# Obtaining More Answers From Information Integration Systems

Gösta Grahne and Victoria Kiricenko
Department of Computer Science, Concordia University
Montreal, Quebec, Canada H3G 1M8
{grahne,kiricen}@cs.concordia.ca

**Abstract**

The current generation of rewriting-algorithms in source-centric (local-as-view) information integration systems all produce a reformulated query that retrieves what has been thought of as "the best obtainable" answer, given the circumstances that the source-centric approach introduces incomplete information into the virtual global relations. This "best obtainable" answer does not however allow partial information. We define the semantics of partial facts, and provide two methods for computing partial answers. The first method is tableau-based and is a generalization of the "inverse-rules" approach. The second method is a generalization of the rewriting approach, and is based on partial containment mappings introduced in the paper.

## 1 Introduction

Information Integration systems aim to provide a uniform query interface to multiple heterogeneous sources. One of the ways to view these systems is to postulate a *global schema* (called a world view) that provides a unifying data model for all the information sources (see e. g. [Len02]). A query processor is in charge of reformulating queries written in terms of this global schema to queries on the appropriate sources and assembling the answers into a global answer. Each source is modeled as a *materialized view* defined in terms of the global relations, which are virtual.

To illustrate the problem with current information integration methodologies let us consider a simple example. Suppose the global schema contains two relations:

*Prof(Pname, Email, Office, Area)*, that is professor's name, email, office number, and research area.

*Dept(Pname, Dname)*, that is professor's name and department.

The available sources are $\{S_1, S_2, S_3, S_4\}$. These sources have the following definitions:

$S_1(Pname, Email) \leftarrow Prof(Pname, Email, Office, Area)$

$S_2(Pname, Office) \leftarrow Prof(Pname, Email, Office, Area)$

$S_3(Pname, Area) \leftarrow Prof(Pname, Email, Office, Area)$

$S_4(Pname, Dname) \leftarrow Dept(Pname, Dname)$

Suppose the user issues the query

$Q(Pname, Email, Office, Area) \leftarrow Prof(Pname, Email, Office, Area), Dept(Pname, compsci)$

That is, the user is interested in obtaining all information available about professors in the *compsci* department. Since the information integration system does not have a way to get tuples for the subgoal *Prof* it would produce an empty rewriting and, thus, an empty answer for the user.

From the point of view of the user, it would be much more useful to get at least partial information about professors in the *compsci* department. This is feasible, the query could be rewritten as union of the following unsafe conjunctive queries.

$$Q_1(Pname, Email, X, Y) \leftarrow S_1(Pname, Email), S_4(Pname, compsci)$$

$$Q_2(Pname, X, Office, Y) \leftarrow S_2(Pname, Office), S_4(Pname, compsci)$$

$$Q_3(Pname, X, Y, Area) \leftarrow S_3(Pname, Area), S_4(Pname, compsci)$$

The unrestricted variables $X$ and $Y$ represent unknown values. The answer can then be presented to the user as a table with some values missing, for example as the table below. (The contents of the table will, obviously, depend on the data provided by the sources.)

| Pname | Email | Office | Area |
|---|---|---|---|
| Murphy | murphy@cs.toronto.edu | $\perp$ | $\perp$ |
| Murphy | $\perp$ | SF322 | $\perp$ |
| Smith | $\perp$ | $\perp$ | DB |
| Jones | jones@cs.concordia.ca | $\perp$ | $\perp$ |
| Brown | $\perp$ | $\perp$ | AI |

This aspect of information integration has not been studied in the current literature despite it's importance especially in the case of the world wide web, where we can expect lots of partially overlapping sources. In this paper we define answers containing null values, and we give two algorithms for computing them. One algorithm is a generalization of the "inverse-rules" approach [DG97], and involves explicitly computing a syntactic representation of the set of global databases implicitly defined by the sources. The other algorithm is a generalization of the rewriting technique (see e.g. [LMSS95], [Ull97]). This algorithm reformulates a query in terms of the source relations, and, thus, avoids inverting the entire source collection.

## 2 Conjunctive Queries and Projection-Containment

Let **rel** be a countably infinite set $\{R, S, \ldots, R_1, S_1, R_2, S_2, \ldots\}$ of *relation names*, let **dom** be a countably infinite set of *constants*, and let **var** be a countably infinite set of *variables*. Constants will be denoted by lower case letters and variables by upper case letters.

Associated with each relation name $R$ is a positive integer $arity(R)$, which is the arity of $R$. A *fact* over $R$ is an expression of the form $R(a_1, \ldots, a_k)$, where $k = arity(R)$, and each $a_i$ is in **dom**.

Let $\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$ be a set of relation names. A finite set of relation names will sometimes also be called a *schema*. A *database* $d$ over $\mathbf{R}$ is a finite set of facts, each fact being over some $R_i \in \mathbf{R}$.

An *atom* over a relation name $R$ is an expression of the form $R(e_1, \ldots, e_k)$, where $k = arity(R)$, and each $e_i$ is either a constant in **dom** or a variable in **var**.

A *conjunctive query* $\varphi$ (over $\mathbf{R}$) has the form

$$head(\varphi) \leftarrow body(\varphi),$$

where $body(\varphi)$ is a set of atoms $b_1, b_2, \ldots, b_n$, each over a relation name in $\mathbf{R}$, and $head(\varphi)$ is an atom over an answer relation name not in $\mathbf{R}$. We assume that all variables occurring in $head(\varphi)$ also occur in $body(\varphi)$, i. e. that the query $\varphi$ is *safe*. The variables occuring in $head(\varphi)$ are the *distinguished* variables of the query, and all the others are *existential* variables.

A conjunctive query $\varphi$ can be applied to a database $d$ over $\mathbf{R}$, resulting in a set of facts

$$\varphi(d) = \{\sigma(head(\varphi)) : \sigma(body(\varphi)) \subseteq d \text{ for some valuation } \sigma\}.$$

A *valuation* $\sigma$, is formally a finite partial mapping from $\mathbf{var} \cup \mathbf{dom}$ to $\mathbf{dom}$ that is the identity on $\mathbf{dom}$. Valuations, like $X_i \mapsto a_i$, for $i \in [1, p]$, will usually be given in the form $\{X_1/a_1, \ldots, X_p/a_p\}$. The identity on constants is omitted in this notation.

The notion of query containment enables comparisons between different reformulations of queries. We can broaden query containment as follows.

Let $\varphi_1$ and $\varphi_2$ be conjunctive queries. A query $\varphi_1$ is said to be *p-contained* in $\varphi_2$, denoted $\varphi_1 \subseteq_p \varphi_2$, if and only if there exists a conjunctive query $\phi$, where $\varphi_1$ is equivalent to $\pi_L(\phi)$ ($\pi$ is relational projection), for some list $L$ of columns in $head(\phi)$ taken in the original order, such that for all databases $d$, $\phi(d) \subseteq \varphi_2(d)$. Note that p-containment is a generalization of query containment since $L$ can be the list of all columns in $\varphi_1$.

**Testing p-containment of conjunctive queries**

The classical notion of a containment mapping can be generalized to define p-containment mappings. A *p-containment mapping* from a conjunctive query $\varphi_2$ to a conjunctive query $\varphi_1$ is a mapping $\mu$, from variables of $\varphi_2$ to variables and constants of $\varphi_1$, such that

1. $\mu(body(\varphi_2)) \subseteq body(\varphi_1)$, and
2. for every variable $X$ in $head(\varphi_1)$ there is a variable $Y$ in $head(\varphi_2)$, such that $\mu(Y) = X$.

Consider the following example

$$\varphi_1 = Q_1(X) \leftarrow R(X, Y), S(Y, Y), T(Y, Z)$$

$$\varphi_2 = Q_2(A, B) \leftarrow R(A, B), S(B, C)$$

There is a p-containment mapping $\mu = \{A/X, B/Y, C/Y\}$ from $\varphi_2$ to $\varphi_1$.

We can now use p-containment mappings to test p-containment of conjunctive queries.

**Theorem 1** *A query $\varphi_1$ is p-contained in a query $\varphi_2$ if and only if there is a p-containment mapping from $\varphi_2$ to $\varphi_1$.*

**Proof.**

Let $\mu$ be a p-containment mapping from $\varphi_2$ to $\varphi_1$, and let $d$ be an arbitrary database. A fact $t_1$ in $\varphi_1(d)$ is generated by some valuation $\sigma$. Then $\sigma \circ \mu$ is a valuation that generates the corresponding fact $t_2$ in $\varphi_2(d)$. To see that this is indeed so, let $b \in body(\varphi_2)$. Then $\sigma \circ \mu(b) = \sigma(c) \in d$, for some $c \in body(\varphi_1)$. Therefore, $\sigma \circ \mu(b) \in d$. From requirement 2 of a p-containment mapping it follows that $\sigma \circ \mu(head(\varphi_2)) = \pi_L(\sigma \circ \mu(head(\varphi_1)))$, where L is a list of variables in $head(\varphi_2)$ in the original order. Thus, $\varphi_1 \subseteq_p \varphi_2$.

Let $\varphi_1 \subseteq_p \varphi_2$. Let $d$ be the canonical database that is the "frozen" $body(\varphi_1)$. By the definition of p-containment there exist a conjunctive query $\psi$, such that $\psi(d) \subseteq \varphi_2(d)$ and $\varphi_1 = \pi_L(\psi)$ for some ordered list $L$ of columns in $head(\psi)$. Obviously, $\varphi_1(d)$ contains a fact $t_1$, which is the "frozen" $head(\varphi_1)$. Since $\varphi_1 = \pi_L(\psi)$ there must be a fact $t_2$ in $\psi(d)$, such that $\pi_L(t_2) = t_1$. Since $\psi(d) \subseteq \varphi_2(d)$, we have $t_2 \in \varphi_2(d)$.

Let $\sigma$ be a valuation that generates the fact $t_2$ in $\varphi_2(d)$. Let $\rho$ be the the "freezing" mapping, which also is a valuation that generates the fact $t_1$ in $\varphi_1(d)$. Then $\rho^{-1} \circ \sigma$ is a p-containment mapping from $\varphi_2$to $\varphi_1$.

To see that this is indeed so note two things. First, that each subgoal $b \in body(\varphi_2)$ is mapped by $\sigma$ to some fact in $d$, which is a frozen version of some subgoal $c \in body(\varphi_1)$, so $\rho^{-1} \circ \sigma$ maps $b$ to the unfrozen fact, that is to $c$ itself.

Second, note that those variables in $head(\varphi_2)$ that are also in $head(\varphi_1)$ are mapped by $\sigma$ to constants in the fact $t_1$, which is the frozen $head(\varphi_1)$, so that all of the head variables in $\varphi_1$ are covered. Thus $\rho^{-1} \circ \sigma$ maps corresponding variables in $head(\varphi_2)$ to the unfrozen $head(\varphi_1)$. Thus, $\rho^{-1} \circ \sigma$ is a p-containment mapping from $\varphi_2$ to $\varphi_1$. ■

# 3 Source Collections and Tableaux

Let **loc** be a countably infinite set $\{V, V_1, V_2, \ldots\}$ of *local relation names*. The local relation names have arities, and atoms over local relation names are defined in the same way as atoms over relation names in **rel**. To distinguish between relations (relation names) in **rel** and in **loc** we will henceforth call the former *global* relations (relation names).

A *source* $S$ is a pair $(\varphi, v)$, where $\varphi$ is a conjunctive query and $v$ is a finite set of facts over $head(\varphi)$. A *source collection* $\mathcal{S}$ is a finite set of sources. The *(global) schema* of $\mathcal{S}$, denoted $sch(\mathcal{S})$ is the set consisting of all the global relation names occurring in the bodies of the defining conjunctive queries of the sources in $\mathcal{S}$. The *description* of $\mathcal{S}$, denoted $desc(\mathcal{S})$ is obtained from $\mathcal{S}$ by dropping the extension from every pair in $\mathcal{S}$. In other words, a source collection $\mathcal{S}$ has *two* "schemas," $sch(\mathcal{S})$ which is the "global world view" and $desc(\mathcal{S})$, which describes the defining views. The *extension* of a source collection $\mathcal{S}$, denoted $ext(\mathcal{S})$, is the union of all facts in sources in $\mathcal{S}$.

A source collection $\mathcal{S}$ defines a set of possible databases, denoted $poss(\mathcal{S})$, as follows:

$$poss(\mathcal{S}) = \{d \text{ over } sch(\mathcal{S}) \ : \ v_i \subseteq \varphi_i(d) \text{ for all sources } S_i = (\varphi_i, v_i) \text{ in } \mathcal{S}\}.$$

Note that $poss(\mathcal{S})$ is infinite. We will now consider the problem of finitely representing an infinite set of databases. For this we invoke the venerable tableau.

### Sets of databases and tableaux

In order to reason about tableaux we have to define a few concepts that are applicable to sets of global databases represented by tableaux.

Let $\mathcal{X}$ and $\mathcal{Y}$ be two enumerable sets of global databases over $\mathbf{R}$. We say that $\mathcal{X}$ and $\mathcal{Y}$ are *coinitial* if they have the same $\subseteq$-minimal elements. Coinitiality is denoted $\mathcal{X} \approx \mathcal{Y}$.

Let $\Omega$ be the set of all queries expressible in a query language that we by abuse of notation also call $\Omega$. Then $\mathcal{X}$ and $\mathcal{Y}$ are said to be $\Omega$-*equivalent*, denoted $\mathcal{X} \equiv_\Omega \mathcal{Y}$, if for all queries $Q \in \Omega$ we have

$$\bigcap_{d \in \mathcal{X}} Q(d) = \bigcap_{d \in \mathcal{Y}} Q(d).$$

The intuition behind $\Omega$-equivalence is that $\mathcal{X}$ and $\mathcal{Y}$ are indistinguishable as far as the certain information derivable by queries in $\Omega$ are conserned. Thus, if a user can only pose queries in $\Omega$, he cannot distinguish between $\mathcal{X}$ and $\mathcal{Y}$.

The following lemma is proved in the seminal paper [IL84].

**Lemma 1** *Let $\Omega$ be a monotone query language. If $\mathcal{X} \approx \mathcal{Y}$, then $\mathcal{X} \equiv_\Omega \mathcal{Y}$.* ■

Of particular interest to us is of course choosing $\Omega$ to be the set of all unions of conjunctive queries, which, it goes without saying, is a monotone query language.

We now define tableaux [Men84], which are intended to concisely and finitely represent a large or infinite set of possible instances.

Let $\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$ be a set of relation names. A *tableau* $T$ over $\mathbf{R}$ is a finite set of atoms over the $R_i$'s. Note that the same variable might appear in several atoms in $T$.

A tableau $T$ over schema $\mathbf{R}$ represents a set of databases over $\mathbf{R}$. This set is denoted $rep(T)$, and it is defined by

$$rep(T) = \{d \ : \text{ there is a valuation } \sigma \text{ such that } \sigma(T) \subseteq d\}.$$

The definition says that a database $d$ is represented by a tableau $T$, if there is a valuation $\sigma$ such that when all variables in $T$ are replaced by their image under $\sigma$, the set of facts thus obtained is a subset of $d$.

In order to compare tableaux, we need the concept of substitution. A *substitution* is a valuation, except that we allow variables to be mapped into variables, not only constants. Thus, a substitution $\theta$

is a function from (a subset of) $\mathbf{dom} \cup \mathbf{var}$ to $\mathbf{dom} \cup \mathbf{var}$, keeping in mind that constants have to be mapped to themselves.

Given two atoms $t$ and $u$ over the same relation name, we say that $t$ is *subsumed* by $u$, denoted $t \leq_{sub} u$, if there is a substitution $\theta$, such that $\theta(u) = t$. A tableau $T$ is *subsumed* by a tableau $U$ denoted $T \leq_{sub} U$, if for every atom $t \in T$ there is an atom $u \in U$, such that $t \leq_{sub} u$, A tableau $T$ is *subsumtion equivalent* to a tableau $U$ denoted $T \equiv_{sub} U$, if $T \leq_{sub} U$ and $U \leq_{sub} T$,

The following little lemma gives a semantic characterization of subsumption equivalence.

**Lemma 2** *Let $\Omega$ be a set of all projection queries. Then $rep(T) \equiv_\Omega rep(U)$ if and only if $T \equiv_{sub} U$.* ■

In other words, if $T$ and $U$ are subsumption equivalent, then for any subset of the columns, $T$ and $U$ contain the same facts over these columns. Compared to letting $\Omega$ be the set of all (unions of) conjunctive queries, subsumption equivalence cannot account for repeated variables. Such repeated variables might allow the user to infer more certain information using subsequent joins. We will return to this point in the conculding section.

### Representing $poss(\mathcal{S})$ by a tableau

Now the set $poss(\mathcal{S})$ can be conveniently represented by a tableau over schema $sch(\mathcal{S})$, denoted $T(\mathcal{S})$, such that $rep(T) = poss(\mathcal{S})$. To construct $T$ we shall follow the approach in [GM99]. We define a function, which we by abuse of notation also denote $T$, from sources with defining view $\varphi$, where the body of $\varphi$ consists of atoms over relation names in $\mathbf{R}$, to tableau over $\mathbf{R}$. We also need an auxiliary function *refresh*, that, when applied to a set of atoms, replaces all variables with fresh ones. Given a source $S = (\varphi, v)$, we set

$$T(S) = \bigcup_{u \in v} \{refresh(\sigma(body(\varphi))) \ : \ \sigma(head(\varphi)) = u \text{ for some valuation } \sigma\}.$$

For example, if $S = (V(X,Z) \rightarrow R(X,Y), S(Y,Z), \{V(a,b), V(c,d)\})$, then $T(S) = \{R(a,Y_1), S(Y_1,b), R(c,Y_2), S(Y_2,d)\}$, where $Y_1$ and $Y_2$ are fresh variables. When there are several sources in $\mathcal{S}$ we set

$$T(\mathcal{S}) = \bigcup_{S \in \mathcal{S}} T(S).$$

The tableau constructed by the function $T$ has the following desirable property.

**Theorem 2** *$rep(T(\mathcal{S})) = poss(\mathcal{S})$.*

**Proof.** Let $d \in rep(T(\mathcal{S}))$. To prove that $d \in poss(\mathcal{S})$ we need to show that for all sources $S_i = (\varphi_i, v_i)$ in $\mathcal{S}$, we have $v_i \subseteq \varphi_i(d)$. Since $d \in rep(T(\mathcal{S}))$ there is a valuation $\sigma$ such that $\sigma(T(\mathcal{S})) \subseteq d$. Let $S_i = (\varphi_i, v_i)$ be an arbitrary source in $\mathcal{S}$, and let $t$ be an arbitrary fact in $v_i$. Then there must be a substitution $\theta$, such that $t = \theta(head(\varphi_i))$ and all facts in $\theta(body(\varphi_i))$ are in $T(\mathcal{S})$. It follows that $\theta(\sigma(body(\varphi_i))) \subseteq d$ and, consequently, $\theta(\sigma(head(\varphi_i))) \in d$. Since $\theta(\sigma(head(\varphi_i))) = t$, we have $v_i \subseteq \varphi_i(d)$ as desired.

For inclusion in the other direction, let $d \in poss(\mathcal{S})$. From construction of $T(\mathcal{S})$ it immediately follows that there is a valuation $\sigma$ such that $\sigma(T(\mathcal{S})) \subseteq d$ and, thus, that $d \in rep(T(\mathcal{S}))$. ■

# 4 Querying Source Collections

Let $\mathcal{S}$ be source collection, and $\varphi$ a conjunctive query, such that the body of $\varphi$ consists of atoms over relation names in $sch(\mathcal{S})$. Now $\varphi$ applied to $\mathcal{S}$ defines the *exact answer*:

$$\varphi(\mathcal{S}) = \{\varphi(d) \ : \ d \in poss(\mathcal{S})\}.$$

The definition essentially says that since the source collection corresponds to a set of databases, the answer should also correspond to a set, obtained by evaluating the query pointwise.

**Computing the exact answer from the tableau**

Now since we are able to construct a database template $T$ representing all databases in $poss(\mathcal{S})$ it is natural to extend the standard query evaluation mechanism to operate on database templates.

Let $T$ be a tableau over $\mathbf{R}$. Given a conjunctive query $\varphi$ over $\mathbf{R}$, our evaluation $\widehat{\varphi}$ (which is basically the "naive evaluation" of [IL84]) is as follows. Recall that a *substitution* is a valuation, except that variables can be mapped into variables, not only constants. Then

$$\widehat{\varphi}(T) = \{\theta(head(\varphi)) : \theta(body(\varphi)) \subseteq T \text{ for some substitution } \theta\}.$$

For example, let $\varphi = Q(X,Y,Z) \leftarrow R(X,Y), S(Y,Z)$ and $T = \{R(a,b), R(d,X), S(b,c), S(X,e), S(Y,f)\}$. Then $\widehat{\varphi}(T) = \{Q(a,b,c), Q(d,X,e)\}$.

Clearly, if our definition of $\widehat{\varphi}$ is semantically meaningful, then we should expect that it approximates the information given by $\varphi(\mathcal{S})$ in some natural sense. Indeed, our extended semantics has the following property:

**Theorem 3** $rep(\widehat{\varphi}(T)) \approx \varphi(rep(T))$.

**Proof.** Let $d$ be a $\subseteq$-minimal element in $rep(\widehat{\varphi}(T))$. Then there exist a valuation $\sigma$ such that $\sigma(\widehat{\varphi}(T)) = d$. Let $t$ be an arbitrary fact in $d$. Then there is a fact $u \in \widehat{\varphi}(T)$ such that $\sigma(u) = t$, and there is a substitution $\theta$ such that $\theta(body(\varphi)) \in T$ and $u = \theta(head(\varphi))$.

Let $\sigma'$ be an extension of $\sigma$ that maps every variable that is in $T$ but not in $\widehat{\varphi}(T)$ to a distinct new constant. Since $\theta(body(\varphi)) \subseteq T$, we have $\sigma'\theta(body(\varphi)) \subseteq \sigma'(T)$. It now follows that $t = \sigma'(\theta(head(\varphi))) \in \varphi(\sigma'(T))$. Note that $\sigma'(T)$ is a $\subseteq$-minimal element in $rep(T)$. From the monotonicity of $\varphi$ it follows that $\varphi(\sigma'(T))$ is a $\subseteq$-minimal element in $\varphi(rep(T))$. We have established that $d \subseteq \varphi(\sigma'(T))$

That concludes the proof that any $\subseteq$-minimal element $d$ in $rep(\widehat{\varphi}(T))$ is in $\varphi(rep(T))$.

For inclusion in the other direction let $d$ be a $\subseteq$-minimal element in $\varphi(rep(T))$. Then there is a valuation $\sigma$, such that $d = \varphi(\sigma(T))$. Let $t$ be an arbitrary tuple in $d$. Then there is a valuation $\rho$, such that $t = \rho(head(\varphi))$ and all facts in $\rho(body(\varphi))$ are in $\sigma(T)$. Now we have two cases to consider.

*Case 1:* The valuation $\sigma$ is one-to-one. Then there is an inverse $\sigma^{-1}$, and hence $\sigma^{-1}(\rho(body(\varphi))) \subseteq \sigma^{-1}(\sigma(T)) = T$, and consequently $\sigma^{-1}(t) = \sigma^{-1}(\rho(head(\varphi)))$ is in $\widehat{\varphi}(T)$. Since $\sigma(\widehat{\varphi}(T)) \in rep(\widehat{\varphi}(T))$, it follows that $t \in \sigma(\widehat{\varphi}(T)) \in rep(\widehat{\varphi}(T))$. Likewise, if $t'$ is any other tuple in $d = \varphi(\sigma(T))$, it is generated by some valuation $\rho'$, and we have $\sigma^{-1}(t') = \sigma^{-1}(\rho'(head(\varphi))) \in \widehat{\varphi}(T)$. Therefore $d \subseteq \sigma(\widehat{\varphi}(T))$.

*Case 2:* There is (at least one) pair of distinct variables $X$ and $Y$ in $T$, such that $\sigma(X) = \sigma(Y)$. If $\sigma(X) = \rho(U)$, and $\sigma(Y) = \rho(W)$, for $U \neq W$, then the valuation $\omega$, that is like $\sigma^{-1} \circ \rho$, except $\omega(U) = X$, and $\omega(V) = Y$, gives us $\omega(body(\varphi)) \subseteq T$, and $\sigma^{-1}(t) = \omega(head(\varphi)) \in \widehat{\varphi}(T)$. Consequently $t = \sigma(\sigma^{-1}(t)) \in \sigma(\widehat{\varphi}(T))$.

Suppose then that $\sigma(X) = \sigma(Y) = \rho(W)$, and that there are (at least) two occurrences of $W$ in $body(\varphi)$. Consider now the valuation $\sigma'$, that is exactly like $\sigma$, except it maps $Y$ to a fresh constant, say $a$. Clearly $t \notin \varphi(\sigma'(T))$, and any fact in $\varphi(\sigma'(T))$ is also in $\varphi(\sigma(T))$ (because there is an embedding of $\sigma'(T)$ into $\sigma(T)$.) Therefore we have a contradiction to the assumption that $t$ belonged to a $\subseteq$-minimal element of $\varphi(rep(T))$. ■

As a consequence we now have a method for computing an $\approx$-approximation of $\varphi(\mathcal{S})$.

**Corollary 1** $rep(\widehat{\varphi}(T(\mathcal{S}))) \approx \varphi(\mathcal{S})$.

In other words, first invert the source extensions through their definitions, then apply the $\widehat{\varphi}$-evaluation of the user query $\varphi$ on the resulting tableau. The result of the evaluation is another tableau, which the user perceives as a relation with nulls.

The problem of computing exact answer to a user query was not addressed in the literature except for a brief discussion in [GM99], instead all of the algorithms are aimed at computing the possible or, most commonly, certain answer.

The *possible answer* can be defined as $\varphi^*(\mathcal{S}) = \bigcup\{\varphi(d) : d \in poss(\mathcal{S})\}$. The *certain answer* can be defined as $\varphi^*(\mathcal{S}) = \bigcap\{\varphi(d) : d \in poss(\mathcal{S})\}$. Given the exact answer that is obviously most informative of all answers, we can obtain the possible answer and the exact answer as follows.

**Lemma 3** *$\varphi_*(\mathcal{S}) = \cap\, rep(\widehat{\varphi}(T(\mathcal{S})))$, and $\varphi^*(\mathcal{S}) \approx \cup\, rep(\widehat{\varphi}(T(\mathcal{S})))$.* ■

However, computing $\widehat{\varphi}(T(\mathcal{S}))$ might involve a lot of redundant work, since it amounts to constructing the tableau corresponding to the entire $ext(\mathcal{S})$, whereas the global relations that are in $body(\varphi)$ might be mentioned in only few source definitions. Furthermore, the query might have selections and joins that could be computed directly at the sources.

### Computing the exact answer directly on the source collection

In view-centric information integration systems a query processor is in charge of reformulating queries written in terms of this global schema to queries on the appropriate sources. This process is also known as *query rewriting*. We can extend the notion of rewriting to *p-rewriting*.

To this end we need a few concepts. The *expansion* of a query $\varphi$ over $desc(\mathcal{S})$, denoted $\varphi^{exp}$, is obtained from $\varphi$ by replacing all the sources in $\varphi$ with their definitions. Existential variables in a source definition are replaced by fresh variables in $\varphi^{exp}$.

Let $\mathcal{S}$ be a source collection and $\varphi$ be a conjunctive query over $desc(\mathcal{S})$. The query $\psi$ is a *p-contained rewriting* of $\varphi$ using $\mathcal{S}$ if $\psi^{exp} \subseteq_p \varphi$. Let $\psi$ be a p-contained rewriting of $\varphi$. We define the $\varphi$-evaluation of $\psi$, denoted $\psi_\varphi$, as follows

$$\psi_\varphi(\mathcal{S}) = \{\sigma_\mu(head(\varphi)) : \sigma(body(\psi)) \subseteq ext(\mathcal{S})\},$$

where $\mu$ is a p-containment mapping from $\varphi$ to $\psi^{exp}$, $\sigma$ is a valuation we extend to $\sigma_\mu$ by setting

$$\sigma_\mu(X) = \begin{cases} \sigma(\mu(X)), \text{ if } \mu(X) \text{ occurs in } head(\psi) \\ \text{a fresh variable, otherwise} \end{cases}$$

Note that it is possible that there is more than one p-containment mapping from $\varphi$ to $\psi^{exp}$. However, as formalized in the following Lemma, choosing one mapping over the other does not affect $\psi_\varphi(\mathcal{S})$.

**Lemma 4** *Let $\varphi$, and $\phi$ be conjunctive queries, such that $\phi^{exp} \subseteq \varphi$, and let $\mu_1$ and $\mu_2$ be containment mappings from $\varphi$ to $\phi^{exp}$. Then for all source collections $\mathcal{S}$, $\{\sigma_{\mu_1}(head\,\varphi) : \sigma_{\mu_1}(\varphi) \subseteq ext(\mathcal{S})\} = \{\sigma_{\mu_2}(head\,\varphi) : \sigma_{\mu_2}(\varphi) \subseteq ext(\mathcal{S})\}$, up to renaming of the fresh variables.*

Now we can define $\widetilde{\varphi}(\mathcal{S})$ as

$$\widetilde{\varphi}(\mathcal{S}) = \bigcup\{\psi_\varphi(\mathcal{S}) : \psi^{exp} \subseteq_p \varphi\},$$

and state the following important result:

**Theorem 4** *$\widetilde{\varphi}(\mathcal{S}) \equiv_{sub} \widehat{\varphi}(T(\mathcal{S}))$.*

**Proof.** Let $t$ be an arbitrary atom in $\widetilde{\varphi}(\mathcal{S})$. Then there was conjunctive query $\psi$ (over $desc(S)$) in the union of maximally-contained p-rewritings of $\varphi$ and a $\varphi$-evaluation of $\psi$, using a valuation $\sigma_\mu$ such that $t = \sigma_\mu(head(\varphi))$, and $\sigma(body(\psi)) \subseteq ext(\mathcal{S})$, where $\mu$ is a containment mapping from $\varphi$ to $\psi^{exp}$.

Since $\sigma(body(\psi)) \subseteq ext(\mathcal{S})$, it means that all atoms in $\sigma(body(\psi^{exp}))$ are in $T(\mathcal{S})$ (with fresh existential variables). Since $\mu$ is a containment mapping from $\varphi$ to $\psi^{exp}$, we have that $\sigma(\mu(body(\varphi))) \subseteq T(\mathcal{S})$. Thus $\sigma(\mu(head(\varphi))) \in \widehat{\varphi}(T(\mathcal{S}))$. Now $\sigma(\mu(head(\varphi)))$ is equal to $\sigma_\mu(head(\varphi))$, except for positions that don't occur in $head(\psi)$, these have been replaced by fresh variables in $\sigma_\mu(head(\varphi))$. If we now define a substitution $\theta$ that maps each of these fresh variables to the variable or constant in the corresponding position in $\sigma(\mu(head(\varphi)))$, we get that $\theta(\sigma(\mu(head(\varphi)))) = \sigma_\mu(head(\varphi))$, and consequently $\sigma_\mu(head(\varphi)) \leq_{sub} \sigma(\mu(head(\varphi)))$. This means that $\widetilde{\varphi}(\mathcal{S}) \leq_{sub} \widehat{\varphi}(T(\mathcal{S}))$.

For the proof of inclusion in the other direction, let $t$ be an arbitrary atom in $\widehat{\varphi}(T(\mathcal{S}))$. Suppose $body(\varphi)$ consists of atoms $b_1, b_2, \ldots, b_n$. Then there is a substitution $\theta$, such that $\theta(b_i) \in T(\mathcal{S})$, for all $i \in \{1, \ldots, n\}$. But each atom $\theta(b_i)$ is in $T(\mathcal{S})$ because there is a source $S_{i_j} = (\varphi_{i_j}, v_{i_j})$, a valuation $\sigma_{i_j}$, and a fact $t_{i_j} \in v_{i_j}$, such that $t_{i_j} = \sigma_{i_j}(head(\varphi_{i_j}))$ and $\theta(b_i) \in refresh(\sigma_{i_j}(body(\varphi_{i_j})))$. If we take $\psi$ to be the query with $body(\psi) = \sigma_{1_j}(head(\varphi_{1_j})), \sigma_{2_j}(head(\varphi_{2_j})), \ldots, \sigma_{n_j}(head(\varphi_{n_j}))$, and $head(\psi) = (\sigma_{1_j} \cup \sigma_{2_j} \cup \cdots \cup \sigma_{n_j})(head(\varphi))$, we have a containment mapping (namely $\theta$), from $\varphi$ to $\psi^{exp}$. Consequently $\psi$ will be an element in the union of queries $\widetilde{\varphi}$, and obviously $\psi$ generates the fact $t$ when applied to $\mathcal{S}$. Since $t \leq_{sub} t$, we have established that $\widetilde{\varphi}(\mathcal{S}) \leq_{sub} \widehat{\varphi}(T)$. ■

In the next section we give an algorithm that, for a given conjunctive query $\varphi$, computes a finite union of conjunctive queries equivalent to $\widetilde{\varphi}$.

# 5 The P-bucket Algorithm

Since we generalized the notion of containment mapping to p-containment mapping it is only natural that any rewriting algorithm, which is based on containment mappings, can be extended to produce p-rewriting.

The following algorithm is a straightforward modification of bucket algorithm [LRO96] and, therefore, we call it p-bucket algorithm. Given a query $\varphi$ the p-bucket algorithm proceeds in two steps. In the first step, the algorithm creates a bucket for each subgoal in $\varphi$. Then the buckets are populated by source atoms (subgoals) that are relevant to answering the particular subgoal. More specifically, consider a bucket for a subgoal $b_\varphi$ of $\varphi$, and a source $S_i = (\varphi_i, v_i)$. If $body(\varphi)$ contains a subgoal $b_{\varphi_i}$ such that there is a (most general) unifier $\theta$ for $b_\varphi$ and $b_{\varphi_i}$, then $\theta(head(\varphi_i))$ is put in the bucket of subgoal $b_\varphi$. In case the subgoal $b_\varphi$ unifies with more than one subgoal in a source $S_i$ the bucket of $b_\varphi$ will contain multiple occurrences of unified $head(\varphi_i)$.

In the second step, the algorithm considers query rewritings that are conjunctive queries, each consisting of one conjunct from every bucket. The head of each rewriting is the projection of variables that are in the body of this rewriting, if the projection results in an empty list the rewriting is discarded. For each rewriting, the algorithm checks whether it's expansion is p-contained in the query. If so, the rewriting is added to the answer. Though not required, a check can be added to determine that the resulting rewriting is not redundant. Hence, the result of algorithm is a union of conjunctive rewritings.

**Theorem 5** *The union of all rewritings produced by the p-bucket algorithm relative to a query $\varphi$ is equivalent to the union of all p-contained rewritings of $\varphi$.*

**Proof.** (*Sketch.*) The p-bucket algorithm produces *only* semantically correct rewritings since it tests for p-containment of each of candidate solutions.

For the proof that the output of the algorithm contains *all* semantically correct rewritings we have to proove that if there exists a p-rewriting $\psi$ of a given conjunctive query $\varphi$ then there will be a p-rewriting $\chi$ in the output of the p-bucket algorithm, such that $\psi \subseteq_p \chi$.

It has been known since Chandra and Merlin's paper [CM77] that every conjunctive query has a unique (up to renaming of variables) minimal equivalent query that can be obtained by deletion of zero or more atoms. Let us call the minimal equvalent of $\psi$ $\psi_{min}$ and the minimal equvalent of $\chi$ $\chi_{min}$. Since $\psi$ is a p-rewriting of $\varphi$ ${\psi^{exp}} \subseteq_p \varphi$ and consequently ${\psi_{min}}^{exp} \subseteq_p \varphi$. It is easy to see that $\psi_{min}$ cannot have more subgoals than $\varphi$ becuase each subgoal of $\psi_{min}$ covers atleast one subgoal of $\varphi$. We now have two cases: either $\psi_{min}$ has the same nubmer of subgoals as $\varphi$, or $\psi_{min}$ has fewer subgoals then $\varphi$.

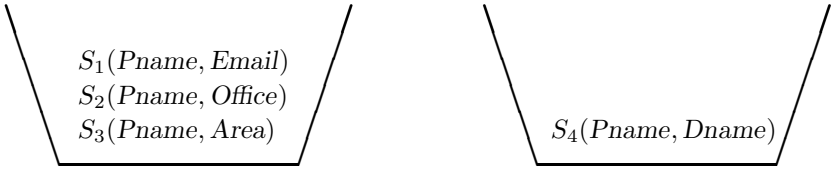
If $\psi_{min}$ has the same number of subgoals as $\varphi$ then each subgoal of $\psi_{min}$ would be placed in the corresponding bucket by the first phase of p-bucket algorithm. In the second phase the algorithm produces cross-product of the contents of all buckets and, thus, it would produce $\chi$ that has all of $\psi_{min}$

subgoals. Then the algorithm would compute the $head(\chi)$ that would be the longest possible list of varables of the $head(\varphi)$. Therefore, the p-bucket algorithm would produce $\chi$ such that $\psi_{min} \subseteq_p \chi$.

If $\psi_{min}$ has fewer subgoals then $\varphi$ then each subgoal of $\psi_{min}$ would be placed in the corresponding bucket by the first phase of p-bucket algorithm. In the second phase the algorithm produces cross-product of the contents of all buckets and, thus, it would produce $\chi$ that has all of $\psi_{min}$ subgoals plus some redundant subgoals. Then the algorithm would compute the $head(\chi)$ that would be the longest possible list of varables of the $head(\varphi)$. Therefore, the p-bucket algorithm would produce $\chi$ such that $\psi_{min} \subseteq_p \chi_{min}$.

The equivalence of the claim of the theorem now follows from the characterization of equivalence of unions of conjunctive queries given in [SY80]. ■

We illustrate the algorithm with the example given in the introduction. The first step of the algorithm will construct and populate two buckets, one for each of the subgoals in the query:




The second step of the algorithm produces the following p-rewritings:

$Q_1(Pname, Email) \leftarrow S_1(Pname, Email), S_4(Pname, compsci)$

$Q_2(Pname, Office) \leftarrow S_2(Pname, Office), S_4(Pname, compsci)$

$Q_3(Pname, Area) \leftarrow S_3(Pname, Area), S_4(Prof, compsci)$

Note that the second step of the algorithm can be easily modified to insert the fresh variables representing unknown values in the head of the rewritings. The result of the algorithm can then be presented to the user as a table with nulls as in the example in the introduction.

# 6 Summary

For an illustrative example, let $\mathcal{S}$ consist of a single source with definition $V(D, A, C) \leftarrow R(D, A, B), S(B, C)$ and extension $\{V(c, a, a), V(d, a, b)\}$.

- $poss(\mathcal{S})$ is equal to the closure under supersets of $\{\{R(c, a, a_0), S(a_0, a), R(c, a, a_0), S(a_0, a)\}, \{R(c, a, a_0), S(a_0, a), R(c, a, a_1), S(a_1, a)\}, \ldots\}$, where $a_0, a_1, \ldots$ is an enumeration of **dom**.
- $T(\mathcal{S}) = \{R(c, a, X), S(X, a), R(d, a, Y), S(Y, b)\}$. It is easy to see that $rep(T(\mathcal{S})) = poss(\mathcal{S})$.

Consider now $\varphi = Q(W, X, Z) \leftarrow R(W, X, Y), S(Z, X)$.

- We have $\varphi(\mathcal{S}) \approx \widehat{\varphi}(T(\mathcal{S})) = \{Q(c, a, X), Q(d, a, X)\}$. Let's call this tableau $T$.
- The p-bucket algorithm will produce the rewriting $\psi = Q(W, X, E) \leftarrow V(W, X, Y)$, where $E$ is a fresh variable. Consequently, $\varphi(\mathcal{S}) \equiv_{sub} \widetilde{\varphi}(\mathcal{S}) = \{Q(c, a, X), Q(d, a, Y)\}$. We call this tableau $U$
- Note that $T \equiv_{sub} U$, but $T \not\equiv_\Omega U$ when $\Omega$ is equal to all (unions of) conjuctive queries. Although $T$ and $U$ have the same partial facts, $T$ contains the additional information that the two variables represent the same unknown value. This allows the user to extract more certain information from $T$ than from $U$. For example, let $\psi = P(U, W) \leftarrow Q(U, Y, X), Q(W, Z, X)$. Then the certain answer to $\psi(rep(T))$ is $\{P(c, d)\}$, and the certain answer to $\psi(rep(U))$ is empty.

Thus, unless the user wants to materialize the exact answer and obtain certain answers for subsequent queries on it, we conclude that for practical purposes in information integration systems, the tableau $\widetilde{\varphi}(\mathcal{S})$ is a sufficient approximation of $\varphi(\mathcal{S})$.

Note that in the introductory example the tableau representing the inversion of sources $S_1, S_2, S_3$, and $S_4$ will never contain repeated occurences of any variable. In this example the $\widetilde{\varphi}$ and $\widehat{\varphi}$ evaluations of the user query $\varphi = Q(Pname, Email, Office, Area) \leftarrow Prof(Pname, Email, Office, Area)$, $Dept(Pname, compsci)$ both produce the same tableau.

# References


[DG97] O. M. Dushka, M. R. Genesereth. Answering recursive queries using views. In *Proc. 16th ACM Symp. on Principles of Database System (PODS '97)*, pp. 109–116 Tuscon, Arizona, 1997.

[CM77] A. K. Chandra, P. M. Merlin. Optimal implementation of conjunctive queries. In *Proc. ACM SIGACT Symp. on the Theory of Computing (STOC '77)*, pp. 77–90, 1977.

[GM99] G. Grahne, A. O. Mendelzon. Tableau Techniques for Querying Information Sources through Global Schemas. In *Proc. 7th International Conference on Database Theory (ICDT '99)*. pp. 332–347, Delphi, Greece 1999.

[IL84] T. Imielinski, W. Lipski Jr. Incomplete Information in Relational Databases. In *J. ACM* **31**:4, 1984, pp. 761–791.

[Len02] M. Lenzerini. Data Integration: A Theoretical Perspective. Invited tutorial in *Proc. 21st ACM Symp. on Principles of Database Systems (PODS '02)*, Madison, Wisconsin 2002.

[LMSS95] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, D. Srivastava. Answering Queries Using Views. In *Proc. 14th ACM Symp. on Principles of Database Systems (PODS '95)*, pp. 95–104, San Jose, California 1995.

[LRO96] A. Y. Levy, A. Rajaraman, J. J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. *Proc. 22nd Int'l. Conf. on Very Large Databases (VLDB '96)*, pp. 251–262, Mumbai (Bombay), India 1996.

[Men84] A. O. Mendelzon. Database States and Their Tableaux. In *ACM Trans. on Databases Systems* **9**:2, 1984, pp. 264–282.

[SY80] Y. Sagiv, M. Yannakakis Equivalence among relational expressions with the union and difference operators. In *J. ACM* **27**:4, 1980, pp. 633–655.

[Ull97] J. D. Ullman. Information Integration Using Logical Views. In *Proc. 6th International Conference on Database Theory (ICDT '97)*. pp. 19–40, Delphi, Greece 1997.