Equivalence and Minimization of Conjunctive Queries Under Combined Semantics (extended abstract)

Rada Chirkova
Department of Computer Science
NC State University, Raleigh, NC 27695, USA
chirkova@csc.ncsu.edu

ABSTRACT

The problems of query containment, equivalence, and minimization are fundamental problems in the context of query processing and optimization. In their classic work [2] published in 1977, Chandra and Merlin solved the three problems for the language of conjunctive queries *(CQ queries)* on relational data, under the "set-semantics" assumption for query evaluation. While the results of [2] have been very influential in database research, it was recognized long ago that the set semantics does not correspond to the semantics of the standard commercial query language SQL. Alternative semantics, called *bag* and *bag-set semantics*, have been studied since 1993; Chaudhuri and Vardi in [5] outlined necessary and sufficient conditions for equivalence of CQ queries under these semantics. (The problems of containment of CQ bag and bag-set queries remain open to this day.) More recently, Cohen [7, 8] introduced a formalism for treating (generalizations of) CQ queries evaluated under each of set, bag, and bag-set semantics uniformly as special cases of the more general *combined semantics*. This formalism provides tools for studying broader classes of practical SQL queries, specifically important types of queries that arise in on-line analytical processing (OLAP). Cohen in [8] provides a sufficient condition for equivalence of (generalizations of) combined-semantics CQ queries, as well as sufficient and necessary equivalence conditions for several proper sublanguages of the query language of [8]. To the best of our knowledge, no results on minimization of CQ queries beyond set-semantics queries have been reported in the literature.

Our goal in this paper is to continue the study of equivalence and minimization of CQ queries. We consider the problems of (i) finding minimized versions of combined-semantics CQ queries, and of (ii) determining whether two CQ queries are combined-semantics equivalent. We continue the tradition of [2, 5, 8] of studying these problems using the tool of containment between queries. We extend the containment, equivalence, and minimization results of [2] to general combined-semantics CQ queries, and show the limitations of each extension. We show that the minimization approach of [2] can be extended to general CQ queries without limitations. We also propose a necessary and sufficient condition for equivalence of queries belonging to a large natural sublanguage of combined-semantics CQ queries; this sublanguage encompasses (but is not limited to) all set, bag, and bag-set queries. Our equivalence and minimization results, as well as our general sufficient condition for containment of combined-semantics CQ queries, reduce correctly to the special cases reported in [5] for bag and bag-set semantics. Our containment and equivalence conditions also properly generalize the results of [8], provided the latter are restricted to the language of (combined-semantics) CQ queries.

ICDT 2012, March 26–30, 2012, Berlin, Germany.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*query processing*

General Terms

Algorithms, performance, theory

Keywords

query equivalence and minimization; query optimization; conjunctive queries; set, bag, bag-set, combined semantics for query evaluation

1. INTRODUCTION

Query containment and equivalence are recognized as fundamental problems in evaluation and optimization of database queries. The reason is, for conjunctive queries *(CQ queries)* — a broad class of frequently used queries, whose expressive power is sufficient to express select-project-join queries in relational algebra — query equivalence can be used as a tool in query optimization. Specifically, to find a more efficient *and* answer-preserving formulation of a given CQ query, it is enough to "try all ways" of arriving at a "shorter" query formulation, by removing query subgoals, in a process called query minimization [2]. A subgoal-removal step succeeds only if equivalence (via containment) of the "original" and "shorter" query formulations can be ensured. The equivalence test of [2] for CQ queries is NP complete, whereas equivalence of general relational queries is undecidable.

The query-minimization algorithm of [2] works under the assumption of *set semantics* for query evaluation, where

both the database (stored) relations and query answers are treated as sets. Query answering and reformulation in the set-semantics setting have been studied extensively in the database-theory literature. As a basis, these studies have all used the necessary and sufficient containment condition of [2] for CQ queries. At the same time, the set semantics is not the default query-evaluation semantics in database systems in practice. For instance, in the standard relational query language SQL, duplicates are removed from the answer to a SQL query *only* if the query uses the `DISTINCT` keyword in its `SELECT` clause. This and other discrepancies between the set semantics for query evaluation and the standard of SQL have prompted researchers [5, 12] to consider "bag semantics" and "bag-set semantics" for query evaluation. Under *bag semantics,* both query answers and stored relations are treated as *bags* (that is, *multisets).* Under *bag-set semantics,* query answers are treated as bags, whereas the database relations are assumed to be sets.

In an extended abstract [5] published in PODS in 1993, Chaudhuri and Vardi focused on the hard problem of bag containment for CQ queries. The paper [5] formulates containment and equivalence results, including equivalence tests, for bag and bag-set queries. However, the full version of the paper [5] has never appeared, and the problems of bag and bag-set containment for CQ queries remain open to this day.

The seminal work by Cohen [7, 8] has provided a Datalog-based formalism for treating queries evaluated under each of set, bag, and bag-set semantics uniformly as special cases of the more general *combined semantics.* To show the practical value of the combined-semantics formalism, Cohen exhibited in [8] a number of real-life SQL queries that can be expressed as combined-semantics CQ queries, but cannot be expressed using set, bag, or bag-set semantics. In the following example we show another realistic combined-semantics query, and use it to illustrate issues in equivalence and minimization of combined-semantics queries.

EXAMPLE 1.1. *The application domain used in this example is based on a data-warehousing example from [11]. Consider a retailer that has multiple stores. The retailer carries many items and has an elaborate relational database/warehouse for analysis, marketing, and promotion purposes. One of the tables in the database has the schema* `pos(transactionID, itemID, storeID, date, amount)`*; the table has one million rows. This table represents point-of-sale transactions, with one tuple for every item sold in a transaction. Each tuple has the transaction ID, the ID of the item sold, the ID of the store selling it, the date, and the amount of the sale.*

Suppose that the business-development division of this store chain would like to study the impact, on the total sales, of those transactions in the stores where the item prices are the same as on some fixed date.[1] *We denote this fixed date of interest by the constant* `d1`*. Consider SQL query* `Q` *that could be used for the purpose of this analysis.*

```
(Q) SELECT storeID, amount FROM pos P WHERE EXISTS
(SELECT * FROM pos WHERE itemID = P.itemID AND
storeID = P.storeID AND amount = P.amount AND date = 'd1')
```

[1]We assume that the transaction amount can be used to determine the price of the item. This is true, for instance, for sales of big-ticket items, where each transaction typically records the sale of one such item. We also assume that item prices do not change in the middle of a business day.

For each store ID, query `Q` *returns separately the amount for each transaction that took place in that store on the date* `d1`*. Moreover, for each item that was sold in the store on the date* `d1`*,* `Q` *returns all the* same *purchase amounts for the same item in the same store, as many times as the purchases have happened, regardless of the date. If the analysts want to calculate correctly the total per-store returns for all the transactions that have the same item prices as the same-store transactions for the date* `d1`*, then all they have to do to write the query is to (i) add to the query* `Q` *the clause* `GROUP BY storeID`*, and to (ii) replace* `amount` *by* `sum(amount)` *in the* `SELECT` *clause of* `Q`*. To evaluate the resulting analysis query, the query processor would evaluate the query* `Q`*, and would then apply to the answer to* `Q` *the above grouping and aggregation.*

Due to the large size of the relation `pos`*, the self-join of* `pos` *in the query* `Q`*, via a correlated subquery, should be avoided if at all possible. However, as we show in the full version [6] of this paper,*[2] *it turns out that the query* `Q` *cannot be expressed equivalently as a set, bag, or bag-set CQ query. (The reason that we cannot find an equivalent set, bag, or bag-set version of* `Q` *is that each such version would return the (store, transaction-amount) pairs with different multiplicities than* `Q` *does. Thus, the total per-store returns analysis using* `Q` *as discussed above, cannot be done correctly using a set, bag, or bag-set CQ query.) It follows that optimizing the query* `Q` *by finding an equivalent efficient rewriting cannot be done using the results of past work. At the same time, we can use the results of this current paper to show that the* minimized *– and thus the most efficient-in-execution – version of the query* `Q` *is the query itself.* □

Combined-semantics CQ queries such as the query `Q` of Example 1.1 with grouping and aggregation added, arise naturally in on-line analytical processing *(OLAP)* applications [14, 15]. Such queries occur whenever a data-analysis task calls for a query structure with nested subqueries. Such queries also arise due to joins that go beyond "star-schema joins" [3, 4], which are the only well-understood joins in the literature on OLAP query optimization. See [14, 15, 16, 17] for more detailed discussions of why queries with nested subqueries and with "non-star joins" are natural and frequent in OLAP. (For an additional extended illustration of such queries, see the full version [6] of this paper.) We can use the results of [7, 8] to show that the query `Q` of Example 1.1 cannot be represented equivalently as a SQL query without subqueries, that is as a CQ set, bag, or bag-set query. At the same time, to the best of our knowledge, past work cannot help us determine the most efficient equivalent SQL representation of the query `Q`. It turns out that, by the results in this current paper, the query `Q` cannot be simplified further by equivalent reformulation, specifically by minimization.

Our contributions.

We study equivalence and minimization of unaggregated SQL queries with equality comparisons and possibly with subqueries. We follow the approach of [8], where the study concentrates on Datalog translations of such queries, that is on combined-semantics CQ queries. The requisite translations from SQL to Datalog are straightforward ("as ex-

[2]For all the details that are omitted from this extended abstract, please see the full version [6] of this paper.

pected").[3] In the remainder of this paper, all queries are expressed using the Datalog-based formalism of [8].

We focus on the problems of (i) finding minimized versions of combined-semantics CQ queries, and of (ii) determining whether two CQ queries are combined-semantics equivalent. We continue the tradition of [2, 5, 8] of studying these problems using the tool of containment between queries. All the results in this paper hold for queries that may have constants. Our specific contributions are as follows:

- For combined-semantics containment of CQ queries, in Section 3 we introduce two necessary conditions and a sufficient condition. The latter result properly generalizes both (i) the sufficient condition outlined in [5] for bag containment of CQ queries, and (ii) the general sufficient containment condition that can be obtained from [8] for CQ queries. To formulate our sufficient condition, we introduce "covering mappings" *(CVMs)* between CQ queries. We use CVMs in our results throughout the remainder of the paper.

- In Section 4 we present a necessary condition for CQ-query equivalence. To formulate this condition, we isolate a large class of CQ queries, which we call "explicit-wave queries".[4] We show that this class of queries encompasses, but is not limited to, (i) all CQ set, bag, and bag-set queries, and (ii) all CQ queries for which [8] provides its sufficient and necessary equivalence tests. We refer to all combined-semantics CQ queries that are not explicit-wave queries as "implicit-wave queries." Our necessary condition for query equivalence is asymmetric – it states that if for CQ queries Q and Q' we have the combined-semantics equivalence $Q \equiv_C Q'$, *and* Q is an explicit-wave query, then there exists a CVM from Q' to Q. We discuss why establishing this result is not trivial.

- In Section 5 we propose a sound and complete algorithm for minimizing combined-semantics CQ queries. We also show that for all CQ queries, including all implicit-wave queries, the minimized version of the query exists and is unique up to an isomorphism CVM.

- Finally, in Section 6 we study our proposed conditions for equivalence of CQ queries. Our main focus is on reformulating these conditions using minimized versions of the queries. The reformulations tie our equivalence conditions together with the results of [2, 5]. Our sufficient and necessary condition for equivalence of explicit-wave CQ queries is strictly more powerful than each of the equivalence tests of [8], provided that the latter are applied to CQ queries only.

Due to the space limit, we present here only an extended abstract of our results. All the details are available in the full version [6] of this paper.

The results of this paper can be used directly in query optimizers for database-management systems, as well as for developing minimization methods for queries in more expressive languages than CQ queries and in presence of integrity constraints. Our results can also be used for developing algorithms for rewriting queries using views, and for view selection under combined semantics.

[3]Section 1 in [8] provides some details of the translations.

[4]The term "explicit-wave query" is due to the structures generated by the proof of the main result of Section 4.

2. PRELIMINARIES

2.1 Combined semantics: The framework [8]

2.1.1 Syntax of queries

Predicate symbols are denoted as p, q, r. Databases contain ground atoms for a given set of predicate symbols; we consider finite-size databases only. A database may have several copies of the same atom. To denote this fact, each atom in the database is associated with a *copy number* N. Formally, if p is an n-ary predicate, for an $n \in \mathbb{N}_+$ (with $\mathbb{N}_+$ the set of natural numbers), we write $p(c_1, \ldots, c_n; N)$, with $N \in \mathbb{N}_+$, to denote that there are precisely N copies of $p(c_1, \ldots, c_n)$ in the database. As a shorthand, if $N = 1$, we often omit the copy number N. The *active domain of database* D, denoted $adom(D)$, is the set of all constants mentioned in the ground atoms of D. We adopt a convention by which, for each atom of the form $p(c_1, \ldots, c_n)$ such that database D has $N \geq 1$ copies of that atom, N is an element of $adom(D)$ only if there exists in D an atom $r(c'_1, \ldots, c'_m)$ (where r and p may or may not be the same predicate) such that N is one of c'_1, ..., c'_m.

For query syntax, we denote variables using X, Y, Z, possibly with subscripts, and i, j, k. The former range over constants in the database (i.e., over $adom(D)$), whereas the latter range over copy numbers. For this reason, we call the former *regular variables* (or simply *variables* for short), and we call the latter *copy variables.* We use c, d to denote constants. A *term,* denoted as S, T, is a variable or a constant.

A *relational atom* has the form $p(S_1, \ldots, S_n)$, where p is a predicate of arity n. We also use the notation $p(\bar{S})$, where $\bar{S}$ stands for a sequence of terms $S_1, \ldots, S_n$. A *copy-sensitive atom* has the form $p(\bar{S}; i)$, and is simply a relational atom with copy variable i. We call relational atom $p(\bar{S})$ *the relational template of copy-sensitive atom* $p(\bar{S}; i)$. For each relational atom, its relational template is the atom itself. A *condition,* denoted as L, is a conjunction of relational and copy-sensitive atoms, with duplicate atoms allowed, such that all copy variables in L are unique (i.e., appear in a single copy-sensitive atom, and do not appear in other atoms). Sometimes it will be convenient for us to view condition L as a bag of all and only the elements in the conjunction L.

We distinguish between the variables that appear in the head of a query, and those that only appear in the body. The former are *distinguished (head) variables,* and the latter are *nondistinguished (nonhead) variables.* Nondistinguished variables come in two flavors: *set variables* and *multiset variables.* The intuition for the difference between these two types of variables is as follows. When evaluating a query, different assignments for set variables do not contribute to the multiplicity in which a particular answer is returned by the query. On the other hand, different assignments for multiset variables do contribute to the multiplicity of the returned answers. Technically, in order to differentiate between set variables and multiset variables, we always specify the set of multiset variables in each condition immediately to the right of the condition. As a syntactic requirement, all copy variables must be in the set of multiset variables.

DEFINITION 2.1. *(**Query syntax: CCQ query**) A copy-sensitive conjunctive query (CCQ query) is a nonrecursive expression of the form*

$$Q(\bar{X}) \leftarrow L, M,$$

where $\bar{X}$ contains at least one term, L is a nonempty condition, and M is a set of variables, such that:

- *L contains all the variables in $\bar{X}$; that is, Q is* safe*;*
- *M is a subset of the set of nondistinguished variables of L and contains all copy variables of L. We denote all the copy variables of Q collectively as $M_{copy} \subseteq M$, and all the remaining ("multiset noncopy") variables in M as $M_{noncopy} := M - M_{copy}$.* □

We call each element of the condition L a *subgoal* of Q. The variables in M are the *multiset variables* of Q. The variables in L that are not in $\bar{X}$ or in M are the *set variables* of Q. Consider an illustration.

EXAMPLE 2.1. *Let CCQ query Q be as follows.*

$$Q(X) \leftarrow p(X, X, Y; i), p(X, Z, Y), \{Y, i\}.$$

The condition L of the query Q is the conjunction of the two subgoals of Q with the predicate p. X is the (only) head variable of the query Q, and Z is its (only) set variable. The set $\{Y, i\}$ is the set M of multiset variables of Q; the set M_{copy} of Q comprises the (only) copy variable i of Q, and the set $M_{noncopy}$ of Q has the multiset noncopy variable Y of Q. By definition, $M = M_{copy} \cup M_{noncopy}$. □

We use $S(Q)$ to denote an arbitrary vector, without repetitions, of the set variables of Q, and $\bar{S}(Q)$ to denote an arbitrary vector, without repetitions, of the remaining variables of Q (i.e., the distinguished and multiset variables of Q). By abuse of notation, we will often refer to a query by its head $Q(\bar{X})$ or simply by its head predicate Q. For a vector of terms $\bar{X}$ with $k \geq 1$ elements, we say that a CCQ query with head $Q(\bar{X})$ is a *CCQ k-ary query.*

We will sometimes be interested in special types of queries. A CCQ query Q is a *set query* if it has no multiset variables, that is, if $M = \emptyset$. Query Q is a *multiset query* if Q has no set variables. Further, a multiset query Q is (i) a *bag query* if Q has only copy-sensitive subgoals, and is (ii) a *bag-set query* if Q has only relational subgoals.

2.1.2 Combined semantics for queries

We define how CCQ query $Q(\bar{X}) \leftarrow L, M$ yields a *multiset* of tuples on database D. Intuitively, we start by considering satisfying assignments of the condition L. We then restrict these assignments to the *nonset variables* of L, that is to $\bar{S}(Q)$. Each of these restricted assignments yields a tuple in the result. A formal description of the semantics follows.

Let γ be a mapping of the terms in condition L to values. We will also apply γ to a sequence of terms to derive a sequence of values, in the obvious way. We say that γ is a *satisfying assignment* of L with respect to database D if all of the following conditions hold:

- γ is the identity mapping on constants;
- for all relational atoms $p(\bar{T}) \in L$, there exists an $N \in \mathbb{N}_+$ such that we have $p(\gamma\bar{T}; N) \in D$; and
- for all copy-sensitive atoms $p(\bar{T}; i) \in L$, the following two conditions hold:
  - $\gamma i \in \mathbb{N}_+$ (i.e, γi is a positive natural number);
  - there is an $N \geq \gamma i$ such that $p(\gamma\bar{T}; N) \in D$.

Let $\Gamma(Q, D)$ denote the set of satisfying assignments of L with respect to database D. Let γ be an assignment of the variables in $\bar{S}(Q)$ to constants. We say that γ is *satisfiably extendible* if there is an assignment $\gamma' \in \Gamma(Q, D)$ such that γ and γ' coincide on all terms for which γ is defined, that is, $\gamma'(X) = \gamma(X)$ for all $X \in \bar{S}(Q)$. Intuitively, this means that it is possible to extend γ to derive a satisfying assignment of L. We use $\Gamma_{\bar{S}}(Q, D)$ to denote the set of satisfiably extendible assignments of $\bar{S}(Q)$ with respect to D. For the $\gamma \in \Gamma_{\bar{S}}(Q, D)$ and for the $\gamma' \in \Gamma(Q, D)$ as specified in this paragraph, we say that γ' *contributes* γ *to* $\Gamma_{\bar{S}}(Q, D)$.

We now define the result of applying a query Q to a database D. (We use $\{\!\{\ldots\}\!\}$ to denote a bag of values.)

DEFINITION 2.2. *(**Combined semantics**) Let $Q(\bar{T}) \leftarrow L, M$ be a CCQ query and let D be a database. The* result of applying Q to D under combined semantics, *denoted $Res_C(Q, D)$, is defined as*

$$Res_C(Q, D) \ := \ \{\!\{ \ \gamma(\bar{T}) \mid \gamma \in \Gamma_{\bar{S}}(Q, D) \ \}\!\} \ .$$ □

Note that $Res_C(Q, D)$ is a bag of tuples, that is, $Res_C(Q, D)$ may contain multiple occurrences of the same tuple. Consider an illustration.

EXAMPLE 2.2. *Let CCQ query Q be as in Example 2.1, and consider database $D = \{ p(1, 1, 3; 2), p(1, 1, 4; 3) \}$. Then $Res_C(Q, D)$ is a bag of exactly five copies of tuple* (1)*. Two of these copies in $Res_C(Q, D)$ are due to the atom $p(1, 1, 3; 2)$ in the database D, and the three remaining copies of* (1) *in $Res_C(Q, D)$ are due to to the atom $p(1, 1, 4; 3)$ in D.* □

Under certain circumstances, combined semantics coincides with set, bag, or bag-set semantics. Please see [8] for the details on the three traditional query semantics, specifically on how these semantics can be formulated as special cases of combined semantics.

2.1.3 Query containment and equivalence

Query containment under combined, set, bag, and bag-set semantics is defined in the standard manner. Formally, *Q is contained in Q'* under a given semantics if, for all databases, the bag of values returned by Q is a subbag of the bag of values returned by Q'. We write $Q \sqsubseteq_C Q'$, $Q \sqsubseteq_S Q'$, $Q \sqsubseteq_B Q'$, and $Q \sqsubseteq_{BS} Q'$ if Q is contained in Q' under combined, set, bag, and bag-set semantics, respectively. Similarly, we use $Q \equiv_C Q'$, $Q \equiv_S Q'$, $Q \equiv_B Q'$, and $Q \equiv_{BS} Q'$ to denote the fact that Q is equivalent to Q' under each semantics. $Q \equiv_C Q'$ holds if and only if $Q \sqsubseteq_C Q'$ and $Q' \sqsubseteq_C Q$ both hold. The definitions of $Q \equiv_S Q'$, $Q \equiv_B Q'$, and $Q \equiv_{BS} Q'$ parallel that of $Q \equiv_C Q'$ in the obvious manner.

For CCQ queries Q and Q', we have that (1) $Q \sqsubseteq_S Q'$ iff $Q \sqsubseteq_C Q'$, in case Q and Q' are set queries; (2) $Q \sqsubseteq_B Q'$ iff $Q \sqsubseteq_C Q'$, in case Q and Q' are bag queries; and (3) $Q \sqsubseteq_{BS} Q'$ iff $Q \sqsubseteq_C Q'$, in case Q and Q' are bag-set queries.

For a class $\mathcal{Q}$ of queries: The *$\mathcal{Q}$-containment problem for combined semantics* is: *Given queries Q and Q' in $\mathcal{Q}$, determine whether $Q \sqsubseteq_C Q'$.* The *$\mathcal{Q}$-equivalence problem* is defined similarly using $\equiv_C$ instead of $\sqsubseteq_C$. The two problems can be defined similarly for other semantics.

2.2 Equivalence and minimization results

Homomorphisms and set queries. Given two conditions $\phi(\bar{U})$ and $\psi(\bar{V})$, a *homomorphism* from $\phi(\bar{U})$ to $\psi(\bar{V})$ is a mapping h from the set of terms in $\bar{U}$ to the set of terms in $\bar{V}$ such that (1) $h(c) = c$ for each constant c, (2) for each relational atom $r(U_1, \ldots, U_n)$ of ϕ we have that $r(h(U_1), \ldots, h(U_n))$ is in ψ, and (3) for each copy-sensitive atom $p(W_1, \ldots, W_n; i)$ of ϕ we have that $p(h(W_1), \ldots, h(W_n); h(i))$ is in ψ. Given two CCQ k-ary queries $Q_1(\bar{X}) \leftarrow \phi(\bar{X}, \bar{Y}), M_1$ and $Q_2(\bar{X}') \leftarrow \psi(\bar{X}', \bar{Y}'), M_2$, a *containment mapping* from Q_1 to Q_2 is a homomorphism h from $\phi(\bar{X}, \bar{Y})$ to $\psi(\bar{X}', \bar{Y}')$ such that $h(\bar{X}) = \bar{X}'$.

THEOREM 2.1. *[2] Given two CCQ set queries Q_1 and Q_2 of the same arity, $Q_1 \sqsubseteq_S Q_2$ holds if and only if there is a containment mapping from Q_2 to Q_1.* □

This classic result of [2] forms the basis for a sound and complete test for set-equivalence of CCQ set queries Q and Q', by definition of set-equivalence $Q \equiv_S Q'$.

We now introduce the notion of a "reduced-condition query" for CCQ query. Given a CCQ query $Q(\bar{X}) \leftarrow L, M$, a CCQ query $Q'(\bar{X}) \leftarrow L', M'$ is a *(proper) reduced-condition query for Q* if (i) L' is a (proper) subbag of L, (ii) the set M' is the set of all elements of M that occur in L', and (iii) Q' is a safe query (i.e., L' contains all the variables in $\bar{X}$).

DEFINITION 2.3. *(***Minimized CCQ query; minimized version of CCQ query***) A CCQ query Q is a* minimized CCQ query *if for each subgoal s of Q, the removal of s from the condition of Q results in a query Q' such that $Q \not\equiv_C Q'$. A CCQ query Q is a* minimized version of CCQ query Q' *if (1) Q is a reduced-condition query for Q', (2) Q is a minimized query, and (3) $Q \equiv_C Q'$.* □

THEOREM 2.2. *[2] Given two CCQ set queries Q_1 and Q_2 of the same arity:*

(1) The minimized version of Q_1 exists and is unique up to isomorphism; and

(2) $Q_1 \equiv_S Q_2$ holds if and only if the minimized versions of Q_1 and of Q_2 are isomorphic. □

(Two CCQ queries Q_1 and Q_2, with respective sets of multiset variables M_1 and M_2, are *isomorphic* if there exists a one-to-one containment mapping from Q_1 *onto* Q_2 such that the mapping induces a bijection from M_1 to M_2, and there exists another containment mapping from Q_2 onto Q_1, with (symmetrically) the same properties.)

Bag and bag-set queries. For bag and bag-set semantics, the following conditions are known for CCQ query equivalence. (Query Q_c is a *canonical representation* of query Q if Q_c is the result of removing all duplicate atoms from the condition of Q.)

THEOREM 2.3. *[5] Let Q and Q' be CCQ queries. Then (1) When Q and Q' are bag queries, $Q \equiv_B Q'$ iff Q and Q' are isomorphic. (2) When Q and Q' are bag-set queries, $Q \equiv_{BS} Q'$ iff Q_c and Q'_c are isomorphic.* □

Combined-semantics queries. The next result is a sufficient condition of [8] for equivalence of two queries under combined semantics. In [8], Cohen formulates each of Definition 2.4 and Theorem 2.4 for CCQ queries that may also contain negation and inequality comparisons. (In condition (3) of Definition 2.4 we treat the query conditions, which are conjunctions of atoms, as bags of the same atoms. Given a bag B, we call a set S the *core-set* of B if S is the result of dropping all duplicates of all elements of B.)

DEFINITION 2.4. *(***Multiset-homomorphism [8]***) Let $Q(\bar{X}) \leftarrow L, M$ and $Q'(\bar{X}') \leftarrow L', M'$ be two k-ary CCQ queries, for $k \geq 1$. Let φ be a mapping from the terms of Q' to the terms of Q.*[5] *We say that φ is a* multiset-homomorphism from Q' to Q *if φ satisfies all of the following conditions:*

1. *$\varphi\bar{X}' = \bar{X}$;*
2. *φ is the identity mapping on constants;*
3. *the core-set of $\varphi L'$ is a subset of the core-set of L ;*
4. *$\varphi M' \subseteq M$; and*
5. *$\varphi Y \neq \varphi Y'$ for every two variables $Y \neq Y' \in M'$.* □

For every mapping φ that satisfies conditions 1–3 of Definition 2.4, we call φ a *generalized containment mapping (GCM)*.

We say that two CCQ queries Q and Q' are *multiset homomorphic* whenever there is a multiset-homomorphism from Q to Q' and another from Q' to Q.

THEOREM 2.4. *[8] Given CCQ queries Q and Q'. If Q and Q' are multiset homomorphic then $Q \equiv_C Q'$.* □

Note 1. Theorem 2.4 is proved in [8] via showing that for two (generalized) CCQ queries Q and Q', the existence of a multiset-homomorphism from Q' to Q implies $Q \sqsubseteq_C Q'$.

It is shown in [8] that the sufficient equivalence condition of Theorem 2.4 is not necessary for the query classes considered in [8].

3. CONTAINMENT AND MAPPINGS

In this section, for combined-semantics containment of CCQ queries, we introduce two necessary conditions, Theorems 3.1 and 3.2, and a sufficient condition, Theorem 3.3. The latter result properly generalizes both (i) the sufficient condition outlined in [5] for bag containment of CQ queries, and (ii) the general sufficient containment condition for CCQ queries that can be obtained from [8]. To formulate Theorem 3.3, we introduce "covering mappings" *(CVMs)* between CCQ queries. We use CVMs in our results throughout the remainder of this paper.

Throughout this paper, we use the notation $Q(\bar{X}) \leftarrow L, M$ and $Q'(\bar{X}') \leftarrow L', M'$ for the definitions of CCQ queries Q and Q'. The conditions of Q and Q' may have constants.

3.1 Necessary conditions for containment

We introduce two necessary conditions, Theorems 3.1 and 3.2, for a CCQ query Q being combined-semantics contained in CCQ query Q'. (We denote by $|S|$ the cardinality of set S.)

THEOREM 3.1. *Let Q and Q' be two k-ary CCQ queries. Then $Q \sqsubseteq_C Q'$ implies both $|M_{copy}| \leq |M'_{copy}|$ and $|M_{noncopy}| \leq |M'_{noncopy}|$.* □

[5] We also apply φ to atoms and conjunctions of atoms, in the obvious way, e.g., $\varphi(p(\bar{S})) = p(\varphi(\bar{S}))$.

The idea of the proof of Theorem 3.1, see [6], is that we use the definition of query Q to construct a special database D. Some answer to Q on D has a multiplicity (in the bag $Res_C(Q, D)$) that is proportional to $|adom(D)|^{|M|}$. Then we can use several versions of the database D to prove Theorem 3.1 by contradiction: We assume either $|M_{copy}| > |M'_{copy}|$ or $|M_{noncopy}| > |M'_{noncopy}|$, and obtain that $Q \sqsubseteq_C Q'$ cannot hold. The challenge in the proof is in the combination of allowing constants in the condition of Q and of arriving at "the right" multiplicity of the answer to Q on D when we are to show that $|M_{copy}| \leq |M'_{copy}|$ must hold whenever $Q \sqsubseteq_C Q'$.

We call a pair (Q, Q') of CCQ queries a *containment-compatible CCQ pair* if (i) The (positive) head arities of Q and Q' are the same; (ii) $|M_{copy}| \leq |M'_{copy}|$; and (iii) $|M_{noncopy}| \leq |M'_{noncopy}|$. (Note the asymmetry in the notation for the pair.) Further, we call a pair (Q, Q') of CCQ queries an *equivalence-compatible CCQ pair* if each of (Q, Q') and (Q', Q) is a containment-compatible CCQ pair. By Theorem 3.1 we have that, whenever $Q \sqsubseteq_C Q'$ ($Q \equiv_C Q'$, respectively) holds, then (Q, Q') is a containment-compatible (an equivalence-compatible, respectively) CCQ pair.

We now generalize the "only-if" part of the classic result of [2], see Theorem 2.1 in Section 2.2, to CCQ queries. For the definition of generalized containment mapping, *GCM,* see Section 2.2. We begin by introducing another definition that we need to formulate our generalization, Theorem 3.2.

For a CCQ query Q, we say that CCQ query Q_{ce} is a *copy-enhanced version of* Q if Q_{ce} is the result of adding a distinct copy variable to each relational subgoal of Q. (We can show that for a query Q, all copy-enhanced versions of Q are identical up to renaming of the copy variables introduced in the construction of Q_{ce}.) Further, for each CCQ query Q' that is a reduced-condition query for CCQ query Q, we obtain the query Q'_{ce} by removing from Q_{ce} those subgoals that do not correspond to the subgoals of Q'. (See [6] for the formal definition.)

We are now ready to formulate Theorem 3.2.

THEOREM 3.2. *Given CCQ queries Q and Q' such that $Q \sqsubseteq_C Q'$. Then there exists a GCM from Q'_{ce} to Q_{ce}.* □

The proof of Theorem 3.2, see [6], is a generalization of the proof, via canonical databases, of the result of [2].

Neither Theorem 3.1 not Theorem 3.2 provides a *sufficient* condition for combined-semantics containment of CCQ queries: Example 3.1 is a counterexample in both cases.

EXAMPLE 3.1. *Consider CCQ queries Q and Q':*

$Q(X) \leftarrow p(X,Y), p(Y,Z), p(Z,X;i), \{Y,i\}.$
$Q'(X) \leftarrow p(X,Y), p(Y,Z), p(Z,X;i), \{Z,i\}.$

Apart from the choice of multiset variables, Q and Q' are clearly isomorphic. However, $Q \equiv_C Q'$ does not hold, as witnessed by database $D = \{p(1,2), p(2,3), p(3,1), p(1,4), p(4,3)\}$. Our results in this paper permit us to determine $Q \not\equiv_C Q'$ syntactically, see Section 4. To the best of our knowledge, no previous work provides a formal procedure to determine $Q \not\equiv_C Q'$ for queries such as in this example. □

Each of Theorem 3.1 and Theorem 3.2 yields a necessary condition for combined-semantics *equivalence* of CCQ queries in a natural way. For instance:

COROLLARY 3.1. *Let Q and Q' be two k-ary CCQ queries such that $Q \equiv_C Q'$. Then we have $|M_{copy}| = |M'_{copy}|$ and $|M_{noncopy}| = |M'_{noncopy}|$.* □

3.2 Covering mappings for CCQ queries

In this subsection, we define covering mappings *(CVMs)* between CCQ queries, and study properties of CVMs. (See [6] for the proofs of all the results of this subsection.)

DEFINITION 3.1. *(**Covering mapping (CVM)**) For CCQ queries Q and Q', a mapping, μ, from the terms of Q' to the terms of Q is called a* covering mapping (CVM) from Q' to Q *whenever μ satisfies all of the following conditions:*

(1) μ maps each constant (if any) in Q' to itself;

(2) applying μ to the vector $\bar{X}'$ yields the vector $\bar{X}$;

(3) the set of terms in $\mu M'_{copy}$ is exactly M_{copy}, and the set of terms in $\mu M'_{noncopy}$ includes all of $M_{noncopy}$;

(4) for each relational subgoal of Q', of the form $s(\bar{Y})$, there exists in Q a relational subgoal $s(\mu(\bar{Y}))$ or a copy-sensitive subgoal $s(\mu(\bar{Y}); i)$, with $i \in M_{copy}$; and

(5) for each copy-sensitive subgoal of Q' of the form $s(\bar{Y}; i)$, there exists in Q a subgoal $s(\mu(\bar{Y}); \mu(i))$. □

By Definition 3.1, if there exists a CVM from CCQ query Q' to CCQ query Q, then (Q, Q') is a containment-compatible CCQ pair. It is immediate from Definition 3.1 that if a mapping μ is a CVM from Q' to Q, then μ induces a surjection from the set of copy-sensitive subgoals of Q' to the set of copy-sensitive subgoals of Q. Observe also that in case both Q and Q' are set queries, Definition 3.1 becomes the definition of containment mapping [2] from Q' to Q.

For the special case where (Q, Q') is an *equivalence*-compatible CCQ pair, we call each CVM from Q' to Q a *same-scale covering mapping (SCVM) from Q' to Q.* By definition, each SCVM from Q' to Q is a bijection from the set M' to the set M when restricted to the domain M'.

The intuition for Definition 3.1 comes from our use of CVMs later in this paper (Section 5) as a tool for minimizing CCQ queries. Consider the following illustration.

EXAMPLE 3.2. *Let queries Q and Q' be as follows.*

$Q(X) \leftarrow p(X,X,Y;i), p(X,Z,Y), \{Y,i\}.$
$Q'(X) \leftarrow p(X,X,Y;i), \{Y,i\}.$

By Definition 2.4, there does not exist a multiset homomorphism [8], or even a GCM, from Q to Q'. At the same time, by our results of Section 5, Q' is a minimized version of Q. We can ascertain this fact by using a CVM, μ, from Q to Q': $\mu = \{ X \to X, Y \to Y, i \to i, Z \to X \}$. □

As illustrated by Example 3.2, CVMs are not GCMs. Indeed, the definition of CVMs gives up explicitly on condition (3) for GCMs (see Definition 2.4); by this condition, for each subgoal s of Q in Example 3.2, we must have that $\mu(s)$ is a subgoal of Q'. While CVMs are not GCMs, a nice relationship exists between CVMs and GCMs, see Proposition 3.2. To formulate Proposition 3.2, we use the following definition, in which we treat query conditions as bags of atoms.

Given CCQ query Q, let $\mathcal{T}(Q)$ be the set of relational templates of all (if any) copy-sensitive subgoals of Q. We recall that CCQ query Q_c is a canonical representation of CCQ query Q if Q_c is the result of removing all duplicate atoms from the condition of Q.

DEFINITION 3.2. *(**(Un)regularizing a CCQ query**) Given CCQ query Q, with canonical representation Q_c. Then (1) A* regularized version of Q *is a CCQ query Q_r obtained by dropping from the condition of Q_c all elements of the set $\mathcal{T}(Q)$; (2) A* deregularized version of Q *is a CCQ query Q_d obtained by adding to the condition of Q_r all elements of the set $\mathcal{T}(Q)$; (3) An* unregularized version of Q *is a CCQ query Q_u obtained by adding to the condition of Q_r one or more duplicates of the existing relational subgoals, and/or one or more elements (possibly with duplicates) of the set $\mathcal{T}(Q)$.* □

The following result is straightforward.

PROPOSITION 3.1. *Given a CCQ query Q. Then (1) Each of Q_r and Q_d is a well defined, unique and polynomial-time computable CCQ query; (2) $Q_r \equiv_C Q$ and $Q_d \equiv_C Q$ both hold; and (3) For each unregularized version Q_u of Q, we have that $Q_u \equiv_C Q$ holds.* □

(See [6] for an illustration and for a discussion of the query versions as specified in Definition 3.2.)

We are now ready to formulate Proposition 3.2.

PROPOSITION 3.2. *Given CCQ queries Q and Q'. Then for each CVM, μ, from Q to Q', we have that (1) μ is a* GCM *from Q to the deregularized version of Q', and (2) μ is a CVM from Q to the regularized version of Q'.* □

In Example 3.2, we are given the regularized version Q'_r of the query Q'. The deregularized version of Q' is $Q'_d(X) \leftarrow p(X,X,Y;i), p(X,X,Y), \{Y,i\}$. The mapping μ of Example 3.2 (i) is a GCM from Q to Q'_d, (ii) is a CVM from Q to Q'_r, and (iii) is not a GCM from Q to Q'_r.

It turns out that CVMs furnish a rather general sufficient condition for CCQ combined-semantics containment:

THEOREM 3.3. *Given CCQ queries Q and Q', such that there exists a CVM from Q' to Q. Then $Q \sqsubseteq_C Q'$ holds.* □

Theorem 3.3 generalizes properly both (i) the sufficient condition of [2] for containment between CCQ set queries, see Theorem 2.1, and (ii) the well-known result of [5] stating that a containment mapping[6] from CCQ bag query Q' *onto* CCQ bag query Q ensures containment $Q \sqsubseteq_B Q'$. In fact, to the best of our knowledge, the proof of our Theorem 3.3 is the first formal proof of the latter result from [5].

The condition of Theorem 3.3 does not appear to be a necessary condition for containment of CCQ queries. Indeed, a well-known example of [5] claims (without proof) containment $Q \sqsubseteq_C Q'$, but no CVM exists from Q' to Q in that example. (See [6] for the details.)

Finally, we compare CVMs with multiset homomorphisms [8], see Definition 2.4. For a fixed pair of CCQ queries Q and Q', with respective sets of multiset variables M and M', each CVM from Q' to Q has the range *at least* M when restricted to the domain M', and each multiset homomorphism from Q' to Q has the range *at most* M when restricted to the domain M'. Therefore, general CVMs and multiset homomorphisms are incomparable when applied to pairs of CCQ queries. At the same time, we have the following result for *SCVMs* and multiset-homomorphisms.

[6]The "containment mapping" terminology of [5] results from the use in that paper of a syntax for bag queries that does not coincide with the syntax of [8] used in this current paper. See [6] for a detailed discussion.

PROPOSITION 3.3. *Given an equivalence-compatible CCQ pair (Q, Q'). Then each SCVM from Q' to Q is a multiset-homomorphism from Q' to* the deregularized version *of Q, and vice versa.* □

For instance, consider the mapping μ of Example 3.2 from the terms of the query Q to the terms of the query Q' of the example. This mapping is a CVM from Q to Q' and is also a multiset-homomorphism from Q to the deregularized version Q'_d of Q', $Q'_d(X) \leftarrow p(X,X,Y;i), p(X,X,Y), \{Y,i\}$. (Observe that there is no GCM from query Q'_d to query Q'.)

As an immediate corollary of Propositions 3.2 and 3.3, we have that for each equivalence-compatible CCQ pair (Q, Q'), the existence of a multiset-homomorphism from Q' to Q implies the existence of a CVM from Q' to Q. From this result and from Example 3.2, we obtain that the restriction of Theorem 2.4 (due to [8]) to CCQ queries does not have quite the same power as the sufficient condition for equivalence of CCQ queries that is immediate from Theorem 3.3. (See Theorem 6.1 here for an explicit formulation; by Theorem 6.1, we have $Q \equiv_C Q'$ for the queries of Example 3.2.) In fact, by Example 3.2 we have that our Theorem 3.3 is a proper generalization of the (implicit) query-containment condition of [8], provided that the latter is applied to CCQ queries only; see Note 1 in Section 2.2. (By Definition 2.4 and by Theorem 3.1, the existence of a multiset-homomorphism from CCQ query Q' to CCQ query Q implies $Q \sqsubseteq_C Q'$ only when (Q, Q') is an equivalence-compatible CCQ pair.)

4. EQUIVALENCE: ASYMMETRIC NECESSARY CONDITION

In this section we present a necessary condition for CCQ query equivalence, Theorem 4.1. To formulate Theorem 4.1, we isolate a large well-behaved class of combined-semantics CQ queries, which we call "explicit-wave queries." Theorem 4.1 is asymmetric: It states that if for CCQ queries Q and Q' the combined-semantics equivalence $Q \equiv_C Q'$ holds, *and* we have that Q is an explicit-wave query, then there exists a CVM from Q' to Q. We discuss why establishing this result is not trivial. (See [6] for the full proof.)

We begin by introducing Definition 4.1. This technical definition is required for the proof of Theorem 4.1 to go through. Given a CCQ query Q, with set $M_{noncopy} \neq \emptyset$ of multiset noncopy variables, we say that a GCM μ from Q to itself is a *noncopy-permuting GCM* if the mapping resulting from restricting the domain of μ to $M_{noncopy}$ is a bijection from $M_{noncopy}$ to itself. For two noncopy-permuting GCMs, μ_1 and μ_2, from Q to itself, we say that μ_1 and μ_2 *agree on $M_{noncopy}$* if μ_1 and μ_2 induce the same mapping from $M_{noncopy}$ to itself. If for CCQ query Q we have $M_{noncopy} = \emptyset$, we say that all GCMs from Q to itself are noncopy-permuting GCMs, and that all pairs of such GCMs agree on $M_{noncopy}$.

In Definition 4.1, for a CCQ query Q and for its copy-enhanced version Q_{ce}, we will call "the original copy-sensitive subgoals of Q" those copy-sensitive atoms that are present in the conditions of both Q and Q_{ce}.

DEFINITION 4.1. *(**Explicit-wave CCQ query**) A CCQ query Q is an* explicit-wave (CCQ) query *if one of the following conditions holds:*

(1) Q has at most one copy-sensitive subgoal; or

(2) For the set $M_{noncopy}$ of multiset noncopy variables of Q, and for each pair (μ_1, μ_2) of noncopy-permuting GCMs from Q_{ce} to itself, such that μ_1 and μ_2 agree on *$M_{noncopy}$, for each original copy-sensitive subgoal, s, of Q we have that $\mu_1(s)$ and $\mu_2(s)$ have the same relational template.* □

The problem of determining whether a given CCQ query is an explicit-wave query can easily be seen to be in co-NP. It is open whether this upper complexity bound is tight.

As an example, any CCQ query Q that has a distinct predicate name for each subgoal (i.e., is a query "without self-joins") can be shown to be an explicit-wave query.

For each CCQ query Q that is not explicit-wave, we call Q an *implicit-wave query.* Consider an illustration.

EXAMPLE 4.1. *Consider CCQ queries Q and Q'.*

$$Q(X_1) \leftarrow r(X_1, Y_1, Y_2, X_2; i), r(X_1, Y_1, Y_2, X_3; j), \{Y_1, Y_2, i, j\}.$$
$$Q'(X_1) \leftarrow r(X_1, Y_1, Y_2, X_2; i), r(X_1, Y_1, Y_2, X_2; j), \{Y_1, Y_2, i, j\}.$$

The only difference between the queries is that the two subgoals of the query Q have different set *variables, X_2 and X_3, whereas the two subgoals of Q' have the same set variable X_2. We can show [6] that Q is an implicit-wave query.*

There exist both a multiset homomorphism and a CVM from the query Q to the query Q'. (Recall that each of the two mappings provides a sufficient condition for $Q' \sqsubseteq_C Q$.) Observe that there is no isomorphism between Q and Q'. The remarkable part is that no multiset homomorphism or CVM exists in the opposite direction, that is from Q' to Q. Yet, $Q \equiv_C Q'$ does hold [6]. It does not help much that there exists a GCM from Q' to Q. By Theorem 3.2, the existence of a GCM is a necessary, rather than sufficient, condition for the containment $Q \sqsubseteq_C Q'$. (To apply Theorem 3.2, observe that Q and Q_{ce} are identical, as are Q' and Q'_{ce}.) □

Queries such as the query Q of Example 4.1 are of the kind that does not seem to have been studied before. For instance, implicit-wave CCQ queries cannot occur under set, bag, or bag-set semantics.[7] By the main result of this section, Theorem 4.1, under these three traditional semantics, as well as in other cases of combined semantics, there exist "symmetric" CVM mappings between equivalent CCQ queries. That is, for each pair (Q, Q') of CCQ queries such that each of Q and Q' is an explicit-wave query, $Q \equiv_C Q'$ implies that a CVM exists from Q to Q'. What is important is that in all such cases, a mapping of the *same type* (i.e., also a CVM) always exists also from Q' to Q. Example 4.1 illustrates that such symmetry does not hold for unrestricted pairs of CQ queries under combined semantics.

We now state Theorem 4.1.

THEOREM 4.1. *Given CCQ queries Q and Q', such that (i) Q is an explicit-wave query, and (ii) $Q \equiv_C Q'$. Then there exists a SCVM from Q' to Q.* □

Theorems 3.3 and 4.1 yield immediately a necessary and sufficient equivalence condition for CCQ explicit-wave queries. We study this equivalence condition in Section 6.

[7]We prove this claim in Section 6.

Due to the well-known example given without proof in [5], it appears that condition (ii) of Theorem 4.1 cannot be replaced by condition $Q \sqsubseteq_C Q'$ (while also replacing SCVMs by CVMs), even when Q is an explicit-wave query. Alternatively, we cannot remove condition (i) of Theorem 4.1. Indeed, in Example 4.1 there is a SCVM from query Q to explicit-wave query Q', but there is no SCVM from Q' to Q, even though $Q \equiv_C Q'$ holds [6]. Thus, Theorem 4.1 provides an *asymmetric* necessary condition for CCQ-query equivalence. This asymmetry does not appear to have been explored in previous work. One reason for this is that, as we have mentioned, under the three traditional semantics all CCQ queries are explicit-wave queries. In [8], Cohen explores query classes that properly subsume the class of CCQ queries. When restricted to CCQ queries, all the necessary and sufficient conditions of [8] for combined-semantics query equivalence require the queries to be explicit-wave queries. We note that none of the necessary and sufficient conditions of [8] applies to our Examples 3.1 or 3.2, even though all the queries in the two examples are explicit-wave queries. Yet, by an equivalence test that is immediate from our Theorems 3.3 and 4.1, we have $Q \not\equiv_C Q'$ for the queries of Example 3.1, and $Q \equiv_C Q'$ for the queries of Example 3.2.

In the remainder of this section we outline the idea of the proof [6] of Theorem 4.1. Intuitively, we generalize the proof, via canonical databases, of the existence of a containment mapping from CCQ set query Q' to CCQ set query Q whenever $Q \equiv_S Q'$. There is a major challenge in the generalization: We are now looking not just for a containment mapping, but for a SCVM from Q' to Q. That is, the desired mapping must map each multiset variable of Q' into a distinct multiset variable of Q. Showing that we have constructed a mapping with this property is thus an essential part of the proof. (Note that in Theorem 4.1, we have no information about the structural, i.e. syntactic, relationship between the given queries Q and Q'.)

For a given CCQ query Q, the proof of Theorem 4.1 constructs an infinite number of databases, where each database $D_{\bar{N}^{(i)}}(Q)$, $i \geq 1$, can be thought of as a union of "extended canonical databases" for Q. Similarly to canonical databases for CCQ set queries, each ground atom in each database $D_{\bar{N}^{(i)}}(Q)$ can be associated, via a mapping that we denote $\nu_Q^{(i)}$, with a unique subgoal of the query Q.

The role of each database $D_{\bar{N}^{(i)}}(Q)$ in the proof of Theorem 4.1 is that the database represents a particular combination of multiplicities of the values of (some of) the multiset variables Y_1, Y_2, ..., Y_n, for some $n \geq 1$, of the query Q. (We have that $n \geq 1$ for all CCQ queries Q and Q' such that $Q \equiv_C Q'$ and at least one of Q and Q' is not a set query.) For each database $D_{\bar{N}^{(i)}}(Q)$, we represent the n respective multiplicities as natural numbers $N_1^{(i)}$ through $N_n^{(i)}$, or equivalently via the n-ary vector $\bar{N}^{(i)}$.

By construction of the databases $D_{\bar{N}^{(i)}}(Q)$, we have that some fixed tuple, t_Q^*, is an element of the bag $Res_C(Q, D_{\bar{N}^{(i)}}(Q))$ for each $i \geq 1$. Moreover, *for all queries Q'' such that (Q, Q'') is an equivalence-compatible CCQ pair,* we have that the multiplicity of the tuple t_Q^* in each bag $Res_C(Q'', D_{\bar{N}^{(i)}}(Q))$ (that is, for each $i \geq 1$) can be expressed using the symbolic representations, N_1 through N_n, of the respective elements $N_1^{(i)}, \ldots, N_n^{(i)}$ of the vector $\bar{N}^{(i)}$. That is, for each such query Q'', we can obtain explicitly a function, $\mathcal{F}_{(Q)}^{(Q'')}$, in terms of the n variables $N_1, \ldots,$

N_n, such that whenever we substitute $N_j^{(i)}$ for N_j, for each $j \in \{1, \ldots, n\}$, the resulting expression in terms of $N_1^{(i)}, \ldots, N_n^{(i)}$ evaluates to the multiplicity of the tuple t_Q^* in the bag $Res_C(Q'', D_{\bar{N}^{(i)}}(Q))$.

A key observation in the proof of Theorem 4.1 is that for our fixed query Q and for each CCQ query Q' such that $Q' \equiv_C Q$, it must be that the functions $\mathcal{F}_{(Q)}^{(Q')}$ and $\mathcal{F}_{(Q)}^{(Q)}$ output the same value on each database $D_{\bar{N}^{(i)}}(Q)$, $i \geq 1$.

Consider the simplest case, where our query Q has no self-joins and has $|M| = n \geq 1$. In this case, by construction of the databases, we have that the function $\mathcal{F}_{(Q)}^{(Q)}$ for the query Q is the monomial $\Pi_{j=1}^n N_j$. Consider an arbitrary assignment, γ, from Q to a $D_{\bar{N}^{(i)}}(Q)$. We have that each such γ has contributed to the construction of the database; we call γ a *generative assignment from Q to $D_{\bar{N}^{(i)}}(Q)$*. We can show that the composition $\nu_Q^{(i)} \circ \gamma$ is a SCVM from Q to itself. (Note the presence in the product $\Pi_{j=1}^n N_j$ of the variables for all the n multiset variables of Q.) Moreover, for each query Q' such that $Q' \equiv_C Q$, the function $\mathcal{F}_{(Q)}^{(Q')}$ is forced (by $Q' \equiv_C Q$ and by $\mathcal{F}_{(Q)}^{(Q)}$ being a multivariate polynomial) to be exactly $\Pi_{j=1}^n N_j$, regardless of the structural relationship between Q and Q'. We show that whenever $\mathcal{F}_{(Q)}^{(Q')} = \Pi_{j=1}^n N_j$, an assignment from Q' to a database $D_{\bar{N}^{(i)}}(Q)$ can be composed with the mapping $\nu_Q^{(i)}$ to yield a SCVM from Q' to Q, precisely due to the presence in the function $\mathcal{F}_{(Q)}^{(Q')}$ of the "representative" N_j of each multiset variable Y_j of the query Q, for $1 \leq j \leq n$.

We now use the discussion of this simplest special case to provide a general high-level intuition of the proof of Theorem 4.1: It turns out that for all CCQ queries Q, there is a monomial, in terms of *all* of $N_1, \ldots, N_n$, that contributes to the construction of the function $\mathcal{F}_{(Q)}^{(Q)}$ and that reflects the multiplicity, in the set[8] $\Gamma_{\bar{S}}^{(t_Q^*)}(Q, D_{\bar{N}^{(i)}}(Q))$, of all generative assignments from Q to databases $D_{\bar{N}^{(i)}}(Q)$. We call this monomial, $\mathcal{P}_*^{(Q)}$, *the wave of the query Q*. Suppose that, for a query Q' such that $Q' \equiv_C Q$, we can show that the function $\mathcal{F}_{(Q)}^{(Q')}$ has, as a term, the wave of Q *backed up by assignments from Q' to the databases $D_{\bar{N}^{(i)}}(Q)$*. Then we can use these assignments and the mapping $\nu_Q^{(i)}$ to construct a SCVM from Q' to Q.

There are two significant challenges in extending this idea to all CCQ queries. First, the term $\mathcal{P}_*^{(Q)}$ may not be "visible" in the expression for $\mathcal{F}_{(Q)}^{(Q)}$. As a result, we can show that $\mathcal{P}_*^{(Q)}$ does not necessarily contribute to the construction of the function $\mathcal{F}_{(Q)}^{(Q')}$, even in case $Q \equiv_C Q'$. Second, in general, function $\mathcal{F}_{(Q)}^{(Q')}$ may have terms that are *not* backed up by assignments from Q' to databases $D_{\bar{N}^{(i)}}(Q)$. Both challenges arise from the fact that the function $\mathcal{F}_{(Q)}^{(Q'')}$, in terms of $N_1, \ldots, N_n$, is, in general, *not* a multivariate polynomial on its entire domain.

To overcome the first challenge, we introduce the restriction that Q be an *explicit-wave query*. (Hence Definition 4.1 is necessarily technical.) Even under this restriction, overcoming the second challenge requires a nontrivial proof.

[8]For a CCQ k-ary query Q ($k \geq 1$), for a database D, and for a fixed k-tuple t, we denote by $\Gamma_{\bar{S}}^{(t)}(Q, D)$ the set of all tuples t' in $\Gamma_{\bar{S}}(Q, D)$ such that the projection of t' on the vector of the head arguments of the query Q is the tuple t.

5. MINIMIZING CCQ QUERIES

In this section we propose a sound and complete algorithm for minimizing CCQ queries. We also show that for *all* CCQ queries, including implicit-wave queries, the minimized version of the query exists and is unique up to an isomorphism SCVM. (An *isomorphism SCVM* from CCQ query Q to CCQ query Q' is a SCVM from Q to Q' that is an isomorphism mapping from the terms of Q to the terms of Q'.) It turns out that minimizing arbitrary CCQ queries is at most as hard as minimizing CQ *set* queries using the approach of [2]. Besides being contributions of this paper in their own right, the results of this section permit us to formulate, in Section 6, equivalence tests for CCQ queries that tie our results together with those of [2, 5].

By Definition 2.3, to find a minimized version of a CCQ query Q, one would remove subsets of subgoals of Q so as to arrive at a minimized query Q' such that $Q' \equiv_C Q$. Our approach is to generalize to the case of CCQ queries the minimization approach that was applied in [2] to CQ set queries: Given a CCQ query Q, the search space of all candidate minimized versions of Q can be restricted to the set $\mathcal{Q}_{min}(Q)$. This set comprises all reduced-condition queries Q' for Q such that (i) Q' has all the multiset variables of the query Q, and such that (ii) there exists a GCM from Q_{ce} *onto* Q'_{ce}. By Theorems 3.1 and 3.2, the set $\mathcal{Q}_{min}(Q)$ contains all minimized versions of the query Q.

It turns out that for every CCQ query Q, the search space $\mathcal{Q}_{min}(Q)$ can be found by generating all CCQ queries obtainable by applying to Q all possible SCVMs of a certain type from Q to itself. This result holds by Proposition 5.1. For CCQ queries $Q(\bar{X}) \leftarrow L, M$ and $Q'(\bar{X}) \leftarrow L', M$ (note the same head vector and the same set M) and for a SCVM μ from Q to Q', we call μ an *M-identity SCVM* whenever μ maps each multiset variable of Q to itself. SCVM μ induces a mapping from L onto some subbag L'' of L'. (Here we treat the conjunctions L', L'' as bags of atoms.) We define $\mu(Q)$ as a CCQ query that is identical to Q except that the condition of $\mu(Q)$ is L''.

PROPOSITION 5.1. *Given a CCQ query Q, with reduced-condition query Q' that retains all the multiset variables of Q. Then there exists a GCM from Q_{ce} onto Q'_{ce} if and only if there exists an M-identity SCVM from Q onto Q'.* □

The intuition for the only-if part of the proof (in [6]) is that each GCM, ν, from Q_{ce} onto Q'_{ce} induces an automorphism from the condition of Q' *as part of the condition of* Q, to the condition of Q' (in Q'). One can use this fact to take an inverse of the mapping resulting from restricting the domain of ν to the range of ν, and by composing that inverse with ν to obtain the desired M-identity SCVM from Q onto Q'. Consider an illustration.

EXAMPLE 5.1. *Let queries Q and Q' be as follows.*

$Q(X) \leftarrow p(X, Y, W; i), p(X, W, Y), p(X, Y, Z), \{Y, i\}.$
$Q'(X) \leftarrow p(X, Y, W; i), p(X, W, Y), \{Y, i\}.$

Q' is a proper reduced-condition query for Q that preserves the multiset variables of Q. Q_{ce} and Q'_{ce} are as follows.

$Q_{ce}(X) \leftarrow p(X,Y,W;i), p(X,W,Y;j), p(X,Y,Z;k),$
$\{Y,i,j,k\}.$
$Q'_{ce}(X) \leftarrow p(X,Y,W;i), p(X,W,Y;j), \{Y,i,j\}.$

There exists a GCM ν from Q_{ce} onto Q'_{ce}: $\nu = \{ X \to X, Y \to W, Z \to Y, W \to Y, i \to j, j \to i, k \to j \}$. Observe that the mapping ν_1 resulting from restricting the domain of ν to the range of ν, that is to the set $\mathcal{X} = \{ X, Y, W, i, j \}$, is a bijection. Thus, there exists an inverse mapping, $\nu_1^{-1} = \{ X \to X, Y \to W, W \to Y, i \to j, j \to i \}$. Further, the mapping $\nu' = \nu_1^{-1} \circ \nu$ is well defined, because the range of ν is the domain of ν_1^{-1}. Finally, when the domain of ν' is restricted to the set $\mathcal{X}$, then the resulting mapping is an identity mapping by definition. Hence, we obtain that ν' is a GCM from Q_{ce} onto Q'_{ce}, and that the mapping resulting from restricting the domain of ν' to the set of variables of the query Q is an M-identity SCVM from Q onto Q'. □

We use the result of Proposition 5.1 to develop algorithm MINIMIZE-CCQ-QUERIES. The pseudocode is as follows.

Algorithm MINIMIZE-CCQ-QUERIES:
Input: CCQ query Q.
Output: CCQ query Q^{min} such that Q^{min} is a minimized version of Q by Definition 2.3.

1. Set Q^{min} to the regularized version of Q;
   // Note that $Q^{min} \in \mathcal{Q}_{min}(Q)$
2. While (there exists an M-identity SCVM μ from Q^{min} to itself such that $\mu(Q^{min})$ has fewer subgoals than Q^{min}) // Note that $\mu(Q^{min}) \in \mathcal{Q}_{min}(Q)$
   3. Set Q^{min} to $\mu(Q^{min})$;
4. Output Q^{min}.

Algorithm MINIMIZE-CCQ-QUERIES is a straightforward generalization to CCQ queries of the minimization algorithm applied by [2] to CCQ *set* queries. That is, our algorithm obtains recursively "shorter-condition" reduced-condition queries Q' for the input query Q, such that each $Q' \in \mathcal{Q}_{min}(Q)$. The algorithm terminates once no more M-identity SCVM can "shorten" any further the condition of Q'.

PROPOSITION 5.2. *Given a CCQ query Q, algorithm* MINIMIZE-CCQ-QUERIES *outputs a minimized version of Q.* □

The proof [6] of Proposition 5.2 is by showing that for each input CCQ query Q, the output of the algorithm MINIMIZE-CCQ-QUERIES satisfies Definition 2.3 with respect to Q.

In addition to being sound, algorithm MINIMIZE-CCQ-QUERIES is also complete, due to the following result:

THEOREM 5.1. *Given a CCQ query Q, the minimized version of Q exists and is unique up to an isomorphism M-identity SCVM.* □

Note that Theorem 5.1 reduces correctly to the special case of set queries, see Theorem 2.2 (1). (Recall that CCQ set queries have zero multiset variables.)

THEOREM 5.2. *Algorithm* MINIMIZE-CCQ-QUERIES *is sound and complete for CCQ queries.* □

The result of Theorem 5.2 is immediate from Proposition 5.2 and from Theorem 5.1.

The asymptotic worst-case time complexity of the algorithm MINIMIZE-CCQ-QUERIES is the same as that for the minimization algorithm of [2] (for CCQ set queries). Indeed, by definition of M-identity SCVMs and from the fact that CCQ set queries have zero multiset variables, finding a minimized version of a set query is at least as hard as finding a minimized version of any CCQ query. This fact is due to the absence, in case of set queries, of any "identity bindings" for multiset variables of the query, in an M-identity SCVM from a set query to itselt. As a result, we obtain the following.

PROPOSITION 5.3. *Finding a minimized version of a CCQ query is NP complete.* □

We now provide an intuition for the proof of Theorem 5.1. In the proof, we use the following result (established in [6]).

PROPOSITION 5.4. *Given CCQ queries Q_1 and Q_2 such that there exists a CVM μ_1 from Q_1* onto *Q_2, and another CVM μ_2 from Q_2* onto *Q_1. Then each of μ_1 and μ_2 is an isomorphism SCVM.* □

The idea of the proof of Theorem 5.1, see [6], is as follows. First, the existence of a minimized version of Q follows from Proposition 5.2. Second, suppose that there exist two distinct minimized versions of Q, Q_1 and Q_2, where each of Q_1 and Q_2 satisfies Definition 2.3 w.r.t. Q. The proof of Theorem 5.1 establishes that there exists an M-identity SCVM from Q_1 *onto* Q_2, and another from Q_2 *onto* Q_1. Then Proposition 5.4 is used to conclude that each of the two M-identity SCVMs is an isomorphism SCVM. Consider an illustration.

EXAMPLE 5.2. *Consider CCQ query Q and two reduced-condition queries for Q, Q_1 and Q_2.*

$Q(X) \leftarrow p(X,Y;i), p(Y,W), p(Y,T), \{Y,i\}.$
$Q_1(X) \leftarrow p(X,Y;i), p(Y,W), \{Y,i\}.$
$Q_2(X) \leftarrow p(X,Y;i), p(Y,T), \{Y,i\}.$

By Definition 2.3 and by Theorem 3.3, each of Q_1 and Q_2 is a minimized version of the query Q.

Consider mapping μ_1 from Q to Q_1: $\mu_1 = \{ X \to X, Y \to Y, i \to i, W \to W, T \to W \}$. This mapping is an M-identity SCVM from Q onto Q_1. When restricted to the domain that is the set of terms of the query Q_2, call this mapping ν_1, mapping μ_1 furnishes an isomorphism M-identity SCVM from Q_2 onto Q_1. The mapping ν_1^{-1} is an isomorphism M-identity SCVM from Q_1 onto Q_2. □

We now contrast these results with our Theorem 4.1. By the results of this current section, a SCVM always exists from an arbitrary CCQ query into its minimized version. This holds even for implicit-wave queries, intuitively because a minimized version Q^{min} of a query Q is a reduced-condition query for Q. Hence, in some sense we know the *structure* of Q^{min}. In contrast, for two general CCQ queries Q and Q' such that $Q \equiv_C Q'$, all we know is that on all databases D, the bags $Res_C(Q,D)$ and $Res_C(Q',D)$ are identical. In general, no information is available about the relationship between the structures of Q and Q'. Thus, Theorem 4.1 does not necessarily hold for the case of implicit-wave queries.

6. CCQ-QUERY EQUIVALENCE

In this section we study those conditions of combined-semantics equivalence of CCQ queries that are immediate

from the results of Sections 3–4. Our focus is on reformulating these conditions using minimized query versions. The reformulations tie our equivalence conditions together with the results of [2, 5]. Our necessary and sufficient query-equivalence condition, Theorem 6.3, applies to the class of all explicit-wave CCQ queries. We show that this class encompasses strictly more queries than (i) all CCQ set, bag, and bag-set queries, and than (ii) all CCQ queries for which [8] provides both sufficient and necessary equivalence conditions.[9]

First, Theorem 3.3 (Section 3) gives us a sufficient condition for combined-semantics equivalence of CCQ queries:

THEOREM 6.1. *Given CCQ queries Q_1 and Q_2: If there exists a CVM from Q_1 to Q_2, and another from Q_2 to Q_1, then we have $Q_1 \equiv_C Q_2$.* □

We reformulate this theorem using the results of Section 5:

THEOREM 6.2. *Given CCQ queries Q_1 and Q_2, with respective minimized versions Q_1^{min} and Q_2^{min}. If there exists an isomorphism SCVM from Q_1^{min} to Q_2^{min}, and another from Q_2^{min} to Q_1^{min}, then we have $Q_1 \equiv_C Q_2$.* □

It turns out that the sufficient query-equivalence conditions of Theorems 6.1 and 6.2 have the same power. (The proof of Theorem 6.2 is immediate from Theorem 6.1 and Proposition 6.1.)

PROPOSITION 6.1. *Given CCQ queries Q_1 and Q_2, with respective minimized versions Q_1^{min} and Q_2^{min}. Then:*

- *There exists a CVM from Q_1 to Q_2, and another from Q_2 to Q_1, if and only if*
- *There exists an isomorphism SCVM from Q_1^{min} to Q_2^{min}, and another from Q_2^{min} to Q_1^{min}.* □

See [6] for the proof of Proposition 6.1. The only-if part of the proof is based on Proposition 5.4.

Neither Theorem 6.1 nor Theorem 6.2 gives us a necessary condition for combined-semantics equivalence of two CCQ queries. (We have Example 4.1 as a counterexample. Observe that both queries in Example 4.1 are represented by their minimized versions.)

At the same time, we use Theorems 4.1 and 6.2, as well as Proposition 6.1, to formulate a sufficient and necessary condition for equivalence of explicit-wave CCQ queries.

THEOREM 6.3. *Given explicit-wave CCQ queries Q_1 and Q_2, with respective minimized versions Q_1^{min} and Q_2^{min}. Then $Q_1 \equiv_C Q_2$ if and only if there exists an isomorphism SCVM from Q_1^{min} to Q_2^{min}, and another from Q_2^{min} to Q_1^{min}.* □

Another sufficient and necessary condition for equivalence of explicit-wave CCQ queries, in terms of CVMs, can be obtained by using only Theorems 4.1 and 6.1; see [6].

We now show that Theorem 6.3 generalizes Theorem 2.2 (2), due to [2], as well as Theorem 2.3, due to [5]. To do this, we show that all CCQ set, bag, and bag-set queries are explicit-wave queries, and then consider minimization of CCQ bag and bag-set queries.

By Condition (1) of Definition 4.1, we have that all set and bag-set CCQ queries are explicit-wave queries. Besides that condition, one can formulate a number of easy syntactic tests, each of which is a sufficient condition for a CCQ query to be an explicit-wave query. (E.g., it is immediate from Definition 4.1 that a CCQ query without self-joins is an explicit-wave query.) One sufficient condition is that a CCQ query Q is an explicit-wave query whenever each copy-sensitive subgoal of Q has no set variables. (In this case, it is easy to see that Condition (2) of Definition 4.1 is always satisfied; see [6].) By this condition, all CCQ bag queries are explicit-wave queries. Other sufficient conditions could generalize the case of the explicit-wave CCQ query Q' of Example 4.1 (note that while being an explicit-wave query, this query does not satisfy any of the above sufficient conditions for a query to be explicit-wave), and so on. As a result, we have the following:

PROPOSITION 6.2. *The set of all CCQ set, bag, and bag-set queries is a proper subset of the set of all explicit-wave CCQ queries.* □

One can argue that CCQ queries that have set variables in copy-sensitive subgoals, such as the implicit-wave query Q of Example 4.1, would not tend to be popular – and may not even be expressible – in practical applications. Hence, we posit that implicit-wave queries may be unlikely to arise in practice.

Now that we know that all CCQ set queries are explicit-wave queries, it is easy to see that Theorem 6.3 generalizes properly Theorem 2.2 (2). (Theorem 2.2 (2) does not generalize to the case of all CCQ queries because not all CCQ queries are explicit wave, see discussion of Example 4.1 earlier in this section.) For instance, by Theorem 6.3 we have that for the queries of Example 3.2, $Q \equiv_C Q'$ holds. The reason is, both Q and Q' in that example are explicit-wave queries and, in addition, Q' is the minimized version of Q.

Observe that Theorem 6.3 is not a trivial generalization of Theorem 2.2 (2). Indeed, the two explicit-wave CCQ queries of Example 3.1 are isomorphic but, by Theorem 6.3, are not combined-semantics equivalent. (Both queries of Example 3.1 are represented by their minimized versions.)

Finally, we consider minimization of CCQ bag and bag-set queries. Recall that each subgoal of a CCQ bag query has a copy variable. Hence, by Theorem 3.1, each CCQ bag query is its unique minimized version. We conclude that the result (due to [5]) of Theorem 2.3 (1) is a special case of Theorem 6.3.

In case of CCQ bag-set queries, the only terms that can appear in such a query are multiset noncopy variables, head variables, and constants. We have from the results of Section 5 that for each CCQ query, there exists an M-identity SCVM from the regularized version of the query to its minimized version. Now the regularized version Q_r of a CCQ bag-set query Q is the canonical representation [5] of Q, that is, the result of dropping all duplicate subgoals from the condition of Q. By definition of M-identity SCVM, we have for all CCQ bag-set queries Q that for each subgoal, s, of Q_r, each M-identity SCVM maps s into itself. It follows that each M-identity SCVM maps Q_r *onto* itself. Thus, for each CCQ bag-set query Q, its canonical representation Q_r is its unique minimized version. Hence Theorem 2.3 (2) is a special case of Theorem 6.3.

[9] In fact, we already showed (ii) in Section 4.

7. RELATED WORK

In their classic paper [2], Chandra and Merlin presented an NP-complete containment test for CQ queries under set semantics. This sound and complete test has been used in optimization, via minimization, of CQ set-semantics queries, as well as in developing algorithms for rewriting queries (both equivalently and nonequivalently) using views. We are not aware of past work that studies minimization of queries beyond the language of CQ set-semantics queries. In this current paper we extend the results of [2] to general CQ combined-semantics queries, and show the limitations of each extension. We show that the minimization approach of [2] can be extended to general CQ queries without limitations. Remarkably, minimizing arbitrary CQ queries is at most as hard as minimizing CQ set-semantics queries.

Equivalence tests for CQ bag and bag-set queries were formulated by Chaudhuri and Vardi in [5]; correctness of the tests follows from the results of [8]. Our equivalence and minimization results for CQ combined-semantics queries reduce correctly to the special cases of CQ bag and bag-set queries, as given in [5]. Further, this current paper provides a nontrivial generalization and the first known proof of the well-known sufficient containment condition for CQ bag queries, as outlined in [5].

Definitive results on containment between CQ queries under bag and bag-set semantics have not been obtained so far. Please see Jayram, Kolaitis, and Vee [13] for original undecidability results on containment of CQ queries with inequalities under bag semantics. The authors point out that it is not known whether the problem of bag containment for *CQ* queries is even decidable. For the case of *bag-set* semantics, sufficient conditions for containment of two CQ queries can be expressed via containment of (the suitable) aggregate queries with aggregate function `count(*)`. The latter containment problem can be solved using the methods proposed in [9]. Please see [5, 1] for other results on bag and bag-set containment of CQ queries. The general problems of containment for CQ bag and bag-set queries remain open.

In her papers [7, 8], Cohen provided an elegant and powerful formalism for treating queries evaluated under each of set, bag, and bag-set semantics uniformly as special cases of the more general combined semantics. The papers also contain a general sufficient condition for combined-semantics equivalence of CQ queries with disjunction, negation, and arithmetic comparisons, as well as necessary and sufficient equivalence conditions for special cases. (When we restrict the language of the queries in question to the language of CQ queries, it turns out that all the necessary and sufficient query-equivalence conditions of [8] hold for queries belonging collectively to a proper subclass of the class of explicit-wave CQ queries, which (class) we introduce in this current paper.) The proof in [8] of its general sufficient condition for equivalence of queries is in terms of containment between the queries under combined semantics. That (implicit) sufficient query-containment condition is proved in [8] for the case where the two queries have the same number of multiset variables. In this current paper we provide proper generalizations of all the results of [8], including its implicit sufficient condition for query containment, provided that the results of [8] are applied to CQ queries only.

In [10], DeHaan presented, among other results, a sufficient and necessary condition for equivalence of all CQ combined-semantics queries. Example 4.1 in this current paper provides a counterexample to the necessary part of the condition of [10]. The reason is, that necessary condition of [10] requires the existence of a certain type of mapping between two equivalent queries, and such a mapping does not exist from the query Q' to the query Q in our Example 4.1.

A discussion of query equivalence and containment for query languages that properly contain the language of CQ queries is beyond the scope of this paper. The interested reader is referred to [8], which contains an excellent overview of the literature in that direction.

8. REFERENCES

[1] F. N. Afrati, M. Damigos, and M. Gergatsoulis. Query containment under bag and bag-set semantics. *Information Processing Letters*, 110(10):360–369, 2010.

[2] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *ACM STOC*, 1977.

[3] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1):65–74, 1997.

[4] S. Chaudhuri, U. Dayal, and V. R. Narasayya. An overview of business intelligence technology. *Commun. ACM*, 54(8):88–98, 2011.

[5] S. Chaudhuri and M. Y. Vardi. Optimization of *real* conjunctive queries (extended abstract). In *PODS*, 1993.

[6] R. Chirkova. Equivalence and minimization of conjunctive queries under combined semantics. Technical Report TR-2010-24, NCSU, 2010. Available from `http://www.csc.ncsu.edu/research/tech/reports.php`.

[7] S. Cohen. Equivalence of queries combining set and bag-set semantics. In *PODS*, pages 70–79, 2006.

[8] S. Cohen. Equivalence of queries that are sensitive to multiplicities. *The VLDB Journal*, 18:765–785, 2009.

[9] S. Cohen, W. Nutt, and Y. Sagiv. Containment of aggregate queries. In *ICDT*, pages 111–125, 2003.

[10] D. DeHaan. Equivalence of nested queries with mixed semantics. In *PODS*, pages 207–216, 2009.

[11] A. Gupta and I. S. Mumick, editors. *Materialized Views: Techniques, Implementations, and Applications*. The MIT Press, 1999.

[12] Y. Ioannidis and R. Ramakrishnan. Containment of conjunctive queries: beyond relations as sets. *ACM TODS*, 20(3):288–324, 1995.

[13] T. Jayram, P. Kolaitis, and E. Vee. The containment problem for *real* conjunctive queries with inequalities. In *PODS*, pages 80–89, 2006.

[14] W. Lehner. Query processing in data warehouses. In *Encyclopedia of Database Systems*, pages 2297–2301. Springer, 2009.

[15] N. Pendse and R. Creeth. The OLAP report. *Business Intelligence*, 1995. The 2008 update available at `http://www.bi-verdict.com/fileadmin/FreeAnalyses/fasmi.htm`.

[16] A. Shukla, P. Deshpande, and J. F. Naughton. Materialized view selection for multi-cube data models. In *EDBT*, 2000.

[17] M. Zaharioudakis, R. Cochrane, G. Lapis, H. Pirahesh, and M. Urata. Answering complex SQL queries using automatic summary tables. In *SIGMOD*, pages 105–116, 2000.