Tableau Techniques For Querying Information Sources Through Global Schemas

Gösta Grahne[1] and Alberto O. Mendelzon[2]

[1] Department of Computer Science, Concordia University
Montreal, Quebec, Canada H3G 1M8
grahne@cs.concordia.ca

[2] Department of Computer Science, University of Toronto
Toronto, Ontario, Canada M5S 3H5
mendel@cs.toronto.edu

Abstract. The foundational homomorphism techniques introduced by Chandra and Merlin for testing containment of conjunctive queries have recently attracted renewed interest due to their central role in information integration applications. We show that generalizations of the classical *tableau* representation of conjunctive queries are useful for computing query answers in information integration systems where information sources are modeled as views defined on a virtual global schema. We consider a general situation where sources may or may not be known to be correct and complete. We characterize the set of answers to a global query and give algorithms to compute a finite representation of this possibly infinite set, as well as its *certain* and *possible* approximations. We show how to rewrite a global query in terms of the sources in two special cases, and show that one of these is equivalent to the Information Manifold rewriting of Levy *et al.*

1 Introduction

Information Integration systems [Ull97] aim to provide a uniform query interface to multiple heterogeneous sources. One particular and useful way of viewing these systems, first proposed within the Information Manifold project [LRO96], is to postulate a *global schema* (called a world view) that provides a unifying data model for all the information sources. A query processor is in charge of accepting queries written in terms of this global schema, translating them 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. Note the reversal of the classical model: instead of thinking of views as virtual artifacts defined on stored relations, we think of the views as stored, and the relations that the views are defined on as virtual. [1]

[1] Whether the views are actually stored at each source or materialized by other means, and whether they consist of relations or semi-structured objects or files are issues irrelevant to our discussion and hidden from the query processor by appropriate wrappers.

A question of semantics now arises: what is the meaning of a query? Since a query is expressed in terms of the global schema, and the sources implicitly represent an instance of this global schema, it would be natural –at least conceptually– to reconstruct the global database represented by the views and apply the query to this global database. There are at least two issues that must be resolved for this to work.

First, the database represented by a set of sources may not be unique; in fact, it may not even exist. For two trivial examples: first, suppose we have a single information source, which is defined as the projection on attribute A of the global binary relation $R(A, B)$. For any given set of tuples stored at the source, there are many (perhaps an infinite number) of possible global databases. Second, suppose we have not one, but two sources, both storing the projection on A as before; one contains the single tuple $\langle a_1 \rangle$, and the other one the single tuple $\langle a_2 \rangle$; then there is *no* global database whose projections equal these two sources. In sum, the first issue is: what database or databases are represented by a given set of sources?

Second, suppose we have identified the set of databases that are represented by a given set of sources. Applying the query to each database and producing all possible answers may be impossible (e.g. if there is an infinite number of such answers) or undesirable. The second issue is: how do we produce a single compact representation of these multiple answers, or an approximation to them if we so desire?

In this paper, we explore answers to both questions. Before overviewing the results, let us make the example above more concrete. Consider a schema for storing information about the first round of the World Cup Soccer Tournament. Suppose global relation *Team(Country,Group)* represents a list of all teams giving the name of the country and the group to which the country has been assigned for first round play.

Suppose first that the only source, S_{Team}, stores a unary relation listing all the countries that are participating in the first round. The corresponding view mapping is given by the conjunctive query:

$$S_{Team}(x) \leftarrow Team(x, y).$$

What global databases are represented by S_{Team}? They are all the relations *Team* such that the view mapping applied to *Team* produces exactly the given instance of S_{Team}, that is:

$$S_{Team} = \pi_{Country}(Team).$$

In this case, we say the view is both *sound* and *complete*.

On the other hand, suppose the only source is S_{Qual}, which contains the list of all teams that participated in the qualifying round. This is a strict superset of the teams that will actually be playing in the tournament. Since the global schema says nothing about the qualifying round, the only reasonable view mapping is still

$$S_{Qual}(x) \leftarrow Team(x, y).$$

However, now we understand that this is just an approximation, and that the actual database could be any relation whose projection on *Country* produces a *subset* of the source S_{Qual},[2] that is:

$$S_{Qual} \supseteq \pi_{Country}(Team).$$

In this case, we say the view is *complete* (since it lists every possible team) but not sound (since it lists teams that are not in the first round.)

Finally, suppose the only source is S_{Tube}, listing those teams whose games will be televised. Again, the best way to represent the view mapping (since there is no information about television in the global schema) is by

$$S_{Tube}(x) \leftarrow Team(x, y).$$

In this case, every team listed in S_{Tube} corresponds to some tuple in the ideal *Team* relation, but there are tuples in this ideal relation not represented in S_{Tube}. Thus, we take as the set of represented databases all the relations *Team* that satisfy

$$S_{Tube} \subseteq \pi_{Country}(Team).$$

In this case, we say the view is sound, but not complete.

In sum, we will annotate each source, not only with the view mapping that is associated with it, but also with the knowledge of whether this view is guaranteed to be sound, or complete, or both. We will say a global database is *represented* by a given set of (annotated) sources when applying the view mappings to this database produces a subset of every complete source and a superset of every sound source.

In the example, if we had *both* the complete source S_{Qual} and the sound source S_{Tube}, the databases represented by them would be all those that contained *at least* all the countries in S_{Tube} and *at most* those in S_{Qual}. If we had the sound and complete source S_{Team} (with or without the other two), then databases would be constrained to contain exactly the teams listed in this source. Note that it is possible for a given set of source instances to be inconsistent with the source annotations; for example, if S_{Team} mentions a team not listed in S_{Qual}, or omits one mentioned in S_{Tube}. In such a case, the set of represented databases is empty.

The outline of the paper is as follows. In Section 2 we formalize the model above. In order to finitely represent infinite sets of databases, we introduce *templates*, and show how to associate with each set of source instances a template that characterizes all databases represented by the sources. A template is a vector of consisting of tableaux [ASU79] and a set of constraints. Our constraints are related to the egd's of Beeri and Vardi [BV81]. In Section 3 we present our results. First we give a syntactic characterization of consistent sets of sources in terms of their templates. Then, we give an algorithm for evaluating a conjunctive

[2] Actually, only relations with 32 tuples in them qualify, since there are 32 teams in the first round, but we ignore this level of detail in our model.

query, expressed on the global schema, using only the given source instances. The algorithm uses the template generated from the source instances and produces as output another template that describes all the answers, one for each represented database. We also show how to compute two natural approximations to this set: the union (the approximation by *possibility*) and the intersection (the approximation by *certainty*) of all the answers.

Next, we study the complexity of conjunctive query evaluation. We show that consistency is coNP-complete in general, but consistency and query evaluation become polynomial when we have only sound or only complete sources. These results extend and generalize earlier ones of Abiteboul and Duschka [AD98].

Finally, in Section 4 we show how to rewrite a query on the global schema in terms of the sources, in two cases: when all sources are sound and the certain semantics is of interest, and when all sources are complete and the possible semantics is sought. This is an optimization over building the template and applying the generic method, since the cost of building the template is linear in the size of all the source instances but there may be only a few out of many sources that are relevant to the query; the rewritings will only use the relevant sources. We show that the all-sound, certain-answer rewriting is equivalent to (although different from) the Information Manifold rewriting described by Levy *et al.* in [LM*95] without any formal justification. In fact, this was the original motivation for our work: providing semantics for the mysterious Information Manifold algorithm.

2 The Model

In this section we first introduce global databases. A global database is a standard multirelational database. We then introduce database templates, a tool for representing sets of global databases. Finally, we define the concept of source collections, modeling a set of sources that we want to amalgamate.

For basic terminology and concepts regarding databases we refer to [AHV95].

Global Databases

Let **rel** be a countably infinite set $\{R, S, \ldots, R_1, R_2, \ldots, S_1, S_2 \ldots\}$ of *global relation names*, Let **dom** be a countably infinite set $\{a, b, \ldots, a_1, a_2, \ldots, b_1, b_2, \ldots\}$ of *constants*, and let **var** be a countably infinite set $\{x, y, \ldots, x_1, x_2, \ldots, y_1, y_2, \ldots\}$ of *variables*.

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 $k = arity(R)$.

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

Template Databases

We now introduce templates, which are a generalization of tableaux intended to represent concisely a large or infinite number of possible database instances.

Let R be a global relation name. An *atom* over R is an expression of the form $R(e_1, \ldots, e_k)$, where $k = arity(R)$ and e_i is in $\mathbf{dom} \cup \mathbf{var}$. Atoms containing only constants will sometimes be called *ground atoms*. Note that a ground atom is the same as a fact.

Let $\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$ be a set of global 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 *constraint* over $\mathbf{R}$ is a pair (U, Θ), where U is a tableau over $\mathbf{R}$ and Θ is a finite set of substitutions of the form $\{x_1/a_1, x_2/a_2, \ldots, x_p/a_p\}$, where all the x_i:s are distinct variables appearing in U.

Finally, a *database template* $\mathcal{T}$ over $\mathbf{R}$ is a tuple $\langle T_1, T_2, \ldots, T_m, C\rangle$, where each T_i is a tableau over $\mathbf{R}$ and C is a finite set of constraints over $\mathbf{R}$.

Example 1. Let $\mathcal{T} = \langle T_1, T_2, C\rangle$, where $T_1 = \{R(a, x), S(b, c), S(b', c)\}$, $T_2 = \{R(a', b'), S(b', c)\}$, and $C = \{(R(a, z), \{\{z/b\}, \{z/b'\}\})\}$. This template contains two tableaux, and one constraint with two substitutions.

A *valuation*, is 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 of substitutions $\{x_1/a_1, \ldots, x_p/a_p\}$. The identity on constants is omitted in this notation. Two valuations are *compatible* if they agree on the variables on which both are defined.

A database template $\mathcal{T}$ on schema $\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$ represents a set of global databases on $\mathbf{R}$. This set is denoted $rep(\mathcal{T})$, and it is defined by

$$\begin{aligned} rep(\mathcal{T}) = \{D \ :\ & \text{there is a valuation } \vartheta \text{ and a tableau } T_i \text{ in } \mathcal{T}, \\ & \text{such that } \vartheta(T_i) \subseteq D, \\ & \text{and for all } (U, \Theta) \in C \text{ in } \mathcal{T}, \text{ and valuations } \sigma, \\ & \text{if } \sigma(U) \subseteq D \text{ then there is a } \theta \in \Theta \text{ such that } \sigma \text{ and } \theta \text{ are compatible}\}. \end{aligned}$$

The definition says that a database D is represented by template $\mathcal{T}$ if there is a tableaux T_i in $\mathcal{T}$, such that when all variables in T_i are replaced by some constants, the set of facts thus obtained is a subset of D. This is the "sound" part of $\mathcal{T}$. Furthermore, the database D has to satisfy the "completeness" restrictions encoded in the constraints (U, Θ) in C: Whenever it is possible to embed the tableau U in D, the mapping used for the embedding must be extendible to one of the substitutions in Θ.

Example 2. Consider the template $\mathcal{T} = \langle T_1, T_2, C\rangle$ from Example 1. The template $\mathcal{T}$ represents the three databases $\{R(a, b), S(b, c), S(b', c)\}$, $\{R(a, b'), S(b, c), S(b', c)\}$, $\{R(a', b'), S(b', c)\}$, and any of their supersets satisfying the constraint saying that whenever a occurs in the first component of R, then the second component has to be b or b'. For instance, $\{R(a, b), R(a, b'), S(b, c), S(b', c)\}$ is a database in $rep(\mathcal{T})$, while $\{R(a, c), R(a, b'), S(b, c), S(b', c)\}$ is not in $rep(\mathcal{T})$.

Note the disjunctive role of the several tableaux in a template, similar to the *flocks* of [FKUV86]. Note also that the constraint can encode the pure closed world assumption. For instance, the closed world interpretation of a relation with the two facts $R(a,b)$ and $R(c,d)$, can be expressed as the template $\langle T, C\rangle$, with $T = \{R(a,b), R(c,d)\}$, and $C = (\{R(x,y), \{\{x/a, y/b\}, \{x/c, y/d\}\})$.

Chasing a Tableau

We now introduce a binary relation $\mathcal{R}_{(U,\{\theta\})}$ between tableaux, where the relation is parameterized by a constraint with a singleton binding set. The idea is that T and T' are related by $\mathcal{R}_{(U,\{\theta\})}$ when U can be embedded into T by a valuation and then this valuation and θ are applied to T to obtain T'. More formally, we say that U *applies* to T through σ when there is a valuation σ such that $\sigma(U) \subseteq T$. Then T *satisfies* $(U, \{\theta\})$ wrt σ if U applies to T and $\theta(U)$ and $\sigma(U)$ can be unified. Let T and T' be tableaux. Then $T\ \mathcal{R}_{(U,\{\theta\})}\ T'$, if and only if U applies to T through σ and, either T satisfies $(U, \{\theta\})$ wrt σ, in which case $T' = mgu(\theta(U), \sigma(U))(T)$, or T does not satisfy $(U, \{\theta\})$ wrt σ, in which case $T' = \perp$. [3]

Then let (U, Θ) be a general constraint. We define $\mathcal{R}_{(U,\Theta)} = \bigcup_{\theta\in\Theta} \mathcal{R}_{(U,\{\theta\})}$. If C is a set of constraints we set $\mathcal{R}_C = \bigcup_{(U,\Theta)\in C} \mathcal{R}_{(U,\Theta)}$. Finally, let $\mathcal{R}_C^*$ be the transitive closure of $\mathcal{R}_C$.

It is straightforward to show that $\mathcal{R}_C$ is well-founded, i. e. for any tableau T, the set $\{T' : T\ \mathcal{R}_C^*\ T'\}$ is finite. This set therefore contains some maximal elements, i. e. tableaux T', such that for no tableau T'' does $T'\ \mathcal{R}_C\ T''$ hold. We can now define the *chase* of a tableau T with a set of constraints C, denoted $C(T)$, to be the set of maximal elements in $\{T' : T\ \mathcal{R}_C^*\ T'\}$, provided the last set is nonempty. Otherwise $C(T) = \{T\}$. In the template $\langle T_1, T_2, C\rangle$ from Example 1 we have $C(T_1) = \{\{R(a,b), S(b,c), S(b'c)\}, \{R(a,b'), S(b,c), S(b'c)\}\}$, and $C(T_2) = T_2$. For another example, let $T = \{R(a,x), R(y,d)\}$, and let $C = (\{R(a,u)\}, \{u/b\}, \{u/b'\}) \cup (\{R(w,d)\}, \{w/c\}, \{w/c'\})$, Then one maximal chain in the “chasing sequence” is

$$\{R(a,x), R(y,d)\}\ \mathcal{R}_C\ \{R(a,b), R(y,d)\}\ \mathcal{R}_C\ \{R(a,b), R(c,d)\}.$$

Another maximal chain is

$$\{R(a,x), R(y,d)\}\ \mathcal{R}_C\ \{R(x,b), R(c',d)\}\ \mathcal{R}_C\ \{R(a,b), R(c',d)\}.$$

It is easy to verify that

$$\begin{aligned} C(T) = \{&\{R(a,b), R(c,d)\},\\ &\{R(a,b), R(c',d)\},\\ &\{R(a,b'), R(c,d)\},\\ &\{R(a,b'), R(c',d)\}\}. \end{aligned}$$

In general, the chase has the following property.

[3] mgu means *most general unifier*, see e. g. [AHV95], and $\perp$ represents the inconsistent tableaux; no U can be applied to $\perp$, and no valuation can be applied to $\perp$.

Theorem 1. *Let* $\mathcal{T} = \langle T_1, \ldots, T_n, C\rangle$ *be a database template, and let* $\{T'_1, \ldots, T'_s\} = \bigcup_{i \in [1,n]} C(T_i)$*. Then* $rep(\mathcal{T}) = rep(\langle T'_1, \ldots, T'_s, C\rangle)$*.*

A fundamental question for a template database $\mathcal{T}$ is whether $rep(\mathcal{T})$ is non-empty, that is, whether $\mathcal{T}$ is consistent. The next theorem gives a syntactic condition.

Theorem 2. *Let* $\mathcal{T} = \langle T_1, \ldots, T_n, C\rangle$ *be a database template. Then* $rep(\mathcal{T}) \neq \emptyset$ *if and only if there is a tableau* T_i *in* $\mathcal{T}$*, such that* $\perp \notin C(T_i)$*.*

Source collections

Let **loc** be a countably infinite set $\{V, U, \ldots V_1, V_2, \ldots, U_1, U_2, \ldots\}$ 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 global relation names.

A *view definition* φ is a conjunctive query

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

where $body(\varphi)$ is a sequence $b_1, b_2, \ldots, b_n$, where each b_i is an atom over a global relation name, and $head(\varphi)$ is an atom over a local relation name V. We also assume that all variables occurring in $head(\varphi)$ also occur in $body(\varphi)$, i. e. that the query φ is *safe*. The conjunctive query φ can be applied to a database D in the usual way, resulting in a set of facts $\varphi(D)$ over V.

A *view extension* for a view φ is a finite set of facts over the relation name V in the head of the view definition φ. Such an extension will be denoted v.

A *source* S is triple consisting of a view definition φ, a set containing one or both of the labels *open* and *closed*, and a view extension v for φ. Sources are called open, closed, or clopen (closed and open) according to their labels. Sometimes open sources will be called *sound*, and closed sources called *complete*. Clopen sources will be called sound and complete.

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 views 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 triple in $\mathcal{S}$. The *extension* of a source collection $\mathcal{S}$, denoted $ext(\mathcal{S})$, is the union of all view extensions in it. 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 sources' soundness/completeness and defining views. Source collections $\mathcal{S}$ have *one* extension, $ext(\mathcal{S})$, which for each source S_i is a set of facts over the defining view-head relation name.

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

$$\begin{aligned} poss(\mathcal{S}) = \{D \text{ over } & sch(\mathcal{S}) \ : \\ & v_i \subseteq \varphi_i(D) \text{ for all open sources } S_i \in \mathcal{S} \\ & v_j \supseteq \varphi_j(D) \text{ for all closed sources } S_j \in \mathcal{S}\} \\ & v_k = \varphi_k(D) \text{ for all clopen sources } S_k \in \mathcal{S}\} \end{aligned}$$

Note that there are source collections, such as $\mathcal{S}$ containing the two sources $\langle\{V_1(x) \leftarrow R(x,y), \{open\}, \{V_1(a)\}\rangle$, and $\langle V_2(u) \leftarrow R(u,w), \{closed\}, \{V_2(b)\}\rangle$, for which $poss(\mathcal{S})$ is empty. This is because the extension of V_1 requires that there be a tuple in every possible database with an a in its first column, while the extension of V_2 requires that every possible database must have only b's in its first column. Such source collections are called *inconsistent.* If $poss(\mathcal{S})$ is nonempty the source collection is said to be *consistent.* We will see in Theorem 4 below that consistency of source collections can be characterized syntactically through database templates and the chase.

3 Querying source collections

In this section we first show how templates can be used to query source collections, by extending conjunctive queries to operate over templates. We then analyze the computational complexity of querying source collections. The complexity turns out to be coNP-complete.

Query evaluation

Let $\mathcal{S}$ be source collection, and Q a conjunctive query, such that the atoms in the body of Q are atoms over relation names in $sch(\mathcal{S})$, and the head of Q is an atom over an relation name in the set $\{ans, ans_1, ans_2, \ldots\}$. Now Q applied to $\mathcal{S}$ defines the following set of possible answers:

$$Q(\mathcal{S}) = \{Q(D) \ : \ D \in poss(\mathcal{S})\}.$$

This (exact) answer can be approximated from below as

$$Q_*(\mathcal{S}) = \bigcap_{D \in poss(\mathcal{S})} Q(D),$$

and from above as

$$Q^*(\mathcal{S}) = \bigcup_{D \in poss(\mathcal{S})} Q(D).$$

The problem begging for a solution is how to compute the answers to queries on source collections. Towards this end we shall first construct a database template over the global relation names. This database template will represent the set of possible databases for $\mathcal{S}$. Let $\mathbf{R}$ be a global schema. We define a function (T, C) from sources with defining view φ, where the body of φ consists of atoms over relation names in $\mathbf{R}$, to template databases over $\mathbf{R}$. Given a source $S_i = \langle \varphi_i, labels, v_i\rangle$, we set $T(S_i) = \{t \ : \ t$ in $body(\varphi_i)\theta$ and $head(\varphi_i)\theta = u$, for some $u \in v_i$ and assignment $\theta\}$, provided $open \in labels$. Otherwise $T(S_i) = \emptyset$. If $closed \in labels$ we set $C(S_i) = (U, \Theta)$, where U consists of all atoms in the body of φ_i, and $\Theta = \{\theta : head(\varphi_i)\theta = u$, for some $u \in v_i\}$. Otherwise $C(S_i) = \emptyset$. Finally, we set

$$\mathcal{T}(\mathcal{S}) = \left\langle \bigcup_{S_i \in \mathcal{S}} T(S_i), \bigcup_{S_i \in \mathcal{S}} C(S_i) \right\rangle.$$

Example 3. Consider $\mathcal{S} = \{S_1, S_2, S_3\}$, where $S_1 = \langle V_1(u,w) \leftarrow S(u,w), \{open\}, \{V_1(b,c), V_1(b',c)\}\rangle$, $S_2 = \langle V_2(v) \leftarrow R(v,x), \{open\}, \{V_2(a)\}\rangle$, and $S_3 = \langle V_3(z) \leftarrow R(a,z), \{closed\}, \{V_3(b), V_3(b')\}\rangle$. Then $\mathcal{T}(\mathcal{S}) = \langle T_1, C\rangle$, where $T_1 = \{R(a,x), S(b,c), S(b',c)\}$, and $C = \{(R(a,z), \{\{z/b\}, \{z/b'\}\})\}$.

Note that T_1 and C above are from Example 1. The $\mathcal{T}$-function will always construct templates with a single tableau. Templates with multiple tableaux are produced by the chase prosess or by query evaluation. Query evaluation is explained below.

The database templates constructed by the function $\mathcal{T}$ have the following desirable property.

Theorem 3. *$rep(\mathcal{T}(\mathcal{S})) = poss(\mathcal{S})$.*

We will now return to the issue of consistency of a source collection. Using the characterization in Theorem 2 together with Theorem 3 we can prove the following.

Theorem 4. *Let $\mathcal{S}$ be a source collection and $\mathcal{T}(\mathcal{S}) = \langle T, C\rangle$. Then $\mathcal{S}$ is consistent if and only if $\perp \notin C(T)$.*

Corollary 1. *If no source in a collection $\mathcal{S}$ is closed, then $\mathcal{S}$ is consistent. Likewise, if no source in a collection $\mathcal{S}$ is open, then $\mathcal{S}$ is consistent.*

For an example of an inconsistent source collection, consider the collection $\mathcal{S}$ from Section 2, where $\mathcal{S}$ contains the two sources $\langle\{V_1(x) \leftarrow R(x,y), \{open\}, \{V_1(a)\}\rangle$, and $\langle V_2(u) \leftarrow R(u,w), \{closed\}, \{V_2(b)\}\rangle$. We will now have $\mathcal{T}(\mathcal{S}) = \langle T, C\rangle$, where $T = \{R(a,y)\}$, and $C = \langle\{R(u,w)\}, \{\{u/b\}\}\rangle$. It is easy to see that $C(T) = \perp$, and that $poss(\mathcal{S})$ is empty.

We then return to the issue of evaluating queries over global relations given only the local relations in a source collection $\mathcal{S}$. Since we are able to construct a database template $\mathcal{T}$ representing all databases in $poss(\mathcal{S})$ it is is natural to extend the standard query evaluation mechanism to operate on database templates.

Let $\mathcal{T} = \langle T_1, \ldots, T_n, C\rangle$ be a database template. Given a conjunctive query $Q = ans(\boldsymbol{x}) \leftarrow R_1(\boldsymbol{x}_1), \ldots, R_n(\boldsymbol{x}_m)$ where each R_i is a relation name in $\mathbf{R}$, our evaluation proceeds as follows. Let $\{T_1', \ldots, T_s'\} = \bigcup_{i \in [1,n]} C(T_i)$. Then define $\widehat{Q}(\mathcal{T}) = \langle T_1'', \ldots, T_q'', \emptyset\rangle$, where each T_j'', $j \in [1,q]$ is equal to a $\widehat{Q}_\kappa(T_i')$, for some $i \in [1,s]$ and substitution κ. Here

$$\begin{aligned}\widehat{Q}_\kappa(T_i') = \{ans(\boldsymbol{x})\sigma : \sigma \text{ is an } mgu \text{ for } \{R_i(\boldsymbol{x}_i)\}_{i=1}^m \text{ and subset of } T_i', \\ \text{and } \sigma \text{ is compatible with } \kappa\},\end{aligned}$$

Note that there can be only finitely many distinct T_j'':s (modulo renaming). In computing the unifiers used in $\widehat{Q}_\kappa(T_i')$ we always try to substitute variables or constants from T_i' for variables in the query. For the inconsistent tableaux $\perp$, we set $\widehat{Q}_\kappa(\perp) = \perp$.

As an illustration, suppose our query Q is $ans(z,u,w) \leftarrow R(z,u), S(u,w)$. If our database template $\mathcal{T}$ is $\langle T_1, T_2, C\rangle$ from Example 1, then the query evaluation proceeds as follows: First we chase the tableaux in $\mathcal{T}$ with C obtaining the three tableaux $\{R(a,b), S(b,c), S(b',c)\}$, $\{R(a,b'), S(b,c), S(b'c)\}$, and $\{R(a',b'), S(b',c)\}$. Then we apply $\widehat{Q}_\kappa$ to each of these tableaux, getting the results $\{ans(a,b,c)\}$, $\{ans(a,b',c)\}$, and $\{ans(a',b',c)\}$. In this example we only need to use one substitution κ, namely the identity substitution $\{\}$. For another example, let $\mathcal{T} = \langle T, C\rangle$, where $T = \langle\{R(a,x), R(a',x), S(b,c), S(y,c')\}, C\}$, and $C = \emptyset$. For our example query Q, we get $\widehat{Q}_{\{x/b\}}(T) = \{ans(a,b,c), ans(a',b,c)\}$, $\widehat{Q}_{\{x/y\}}(T) = \{ans(a,y,c'),\ ans(a',y,c')\}$, and $\widehat{Q}_{\{\}}(T) = \emptyset$. These are all the answer tableaux; no other substitutions κ will yield more.

The query evaluation uses the constraints in that the constraints are chased into the tableaux before querying. The template resulting from the query evaluation will have the empty constraint. It is clear that we thus loose some information. However, the query result will be *coinitially* equivalent to the desired one. Two sets of databases are coinitial if the have the same set of $\subseteq$ minimal elements (see [IL84]). The coinitiality relation will be denoted $\approx$. Our extended semantics now has the following correctness property:

Theorem 5. $rep(\widehat{Q}(\mathcal{T})) \approx Q(rep(\mathcal{T}))$,

where the query on the right-hand side is evaluated pointwise.

Let $\mathcal{S}$ be a source collection and Q be a query on $sch(\mathcal{S})$. Together Theorems 3 and 5 give the following method for evaluating Q on $\mathcal{S}$.

Theorem 6. *1.* $Q(\mathcal{S}) \approx \widehat{Q}(\mathcal{T}(\mathcal{S}))$.
2. $Q_*(\mathcal{S}) = \cap\, rep(\widehat{Q}(\mathcal{T}(\mathcal{S})))$.
3. $Q^*(\mathcal{S}) \approx \cup\, rep(\widehat{Q}(\mathcal{T}(\mathcal{S})))$.

The computational complexity of querying sources

In general, template databases have the following complexity.

Theorem 7. *Given a database template $\mathcal{T}$, testing whether $rep(\mathcal{T}) \neq \emptyset$ is coNP-complete.*

For the lower bound we use a reduction from the non-containment of tableaux queries, which is known to be a coNP-complete problem [CM77,ASU79]. Let U_1 and U_2 be tableaux queries. Without loss of generality we can assume that the summary rows of U_1 and U_2 are the same, say $\langle x_1, \ldots, x_p\rangle$. Our task is to construct a template $\mathcal{T}$, such that $rep(\mathcal{T}) \neq \emptyset$ if and only if U_1 is not contained in U_2. Let therefore $\mathcal{T} = \langle T, (U, \{\theta\})\rangle$, where T equals the rows of U_1, except that each summary variable x_i has been substituted with a distinct constant a_i. Let U be the rows of U_2, and $\theta = \{x_1/b_1, \ldots, x_p/b_p\}$, where $a_i \neq b_i$. Now it is easy to see that $\mathcal{T}$ is inconsistent if and only if there is a containment mapping from U_2 to U_1.

As outlined in Theorem 6 we have a general method that given a source collection $\mathcal{S}$ evaluates the answer (or the exact or possible answer) on the template $\mathcal{T}(\mathcal{S})$. On the other hand, Theorem 7 says that a template database might have an intricate combinatorial structure. In the template resulting from query evaluation we have chased out this combinatorial structure. It is clear the the resulting template can be intractably large. The following theorem says that there probably is no way to overcome this intractability.

Theorem 8. *1. If $\mathcal{S}$ contains both sound and complete sources the problem of testing whether $poss(\mathcal{S})$ is nonempty is coNP-complete.*
2. If $\mathcal{S}$ contains both sound and complete sources the problem of testing whether a given tuple t is in $Q_*(\mathcal{S})$ is coNP-complete.
3. If $\mathcal{S}$ contains both sound and complete sources the problem of testing whether whether a given tuple t is in $Q^*(\mathcal{S})$ is coNP-complete.

As an aside we note that Theorem 5.1 in [AD98] falls out as a special case of Theorem 8.

If $\mathcal{S}$ contains only sound or only complete sources query evaluation becomes tractable.

Theorem 9. *If $\mathcal{S}$ contains only sound or only complete sources the consistency of $poss(\mathcal{S})$ as well as $Q_*(\mathcal{S})$ and $Q^*(\mathcal{S})$ can be tested/computed in time polynomial in the number of tuples in $\mathcal{S}$. (In the case where $Q^*(\mathcal{S})$ is infinite we only report this fact, we don't output all tuples.)*

4 Query optimization through rewriting

In practice one would be most interested in computing $Q_*(\mathcal{S})$ or $Q^*(\mathcal{S})$. It is clear that computing $Q_*(\mathcal{S})$ or $Q^*(\mathcal{S})$ from $\widehat{Q}(\mathcal{T}(\mathcal{S}))$ might involve a lot of redundant work, since computing $\mathcal{T}(\mathcal{S})$ amounts to constructing the tableaux and constraints corresponding to *all* view-extensions, whereas the global relations mentioned in the body of Q might be mentioned in only a few view definitions. Furthermore, the query might have selections and joins that could be computed directly at the sources. Take for example the open view definitions $\varphi_1 =_{df} V_1(x, y) \leftarrow R(x, y)$, $\varphi_2 =_{df} V_2(u, v) \leftarrow S(u, v)$. and the query $Q =_{df} ans(a, z) \leftarrow R(a, w), S(w, z)$. Obviously it is more efficient to "push the selection and join," and evaluate the query $ans(a, z) \leftarrow V_1(a, w), V_2(w, z)$, on the view extensions v_1 and v_2, rather that obtaining $Q_*(\mathcal{S})$ from $\widehat{Q}$ computed on the tableau constructed from v_1 and v_2 and possibly a large number of other sources.

We shall thus consider the following problem:

Let $\Phi = \{\langle\varphi_1, \{.\}\rangle \ldots, \langle\varphi_n, \{.\}\rangle\}$ be a set of source descriptions where the view definitions have heads $V_1, \ldots, V_n$, and let $\{R_1, \ldots, R_m\}$ be the relation names occurring in the bodies of the φ_i:s. Given a conjunctive query Q where $body(Q)$ contains only global relation names in $\{R_1, \ldots, R_m\}$ find a query $rew(Q)$, where the body of $rew(Q)$ only mentions view relation names in $\{V_1, \ldots, V_n\}$, and $head(rew(Q))$ is compatible with $head(Q)$, such that the following condition holds: For all source collections $\mathcal{S}$ with $desc(\mathcal{S}) = \Phi$ and $ext(\mathcal{S}) = \boldsymbol{v}$ it is true that

1. (Certain answer:) $rew(Q)(\boldsymbol{v}) = Q_*(\mathcal{S})$, or
2. (Possible answer:) $rew(Q)(\boldsymbol{v}) = Q^*(\mathcal{S})$.

The queries $rew(Q)$ above are called *rewritings* of Q with respect to Φ. If the rewriting satisfies condition 1 it is called a *certainty rewriting*, and if it satisfies condition 2 it is called a *possibility rewriting*.

Certainty rewritings

Given a query Q on $\{R_1, \ldots, R_m\}$ we shall now construct the desired rewriting on $\{V_1, \ldots, V_n\}$ for the certain answer. It is easy to show, that if all views are closed, then the certain answer is always empty. We will consider the case where all views are open.

The rewriting for the certain answer can now be obtained according to the following recipe: Let L be a subset of the atoms in the bodies of the view-definitions $\varphi_1, \ldots, \varphi_n$, such that L unifies with the set of atoms in $body(Q)$. Let the *mgu* that achieves this unification be θ. If θ does not equate any view variable x to any other view variable or constant, unless x appears in the head of its view, then do the following. Choose a subset $\{\varphi_{i_1}, \ldots, \varphi_{i_k}\}$ of the view queries such that every atom in L occurs in some $body(\varphi_{i_j})$, every $body(\varphi_{i_j})$ contains at least one atom in L. and such that the query

$$Q' = head(Q)\theta \leftarrow head(\varphi_{i_1})\theta, \ldots, head(\varphi_{i_k})\theta,$$

is safe (i. e. all variables in the head occur in the body). Consider now the queries Q' and

$$Q'' = head(Q)\theta \leftarrow body(\varphi_{i_1})\theta, \ldots, body(\varphi_{i_k})\theta.$$

It is clear that if Q' exists then the query Q'' is the composition of the query Q' with the view-definition $\langle \varphi_{i_1}, \ldots, \varphi_{i_k} \rangle$. Let $rew_1(Q)$ be the union of all all possible Q':s thus constructed.[4]

The query $rew_1(Q)$ is the rewriting of Q achieving the desired result. Formally we have:

Theorem 10. *Let $\mathcal{S}$ be a collection of open sources where $ext(\mathcal{S}) = \boldsymbol{v}$, and let Q be a query on $sch(\mathcal{S})$. Then*

$$rew_1(Q)(\boldsymbol{v}) = Q_*(\mathcal{S}).$$

[4] There is a finite number (modulo renaming) of such Q':s since each one is based on a subset L of the atoms in the bodies of the view queries, and there is a unique (up to renaming) *mgu* for each pair of atom sets. Furthermore, there is a finite number of ways of choosing a "covering" subset of the view-queries given L.

For a simple example, suppose the global database schema has one relation *Flights(Source, Destination, Airline)*. We have two open sources: The flights by Air Canada, and the ones by Canadian Airlines. The definitions of these views are $AC(x, y) \leftarrow Flights(x, y, \mathrm{ac})$, and $CP(x', y') \leftarrow Flights(x', y', \mathrm{cp})$. Suppose we would like to find all connections from Toronto to Jerusalem with one change of planes. The query Q is[5]

$$ans(\mathrm{toronto}, u, v, u, \mathrm{jerusalem}, w) \leftarrow Flights(\mathrm{toronto}, u, v), Flights(u, \mathrm{jerusalem}, w).$$

Now $rew_1(Q)$ will be the union of four queries, of which for instance

$$ans(\mathrm{toronto}, u, \mathrm{ac}, u, \mathrm{jerusalem}, \mathrm{cp}) \leftarrow AC(\mathrm{toronto}, u), CP(u, \mathrm{jerusalem}),$$

is obtained through the *mgu* $\{x/\mathrm{toronto}, y/u, v/\mathrm{ac}, x'/u, y'/\mathrm{jerusalem}, w/\mathrm{cp}\}$. The union of the four queries gives the certain answer: indeed, the only connections we can for sure assert are those by Air Canada – Air Canada, Air Canada – Canadian Airlines, Canadian Airlines – Air Canada, and Canadian Airlines – Canadian Airlines.

We have seen that $rew_1(Q)$ is a "sound and complete" rewriting of a query Q with respect to the certain answer on a collection of open views. It turns out that $rew_1(Q)$ is equivalent to a rewriting technique used in the Information Manifold project [LM*95]. Formally, let $manifold(Q)$ be the union of all conjunctive queries P with no more atoms in their bodies than Q and no constants other than those appearing in Q or the view definitions, such that the expansion P' of P through $\{\phi_i\}$ is contained in Q. Then we have:

Theorem 11. $rew_1(Q) \equiv manifold(Q)$.

For algorithmic purposes note the following: Suppose that there are n sources, and that the size of the body of the view-definition φ_i is b_i, with $B = max\{b_i : i \in [1, n]\}$, $b = \Sigma_{i=1}^{n} b_i$, and the global query has a body consisting of m atoms. A straightforward implementation of the Information Manifold rewriting technique runs in time $\mathcal{O}(n^{n \times B} \times n^m)$, whereas a straightforward implementation of our technique based on unification requires time $\mathcal{O}(b^m)$. We see that our unification-based rewriting technique is polynomial in the source descriptions and the the number of sources and exponential in the size of the global query. The information manifold rewriting technique is polynomial in the number of sources and exponential both in the size of the source descriptions and the global query. However, the following result –indicating that there is little chance of getting rid of the global query size in the exponent– follows easily from [LM*95].

Theorem 12. *Let $\Phi = \{\langle\varphi_1, \{\mathrm{open}\}\rangle, \ldots, \langle\varphi_n, \{\mathrm{open}\}\rangle\}$, be a set of source descriptions. Given a conjunctive query Q using only the relation names in the bodies of the φ_i:s, the problem of whether there exists a certainty rewriting of Q with respect to Φ is NP-complete.*

[5] In the example toronto, jerusalem, ac, and cp are constants.

Possibility rewritings

For the possible answer we note that it is straightforward to show that if all sources in $\mathcal{S}$ are open, then $Q^*(\mathcal{S})$ is always infinite. Here we treat the case where all views are closed.

For this consider each $body(\varphi_i)$ of the view definitions φ_i. For each containment mapping h_{i_j} from $body(\varphi_i)$ to $body(Q)$ set Q'_{i_j} to $head(Q) \leftarrow h_{i_j}(head(\varphi_i))$. If there are no containment mappings h_{i_j}, then no Q'_{i_j} exists. Suppose wlog that we found containment mappings for $i \in [1, k], k \leq n$, and for each such i we found m_i different containment mappings. We then set $rew_2(Q) =$

$$\begin{aligned} head(Q) \leftarrow\ & h_{1_1}(head(\varphi_1)), \ldots, h_{1_{m_1}}(head(\varphi_1)), \\ & \ldots, \\ & h_{k_1}(head(\varphi_k)), \ldots, h_{k_{m_k}}(head(\varphi_k)) \end{aligned}$$

provided this query is safe. If it isn't safe or if there are no Q'_{i_j}:s, then $rew_2(Q)$ is undefined.

We can now show that the rewriting has the desired property.

Theorem 13. *Let $\mathcal{S}$ be a collection of closed sources where $ext(\mathcal{S}) = \boldsymbol{v}$, and let Q be a query on $sch(\mathcal{S})$. Then*

$$rew_2(Q)(\boldsymbol{v}) = Q^*(\mathcal{S}).$$

For an example, let us return to the World Cup Soccer Tournament from the introduction. Suppose we only have the closed source S_{Qual} available. Recall that this source was defined by

$$S_{Qual}(x) \leftarrow Team(x, y).$$

Suppose we would like to have a list of all possible matches in the first round. Our query Q is then

$$ans(u, v) \leftarrow Team(u, w), Team(v, w).$$

We are now able to compute the possible answer Q^* by evaluating $rew_2(Q) =$

$$ans(u, v) \leftarrow S_{Qual}(u), S_{Qual}(v)$$

on the closed source. The rewriting was obtained as $ans(u, v) \leftarrow h_1(S_{Qual}(x))$, $h_2(S_{Qual}(y))$, where $h_1 = \{x/u, y/w\}$ and $h_2 = \{x/v, y/w\}$. Clearly, the pair of teams actually playing against each other in the first round will be found only among the elements of the Cartesian product $S_{Qual} \times S_{Qual}$.

As in Theorem 12 it turns out that the rewriting is intractable in the global query size.

Theorem 14. *Let $\Phi = \{\langle \varphi_1, \{\text{closed}\}\rangle, \ldots, \langle \varphi_n, \{\text{closed}\}\rangle\}$, be a set of source descriptions. Given a conjunctive query Q using only the relation names in the bodies of the φ_i:s, the problem of whether there exists a possibility rewriting of Q with respect to Φ is NP-complete.*

5 Related Work

Many works deal with finite representations of infinite sets of databases, both in the constraint database literature [KKR90,vdM93] and in the literature on uncertain and indefinite databases [AKG91,Gra91,IL84,Men84]. We will not go into details; for surveys of these fields see [Kan95] and [vdM98].

The work of Motro [Mot97] is close in spirit and motivation to ours; however, Motro assumes the existence of a "real world" global database instead of modeling the uncertainty in the sources by associating a set of databases with them; this makes it difficult to give precise meaning to his ideas.

In a recent paper [AD98], Abiteboul and Duschka study the special case in which all sources are sound, or all sources are sound and complete, but do not formulate the more general problem. On the other hand, they consider more general view definitions and queries than conjunctive ones.

The idea of sources as views on the global schema goes back to the GMAP project [TSI94]. It was subsequently popularized by the Information Manifold project [LRO96,LM*95], which introduced the query algorithm that we characterize in this paper. In a related paper, Levy [Lev96] studies sources that are complete over some subset of their domain and show how to use them to obtain exact answers.

Finally, we mention two special cases: Suppose the global database consists of one relation, and all sources are defined by projections. Then if all sources are open the certain answer corresponds to the *weak instance window function* [MRW86] with no dependencies, and if all sources are clopen, then the only possible global database is the notorious *universal relation* [Ull82].

Acknowledgments

We thank NSERC, IRIS and Concordia FRDP for their support, and Anthony Tomasic for pointing out Motro's work [Mot97].

References

[AD98] Abiteboul S., O. M. Duschka: Complexity of Answering Queries Using Materialized Views. *Proc. 9th Annual ACM Symp. on the Theory of Computing*, pp. 254–263.

[AHV95] Abiteboul S., R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, Reading Ma. 1995.

[AKG91] Abiteboul S., P. C. Kanellakis, G. Grahne. On the Representation and Querying of Sets of Possible Worlds. *Theoretical Computer Science* **78**:1, 1991, pp. 158–187

[ASU79] Aho A. V., Y. Sagiv, J. D. Ullman. Equivalences Among Relational Expressions. *SIAM J. Comput.* **8**:2, 1979, pp. 218–246

[BV81] Beeri C, M. Y. Vardi. The Implication Problem for Data Dependencies. *Proc. 8th International Colloquium on Automata, Languages and Programming*, pp. 73–85

[CM77] Chandra A. K. and Merlin P. M. Optimal implementation of conjunctive queries in relational databases. *Conference Record of the 9th Annual ACM Symposium on Theory of Computing*, pp. 77–90.

[FKUV86] Fagin R., G. M. Kuper, J. D. Ullman, M. Y. Vardi. Updating Logical Databases. In P. C. Kanellakis and F. Preparata (Editors) *Advances in Computing Research, Vol. 3*, pp. 1–18. JAI Press Inc., Greenwich, CT, 1986.

[Gra91] Grahne G. *The Problem of Incomplete Information in Relational Databases.* Lecture Notes in Computer Science, vol. 554. Springer-Verlag, Berlin 1991.

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

[Kan95] Kanellakis P. C. Constraint Programming and Database Languages: A Tutorial. *Proc. 14th ACM Symp. on Principles of Database Systems*, pp. 46–53.

[KKR90] Kanellakis P. C., G. M. Kuper, P. Z. Revesz. Constraint Query Languages. *Proc. 9th ACM Symp. on Principles of Database Systems*, pp. 299–313.

[Lev96] Levy A. Y. Obtaining Complete Answers from Incomplete Databases. *Proc. 22nd Int'l. Conf. on Very Large Databases*, pp. 402–412.

[LM*95] Levy A. Y., A. O. Mendelzon, Y. Sagiv, D. Srivastava: Answering Queries Using Views. *Proc. 14th ACM Symp. on Principles of Database Systems*, pp. 95–104.

[LRO96] Levy A. Y., A. Rajaraman and J. J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. *Proc. 22nd Int'l. Conf. on Very Large Databases*, pp. 251–262.

[MRW86] Maier D., D. Rozenshtein, D. S. Warren. Window Functions. In P. C. Kanellakis and F. Preparata (Editors) *Advances in Computing Research, Vol. 3*, pp. 213–246, JAI Press Inc., Greenwich, CT, 1986.

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

[Mot97] Motro A. Multiplex: A Formal Model for Multidatabases and Its Implementation. Technical Report ISSE-TR-95-103.

[vdM93] van der Meyden R. Recursively Indefinite Databases, *Theoretical Computer Science* 116(1,2), 1993, pp. 151–194

[vdM98] van der Meyden R. Logical Approaches to Incomplete Information: A Survey. In J. Chomicki and G. Saake (Editors) *Logics for Databases and Information Systems.* Kluwer Academic Publishers, 1998, pp. 307–356

[TSI94] Tsatalos, O.G, M.H. Solomon, Y.E. Ioannidis. The GMAP: A Versatile Tool for Physical Data Independence. *Proc. 20th Int'l. Conf. on Very Large Databases*, pp. 367–378.

[Ull82] Ullman J. D. The U. R. Strikes Back. *Proc. 1st ACM Symp. on Principles of Database Systems*, pp. 10–22.

[Ull97] Ullman J. D. Information Integration Using Logical Views. *Proc. 6th Int'l. Symp. on Database Theory*, pp. 19–40.