otherwise.}$$

Let $\mathbf{P}_\alpha$ (resp., $\mathbf{P}_\beta$) be the result of applying α (resp., β) on that instance. Let s be the tuple in these results that has c in all of its arguments. We note that $\mathbf{P}_\alpha(s) = l^m$ and $\mathbf{P}_\beta(s) = l^n$. Since $m > n$, by property (B'1), it follows that $\mathbf{P}_\alpha(s) > \mathbf{P}_\beta(s)$, which implies that $\alpha \not\leq_r \beta$, which is a contradiction. Hence, h must be onto. □

We remark that the choice of h in the above proof was arbitrary. Hence, $\alpha \leq_r \beta$ only if every homomorphism h: $\beta \rightarrow \alpha$ is onto.

As mentioned earlier, commercial relational database systems treat relations as multisets and therefore, operate under the semantics of type B (and, in fact, type B') label systems. Hence, Theorems 5.1–5.3 are applicable to this case.

6. UNIONS OF CONJUNCTIVE QUERIES

In this section we generalize the results presented above and discuss containment not of individual conjunctive queries, but of collections of them. For the classical case, where relations are viewed as sets, the problem has been addressed by Sagiv and Yannakakis [1980], who gave a syntactic characterization to the containment problem. After introducing the necessary terminology, we first show that their theorem holds for all label systems of type A (of which the set-case is one). We then show that the problem is, in general, undecidable for label systems of type B, thus providing more evidence of the differences between the two label systems.

6.1 Basic Definitions

The following definitions are straightforward extensions of the single conjunctive query case:

Definition 6.1. A *union of conjunctive queries* is a collection of first-order formulas with identical consequents. If α_i, $1 \leq i \leq n$, are the conjunctive queries in the collection, then their union is denoted by $\bigcup_{i=1}^{n} \alpha_i$.

Definition 6.2. Consider a union of conjunctive queries $\bigcup_{i=1}^{n} \alpha_i$ and a database instance I. Let $\mathbf{P}_i$ be the relation instance that is the result of applying α_i to I. The *result* of applying $\bigcup_{i=1}^{n} \alpha_i$ to I (denoted by $\bigcup_{i=1}^{n} \alpha_i(I)$) is a relation instance $\mathbf{P}$, that is, a function, such that, for each tuple t in the domain of the consequent of the conjunctive queries,

$$\mathbf{P}(t) = \sum_{i=1}^{n} \mathbf{P}_i(t).$$

Definition 6.3. For two unions of conjunctive queries $\bigcup_{i=1}^{n1} \alpha_i$ and $\bigcup_{j=1}^{n2} \beta_j$ with compatible consequents, $\bigcup_{i=1}^{n1} \alpha_i$ is *more restrictive* than $\bigcup_{j=1}^{n2} \beta_j$,

denoted $\bigcup_{i=1}^{n1} \alpha_i \leq_r \bigcup_{j=1}^{n2} \beta_j$, if, for any database instance I, $\bigcup_{i=1}^{n1} \alpha_i(I) \leq_r \bigcup_{j=1}^{n2} \beta_j(I)$.

6.2 Label Systems of Type A

The following theorem identifies a necessary and sufficient condition for containment of unions of conjunctive queries over databases with label systems of type A:

THEOREM 6.1. *For two unions of conjunctive queries* $\bigcup_{i=1}^{n1} \alpha_i$ *and* $\bigcup_{j=1}^{n2} \beta_i$, *the inequality* $\bigcup_{i=1}^{n1} \alpha_i \leq_r \bigcup_{j=1}^{n2} \beta_i$ *holds with respect to a label system* $\mathcal{L}$ *of type A iff for all* $1 \leq i \leq n1$ *there exists* $1 \leq j \leq n2$ *such that* $\alpha_i \leq_r \beta_j$.

PROOF. For the "if" direction, assume that for all $1 \leq i \leq n1$ there exists $1 \leq j \leq n2$ such that $\alpha_i \leq_r \beta_j$. Consider a database instance and let $\mathbf{P}_\alpha$, $\mathbf{P}_\beta$, $\mathbf{P}_{\alpha i}$, and $\mathbf{P}_{\beta j}$ denote the results of applying $\bigcup_{i=1}^{n1} \alpha_i$, $\bigcup_{j=1}^{n2} \beta_i$, α_i, and β_j to that instance, respectively. For any tuple t in the domain of the consequents of the conjunctive queries, Definition 6.2 implies that

$$\mathbf{P}_\alpha(t) = \sum_{i=1}^{n1} \mathbf{P}_{\alpha i}(t). \tag{7}$$

Properties (A3) and (A4) together with associativity and commutativity of + imply that there exists some $1 \leq I \leq n1$ such that

$$\sum_{i=1}^{n1} \mathbf{P}_{\alpha i}(t) \leq \mathbf{P}_{\alpha I}(t). \tag{8}$$

By the premises of the "if" direction, there exists some $1 \leq J \leq n2$ such that $\alpha_I \leq_r \beta_J$, which implies that

$$\mathbf{P}_{\alpha I}(t) \leq \mathbf{P}_{\beta J}(t). \tag{9}$$

Finally, property (A3) and associativity of + together with Definition 6.2 imply that

$$\mathbf{P}_{\beta J}(t) \leq \sum_{j=1}^{n2} \mathbf{P}_{\beta j}(t) = \mathbf{P}_\beta(t). \tag{10}$$

The combination of (7)–(10) yields that for any tuple t in the domain of the consequents of the conjunctive queries $\mathbf{P}_\alpha(t) \leq \mathbf{P}_\beta(t)$. Since the above holds for arbitrary tuples t and arbitrary database instances, Definitions 2.10 and 2.11 imply that $\bigcup_{i=1}^{n1} \alpha_i \leq_r \bigcup_{j=1}^{n2} \beta_i$.

The proof of the "only-if" direction is very similar to the proof of Lemma 3.1, but is given here in full detail for clarity. Assume that the inequality $\bigcup_{i=1}^{n1} \alpha_i \leq_r \bigcup_{j=1}^{n2} \beta_i$ holds. Let α_i (resp., b_i), $1 \leq i \leq n$, be the distinguished variable in the ith argument position of the conjunctive queries in $\bigcup_{i=1}^{n1} \alpha_i$ (resp., $\bigcup_{j=1}^{n2} \beta_j$). Consider an arbitrary conjunctive query α_I in $\bigcup_{i=1}^{n1} \alpha_i$.

Furthermore, consider a valuation θ of α_I such that θ_v is one-to-one from the variables in α_I onto some set of constants C and θ_l maps all atomic formulas in α_I to nonzero elements of L. Consider a database instance such that for any relation $\mathbf{Q}$ the following is satisfied:

$$\mathbf{Q}(t) = \begin{cases} \theta_l(Q(x_1, \ldots, x_m)), & \text{if } t = \langle \theta_v(x_1), \ldots, \theta_v(x_m) \rangle \text{ for some } Q(x_1, \ldots, x_m) \text{ in } \alpha_I; \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathbf{P}_\alpha$, $\mathbf{P}_\beta$, $\mathbf{P}_{\alpha_I}$ and $\mathbf{P}_{\beta_J}$ denote the results of applying $\bigcup_{i=1}^{n1} \alpha_i$, $\bigcup_{j=1}^{n2} \beta_j$, α_I, and β_J to that instance, respectively. Properties (A1) and (A3), Definition 6.2, and the premise of the "only-if" direction imply that

$$0 < \mathbf{P}_{\alpha I}(\langle \theta_v(a_1), \ldots, \theta_v(a_n) \rangle) \leq \mathbf{P}_\alpha(\langle \theta_v(a_1), \ldots, \theta_v(a_n) \rangle) \leq \mathbf{P}_\beta(\langle \theta_v(a_1), \ldots, \theta_v(a_n) \rangle).$$

Since 0 is the least element of L with respect to $\leq$ and the additive identity, the above implies that there must exist some β_J in $\bigcup_{j=1}^{n2} \beta$ such that $0 < \mathbf{P}_{\beta J}(\langle \theta_v(a_1), \ldots, \theta_v(a_n) \rangle)$ as well. Thus, a valuation θ' of β_J exists that is true with respect to the given database instance that is compatible with θ and such that, for any atomic formula $Q(y_1, \ldots, y_m)$ in β_J, $\mathbf{Q}(\langle \theta'_v(y_1), \ldots, \theta'_v(y_m) \rangle) \neq 0$. By the construction of the database instance, the above implies that θ'_v maps the variables of β_J into the set of constants C. Valuation θ_v is one-to-one and onto, so its inverse θ_v^{-1} is defined. Taking the composition $h = \theta_v^{-1} \circ \theta'_v$, it is easy to verify that it is a homomorphism from the variables of β_J to the variables of α_I. Theorem 4.1 implies that $\alpha_I \leq_r \beta_J$. Since the above was true for an arbitrary conjunctive query α_I in $\bigcup_{i=1}^{n1} \alpha_i$, the theorem follows. □

When considering the label system that captures the usual interpretation of relations as sets, Theorem 6.1 is identical to the corresponding theorem of Sagiv and Yannakakis [1980] on containment of unions of conjunctive queries. Its significance is that it demonstrates that the same characterization of containment exists for arbitrary label systems of type A.

6.3 Label Systems of Type B

We now turn our attention to label systems of type B and the problem of containment of unions of conjunctive queries. We show that there is one such label system for which the problem is undecidable, specifically, the label system that captures the interpretation of relations as multisets. The reduction is from a variant from the Diophantine equation problem, so we first state some definitions related to that.

Let $\Lambda(x_1, x_2, \ldots, x_k)$ be a polynomial in k variables with integer coefficients. The equation $\Lambda(x_1, x_2, \ldots, x_k) = 0$ is referred to as a *Diophantine equation* when only its integer solutions are being sought.

The problem of determining whether there exists a nonnegative integer[3] solution to such an equation is undecidable [Davis 1982]. More formally, there is no decision procedure for the statement

$$(\exists x_1, x_2, \ldots, x_k)\Lambda(x_1, x_2, \ldots, x_k) = 0.$$

The following equivalences are straightforward:

$$\begin{aligned}(\neg\exists x_1, x_2, \ldots, x_k)\Lambda(x_1, x_2, \ldots, x_k) = 0 \\ \Leftrightarrow (\forall x_1, x_2, \ldots, x_k)\Lambda(x_1, x_2, \ldots, x_k) \neq 0, \\ \Leftrightarrow (\forall x_1, x_2, \ldots, x_k)1 - (\Lambda(x_1, x_2, \ldots, x_k))^2 \leq 0.\end{aligned}$$

Therefore, the problem of determining whether a polynomial is nonpositive for all nonnegative integer assignments to its variables is also undecidable. In addition, for a variable x_0 that is distinct from all other variables x_i, $1 \leq i \leq k$, the following equivalence holds:

$$\begin{aligned}(\forall x_1, x_2, \ldots, x_k)\Lambda(x_1, x_2, \ldots, x_k) \leq 0 \\ \Leftrightarrow (\forall x_1, x_2, \ldots, x_k)x_0\Lambda(x_1, x_2, \ldots, x_k) \leq 0.\end{aligned}$$

Note that $x_0\Lambda(x_1, x_2, \ldots, x_k)$ is a polynomial without a constant term. Hence, the above equivalence implies that the problem of determining whether a polynomial is nonpositive for all nonnegative integer assignments to its variables remains undecidable even if the polynomial has no constant terms. This final undecidable problem related to Diophantine equations is the one that we use in the following result:

THEOREM 6.2. *Consider the following label system of type B:* $\mathscr{L} = \langle L, *, +, 0, \leq \rangle$, *where* $L = N_0$, *the set of nonnegative integers,* $*$ *and* $+$ *are the usual integer multiplication and addition, and* $\leq$ *is the usual total order on the integers. Also consider two unions of conjunctive queries* $\bigcup_{i=1}^{n1} \alpha_i$ *and* $\bigcup_{j=1}^{n2} \beta_i$. *The problem of deciding whether or not* $\bigcup_{i=1}^{n1} \alpha_i \leq_r \bigcup_{j=1}^{n2} \beta_i$ *with respect to the above label system* $\mathscr{L}$ *is undecidable.*

PROOF. Let $\Phi(x_1, x_2, \ldots, x_k)$ and $\Psi(x_1, x_2, \ldots, x_k)$ be two polynomials in k variables with positive integer coefficients and with no constant terms. We have established above that there is no decision procedure for the statement[4]

$$(\forall x_1, x_2, \ldots, x_k)(\Phi(x_1, x_2, \ldots, x_k) \leq \Psi(x_1, x_2, \ldots, x_k)).$$

Given the above problem instance, we construct an equivalent instance of the problem of containment of unions of conjunctive queries.

[3]The problem is undecidable independent of whether the solutions sought are nonnegative, positive, or arbitrary integers.

[4]Note that we have rewritten a polynomial with arbitrary integer coefficients as the difference of two polynomials with positive coefficients.

Assume that the polynomial Φ is of the form

$$\Phi(x_1, x_2, \ldots, x_k) = \sum_{i=1}^{s} a_i x_1^{c_{i1}} x_2^{c_{i2}} \cdots x_k^{c_{ik}},$$

where there are s terms in Φ, which have been numbered in some arbitrary way from 1 to s. For all $1 \leq i \leq s$, a_i is a positive integer, and for all $1 \leq j \leq k$, c_{ij} is a nonnegative integer with at least one of them being positive (Φ has no constant terms). From Φ, we construct a union of conjunctive queries as follows: There is a unique, unary relation symbol X_i for each variable x_i, $1 \leq i \leq k$, plus the unary relation symbol P for the query consequents. All of these relations share the same domain D. For each term $a_i x_1^{c_{i1}} x_2^{c_{i2}} \cdots x_k^{c_{ik}}$, we put a_i identical conjunctive queries in the union; that is, there are a total of $n_1 = \sum_{i=1}^{s} a_i$ conjunctive queries. Each such conjunctive query is of the form

$$\overbrace{X_1(v) \wedge X_1(v) \wedge \cdots \wedge X_1(v)}^{c_{i1} \text{ times}} \wedge \cdots \wedge \overbrace{X_k(v) \wedge X_k(v) \wedge \cdots \wedge X_k(v)}^{c_{ik} \text{ times}} \rightarrow P(v).$$

Let $\bigcup_{i=1}^{n1} \alpha_i$ be the union of conjunctive queries thus constructed. Similarly, the union of conjunctive queries $\bigcup_{j=1}^{n2} \beta_j$ can be constructed from the polynomial Ψ.

What remains to be shown is that

$$(\forall x_1, x_2, \ldots, x_k)(\Phi(x_1, x_2, \ldots, x_k) \leq \Psi(x_1, x_2, \ldots, x_k)$$
$$\Leftrightarrow \bigcup_{i=1}^{n1} \alpha_i \leq_r \bigcup_{j=1}^{n2} \beta_j.$$

For the "only-if" direction, consider an arbitrary database instance I, and let $\mathbf{P}_\alpha$ denote the result of applying $\bigcup_{i=1}^{n1} \alpha_i$ to I. If $\mathbf{X}_i$ is the relation (function) associated with the symbol X_i in this instance, then, for any element $d \in D$,

$$\mathbf{P}_\alpha(d) = \Phi(\mathbf{X}_1(d), \mathbf{X}_2(d), \ldots, \mathbf{X}_k(d)). \tag{11}$$

The above is a straightforward consequence of the way $\bigcup_{i=1}^{n1} \alpha_i$ was constructed from the polynomial Φ. Likewise, if $\mathbf{P}_\beta$ denotes the result of applying $\bigcup_{j=1}^{n2} \beta_j$ to I, then, for any element $d \in D$,

$$\mathbf{P}_\beta(d) = \Psi(\mathbf{X}_1(d), \mathbf{X}_2(d), \ldots, \mathbf{X}_k(d)). \tag{12}$$

If $(\forall x_1, x_2, \ldots, x_k)(\Phi(x_1, x_2, \ldots, x_k) \leq \Psi(x_1, x_2, \ldots, x_k))$, then by (11) and (12), for any element of $d \in D$, $\mathbf{P}_\alpha(d) \leq \mathbf{P}_\beta(d)$, which implies that $\mathbf{P}_\alpha \leq_r \mathbf{P}_\beta$. Since the above holds for an arbitrary database instance, we conclude that $\bigcup_{i=1}^{n1} \alpha_i \leq_r \bigcup_{j=1}^{n2} \beta_j$.

For the "if" direction, consider an arbitrary set of nonnegative integers χ_i assigned to the variables x_i, respectively, of Φ and Ψ. Let I be a database instance such that, for some $d \in D$, for all $1 \leq i \leq k$, $\mathbf{X}_i(d) = \chi_i$. If $\mathbf{P}_\alpha$ and $\mathbf{P}_\beta$ denote the results of applying $\bigcup_{i=1}^{n1} \alpha_i$ and $\bigcup_{j=1}^{n2} \beta_j$ to I, respectively, then clearly

$$\mathbf{P}_\alpha(d) = \Phi(\chi_1, \chi_2, \ldots, \chi_k), \tag{13}$$

$$\mathbf{P}_\beta(d) = \Psi(\chi_1, \chi_2, \ldots, \chi_k). \tag{14}$$

If $\bigcup_{i=1}^{n1} \alpha_i \leq_r \bigcup_{j=1}^{n2} \beta_j$, then $\mathbf{P}_\alpha(d) \leq \mathbf{P}_\beta(d)$, which by (13) and (14) implies that $\Phi(\chi_1, \chi_2, \ldots, \chi_k) \leq \Psi(\chi_1, \chi_2, \ldots, \chi_k)$. Since the above holds for an arbitrary set of nonnegative integers χ_i, we conclude that $(\forall x_1, x_2, \ldots, x_k)(\Phi(x_1, x_2, \ldots, x_k) \leq \Psi(x_1, x_2, \ldots, x_k))$.

The two problem instances are equivalent, and the polynomial inequality problem is undecidable. Therefore, the problem of containment of unions of conjunctive queries with respect to the label system of type B described in the theorem is undecidable as well. □

We should emphasize that Theorem 6.2 deals explicitly with the most important label system of type B from a practical perspective. In principle, there could be some such label system for which the problem of containment of unions of conjunctive queries is decidable, but Theorem 6.2 shows that there is no general decision procedure for all such label systems and, in particular, for unions of conjunctive queries in the case of relations as multisets.

7. DISCUSSION

One of the benefits of our general framework is the ability to identify other useful types of label systems and to characterize containment for them syntactically. As an example, we consider a specific new label system in some detail. This is a variant of our type A system and is also motivated by fuzzy-set reasoning.

Definition 7.1. A label system $\mathscr{L} = \langle L, *, +, 0, \leq \rangle$ is of *type* A' if it satisfies the following:

(A1′) $\forall a, b \in L - \{0\}$, $0 < a * b \leq a$.

(A2′) $\forall a \in L$, $a * a = a$.

(A3′) $\forall a, a', b, b' \in L$, $(a \leq a' \text{ and } b \leq b') \Rightarrow a + b \leq a' + b'$.

With respect to type A, condition (A4) has been dropped. Clearly, every type A label system is also a type A′ label system.

Example 7.1. For certainty factors, L is equal to the set of real numbers between 0 and 1, as noted earlier. However, while the operation $*$ is usually *min*, the operation $+$ is not always taken to be *max*. For instance, in the expert system MYCIN, $a + b$ is defined as $a + b - a.b$ (where $+$ and $\cdot$ denote addition and multiplication over numbers, resp.). This label system is of type A′, but not of type A. Many other choices for $+$ also yield type A′ systems.

Lemma 3.1 clearly holds for type A′ systems, giving us a necessary condition for conjunctive query containment. The following theorem establishes a sufficient condition:

THEOREM 7.1. *Consider two conjunctive queries* α: $A_1 \wedge \cdots \wedge A_{m1} \xrightarrow{f_\alpha} c_\alpha$ *and* β: $B_1 \wedge \cdots \wedge B_{m2} \xrightarrow{f_\beta} c_\beta$. *Assume that* $\forall a, b \in L$, $a \leq b \Rightarrow f_\alpha(a) \leq f_\beta(b)$.

Then, the inequality $\alpha \leq_r \beta$ *holds with respect to a label system* $\mathcal{L}$ *of type* A' *if there exists a variable-onto homomorphism* h: $\beta \rightarrow \alpha$.

PROOF. Let a_i (resp., b_i), $1 \leq i \leq n$, be the distinguished variable in the ith argument position of α (resp., β). Assume that there exists a variable-onto homomorphism h: $\beta \rightarrow \alpha$. Consider a database instance I and the set Θ_α (resp., Θ_β) of all valuations of α (resp., β) that are true with respect to I. Let F be defined as in Lemma 3.2. Then, F has the following additional properties.

(a) For all $\theta \in \Theta_\alpha$, $\theta_l(c_\alpha) \leq (F(\theta_l))(c_\beta)$; that is, $\theta_l(c_\alpha) \leq \theta_l \circ h(c_\beta)$.
(b) F is one-to-one from Θ_α to Θ_β.

The proof of property (a) is identical to the corresponding part of the proof of Theorem 4.1. The proof of property (b) is identical to the corresponding part of the proof of Theorem 5.1. The rest of the proof is identical to the corresponding part of the proof of Theorem 5.1, observing that condition (B2) is identical to condition (A3′). □

The proof of Theorem 7.1 illustrates an important benefit of our algebraic framework: The exact properties upon which each part of a proof rests can be identified precisely, and this is of great help in identifying new label systems of interest for which useful results can be established.

This still leaves open the interesting question of whether Theorem 7.1 can be strengthened to provide a necessary and sufficient condition. Of course, studying equivalence and the case of unions of conjunctive queries for the various label systems considered here, and possibly other interesting label systems, pose further interesting problems in this area.

A more basic question is whether or not our conditions on label systems can be made more liberal. In particular, can we relax the requirement that the least element should be both the additive identity and the multiplicative annihilator? Our final example provides some motivation for such fundamental extensions and also illustrates that our generalization of conjunctive queries using label systems is equally useful in the context of recursive queries.

Example 7.2. Several generalizations of the transitive closure of a graph seek to perform path computations by associating labels with paths and by performing label computations as paths are enumerated. Typically, the set of labels is either the set of nonnegative integers or the set of nonnegative reals. The operations correspond to computing a new label for a path by "multiplying" ($*$) the labels for the subpaths used to generate the path and to computing a new label for a set of paths by "adding" ($+$) the labels for the paths in the set. By making different choices for $*$ and $+$, a variety of path problems can be stated in this framework. Table IX contains some characteristic examples.

Although we do not consider recursive queries, as Table IX shows, we can use any of these label systems to define useful databases and queries. For example, the label system in reachability is the traditional relation-as-set

Table IX. Characteristic Examples of Path Problems

Problem	$*$	$+$	L	*Label system type*
Reachability	*and*	*or*	{"false," "true"}	A
Maximum capacity path	*min*	*max*	Nonnegative integers	A
Bill of materials	*product*	*sum*	Nonnegative integers	B
Shortest path	*sum*	*min*	Nonnegative integers	?
Most reliable path	*product*	*max*	[0, 1]	?
Critical (longest) path	*sum*	*max*	Nonnegative integers	?

view, whereas bill of materials, when restricted to nonnegative edge labels, essentially treats relations as multisets. Unfortunately, not all proposed path systems can be viewed as type A or type B label systems, or even as label systems! Most reliable path is not a type A label system because $*$ is not idempotent, and it is not a type B label system because it does not satisfy (B1). Shortest path and critical path are not even label systems, because neither has a multiplicative annihilator.

8. RELATED WORK

The problem of conjunctive query containment for sets was solved by Chandra and Merlin [1977]. Aho et al. [1979] studied the problem in the presence of functional and multivalued dependencies, while Johnson and Klug [1982] studied it in the presence of functional and inclusion dependencies. Finally, Sagiv and Yannakakis [1980] addressed the union of conjunctive queries for sets and showed that the problem is decidable, essentially proving a special case of Theorem 6.1 (in contrast to the same problem over relations as multisets, which we prove undecidable).

Recently, Chaudhuri and Vardi [1993] independently discovered Theorem 5.1, Proposition 5.1, and Theorem 5.2 (i.e., similar sufficient conditions for conjunctive query containment) for the case of multiset queries. In terms of differences with our work, they also showed that the problem of conjunctive query containment for multiset queries is Π_2^p hard, whereas surprisingly, conjunctive query equivalence is the same as graph isomorphism (and therefore in NP, but not known to be NP-complete). On the other hand, we additionally deal with unions of conjunctive queries and show that containment is undecidable in that case. Moreover, whereas their entire work is on multisets alone, we present a much more general formalism (that of label systems), which captures a variety of possible interpretations of relations (including sets and fuzzy sets). Hence, even the results that are common with those of Chaudhuri and Vardi are in essence more general, since they hold for all label systems of type B.

The algebraic framework that we have used in this paper is inspired by an idea of Ioannidis and Wong [1991]. Specifically, that earlier work talks about associating a real number with each tuple of a relation, so that an algebra

can be formed and a theorem on linearizability of multilinear recursion (over relations as sets of tuples) can be obtained. In this paper we have taken the real-number idea and have generalized it to attach an arbitrary label to a tuple. Based on that, we have then developed the entire formalism of label systems, their properties, and the results on query containment.

There have been a few other pieces of work that deal with relations as multisets, but address problems other than containment of conjunctive queries. Dayal et al. [1982] were probably the first to consider multiset queries in SQL, using a model of relational algebra with control over duplicate elimination. Maher and Ramakrishnan [1989] proposed a multiset semantics for logic programs that generalizes the multiset semantics for conjunctive queries studied in this paper. The main results in that paper were that checking if a general logic program generated duplicates is undecidable and that there is a sufficient condition for ensuring that no duplicates are generated. Mumick et al. [1990] studied the issue of multiset queries in SQL and showed how selection-pushing into nested queries can be accomplished using the magic sets technique. Finally, Negri et al. [1991] recently considered the semantics of general SQL queries.

The area of fuzzy sets has been studied extensively following the seminal work by Zadeh [1965]. Van Emden [1986] proposed an extension of logic programs to deal with quantitative deduction that is closely related to fuzzy sets and uncertainty reasoning in expert systems [Parsaye and Chignell 1988].

9. SUMMARY AND FUTURE WORK

We have generalized the notion of a relational database to cover fuzzy sets, multisets, and other refinements to the concept of a relation as a set. We have examined the problem of conjunctive query containment for two important types of label systems. Specifically, we have shown that earlier theorems that deal with relations as sets are generalized for all label systems of type A, which include sets and fuzzy sets as special cases. We have also provided sufficient conditions, and in special cases necessary and sufficient conditions, for label systems of type B, which include relations as multisets, an important special case due to its significance in SQL. We have also addressed the problem of containment of sets of conjunctive queries, generalized again earlier set-based results for all label systems of type A and proving undecidability for label systems of type B.

An interesting open problem is that of containment of individual conjunctive queries for type B systems. We have presented a necessary and sufficient condition for queries with no repeated predicates and a sufficient condition for the general case. Is there a general necessary and sufficient condition? Is the problem decidable? (Note that Chaudhuri and Vardi [1993] established a lower bound for this problem, but not an upper bound.) Given that the problem is decidable for both individual queries and sets of queries over relations as sets and our undecidability result for sets of queries over

multisets, this raises the possibility that the problem is undecidable even for individual queries.

There are several additional areas that require further exploration. These include dealing with recursive queries in the context of deductive databases, formalizing a label system that captures the traditional notion of probabilities and addressing the query containment problem for it, and incorporating the techniques developed in this work into query optimizers that face issues of query containment.

REFERENCES

AHO, A., SAGIV, Y., AND ULLMAN, J. 1979. Equivalence among relational expressions. *SIAM J. Comput. 8*, 2 (May), 218–246.

BLAKELEY, J. A., LARSON, P. A., AND TOMPA, F. W. 1986. Efficiently updating materialized views. In *Proceedings of the 1986 ACM–SIGMOD Conference on the Management of Data* (Washington, D.C., May). ACM, New York, 61–71.

CHANDRA, A. K., AND MERLIN, P. M. 1977. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing* (Boulder, Colo., May) ACM, New York, 77–90

CHAUDHURI, S., AND VARDI, M. 1993. Optimization of real conjunctive queries. In *Proceedings of the 12th ACM–PODS Conference* (Washington, D.C., June). ACM, New York, 59–70.

CHAUDHURI, S., KRISHNAMURTHY, R., POTAMIANOS, S., AND SHIM, K. 1995. Optimizing queries with materialized views. In *Proceedings of the IEEE International Conference on Data Engineering* (Taipei, Taiwan, Mar.). IEEE, New York, 190–200.

DAVIS, M. 1982. *Computability and Unsolvability*. Dover Publications, New York.

DAYAL, U., GOODMAN, N., AND KATZ, R. H. 1982. An extended relational algebra with control over duplicate elimination. In *Proceedings of the 1st ACM–PODS Conference* (Los Angeles, Calif., Mar.). ACM, New York, 117–123.

GRAEFE, G. 1995. Personal communication, May.

GRANT, J., AND MINKER, J. 1981. Optimization in deductive and conventional relational database systems. In *Advances in Data Base Theory*, Vol. 1, J. Minker, H. Gallaire, and J. M. Nicolas, Eds., Plenum Press, New York, 195–234.

IOANNIDIS, Y. E., AND WONG, E. 1991. Towards and algebraic theory of recursion. *J. ACM 38*, 2 (Apr.), 329–381.

JOHNSON, D. S., AND KLUG, A. 1982. Testing containment of conjunctive queries under functional and inclusion dependencies. In *Proceedings of the 1st ACM–PODS Conference* (Los Angeles, Calif., Mar.). ACM, New York, pp. 164–169.

KING, J. J. 1981. Quist: A system for semantic query optimization in relational databases. In *Proceedings of the 7th International VLDB Conference* (Cannes, France, Aug). Pp. 510–517

MAHER, M. J., AND RAMAKRISHNAN, R. 1989. Dèjá vu in fixpoints of logic programs. In *Proceedings of the ALP Symposium on Logic Programming* (Cleveland, Ohio, Oct. 1), 963–980.

MUMICK, I. S., PIRAHESH, H., AND RAMAKRISHNAN, R. 1990. Duplicates and aggregates in deductive databases. In *Proceedings of the 16th International VLDB Conference* (Brisbane, Australia, Aug.). 264–277.

NEGRI, M., PELAGATTI, G., AND SBATTELLA, L. 1991. Formal semantics of SQL queries. *ACM Trans. Database Syst. 16*, 3 (Sept.), 513–534.

PARSAYE, K., AND CHIGNELL, M. 1988. *Expert Systems for Experts*. Wiley, New York.

SAGIV, Y., AND YANNAKAKIS, M. 1980. Equivalences among relational expressions with the union and difference operators. *J. ACM 27*, 4 (Oct.), 633–655.

SELLIS, T. K. 1986. Global query optimization. In *Proceedings of the 1986 ACM–SIGMOD Conference on the Management of Data* (Washington, D.C., May). ACM, New York, 191–205.

SHENOY, S., AND OZSOYOGLU, Z. M. 1987. A system for semantic query optimization. In *Proceedings of the 1987 ACM–SIGMOD Conference on the Management of Data* (San Francisco, Calif., May). ACM, New York, 181–195.

TSATALOS, O., SOLOMON, M., AND IOANNIDIS, Y. 1994. The gmap: A versatile tool for physical data independence. In *Proceedings of the 20th International VLDB Conference* (Santiago, Chile, Sept.), 367–378

VAN EMDEN, M. H. 1986. Quantitative deduction and its fixpoint theory *J. Logic Program. 3*, 1 (Jan.), 37–53.

ZADEH, L. 1965. Fuzzy sets *Inf. Control 8*, 3 (June), 338–353.

Received January 1992; revised June 1993; August 1994, and May 1995; accepted May 1995