Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information and Computation 194 (2004) 79–100
Information and Computation
www.elsevier.com/locate/ic

Towards an algebraic theory of information integration

Gösta Grahne*, Victoria Kiricenko

Department of Computer Science, Concordia University Montreal, Que., Canada H3G 1M8

Received 3 October 2003; revised 23 June 2004
Available online 29 September 2004

Abstract

Information integration systems provide uniform interfaces to varieties of heterogeneous information sources. For query answering in such systems, the current generation of query answering algorithms in local-as-view (source-centric) information integration systems all produce 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. However, this "best obtainable" answer does not include all information that can be extracted from the sources because it does not allow partial information. Neither does the "best obtainable" answer allow for composition of queries, meaning that querying a result of a previous query will not be equivalent to the composition of the two queries. In this paper, we provide a foundation for information integration, based on the algebraic theory of incomplete information. Our framework allows us to define the semantics of partial facts and introduce the notion of the exact answer—that is the answer that includes partial facts. We show that querying under the exact answer semantics is compositional. We also present two methods for actually computing the exact answer. The first method is tableau-based, and it is a generalization of the "inverse-rules" approach. The second, much more efficient method, is a generalization of the rewriting approach, and it is based on partial containment mappings introduced in the paper.

Keywords: Information integration; Incomplete information; Query rewriting

* Corresponding author.
E-mail addresses: grahne@cs.concordia.ca (G. Grahne), kiricen@cs.concordia.ca (V. Kiricenko).

0890-5401/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2004.07.003

1. Introduction

In the mid 80s the first author, together with Serge Abiteboul, interested Paris Kanellakis in incomplete information in relational databases. Paris immediately realized the importance of the problem, and the profoundness of the algebraic foundation laid by Imielinski and Lipski [16] in their landmark paper. Paris also saw that an understanding of the complexity theoretic aspects, in particular data complexity, was lacking. This resulted in the paper [4], which has become the standard reference for the computational complexity of incomplete information.

In this paper, we show that the problem of incomplete information is still highly relevant, due to its application to information integration. By extending the classical framework, we are able to provide a solid algebraic foundation for reasoning about information integration. As an additional benefit, our approach allows an important extension of the capabilities of these applications, namely to provide partial answers to user queries. These partial answers are important, not only because they are more informative than the types of answers considered before, but also because they enable query evaluation to be compositional. Without compositional query semantics, a user would not be able to "search within the result," i.e., pose a second query on the result of a first query, nor would it be possible to have a view mechanism.

To illustrate the problem and to make this discussion more concrete let us consider a simple example. Suppose we want to integrate three sources, let us call them S_1, S_2, and S_3. The sources S_1 and S_3 are Internet movie databases, and source S_2 is a Hollywood gossip website that provides information about the cohabitation of movie stars. A small, but illustrative example of the sources is given below.

Source S_1, an Internet movie database about movie stars.

Actor	Origin	Domicile
Gloria Swanson	US	Sunset Blvd., Hollywood
Eric von Stroheim	Germany	Sunset Blvd., Hollywood
Greta Garbo	Sweden	Manhattan, New York

Source S_2, a Hollywood gossip database about cohabitation of movie stars.

Actor1	Type1	Actor2	Type2
Britt Ekland	Comedienne	Peter Sellers	Comedian
Elisabeth Taylor	Tragedienne	Richard Burton	Tragic hero

Source S_3, an Internet movie database about affiliations of movie stars.

Actor	Affiliation
Gloria Swanson	UA
Eric von Stroheim	UA
Britt Ekland	UA
Peter Sellers	UA
Charlie Chaplin	UA
Greta Garbo	MGM

A plausible interpretation of this source collection is to stipulate the global schema as

`Star(Name, Origin, Type, Domicile)`, that is actor's name, role type, and address; and
`Affiliation(Actor, Studio)`, that is information of the studio of the star.

The relationship between the global schema and the schemas of S_1, S_2, and S_3 could then be expressed by the following definitions:[1]

$$\begin{aligned} S_1(Actor, Origin, Domicile) &\leftarrow Star(Actor, Origin, Type, Domicile). \\ S_2(Actor_1, Type_1, Actor_2, Type_2) &\leftarrow Star(Actor_1, Origin_1, Type_1, Domicile), \\ &\quad\ Star(Actor_2, Origin_2, Type_2, Domicile). \\ S_3(Actor, Studio) &\leftarrow Affiliation(Actor, Studio). \end{aligned}$$

Suppose now the user issues the query

$$\begin{aligned} Q_1(Actor, Origin, Type, Domicile) &\leftarrow Star(Actor, Origin, Type, Domicile), \\ &\quad\ Affiliation(Actor, \text{`}UA\text{'}). \end{aligned}$$

That is, the user is interested in obtaining all available information about stars affiliated with the United Artists studio. In all current generation information integration systems, such as [19,18], the user would be returned an empty answer. This is because these systems do not have a way to get tuples for the subgoal *Star*, and would produce an empty rewriting.

Everybody who has ever queried integrated information, especially in the case of the World Wide Web, knows this situation very well. The next logical action of the user, who wants to get at least partial information, is to try to make his/her query less restrictive. In our example, to get at least some information about the stars affiliated with United Artists, the user has to modify the original query by projecting out some of the attributes of the relation *Star*. Since the user is unaware of the internals or the system, he/she has to try all possible combinations of the attributes and then manually assemble the final answer. From the point of view of the user, the system should take on this burden and compute the answer containing this partial information. This is feasible, since the query could be rewritten as the union of the following unsafe conjunctive queries.

$$\begin{aligned} Q_1(Actor, Origin, Type, X) &\leftarrow S_1(Actor, Origin, Type), S_3(Actor, \text{`}UA\text{'}.) \\ Q_1(Actor_1, X, Type_1, Y) &\leftarrow S_2(Actor_1, Type_1, Actor_2, Type_2), S_3(Actor_1, \text{`}UA\text{'}.) \\ Q_1(Actor_2, X, Type_2, Y) &\leftarrow S_2(Actor_1, Type_1, Actor_2, Type_2), S_3(Actor_2, \text{`}UA\text{'}.) \end{aligned}$$

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 our example as the table below.

[1] Note that we are using the positional version of the relational model in our queries.

Actor	Origin	Type	Domicile
Gloria Swanson	US	⊥	Sunset Blvd., Hollywood
Eric von Stroheim	Germany	⊥	Sunset Blvd., Hollywood
Britt Ekland	⊥	Comedienne	⊥
Peter Sellers	⊥	Comedian	⊥

Producing such partial answers is not without its intricacies. Since the missing values obviously represent nulls of the type "unknown," there is a connection to incomplete information. This connection was first utilized by Abiteboul and Duschka [2], who used the conditional tables of Imielinski and Lipski [16], as extended in [9], as a tool to obtain an effective query evaluation mechanism and complexity theoretic results for information integration. Notably, the complexity results for non-recursive queries in [2] are closely related to those in [4]. Here we see again an example of the importance of incomplete information, and Abiteboul subsequently argues in his PODS 1999 invited talk [1] that the problem of information integration "should be approached with an incomplete information model," and he recalls Imileinski's and Lipski's contribution as "*the* model for incomplete information."

However, in [2], Abiteboul and Duschka only use the *certain answer* aspect of incomplete information and conditional tables. The framework of Imielinski and Lipski is much richer than that, and we shall see below that a notion called the *exact answer* will play a crucial role in extending the theory of information integration to take partial information into account.

First of all, the answer above containing nulls is essentially the exact answer, an alternative semantic to the certain answer. Note that all previous approaches, including [2], explicitly or implicitly use the certain answer. Second, the exact answer allows for composition of queries. One of the important benefits of compositionality of queries is that it allows the user to "search within the results." For example, in a web based setting, the number of tuples in an answer typically is much larger than the user wants. In such a case, to reduce the size of the answer, the user would most likely want to make his/her query more restrictive. This means that an information integration system, in order to be of any practical use, has to provide the user with the ability to search within the answers.

Lets assume that in our example the user is interested in knowing about the cohabitation of MGM stars. In current generation information integration systems the user would have to write the query as

$$\begin{aligned} Q_2(Actor_1, Actor_2) \leftarrow\ & Star(Actor_1, Origin_1, Type_1, Domicile), \\ & Affiliation(Actor_1, \text{'}UA\text{'}), \\ & Star(Actor_2, Origin_2, Type_2, Domicile), \\ & Affiliation(Actor_2, \text{'}UA\text{'}). \end{aligned}$$

Since we are able to provide the user with partial information about the join of the two first (and two last) subgoals as relation Q_1, it is easy to see that Q_2 is equivalent to

$$\begin{aligned} Q_2'(Actor_1, Actor_2) \leftarrow\ & Q_1(Actor_1, Origin_1, Type_1, Domicile), \\ & Q_1(Actor_2, Origin_2, Type_2, Domicile). \end{aligned}$$

From the users point of view it is obvious that he/she should "search within the results," in other words, issue query Q_2'.

Can we use the results of the previous query in the computation of the answer? The answer is yes but only if the system remembers that some of the null values in this result represent the same unknown values. In particular, the unknown domiciles of Britt Ekland and Peter Sellers are the *same* domicile. Intuitively, this is because only source S_2 mentions these actors, and according to the definition of S_2 they would live at the same unknown address. Therefore, the information integration system will now have to account for this fact in some way, for instance by presenting the answer with marked nulls instead.

Actor	Origin	Type	Domicile
Gloria Swanson	US	$\perp_1$	Sunset Blvd., Hollywood
Eric von Stroheim	Germany	$\perp_2$	Sunset Blvd., Hollywood
Britt Ekland	$\perp_3$	Comedienne	$\perp_4$
Peter Sellers	$\perp_5$	Comedian	$\perp_4$

Then we could evaluate Q_2' of the result of Q_1 above, and obtain the table below. Note that Britt Ekland and Peter Sellers occur in the output since they join on the value $\perp_4$ in the fourth column. For an exposition on marked nulls, see [26].

Actor1	Actor2
Gloria Swanson	Eric von Stroheim
Britt Ekland	Peter Sellers

The question that now arises is how these marked nulls should be accounted for.

The rest of this paper is organized as follows. In Section 2, we review the algebraic approach to incomplete information. In Section 3, we describe how querying in information integration systems amounts to querying incomplete databases, producing partial facts. In Section 4, we extend the notion of rewriting to take into account the partial facts, and in Section 5, we give a unification based method for computing the extended rewritings. Section 6, contains the conclusions.

2. Basic definitions and results

2.1. Relational databases

The Relational Model structures information in the form of tables. There are various ways to formalize this notion. The first is a direct adaptation of the mathematical concept of a relation: let the atomic values come from a countably infinite set **dom**.[2] Let k be a positive integer. A k-ary relation is then a finite subset of $\mathbf{dom} \times \mathbf{dom} \times \cdots \times \mathbf{dom}$, k-times, sometimes denoted $\mathbf{dom}^k$. To have names

[2] We assume for simplicity and clarity of exposition that there is only one domain from which all values are drawn.

for relations, such as R, R_1, R_2, etc., we assume a function **db** that when applied to a relation name R gives the *instance* $\mathbf{db}(R)$ of the actual tuples from $\mathbf{dom}^k$. For a relational "schema" $\{R_1, R_2, \ldots, R_n\}$, the database instance consists of an instance $\mathbf{db}(R_i)$ for each relation name R_i, $i \in \{1, \ldots, n\}$. If the schema is understood, we sometimes write simply **db**. Note that there are no column names in this formalization, only positions (first column, second column, etc.).

In this paper, we adopt a variation of this modeling influenced by the logic programming approach to databases. From the domain **dom** and the relation names we build up a (Herbrand) universe consisting of all expressions of the form $R(a_1, a_2, \ldots, a_k)$, where R is a relation name and the a_i's are values in **dom**. Such expressions are called *facts*. A database instance is then simply a finite set of facts, i.e., a finite subset of the universe, such as $\{R_1(a_1, a_3), R_1(a_2, a_3), R_2(a_3, a_4)\}$. Since relation names are part of the facts, we do not need to list facts from different relations separately.

For more information on the various ways to formalize the relational model, see [3].

A *query* is an expression that defines a function from all databases to (usually) a single relation. In this paper we focus on a simple but the most practical class of queries, namely the conjunctive queries. For this we need the concept of an *atom*. An atom is like a fact, except that we allow *variables*, taken from an infinite set **var**, as well as constants. For instance, $R(a, X)$ is an atom, whereas $R(a, b)$ is a fact.[3] The variables in the atoms are placeholders, and they stand for any domain value. Variables will be denoted by uppercase letters, such as X and Y.

A *conjunctive query* φ is an expression of the form

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

where $body(\varphi)$ is a set of atoms over relation names in the database schema, and $head(\varphi)$ is an atom over an answer relation name not used in the database. We assume that all variables occurring in $head(\varphi)$ also occur in $body(\varphi)$, i.e., that the query φ is *safe*.

A conjunctive query φ can be applied to a database **db** resulting in a set of facts

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

A *valuation* σ, is formally a finite partial mapping from $\mathbf{var} \cup \mathbf{dom}$ to **dom** that is the identity on **dom**.

Example 1. If the database **db** is $\{R_1(a, b), R_1(c, b), R_2(b, d)\}$, and φ is $Q(X) \leftarrow R_1(X, Y), R_2(Y, Z)$, then $\varphi(\mathbf{db}) = \{Q(a), Q(c)\}$. On the other hand, if φ is $Q(X) \leftarrow R_1(a, Y), R_2(Y, X)$, then $\varphi(\mathbf{db}) = \{Q(d)\}$.

2.2. Incomplete databases and tableaux

Here we shall briefly review Imielinski and Lipski [16] algebraic approach to incomplete information. We shall present the approach in our uniform framework.

Usually a database is a complete description of the state of the world modeled by it. This is actually true only if we adopt the *Closed World Assumption*, CWA [24], according to which facts not explicitly stored in the database are false. For example, in the database $\{R_1(a_1, a_3), R_1(a_2, a_3), R_2(a_3, a_4)\}$,

[3] $R(a, b)$ is of course also an atom, since a fact is a special case of an atom.

the fact $R_1(a_3, a_1)$ is false. The closed world assumption is convenient since there is, in general, an infinitude of false facts. The CWA is, however, not appropriate for modeling all situations. We might for instance have a database $\mathbf{db} = \{$*Affiliation(Greta Garbo, MGM)*, *Affiliation(Charlie Chaplin, UA)*$\}$. If we consider the fact *Affiliation(Elisabeth Taylor, UA)* we might not want to draw the conclusion that Elisabeth Taylor is not affiliated with UA, we just do not have evidence recorded about it. We then adopt the *Open World Assumption*, OWA, which regards the database as an *incomplete* description of the world: all facts stored in the database are true, the truth value of any other fact is *unknown*. Semantically this means that the stored database **db** actually is a finite description of a *set of possible worlds*, defined as

$$\{\mathbf{db}' : \mathbf{db} \subseteq \mathbf{db}'\}.$$

Each $\mathbf{db}'$ represents one possible complete state of affairs (or world). Thus for instance the fact *Affiliation(Elisabeth Taylor, UA)* will be true in some possible worlds, and false in others. Facts that are true in *some* possible worlds are called *possible* facts, and facts true in *all* possible worlds are called *certain* facts.

A database under the OWA is the simplest example of an incomplete database. To go a step further, suppose we know that Elisabeth Taylor is affiliated with some studio, but we do not know which one. This could be represented by the atom *Affiliation(Elisabeth Taylor, X)*. Here again, we have an incomplete description of the world. The world could correspond to *Affiliation(Elisabeth Taylor, MGM)*, or to *Affiliation(Elisabeth Taylor, UA)*, and so on.

A very simple (and efficient) way to represent possible worlds is to allow the database to consist of atoms, as opposed to pure facts only. A set of atoms is called a *tableau* [21], also known as *naive tables* [16], and *equality tables* [4]. A tableau is denoted T, as opposed to **db** which stands for a finite set of facts (not atoms).

Example 2. Let $T = \{$*Affiliation(Greta Garbo, MGM), Affiliation(Elisabeth Taylor, X)*$\}$. This tableau contains two atoms, the first of which actually is a fact (a special case of an atom).

A tableau T represents a set of databases. This set is denoted $rep(T)$, and it is defined by

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

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

Example 3. In our tableau $T = \{$*Affiliation(Greta Garbo, MGM)*, *Affiliation(Elisabeth Taylor, X)*$\}$, we will for instance have $\{$*Affiliation(Greta Garbo, MGM)*, *Affiliation(Elisabeth Taylor, MGM)* $\} \in rep(T)$, $\{$*Affiliation(Greta Garbo, MGM), Affiliation(Elisabeth Taylor, UA)*$\} \in rep(T)$, etc. According to $rep(T)$, *Affiliation(Greta Garbo, MGM)* is a certain fact since it is true in all possible worlds in $rep(T)$, and *Affiliation(Elisabeth Taylor, MGM)* is a possible fact since it is true in some possible worlds represented by T.

There are many other ways to represent sets of possible worlds, see [9,16,4], and [10,3] for an overview.

2.3. Querying incomplete databases and tableaux

Now, what does it mean to query an incomplete database? Since an incomplete database is a set, each element representing a possible database, the natural answer to a query is the *set* of answers obtained by applying the query on each possible database. If φ is a query, and $\mathcal{X}$ is a set of possible databases, then

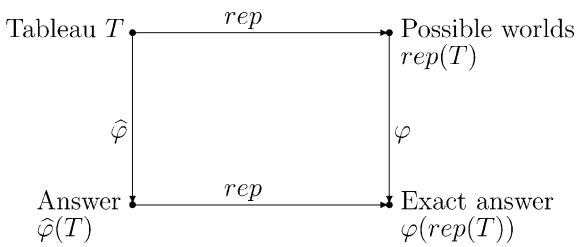
$$\varphi(\mathcal{X}) = \{\varphi(\mathbf{db}) : \mathbf{db} \in \mathcal{X}\}$$

This set of answers is called the *exact answer*.

Suppose now we have a query φ on an incomplete database represented by a tableau T. The exact answer would be $\varphi(rep(T))$. But just as the original database was represented by a tableau T, we would like to represent the exact answer by another tableau U, such that $rep(U) = \varphi(rep(T))$. More practically thinking, given a query φ and a tableau T, we would like to have an algorithm, or evaluation mechanism, call it $\widehat{\varphi}$, applicable on tableau, such that

$$rep(\widehat{\varphi}(T)) = \varphi(rep(T)).$$

This requirement can be illustrated by the following commutative diagram.



Is the requirement in $rep(\widehat{\varphi}(T)) = \varphi(rep(T))$ achievable in general? The answer is yes, but only if we use a representation mechanism more complicated than tableaux (see [16]). If we wish to use tableaux we have to give up the strict equality requirement in the commutative diagram.

Since we cannot represent the exact answer to a query, it is natural to focus on the *certain answer* instead. The certain answer to a query φ consists of those tuples that are in the answer to φ, no matter which possible database we consider. The certain answer is denoted φ_*, and is defined formally as

$$\varphi_*(\mathcal{X}) = \bigcap_{\mathbf{db} \in \mathcal{X}} \varphi(\mathbf{db}).$$

Note that $\varphi_*(\mathcal{X})$ is a relation (a set of facts, no atoms). Then we could require instead that given a query φ and a tableaux T, there is an effectively constructible relation, call it $\varphi_*(T)$, such that

$$\varphi_*(T) = \bigcap_{\mathbf{db} \in rep(T)} \varphi(\mathbf{db}).$$

However, as illustrated in the introduction, if we return the relation $\varphi_*(T)$, then we cannot use the result for subsequent querying. In other words, there are (conjunctive) queries φ and ψ, and tableaux T, such that

$$(\varphi \circ \psi)_*(T) \neq \varphi_*(\psi_*(T)).$$

This also means that we could not have a query evaluation mechanism, found in all real query engines, that would operate in a uniform recursive fashion, i.e. $\varphi \circ \psi$ could be evaluated by first evaluating ψ, and then ϕ.

The reason that the equality $(\varphi \circ \psi)_*(T) = \varphi_*(\psi_*(T))$ fails to hold is that the intermediate result contains some *partial facts* that are discarded by the certain answer of ψ, and that nevertheless contributes to some certain facts found in $\varphi \circ \psi$.

If we like to achieve uniform evaluation and composability of queries we, thus, have to preserve the partial facts in the answer. We shall, therefore, require that given a query φ and a tableau T, there is a tableaux $\widehat{\varphi}(T)$, such that

$$\begin{aligned} &\textit{Truth Preservation} && \cap\, rep(\widehat{\varphi}(T)) = \cap\varphi(rep(T)) \\ &\textit{Query Compositionality} && \widehat{\psi}(\widehat{\varphi}(T)) = \widehat{\psi \circ \varphi}(T) \end{aligned}$$

for all queries ψ applicable at a result of φ.

Lipski [20], in an all but forgotten paper, proved that for monotone query languages, a sufficient condition for truth preservation and compositionality is that

$$\textit{Coinitiality} \quad rep(\widehat{\varphi}(T)) \approx \varphi(rep(T)),$$

where $\approx$-sign means that the two sets have the same $\subseteq$-minimal elements. In the sequel we will use this sufficient condition.

For tableaux, it turns out that if we, for evaluation purposes, treat the variables as pairwise distinct constants, and distinct from all "true" constants, we can apply standard evaluation and obtain a representation of the result that is coinitial with the exact answer. To formally explain the evaluation, we need the concept of a substitution. A *substitution* is a valuation, except that we allow variables to be mapped into variables, not only constants. Thus, a substitution θ is a partial function from **dom** $\cup$ **var** to **dom** $\cup$ **var**, keeping in mind that constants have to be mapped to themselves. Then

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

Theorem 1. $rep(\widehat{\varphi}(T)) \approx \varphi(rep(T))$.

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

Let σ' be an extension of σ 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 φ it follows that $\varphi(\sigma'(T))$ is a $\subseteq$-minimal element in $\varphi(rep(T))$. We have established that $\mathbf{db} \subseteq \varphi(\sigma'(T))$

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

For inclusion in the other direction let **db** be a $\subseteq$-minimal element in $\varphi(rep(T))$. Then there is a valuation σ, such that $\mathbf{db} = \varphi(\sigma(T))$. Let t be an arbitrary tuple in **db**. Then there is a valuation ρ, 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 σ is one-to-one. Then there is an inverse σ^{-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 $\mathbf{db} = \varphi(\sigma(T))$, it is generated by some valuation ρ', and we have $\sigma^{-1}(t') = \sigma^{-1}(\rho'(head(\varphi))) \in \widehat{\varphi}(T)$. Therefore $\mathbf{db} \subseteq \sigma(\widehat{\varphi}(T))$.

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 ω, 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 σ', that is exactly like σ, 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 $\varphi(rep(T))$. □

Theorem 1 means that the diagram on page 8 commutes with respect to $\approx$, i.e., coinitiality. Furthermore, the certain facts of $\varphi(rep(T))$, that is $\cap(\varphi(rep(T)))$, can be obtained from $\widehat{\varphi}(T)$ by retaining only the pure variable-free atoms. Moreover, the resulting tableau $\widehat{\varphi}(T)$ contains all the facts necessary for subsequent query evaluation.

Example 4. Let $T = \{R_1(a,b), R_1(d,X), R_2(b,c), R_2(X,e), R_2(Y,f)\}$, and $\varphi = Q(X,Y,Z) \leftarrow R_1(X,Y), R_2(Y,Z)$. Then $\widehat{\varphi}(T) = \{Q(a,b,c), Q(d,X,e)\}$. The only certain fact in the answer is $Q(a,b,c)$. If $\varphi = Q(X,b) \leftarrow R_1(X,b)$ we have $\widehat{\varphi}(T) = \{Q(a,b)\}$. This fact is also certain.

Theorem 1 was first discovered by Imielinski and Lipski [16], and independently by Vardi [28].

3. Global databases: The meaning of integrated information

Information Integration systems [27,14,15,17] 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 [19], 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.[4]

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.

Well, it turns out that the global database actually is incomplete. In other words, there might be *several* (usually infinitely many) global databases that are consistent with the definition of, and the data in the sources.

Example 5. For a simple example, suppose we have a global relations *Affiliation(Actor, Studio)*, and two sources: Source S_1 has definition $V_1(X, Y) \leftarrow$ *Affiliation (X,Y)* and extension $\mathbf{s}_1 = \{V_1($*Greta Garbo, MGM*$)\}$. Source S_2 has definition $V_2(X) \leftarrow$ *Affiliation (X,Y)* and extension $\mathbf{s}_2 = \{V_2($*Elisabeth Taylor*$)\}$. Then it is natural to think that any database that contains at least the facts *Affiliation(Greta Garbo, MGM)*, and *Affiliation (Elisabeth Taylor,a)*, for some $a \in \mathbf{dom}$, is a possible global database for $\mathcal{S} = \{S_1, S_2\}$.

Formally, for a source collection $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, where each S_i has as definition a conjunctive query ψ_i, and a finite set of facts $\mathbf{s}_i$ over the relation name in $head(\psi_i)$ as extension, we call its set of possible databases $poss(\mathcal{S})$, and define it as

$$poss(\mathcal{S}) = \{\mathbf{db} \ : \ \mathbf{s}_i \subseteq \psi_i(\mathbf{db}), i \in \{1, \ldots, m\}\}.$$

Now the set $poss(\mathcal{S})$ can be conveniently represented by a tableau, denoted $T(\mathcal{S})$, over the relation names in the bodies of the definitions ψ_i, such that $rep(T) = poss(\mathcal{S})$. To construct T we shall follow the approach in [11] (see also [12,13]). We define a function, which, by abuse of notation, we also denote T, from source extensions $\mathbf{s}$, with definition ψ, to tableaux over the relation names in the body of ψ. We also need an auxiliary function *refresh*, that, when applied to a set of atoms, replaces each variable with a distinct unused ("fresh") variable. Given a source $S = (\psi, \mathbf{s})$, we set

$$T(S) = \bigcup_{t \in \mathbf{s}} \{refresh(\sigma(body(\psi))) \ : \ \sigma(head(\psi)) = t\},$$

for some valuation σ.

Example 6. Let S have definition $\psi = V(X, Z) \leftarrow R_1(X, Y), R_2(Y, Z)$ and extension $\mathbf{s} = \{V(a, b), V(c, d)\}$. Then $T(S) = \{R_1(a, Y_1), R_2(Y_1, b), R_1(c, Y_2), R_2(Y_2, d)\}$, where Y_1 and Y_2 are fresh variables.

[4] 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.

When there are several sources in $\mathcal{S}$ we set

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

Example 7. For the source collection $\mathcal{S} = \{S_1, S_2\}$ presented in the Example 5, we have $T(\mathcal{S}) = \{$*Affiliation(Greta Garbo, MGM), Affiliation(Elisabeth Taylor, X)*$\}$.

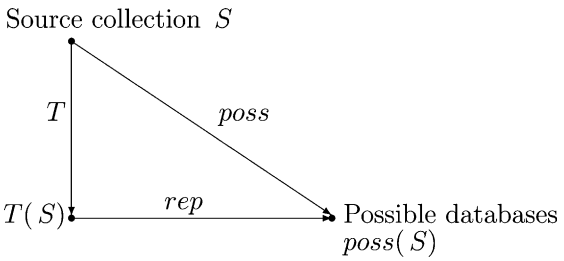
The tableau constructed by the function T has the following desirable property.

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

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

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

Theorem 2 is illustrated in the following diagram.



In definition of the set of possible databases we made a version of the OWA. The facts in the sources are considered to be *sound*, in the sense that they all are generated by facts in the global database. An alternative view would be to state that a source can contain (spurious) facts that do not stem from facts in the global database. In this case the source would be considered *complete*, but not necessarily sound. If the sources are required to be both sound and complete, they are called *exact*.[5] The situations where sources are allowed to be both sound and complete are treated in [2,11]. The presence of complete sources, however, usually increase the computational complexity of decision problems related to integration, see [2,11]. In [22], the authors consider a generalization of the sound/complete view assumptions. Each source is considered to be both *partially sound*, and

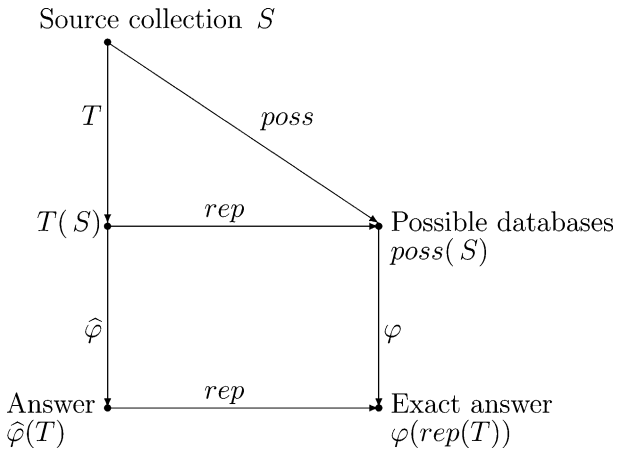
[5] If all sources are complete, the definition of *poss* would be $poss(\mathcal{S}) = \{\mathbf{db} : \mathbf{s}_i \supseteq \psi_i(\mathbf{db}),\ i \in \{1, \ldots, m\}\}$, and for exact sources we would have $poss(\mathcal{S}) = \{\mathbf{db} : \mathbf{s}_i = \psi_i(\mathbf{db}), i \in \{1, \ldots, m\}\}$.

partially complete. A soundness of say 80% would say that the source contains 80% facts that are derived from facts in the global database, and 20% of "spurious" facts. A completeness of 80% says that 80% of the facts derivable from the global database are in the source.

Now that we have seen that a source collection $\mathcal{S}$ actually defines an incomplete database $poss(\mathcal{S})$ that we can represent by a tableau $T(\mathcal{S})$, and that an incomplete database represented as a tableau can be queried using the $\widehat{\varphi}$-evaluation of a query φ, we have a method for computing a representation of the exact answer to a query φ posed on the global database: first compute the tableau $T(\mathcal{S})$, then apply $\widehat{\varphi}$ to obtain $\widehat{\varphi}(T(\mathcal{S}))$. Theorem (1) gave us $rep(\widehat{\varphi}(T)) \approx \varphi(rep(T))$, for any tableau T, and since Theorem (2) gives $rep(T(\mathcal{S})) = poss(\mathcal{S})$, we get

Theorem 3. $rep(\widehat{\varphi}(T(\mathcal{S})) \approx \varphi(poss(\mathcal{S}))$.

In other words, we combine the two previous commutative diagrams, and get the following illustration of Theorem 3.



Example 8. In Examples 5 and 7, if the query φ were the identity, i.e., it asked for the facts in the *Affiliation* -relation, a representation of the exact answer would be the tableau {*Affiliation(Greta Garbo, MGM), Affiliation(Elisabeth Taylor,* X)}. The certain fact in this answer is *Affiliation(Greta Garbo, MGM).*

4. Using rewritings to answer queries on global databases

In the previous section, we saw how to answer a query φ by first constructing the tableau representing all possible global databases. However, computing $T(\mathcal{S})$ might involve a lot of redundant work, since it amounts to constructing the tableau corresponding to the entire source collection $\mathcal{S}$, whereas the global relations that are in the body of φ might be mentioned in only few source definitions. Furthermore, the query might have selections and joins that could be computed directly at the sources.

Let $\{\psi_1, \ldots, \psi_n\}$ be a set of source definitions, and φ a query whose body mentions relation names in $\{body(\psi_i)\}_i$. A *rewriting* of a query φ, with respect to $\{\psi_1, \ldots, \psi_n\}$, is a query χ with the same head-relation name as φ, and body-relation names in $\{head(\psi_i)\}_i$.

The idea of a rewriting is that it can be evaluated on source extensions. It appears that rewritings were first proposed for a restricted setting in [29]. They were generalized in [18,19]. The correctness criteria for rewritings were defined in terms of a certain containment between the rewriting and the original query.

Example 9. Let $\mathcal{S} = \{S_1, S_2\}$, with $V_1(X,Z) \leftarrow R_1(X,Y)$, $R_2(Y,Z)$, and $V_2(X,Z) \leftarrow R_1(X,Y)$, $R_3(Y,Z)$ as definitions. Let φ be the query $Q(X,Y) \leftarrow R_1(X,Y)$. According to the correctness criteria in [19], the desired rewriting φ' would be the union of $Q(X,Y) \leftarrow V_1(X,Y)$, and $Q(X,Y) \leftarrow V_2(X,Y)$.

The correctness criteria did not however state what the answer produced by the rewriting meant. It was subsequently shown in [8,11] that the rewritings actually produced the *certain answer* $\cap(\varphi(poss(\mathcal{S})))$. This is easy to see in Example 9. Source S_1 contains facts of R_1 that join with a fact in R_2, and S_2 contains facts of R_1 that join with a fact in R_3. Since the sources provide us no information about R_2 and R_3, there is a possible database where R_2 and R_3 are empty. Therefore, the facts of R_1 that are in every possible database is the union of the facts in S_1 and S_2, which is exactly what the "correct" rewriting would produce in Example 9.

The first algorithms for constructing rewritings essentially explored the whole (finite) solution space to find all χ's. Later more efficient algorithms were developed. These include the set-covering based algorithm in [11], its more detailed implementation MiniCon [23], the CoreCover algorithm [6], and the algorithm [5] taking arithmetic comparisons into account. Other extensions of the basic technique are surveyed in [14,15]. The cases where the queries defining the sources, as well as the user query, are allowed to be more general than conjunctive ones are analyzed in [2].

However, perhaps since the relationship between information integration and incomplete information had not been clearly articulated, there were no algorithms for computing the exact answer. To see the difference of the certain and exact answers in information integration applications, consider the following example.

Example 10. Suppose that the global schema contains two relations, *Affiliation(Actor, Studio)*, and *Star(Name, Type)*. A fact *Affiliation(Greta Garbo, MGM)* means that Greta Garbo is affiliated with the MGM studio, and a fact such as *Star(Elisabeth Taylor, Tragedienne)* means that Elisabeth Taylor is a Tragedienne. We have two sources, S_1 and S_2, with definitions. $V_1(X,Y) \leftarrow$ *Affiliation*(X,Y), *Star*(X,Z), and $V_2(X,Z) \leftarrow$ *Affiliation*(X,Y), *Star*(X,Z). The user issues the query $\varphi = Q(X,Y,Z) \leftarrow$ *Affiliation*(X,Y), *Star*(X,Z). Using the correctness criteria in [19], we get an empty rewriting, and thus an empty query result for the user. However, suppose the extension $\mathbf{s}_1 = \{V_1($*Greta Garbo, MGM*$)\}$, and $\mathbf{s}_2 = \{V_2($*Elisabeth Taylor, Tragedienne*$)\}$. Then $T(\{S_1,S_2\}) = \{$*Affiliation(Greta Garbo, MGM), Affiliation(Elisabeth Taylor,* X_1*), Star(Elisabeth Taylor, Tragedienne), Star(Greta Garbo,*X_2*)*$\}$, and $\widehat{\varphi}(T(\{S_1,S_2\})) = \{Q($*Greta Garbo, MGM,* $Y_1), Q($*Elisabeth Taylor,* Y_2*, Tragedienne*$)\}$. This exact answer tells the user that Greta Garbo is affiliated with MGM and is an actress of an unknown type, and that Elisabeth Taylor is a Tragedienne and is affiliated with an unknown studio. This is

all the information we can deduce based on the incomplete information about the global relations provided by the sources.

We now proceed as follows. In the remainder of this section we first generalize the notion of query containment, which enables comparisons between different reformulations of queries, to *p-containment*. Next we define *p-rewritings* based on this more general containment. Then we extend the standard query evaluation so that given a p-rewriting, it computes the exact answer. In Section 5, we provide a unification-based method for actually producing the p-rewritings.

4.1. P-containment and p-containment mappings

We can broaden the classical notion of query containment as follows.

Let φ_1 and φ_2 be conjunctive queries. A query φ_1 is said to be *p-contained* in φ_2, denoted $\varphi_1 \subseteq_p \varphi_2$, if and only if there exists a conjunctive query ϕ, where φ_1 is equivalent to $\pi_L(\phi)$ (π is relational projection), for some list L of columns in $head(\phi)$ taken in the original order, such that for all databases **db**, $\phi(\mathbf{db}) \subseteq \varphi_2(\mathbf{db})$. Note that p-containment is a generalization of query containment since L can be the list of all columns in φ_2.

To facilitate testing of p-containment, the classical notion of a containment mapping [7], can be generalized to define p-containment mappings. A *p-containment mapping* from a conjunctive query φ_2 to a conjunctive query φ_1 is a mapping μ, from variables of φ_2 to variables and constants of φ_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 $\mu(Y) = X$.

Example 11. Consider the following queries:

$$\varphi_1 = Q_1(X) \leftarrow R_1(X, Y), R_2(Y, Y), R_3(Y, Z)$$
$$\varphi_2 = Q_2(X', Y') \leftarrow R_1(X', Y'), R_2(Y', Z')$$

There is a p-containment mapping $\mu = \{X'/X, Y'/Y, Z'/Y\}$ from φ_2 to φ_1.

As a consequence, we can now use p-containment mappings to test p-containment of conjunctive queries.

Theorem 4. *A query φ_1 is p-contained in a query φ_2 if and only if there is a p-containment mapping from φ_2 to φ_1.*

Proof. Let μ be a p-containment mapping from φ_2 to φ_1, and let **db** be an arbitrary database. A fact t_1 in $\varphi_1(\mathbf{db})$ is generated by some valuation σ. Then $\sigma \circ \mu$ is a valuation that generates the corresponding fact t_2 in $\varphi_2(\mathbf{db})$. To see that this is indeed so, let $b \in body(\varphi_2)$. Then $\sigma \circ \mu(b) = \sigma(c) \in \mathbf{db}$, for some $c \in body(\varphi_1)$. Therefore, $\sigma \circ \mu(b) \in \mathbf{db}$. 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 **db** be the canonical database that is the "frozen" $body(\varphi_1)$, that is, $\mathbf{db} = \rho(body(\varphi_1))$, where ρ is an injective valuation (the "freezing mapping"). By the definition of p-containment there exist a conjunctive query ϕ, such that $\phi(\mathbf{db}) \subseteq \varphi_2(\mathbf{db})$ and $\varphi_1 = \pi_L(\phi)$ for some ordered list L of columns in $head(\phi)$. Obviously, $\varphi_1(\mathbf{db})$ contains a fact t_1, which is the "frozen" $head(\varphi_1)$. Since $\varphi_1 = \pi_L(\phi)$ there must be a fact t_2 in $\phi(\mathbf{db})$, such that $\pi_L(t_2) = t_1$. Since $\phi(\mathbf{db}) \subseteq \varphi_2(\mathbf{db})$, we have $t_2 \in \varphi_2(\mathbf{db})$.

Let σ be a valuation that generates the fact t_2 in $\varphi_2(\mathbf{db})$. Let ρ be the "freezing" mapping, which also is a valuation that generates the fact t_1 in $\varphi_1(\mathbf{db})$. Then $\rho^{-1} \circ \sigma$ is a p-containment mapping from φ_2 to φ_1.

To see that this is indeed so note two things. First, that each atom $b \in body(\varphi_2)$ is mapped by σ to some fact in **db**, which is a frozen version of some atom $c \in body(\varphi_1)$, 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 σ to constants in the fact t_1, which is the frozen $head(\varphi_1)$, so that all of the head variables in φ_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 φ_2 to φ_1. □

4.2. P-rewritings

We will now extend the classical notion of rewriting [18,27] to *p-rewriting*.

Think of a set of source definitions $\{\psi_1, \ldots, \psi_n\}$ for a class of source collections. Let χ be a query over relation names V_i in $\{head(\psi_i)\}_i$. Then the *expansion* of χ, denoted $\chi^{\exp}$, is defined only if, for each $V_i(.)$ in $body(\chi)$, there is a containment mapping μ from $head(\psi_i)$ to $V_i(.)$. In this case $\chi^{\exp}$ is obtained from χ by replacing all the $V_i(.)$'s in $body(\chi)$ with $\mu(body(\psi_i))$. Existential variables in $\mu(body(\psi_i))$ are replaced by fresh variables when constructing $\chi^{\exp}$.

A query χ is a *contained rewriting* of φ if $\chi^{\exp} \subseteq \varphi$. The query χ is a *p-contained rewriting* of φ if $\chi^{\exp} \subseteq_p \varphi$.

Let χ be a p-contained rewriting of φ, and $\mathcal{S}$ a source collection with definitions $\{\psi_1, \ldots, \psi_n\}$ and extension $\mathbf{s} = \cup\{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$. We define the φ-evaluation of χ on $\mathcal{S}$, denoted $\chi_\varphi(\mathcal{S})$, as follows:

$$\chi_\varphi(\mathcal{S}) = \{\sigma_\mu(head(\varphi)) : \sigma(body(\chi)) \subseteq \mathbf{s}\},$$

where μ is a p-containment mapping from φ to $\chi^{\exp}$ and σ is a valuation. Suppose a head variable X of φ is mapped by μ to a variable in the expansion of the atom $V_i(X_1, \ldots, X_k)$, and $\mu(X)$ is the jth existential variable in the definition ψ_i. Then the extension σ_μ of σ is defined by setting

$$\sigma_\mu(X) = \begin{cases} \sigma(\mu(X)), \text{ if } \mu(X) \text{ occurs in } head(\chi) \\ f_{i,j}(\sigma(X_1), \ldots, \sigma(X_k)) \text{ otherwise.} \end{cases}$$

To define $\widetilde{\varphi}(\mathcal{S})$, the evaluation of conjunctive queries with function symbols in the head, we need an auxiliary function *replace* that, when applied to a set of atoms with function terms, replaces each unique term with a unique variable. Now we set

$$\widetilde{\varphi}(\mathcal{S}) = replace\left(\bigcup\{\chi_\varphi(\mathbf{s}) : \chi^{\exp} \subseteq_p \varphi\}\right),$$

and state the following important result:

Theorem 5. *For all source collections $\mathcal{S}$ with definitions $\{\psi_1, \ldots, \psi_n\}$,*

$$\widetilde{\varphi}(\mathcal{S}) = \widehat{\varphi}(T(\mathcal{S})),$$

up to renaming of the variables.

Proof. Let t be an arbitrary atom in $\widetilde{\varphi}(\mathcal{S})$. Then there was a conjunctive query χ in the union of all p-rewritings of φ and a φ-evaluation of χ, using a valuation σ_μ such that $t = \sigma_\mu(head(\varphi))$, and $\sigma(body(\chi)) \subseteq \mathbf{s}$, where μ is a containment mapping from φ to $\chi^{\exp}$.

Let the extension of $\mathcal{S}$ be $\mathbf{s}$. Since $\sigma(body(\chi)) \subseteq \mathbf{s}$, it means that all atoms in $\sigma(body(\chi^{\exp}))$ are in $T(\mathcal{S})$ (with fresh existential variables). Since μ is a containment mapping from φ to $\chi^{\exp}$, we have $\sigma(\mu(body(\varphi))) \subseteq T(\mathcal{S})$. 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 $head(\chi)$, these have been replaced by function terms in $\sigma_\mu(head(\varphi))$. Now let us consider the case that a given term appears somewhere else in $\widetilde{\varphi}(\mathcal{S})$. There are two cases, either the repeating term appears in the same atom or it appears in a different atom.

Case 1: (See Example 12 further.) The repeating term appears in the same atom. This means that either there were repeating variables in $head(\varphi)$ and it is obvious that in $\widehat{\varphi}(T(\mathcal{S}))$ there are two occurrences of the same variable. Another possibility is that in $\chi^{\exp}$ there were two occurrences of the same existential variable. In this situation in $T(\mathcal{S})$ there would be two occurrences of the same variables and since $\widehat{\varphi}$ always substitutes variables of the query for the variables of the tableau they would also appear in $\widehat{\varphi}(T(\mathcal{S}))$.

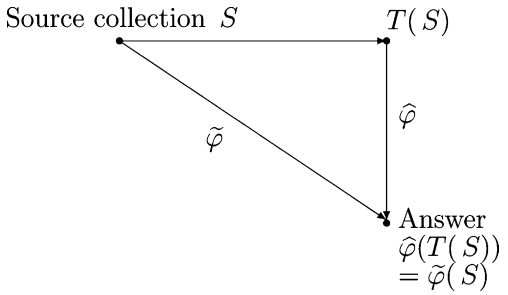
Case 2: (See Example 13 further.) The repeating term appears in some atom t' in $\widetilde{\varphi}(\mathcal{S})$. In this case there are two possibilities, either t' was produced by the φ-evaluation of χ or by the φ-evaluation of some other rewriting χ'. In either situation both terms originated from the same fact in $\mathbf{s}$ and, moreover, from the same existential variable because function terms are subscripted with variable index. It is obvious that there would be two occurrences of the same variables in $T(\mathcal{S})$.

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 θ, such that $\theta(b_i) \in T(\mathcal{S})$, for all $i \in \{1, \ldots, n\}$. But each atom $\theta(b_i)$ is in $T(\mathcal{S})$, which follows from the fact that there is a source S_{i_j}, with definition ψ_{i_j} and extension $\mathbf{s}_{i_j}$, a valuation σ_{i_j}, and a fact $t_{i_j} \in \mathbf{s}_{i_j}$, such that $t_{i_j} = \sigma_{i_j}(head(\psi_{i_j}))$ and $\theta(b_i) \in refresh(\sigma_{i_j}(body(\psi_{i_j})))$. If we take a query ϕ such that $body(\phi) = \sigma_{1_j}(head(\psi_{1_j})), \sigma_{2_j}(head(\psi_{2_j})), \ldots, \sigma_{n_j}(head(\psi_{n_j}))$, and $head(\phi) = (\sigma_{1_j} \cup \sigma_{2_j} \cup \cdots \cup \sigma_{n_j})(head(\varphi))$, we have a p-containment mapping (namely θ), from φ to $\chi^{\exp}$. Consequently χ will be an element in the union of queries that $\widetilde{\varphi}$ considers, and obviously χ generates the fact t when applied to $\mathbf{s}$. Now let us consider the case that a given variable appears somewhere else in $\widehat{\varphi}(T(\mathcal{S}))$. Again there are two cases to consider, either the repeating variable appears in the same atom or it appears in a different atom. In both cases, the same variables are in $\widehat{\varphi}(T(\mathcal{S}))$ because there were the same variables in $T(\mathcal{S})$—remember that $\widehat{\varphi}$ always substitutes variables from the query for the variables from the tableau. The same variables could appear in the tableau only if they come from inversion of the same fact and, moreover, from the same existential variable in the query defining the view. And since this is the case it is obvious that in the corresponding atoms in the $\widetilde{\varphi}$-evaluation on $\mathcal{S}$ there will be the same terms in the corresponding positions. □

Example 12. For example, let the source collection be $\mathcal{S} = \{S_1, S_2\}$, where $\psi_1 = V_1(X_1, X_3) \leftarrow R_1(X_1, X_2), R_2(X_2, X_3)$, $\mathbf{s}_1 = \{V_1(a, c)\}$, $\psi_2 = V_2(X_1) \leftarrow R_3(X_1, X_2)$, and $\mathbf{s}_2 = \{V_2(c)\}$. Let the query φ be $Q(X, Y, Y, Z, W) \leftarrow R_1(X, Y), R_2(Y, Z), R_3(Z, W)$. In this case there is one (minimal) p-contained rewriting of φ, namely $\chi = Q(X, Y) \leftarrow V_1(X, Y), V_2(Y)$. Applying χ_φ gives us the atom $Q(a, f_{1,2}(a), f_{1,2}(a), c, f_{2,2}(c))$, and applying the *replace* function yields the answer $\{Q(a, Y, Y, c, W)\}$. Had we used the tableau method instead, we would have gotten $T(\mathcal{S}) = \{R_1(a, Y), R_2(Y, c), R_3(c, W)\}$. Then applying $\widetilde{\varphi}$ to $T(\mathcal{S})$ would have given $\{Q(a, Y, Y, c, W)\}$.

Example 13. For an additional example, consider the following source collection: $\mathcal{S} = \{S_1, S_2, S_3\}$, where $\psi_1 = V_1(X_1, X_3) \leftarrow R_1(X_1, X_2, X_3), \mathbf{s}_1 = \{V_1(a, b)\}, \psi_2 = V_2(X_1, X_2) \leftarrow R_2(X_1, X_2), \mathbf{s}_2 = \{V_2(c, a)\}$, $\psi_3 = V_3(X_2) \leftarrow R_2(X_1, X_2)$, and $\mathbf{s}_3 = \{V_3(a)\}$. Let the query φ be $Q(X, Y, Z, W) \leftarrow R_2(X, Y), R_1(Y, Z, W)$. The two (minimal) p-contained rewritings of φ are χ_1, which is $Q(X, Y, W) \leftarrow V_2(X, Y), V_1(Y, W)$ and χ_2, which is $Q(Y, W) \leftarrow V_3(Y), V_1(Y, W)$. Applying $(\chi_1)_\varphi$ and $(\chi_2)_\varphi$ gives us two atoms $Q(c, a, f_{1,2}(a, b), b)$ and $Q(f_{3,1}(a), a, f_{1,2}(a, b), b)$, and applying the *replace* function yields the answer $\{Q(c, a, Z, b), Q(X, a, Z, b)\}$. Had we constructed the tableau for this source collection first, we would have gotten $T(\mathcal{S}) = \{R_1(a, Z, b), R_2(c, a), R_2(X, a)\}$. Then applying $\widehat{\varphi}$ to $T(\mathcal{S})$ would have given $\{Q(c, a, Z, b), Q(X, a, Z, b)\}$.

Theorem 5 can be illustrated by the following diagram.



Note that since $\widetilde{\varphi}(\mathcal{S})$ computes a tableau that is equivalent to $\widehat{\varphi}(T(\mathcal{S}))$ the result of this computation can be used for subsequent querying.

5. Unification-based rewriting

The definition of a p-rewriting in the previous section is purely declarative. We now provide a constructive method for producing a p-rewriting given a query φ, and the description $\{\psi_1, \ldots, \psi_n\}$ for a class of source collections.

A *unifier* for two atoms b_1 and b_2 is a substitution θ such that $\theta(b_1) = \theta(b_2)$. A substitution θ is *more general* than a substitution ζ if for some substitution ζ', $\zeta = \theta \circ \zeta'$. A *most general unifier* for a and b is a unifier θ such that, for each unifier ζ of a and b, θ is more general than ζ.

Let α be a subset of the atoms in $\{body(\psi_i)\}_i$, such that α unifies with the set of atoms in $body(\varphi)$. 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 $\{\psi_{i_1}, \ldots, \psi_{i_k}\}$ of the view definitions, such that every atom in α occurs in some $body(\psi_{i_j})$ and every $body(\psi_{i_j})$ contains at least one atom in α. Now, define χ as

$$\pi_L(\theta(head(\varphi)) \leftarrow \theta(head(\psi_{i_1})), \ldots, \theta(head(\psi_{i_k})),$$

where L is a list of all variables in $\theta(head(\varphi))$ that are also in $\theta(head(\psi_{i_1})), \ldots, \theta(head(\psi_{i_k}))$.

The output, $rew(\varphi)$, of the algorithm is the union of all possible χ's thus constructed. Note that there is a finite number (modulo renaming) of such χ's since each one is based on a subset α 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 the "covering" subset α of the view atoms.

Theorem 6. *$rew(\varphi)$ is equivalent to the union of all p-contained rewritings of φ.*

Proof. To prove that $rew(\varphi)$ contains *only* semantically correct rewritings we need to show that for each $\chi \in rew(\varphi)$, there is a p-containment mapping from φ to $\chi^{\exp}$. Let χ be an arbitrary element in $rew(\varphi)$ and let θ be the *mgu* that was used to produce χ. Consider the expansion of χ

$$\chi^{\exp} = \pi_L(head(\theta(\varphi)) \leftarrow \theta(body(\psi_{i_1})), \ldots, \theta(body(\psi_{i_k})).$$

It is obvious that θ also gives a p-containment mapping from φ to $\chi^{\exp}$. Thus χ is p-contained in φ.

To prove that the method computes *all* semantically correct rewritings we will show that for each p-rewriting χ of a given conjunctive query φ, there will be a query χ' in $rew(\varphi)$ such that $\chi \subseteq \chi'$.

Suppose χ is of the form

$$head(\chi) \leftarrow V_{i_1}(.), \ldots, V_{i_k}(.),$$

where each $V_{i_j}(.)$ comes from $head(\psi_{i_j})$. Without loss of generality we assume that no variable is unnecessarily projected out from $head(\chi)$.

Since χ is a semantically correct rewriting, we know that there is a p-containment mapping μ from φ to $\chi^{\exp}$. It now follows that there is a subset α of the atoms in the bodies of $\{\psi_{i_1}, \ldots, \psi_{i_k}\}$, such that μ is an unifier for α and $body(\varphi)$. Suppose that μ is actually the most general unifier. Then the query χ' obtained as $\pi_L(head(\varphi)) \leftarrow \theta(head(\psi_{i_1})), \ldots, \theta(head(\psi_{i_k}))$, where L is a list of variables $X \in head(\theta(\varphi))$, such that $X \in \theta(head(\psi_{i_1})), \ldots, \theta(head(\psi_{i_k}))$, will be in $rew(\varphi)$. Obviously, the identity function will be a containment mapping from χ' to χ.

Suppose then that θ is not the most general unifier for α and $body(\varphi)$. Then there will in $rew(\varphi)$ be a query χ', such that the body of $\chi^{\exp}$ is a more specific instance of the body of the expansion of χ'. This guarantees that there will be a containment mapping from χ' to χ.

The claim of the theorem now follows from the characterization of equivalence of unions of conjunctive queries by Sagiv and Yannakakis [25]. □

Example 14. In Example 10 the unification method would produce as rewriting the union of $Q(X,Y) \leftarrow V_1(X,Y)$ and $Q(X,Z) \leftarrow V_2(X,Z)$. When using the $\widetilde{\varphi}$-evaluation on this rewriting applied to the sources in Example 10 would produce the result $\{Q(\textit{Greta Garbo, MGM}, Y_1), Q(\textit{Elisabeth Taylor, } Y_2\textit{, Tragedienne})\}$.

Obviously it would be desirable to compute a rewriting that encodes the containment mappings μ needed in the $\widetilde{\varphi}$ evaluation in the rewriting directly, rather than recompute the μ's at evaluation time. To do that we can extend our unification based method as follows. Everything up to producing a rewriting χ remains the same but for every rewriting χ, where

$$body(\chi) = \theta(head(\psi_{i_1})), \ldots, \theta(head(\psi_{i_k})),$$

we set $head(\chi) = \eta(\theta(head(\varphi)))$, where η is defined as follows. Suppose a variable X of φ is unified by θ with a variable in an atom in $body(\psi_{i_1})$, $\theta(X)$ is the jth distinguished variable in the $body(\psi_{i_1})$, and existential variables in ψ_{i_1} are $X_1, \ldots, X_k$. Then the

$$\eta(X) = \begin{cases} \theta(X), \text{ if } \theta(X) \text{ occurs in } head(\psi_{i_j}) \\ f_{i,j}(\theta(X_1), \ldots, \theta(X_k)) \text{ otherwise.} \end{cases}$$

The union of all rewritings produced by our extended method can then be evaluated on the source collection in the usual manner and *replace* function can be applied to the resulting set.

The following lemma directly follows from Theorem 6 and the construction of skolemized rewriting by our extended unification based method.

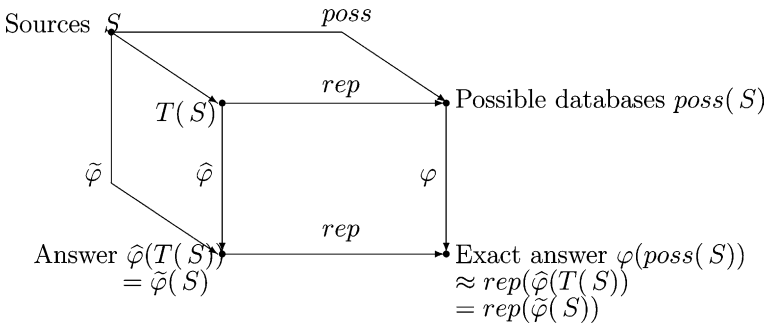
Lemma 1. *The union of all rewritings produced by the extended unification based method relative to a query φ is equivalent to $rew(\varphi)$.*

Proof. It is obvious that the union of all rewritings produced by our extended unification based method contains one skolemized rewriting for each p-contained rewriting produced by the original method. The claim follows from the Theorem 6 and the fact that function terms are inserted by our extended method in the same way as by $\widetilde{\varphi}$. □

Example 15. In Example 14, the extended unification method would produce as rewriting the union of the queries $Q(X, Y, f_{1,1}(X,Y)) \leftarrow V_1(X,Y)$ and $Q(X, f_{2,1}(X,Z), Z) \leftarrow V_2(X,Z)$. The extended evaluation of the rewriting on the source collection in Example 10 will indeed produce $\{Q(GretaGarbo, MGM, Y_1), Q(Elisabeth\ Taylor, Y_2, Tragedienne)\}$.

6. Conclusion

Query answering in Information Integration systems can now be summarized by combining the previous commutative diagrams into the following.



We note that the computational complexity of query evaluation does not change when moving from certain to exact answers. This means that the basic complexity results, such as those in [4,2] carry over to our more general theory. It does not mean that the theory does not open up any new complexity questions, they will be the topic of future papers.

References

[1] S. Abiteboul, On views and XML, in: Proceedings of the 18th Annual ACM Symposium Principles of Databases (PODS '98), Philadelphia, Pennsylvania 1999, pp. 1–9.
[2] S. Abiteboul, O.M. Duschka, Complexity of answering queries using materialized views, in: Proceedings of the 17th Annual ACM Symposium Principles of Databases (PODS '98), Seattle, Washington, 1998, pp. 254–263.
[3] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, Reading, MA, 1995.
[4] S. Abiteboul, P.C. Kanellakis, G. Grahne, On the representation and querying of sets of possible worlds, J. TCS 78 (1) (1991) 158–187.
[5] F. Afrati, C. Li, P. Mitra, Answering queries using views with arithmetic comparisons, in: Proceedings of the 21st ACM Symposium on Principles of Database Systems (PODS '02), Madison, Wisconsin, 2002, pp. 209–220.
[6] F. Afrati, C. Li, J.D. Ullman, Generating efficient plansfor queries using views, in: Proceedings of the ACM SIGMOD International Conference on Managementof Data (SIGMOD '01), Dallas, Texas, 2001.
[7] A.K. Chandra, P.M. Merlin, Optimal implementation of conjunctive queries, in: Proceedings of the ACM SIGACT Symposium on the Theory of Computing (STOC '77), 1977, pp. 77–90.
[8] O.M. Dushka, M.R. Genesereth,Query planning with disjunctive sources, in: Proceedings of the AAAI Workshop on AIand Information Integration, Madison, Wisconsin, 1998.
[9] G. Grahne, Dependency satisfaction in databases with incomplete information, in: Proceedings of the 10th International Conference on Very Large Databases (VLDB '84), Singapore, 1984, pp. 37–45.
[10] G. Grahne, The problem of incomplete information in relational databases, in: Lecture Notes in Computer Science, vol. 554, Springer-Verlag, Berlin, 1991.
[11] G. Grahne, A.O. Mendelzon, Tableau techniques for querying information sources through global schemas, in: Proceedings of the 7th International Conference on Database Theory (ICDT '99), Jerusalem, Israel, pp. 332–347.
[12] G. Grahne, V. Kiricenko, Obtaining more answers from information integration systems, in: Proceedings of the Fifth International Workshop on the Web and Databases (WebDB '02), Madison, Wisconsin, 2002, pp. 67–76.
[13] G. Grahne, V. Kiricenko, Partial answers in information integration systems, in: Proceedings of the 5th ACM CIKM International Workshop on Web Information and Data Management (WIDM '03), New Orleans, Louisiana, 2003, pp. 98–101.
[14] A.Y. Halevy, Theory of answering queries using views, SIGMOD Rec. 29 (4) (2000) 40–47.
[15] A.Y. Halevy, Answering queries using views: a survey, VLDB J. 10 (4) (2001) 270–294.
[16] T. Imielinski, W. Lipski Jr., Incomplete information in relational databases, J. ACM 31 (4) (1984) 761–791.
[17] M. Lenzerini, Data integration: a theoretical perspective, Invited tutorial, in: Proceedings of the 21st ACM Symposium on Principles of Database Systems (PODS '02), Madison, Wisconsin, 2002, pp. 233–246.

[18] A.Y. Levy, A.O. Mendelzon, Y. Sagiv, D. Srivastava, Answering queries using views, in: Proceedings of the 14th ACM Symposium on Principles of Database Systems (PODS '95), San Jose, California, 1995, pp. 95–104.
[19] A.Y. Levy, A. Rajaraman, J.J. Ordille, Querying heterogeneous information sources using source descriptions, in: Proceedings of the 22nd International Conference on Very Large Databases (VLDB '96), Mumbai (Bombay), India, 1996, pp. 251–262.
[20] W. Lipski Jr., On relational algebra with marked nulls, in: Proceedings of the Third ACM Symposium on Principles of Database Systems (PODS '84), Waterloo, Ont., 1984, pp. 201–203.
[21] A.O. Mendelzon, Database states and their tableaux, ACM Trans. Databases Syst. 9 (2) (1984) 264–282.
[22] A.O. Mendelzon, G. Mihaila, Querying partially sound and complete data sources, in: Proceedings of the 20th ACM Symposium on Principles of Database Systems (PODS '01), Santa Barbara, California, 2001, pp. 162–170.
[23] R., Pottinger, A.Y. Levy, A scalable algorithm for answering queries using views, in: Proceedings of the 26th International Conference on Very Large Databases (VLDB '00), Cairo, Egypt, 2000, pp. 484–495.
[24] R. Reiter, On closed world databases, in: H. Gallaire, J. Minker (Eds.), Logic and Databases, Plenum Press, New York, 1978, pp. 56–76.
[25] Y. Sagiv, M. Yannakakis, Equivalence among relational expressions with the union and difference operators, J. ACM 27 (4) (1980) 633–655.
[26] J.D. Ullman, in: Principles of Database and Knowledge-Base Systems, vol. II, Computer Science Press, Rockville, MD, 1989.
[27] J.D. Ullman, Information integration using logical views, in: Proceedings of the 6th International Conference on Database Theory (ICDT '97), Delphi, Greece, 1997, pp. 19–40.
[28] M.Y. Vardi, Querying logical databases, J. Comput. Syst. Sci. 33 (1986) 142–160.
[29] H.Z. Yang, P.A. Larson, Query transformation for PSJ Queries, in: Proceedings of the 13th International Conference on Very Large Data Bases (VLDB '87), Brighton, England, 1987, pp. 245–254.