# Incompleteness in Information Integration

Evgeny Kharlamov

Supervised by: Werner Nutt

Free University of Bozen-Bolzano
39100, Piazza Domenicani, 3
Bozen-Bolzano, Italy

{kharlamov, nutt}@inf.unibz.it

## ABSTRACT

Information integration is becoming a critical problem for both businesses and individuals. The data, especially the one that comes from the Web, is naturally incomplete, that is, some data values may be unknown or lost because of communication problems, hidden due to privacy considerations. At the same time research in (virtual) integration in the community focusses on null-free sources and addresses limited forms of incompleteness only. In our work we aim to extend current results on virtual integration by considering various forms of incompleteness at the level of the sources, the integrated database and the queries (we call this *Incomplete Information Integration*, or III). More specifically, we aim to extend current query answering techniques for *local-*, and *global-as-view* integration to integration of tables with SQL nulls, Codd tables, etc. We also aim to consider incomplete answers as a natural extension of the classical approach. Our main research issues are (i) semantics of III, (ii) semantics of query answering in III, (iii) complexity of query answering, and (iv) algorithms (possibly approximate) to compute the answers.

## Keywords

Information integration, incomplete information, certain answers, query rewriting.

## 1. INTRODUCTION

Information integration (II) is becoming a critical problem for both businesses and individuals [14]. The amounts of data on the Web, in life-science labs, governmental institutions is sky-rocketing and new sources and types of information appear constantly. Processing these data is infeasible without integration approaches. The most adequate integration scenario for these applications is *virtual* or *mediator based* integration.

In virtual integration a user is provided with a query interface over several heterogeneous distributed data sources. The interface allows the user to pose queries to the sources in terms of a global schema. The system (underlying the interface) is responsible for (i) linking the global schema with the sources by means of mappings and (ii) performing query answering. In this integration the sources can be incomplete in two different ways: (i) they may contain null values and (ii) they may be incomplete wrt to the mappings and the global schema.

Most of the research on virtual integration is done under the assumption that sources are null-free and addresses the second type of incompleteness only. In reality nulls in the sources are common, for example, SQL nulls in RDBs or missing values in relational wrappers on top of non-relational Web data.

Therefore we consider the completeness assumption for the sources as a serious limitation of the current approaches to virtual integration. In this work we aim to overcome this limitation by assuming that the sources may have nulls (that is, may be incomplete and modeled using either SQL nulls, or *Codd tables*, or *v-tables*, or *c-tables*) and extend techniques for query answering over different types of virtual integration (*local-*, *global-as-view*) under this assumption. We call this type of integration *Incomplete Information Integration*, or III for short.

We also consider incomplete answers as a natural extension of classical answers that should reflect (i) the incompleteness coming from the sources and (ii) the fact that most approaches to II define this as a task of querying incomplete databases (DBs).

The following **research questions** arise in the work:

- What is the semantics of III?
- What is the semantics of query answering over III?
- What kind of answers are appropriate for III?
- How to compute answers?
- How to integrate integration settings?

The structure of the paper is as follows. In Section 2 we review models of incomplete information: SQL nulls, Codd tables, v- and c-tables and world-set decompositions. In Section 3 we review virtual II and query answering. In Section 4 we present the research questions of our interest. In Sections 5, 6 and 7 are our preliminary results on integration of tables with SQL nulls, semantics for integration of



PVLDB '08, August 23-28, 2008, Auckland, New Zealand
Copyright 2008 VLDB Endowment, ACM 978-1-60558-306-8/08/08

incomplete sources and integration of Codd and v-tables, respectively. In Section 8 we discuss incomplete answers and in Section 9 we conclude and present our next steps.

# 2. INCOMPLETE INFORMATION

The relational data model is based on the idea that the data in a DB is complete, it is usually referred to as *Closed World Assumption*. For example, if we want to set up a DB of employees in a company, we assume that we know all the employees and we know all the relevant facts about them.

In reality the data frequently happens to be incomplete. For example, if we create a DB from a white pages book of Bozen citizens, we face that (i) records about people that just moved in the city are missing and (ii) some facts, like phone numbers of some citizens, may be missing because people rejected to make them public. In this case we can only guess which are the missing records and phone numbers. Each guess gives one completion of the data (that can be represented as a relational DB) and all the possible guesses give a set of data completions and a set of corresponding DBs. This understanding of incomplete data as a set of relational DBs corresponding to the data completions is the most common in the database community and usually referred to as *Open World Assumption*. In the following we will use the term *incomplete DB* as a synonym for incomplete data or a set of relational DBs.

Constraints over relational schemas give rise to another type of incompleteness, namely, when a set of constrains $C$ enforces the presence of certain tuples in all relational instances over a schema $\Sigma$ but in a given instance $D$ over $\Sigma$ the tuples are missing. In this case $D$ is incomplete wrt to $C$. For example, $C$ may consist of tuple generating dependencies. To illustrate the phenomenon let $D$ be a DB about employees with two tables `Manager` and `Secretary` and $C$ consists of a tuple-generating dependency that says that for every manager there is at least one secretary. If $D$ contains a record about a manager *Bob*, but no information about his secretaries, then $D$ is incomplete. Since there are several ways to satisfy this dependency by extending $D$ to $D'$ with records about *Bob*'s secretaries, we have a set of $D'$s that is an incomplete DB that corresponds to $D$ and $C$. We notice that this type of incompleteness is a specific kind of inconsistency in databases, namely, when the inconsistency can be resolved by *only* adding (possibly infinitely many) tuples to the database.

## 2.1 Managing Incomplete Information

As a consequence of the above discussion, in order to manage incomplete data one need to understand:

- how to *store sets of databases* and
- how to *query sets of databases*, that is, what is the answer to a query and how it can be obtained (efficiently).

### 2.1.1 Storing Incomplete Information

Even simple examples of incomplete data comprise too many corresponding DBs to store them explicitly (even infinitely many). Therefore, in order to store sets of DBs, the community proposed several approaches or *systems* how to *represent* the sets in a compact way. The most well known *representation systems* include *Codd tables*, *v-tables* and *c-tables* [16, 3] as well as *world-set decompositions* [4], and *probabilistic DBs* [20].

Codd tables are tables that may contain a symbol $\perp$, which is called *null* and is a kind of existential variable that serves as a syntactic substitute for unknown values. For example, one can use $\perp$ as a value for the unknown phone number of Mary in a table `Person` as follows:

| Name | Tel |
|---|---|
| Mary | $\perp$ |

The semantics of a Codd table is, intuitively, the set of all instances obtained from the Codd table by (i) substituting each occurrence of $\perp$ with a constant and (ii) extending the result with any extra tuples. Formally, the semantics $Rep(T)$ of a Codd table $T$ with the schema $\Sigma$ is defined as

$$Rep(T) = \{D \in I(\Sigma) \mid \text{there exists } \sigma \text{ such that } \sigma T \in D\},$$

where $I(\Sigma)$ is a set of all possible instances over $\Sigma$ and $\sigma$ is a mapping that is the identity on constants and maps each occurrence of $\perp$ in $T$ to some constant.

Codd tables have limitations in modeling incomplete information when extra knowledge about unknown attribute values is to be represented. For example, one may need to model the situation when Mary and John live together and have the same unknown phone number, which may be different from the unknown phone number of a third person, Bob. In order to overcome such problems *v-tables* were proposed.

In v-tables nulls are *labeled* with sub-scripts and nulls with the same label denote the same unknown attribute value. The following v-table illustrates our example.

| Name | Tel |
|---|---|
| Mary | $\perp_1$ |
| John | $\perp_1$ |
| Bob | $\perp_2$ |

It turns out that v-tables can only say that some labeled nulls are the same but not to express more interesting constraints on them, for instance, that Mary's age is between 30 and 35 and that she is younger than John and older than Bob. These constraints can be captured by c-tables which are essentially v-tables equipped with an extra column to store (local) boolean constraints on the labeled nulls and constants. The following c-table illustrates our example.

| Name | Tel | Age | Con |
|---|---|---|---|
| Mary | $\perp_1$ | $\perp_3$ | $\perp_3 \in [30, 35]$ |
| John | $\perp_1$ | $\perp_4$ | $\perp_3 < \perp_4$ |
| Bob | $\perp_2$ | $\perp_5$ | $\perp_5 < \perp_3$ |

In the following we refer to sets of Codd tables as *Codd multi-tables*. Analogously we define *v-* and *c-multi-tables*.

The semantics $Rep(T)$ of v-tables and c-tables $T$ is similar to the one of Codd tables with the difference that in v-tables one should substitute all occurrences of the same labeled null with the same constant and in c-tables after the substitutions one should delete tuples from the obtained table where components of the tuple do not satisfy the constrains associated to the tuple [16].

World-set decompositions (WSDs) were proposed in [4] to represent any finite sets of relational databases. WSDs are equivalent to restricted c-tables, namely, c-tables with finite ranges for the labeled nulls.

SQL nulls, usually denoted as $\omega$, are a standard engineering approach to deal with incomplete information in

RDBMS. Semantically SQL nulls are constants but, while evaluating SQL expressions, SQL interpreters should treat them differently from other constants. For instance, the semantics of Boolean expressions that involve $\omega$ is based on a three-valued logic, join operation on $\omega$ is forbidden [7]. The main difference between SQL nulls and nulls in previously considered formalisms is that $\omega$ is a constant. Consequently, a table with $\omega$ is not a representation of a set of tables, but a single relational table with constants.

We notice that the fact that one cannot join relations on $\omega$ will have a significant impact on the integration of instances that contain $\omega$ (see Section 5).

In the following we discuss how to query incomplete DBs.

### 2.1.2 Semantics of Queries

In the case of complete DBs queries are mappings from DBs to DBs. What should a query $q$ output when the input, say $\mathcal{I}$, is an incomplete DB, that is a set of relational databases? There are two approaches to this question proposed by the community.

- *Incomplete DB as output.* In this case, the answer set of $q$ over $\mathcal{I}$, written $q(\mathcal{I})$, is obtained (conceptually) by applying $q$ to each element $D$ of $\mathcal{I}$ separately, that is,

  $$q(\mathcal{I}) := \{q(D) \mid D \in \mathcal{I}\},$$

  which is a set of relational databases, that is, an incomplete DB.

- *Complete DB as output.* In this case, the answer set of $q$ over $\mathcal{I}$ is a set of tuples. One distinguishes between *certain* and *possible* answers. A tuple of constants is a possible answer if it is returned by $q$ over some $D \in \mathcal{I}$ and it is a certain answer if it is returned by $q$ over every $D \in \mathcal{I}$. Technically, sets of possible and certain answers are defined, respectively, as follows

  $$\mathit{Poss}(q, \mathcal{I}) = \bigcup_{D \in \mathcal{I}} q(D),$$

  $$\mathit{Cert}(q, \mathcal{I}) = \bigcap_{D \in \mathcal{I}} q(D).$$

It is worth noting that that both possible and certain answers can be obtained from $q(\mathcal{I})$ by taking the union $\bigcup q(\mathcal{I})$ or the intersection $\bigcap q(\mathcal{I})$, respectively, of all elements of $q(\mathcal{I})$.

We notice that in information integration the most widely accepted approach is to use certain answers. We now discuss how to compute answers to queries under any of the two semantics.

### 2.1.3 Computing Answers to Queries

In the general case it is obviously infeasible to perform a "naive" query evaluation over an incomplete DB, that is, to query all the DBs in the input set separately. Hence, a natural need is (i) to query a representation of the input in a consistent way, that is, the result of the query should be the same as if the input set was naively queried. If the query result is defined as an incomplete DB, then the next need is (ii) to represent the output in the same representation formalism as the input. Both needs could be illustrated as [2]:

$$\mathit{Rep}(\tilde{q}(T)) = q(\mathit{Rep}(T)), \tag{1}$$

where $\tilde{q}$ is a function that evaluates $q$ over representation $T$ and $\tilde{q}(T)$ is represented in the same formalism as $T$.

In [16] Imielinski and Lipski investigated for which representation systems and which relational algebra operators Equation 1 holds. They proposed techniques for querying the representation systems and showed that Codd tables support projection and selection but do not support projection and join together. V-tables support arbitrary positive queries, that is, projection, positive selection, union, join and renaming of attributes but do not support selection with negative conditions, while c-tables support all of relational algebra. WSDs support projection, product and union [4].

In order to compute certain answers over an incomplete DB $\mathcal{I}$ represented by $T$, one can query $T$ with the relational operators supported by $T$ and then "clean" the resulting representation $T'$ from tuples that contain nulls. For example, if $T$ is a v-multi-table, then certain answers for positive $p$ are tuples from a v-table $T'$ without labeled nulls.

## 3. INFORMATION INTEGRATION

Since the early 1990's, there has been considerable interest in integrating information from heterogeneous data sources. Most of it is in the tradition of Wiederhold's seminal paper, where he envisioned collections of sources being queried via a mediated or "global" schema – as opposed to the "local" schemas of the sources [22]. The global schema is usually denoted as $\Sigma$. To simplify our exposition we assume that there is a single local schema $\mathcal{L}$ which combines the schemas of the relations in the sources. In this framework the question arises how to describe the connections (usually called *mappings*) between the mediated and the source schema. Ullman distinguished two approaches [21], which are called "global-as-view" (GAV) and "local-as-view" (LAV). In GAV the global relations from $\Sigma$ are described in terms of views over relations from $\mathcal{L}$, while in LAV the local relations from $\mathcal{L}$ are described in terms of views over relations from $\Sigma$. Mappings are usually written as:

$$e_1 \colon v_l(\vec{x}, \vec{y}) \rightsquigarrow g(\vec{x}), \qquad\qquad e_2 \colon l(\vec{x}) \rightsquigarrow v_g(\vec{x}, \vec{y}),$$

where the mapping on the left hand side is GAV and the one on the right hand side is LAV, $e_1$, $e_2$ are the names of the mappings, $l$ and $g$ are relations, from the local and global schemas, respectively, and $v_l$ and $v_g$ are views over the local and global schemas, respectively.

The global schema represents a virtual (usually called *global*) DB, the content of which is (not completely) determined by the sources and the mappings, that is, the global DB is an incomplete DB. In order to query sources trough the global schema one can compute the global DB, using the data in the sources and query it directly (this approach is called *materialized integration*) or identify which data in the sources is relevant to a given query and extract this relevant data only (*virtual integration*).

Consider examples of GAV and LAV mappings. Let the local schema consist of

- `Couple`(*Husband*, *Wife*) that lists all married couples,
- `Emp`(*Name*, *Company*, *Tel*) that lists employees with the places of their work and their telephone numbers and
- `Company`(*Name*) that lists all registered companies.

and the global schema consists of `Phone`(*Name*, *Tel*) that lists people with their telephone numbers.

The mappings are in the table below:

| Type | Mapping |
|---|---|
| LAV | $\texttt{Couple}(x, y) \rightsquigarrow \exists z.\texttt{Phone}(x, z), \texttt{Phone}(y, z)$ |
| GAV | $\texttt{Employee}(x, y, z), \texttt{Company}(y) \rightsquigarrow \texttt{Phone}(x, z)$ |

We use Prolog-like notation for the mappings and the views are conjunctive. The LAV mapping describes the local relation `Couple` in terms of a view over the global relation `Phone`. Namely, it "relates" married couples $(x, y)$ and people that have the same phone number $z$ in the phone book. The GAV mapping describes the global relation `Phone` in terms of a view over the local relations `Employee` and `Company`. Namely, it "relates" a person $x$ that works for a registered company $y$ with the phone number $z$ and a person recorded in the phone book. The meaning of "relates" depends on the interpretation of "$\rightsquigarrow$".

Usually "$\rightsquigarrow$" is interpreted as one of the three logical relations "$\rightarrow$", or "$\leftarrow$", or "$\equiv$" [17]. In the first case the mapping says that the global DB contains at least the data from the sources (*sound mapping*). In the second and third it says, respectively, that the global DB contains at most (*complete mapping*) and exactly (*exact mapping*) the data from the sources. If the LAV mapping from the example is sound, the global table `Phones` has at least the phone numbers of all couples from `Couple`.

The classical approach to II defines the semantics of the global DB as an incomplete DB, that is, the global DB is the set of all global instances compatible with the source instance and the mappings. Formally, an *information integration setting* is a pair $\langle I, \mathcal{M}\rangle$, where $I$ is a source instance over $\mathcal{L}$ and $\mathcal{M}$ is a set of mappings [13, 17]. We say that a global instance $J$ over $\Sigma$ is *compatible* with $I$ wrt to a sound LAV mapping $e\colon l(\vec{x}) \rightarrow v_g(\vec{x}, \vec{y})$, where $l \in \mathcal{L}$, if

$$l^I \subseteq v_g(J),$$

where $v_g(J)$ is the extension of the global view $v_g$ over $J$. Analogously, one can define compatibility wrt other types of mappings. The *semantics* of the setting or the *global DB* is defined as a set of instances $\mathcal{G}_{\mathcal{M}}(I)$, or $\mathcal{G}(I)$ when $\mathcal{M}$ is clear from the context, obtained as follows:

$$\mathcal{G}(I) = \{J \mid J \text{ is compatible with } I\}.$$

For example, the semantics of a setting with the instance $I$ that consists of a single relation `Couple` with a single tuple,

$$I = \{\texttt{Couple}(Mary, John)\},$$

and the LAV mapping considered above can be represented by a v-multi-table, that is,

$$\mathcal{G}(I) = \mathit{Rep}(\{\texttt{Phone}(Mary, \bot_1),\ \texttt{Phone}(John, \bot_1)\}).$$

In fact, such a representation by a v-multi-table is possible for arbitrary LAV integration settings (see Theorem 3.1).

The goal of II is to provide a user with a query interface over a set of sources, that is, to enable the user to get answers $q(\langle I, \mathcal{M}\rangle)$ for a global query $q$ from an integration setting $\langle I, \mathcal{M}\rangle$. The classical approach [17] defines $q(\langle I, \mathcal{M}\rangle)$ as the set of certain answers of $q$ over $\mathcal{G}(I)$, that is, as $\mathit{Cert}(q, \mathcal{G}(I))$.

We refer to LAV integration settings where all the views in the mappings are conjunctive as *conjunctive* LAV settings. To summarize, the following theorem holds.

THEOREM 3.1. *Let $\langle I, \mathcal{M}\rangle$ be a sound conjunctive* LAV *integration setting and $q$ be a conjunctive query. Then:*

- *There is a v-multi-table $T$ such that $\mathcal{G}(I) = \mathit{Rep}(T)$,*
- $\mathit{Cert}(q, \mathcal{G}(I)) = \{\vec{t} \mid \vec{t} \in q(T),\ \vec{t}$ *has no labelled nulls*$\}$.

It can be shown that the v-multi-table $T$ from the theorem can be computed by chasing $I$ with $\mathcal{M}$. We denote the function that computes $T$ by chasing $I$ as $\mathtt{T}_{\mathcal{M}}(I)$.

# 4. PROBLEMS TO INVESTIGATE

As we saw in Section 3, in the classical approach the sources are complete, the global DB is incomplete and the answers are certain, that is, complete. Therefore, incompleteness is only in the global DB and it comes from mappings and/or constraints. As a consequence, using the classical approach one cannot perform integration of sources that result from integration, that is, are incomplete databases.

In general, we lacking a uniform theory that combines information integration with the topic of incomplete information by allowing the sources and the answers to be incomplete. We also lacking methods and (efficient) algorithms to perform incomplete information integration. This work aims to propose such a theory and such algorithms.

More precisely, the problems we are interested in are:

- *Semantics of III*: What is the global DB $\mathcal{G}_{\mathcal{M}}(\mathcal{I})$ for incomplete sources $\mathcal{I}$?
- *Semantically correct representations of $\mathcal{G}_{\mathcal{M}}(\mathcal{I})$*: Given that $\mathcal{I}$ is represented by a formalism $A$, what is a good formalism $B$ to represent $\mathcal{G}_{\mathcal{M}}(\mathcal{I})$?
- *Forms of answers for III*: What are meaningful answers (e.g. certain, incomplete) for queries in III?
- *Computation of answers*: How can one retrieve these meaningful answers in III?
- *Integration over integrations*: How can one perform an integration of sources that are themselves results of an integration?

Regarding the related work, there is a body of research done on incompleteness that stems from mappings and constraints [5, 9, 6] but to the best of our knowledge nothing has been done to treat incompleteness originating from sources.

Although a considerable amount of work has been done in the past few years on query processing via rewritings on integration of complete sources by means of LAV mappings (see for example [1, 8, 18]), to the best of our knowledge, there are no attempts to develop rewriting techniques for the integration of incomplete sources. There are also no results on query rewriting in LAV integration when the sources are LAV integration settings themselves.

The question of integrating sources that themselves resulted from integrations have been studied in the context of *data exchange* by Fagin et al. [10]. In this work the authors consider three schemas, namely, a source schema $S$, and two target schemas $T_1$ and $T_2$. They assume to have mappings $\mathcal{M}_{S,T_1}$ from $S$ to $T_1$ and mappings $\mathcal{M}_{T_1,T_2}$ from $T_1$ to $T_2$. The mappings describe how to "ship" the data from instances of $S$ to the schema $T_1$ and from instances of $T_1$ to the schema $T_2$, respectively. The authors investigated the problem of finding mappings $\mathcal{M}_{S,T_2}$ that "ship" the data from instances of $S$ to $T_2$ directly and the result of the "shipping" is the same as if the data was first "shipped"

to $T_1$ by $\mathcal{M}_{S,T_1}$ and then to $T_2$ by $\mathcal{M}_{T_1,T_2}$. In the context of data integration $T_2$ can be seen as a global schema of a data integration setting with a local schema $T_1$, that at the same time is a global schema in another integration setting with the local schema $S$. In this case we have an integration of sources that are integration settings. Fagin et al. considered composition of mappings, while we are interesting in query rewriting, hence, the results of [10] are not directly applicable for our scenario. Moreover, we assume that the sources that are integration settings do not expose the underlying mappings, but only provide a query interface. Consequently, there is no access to $\mathcal{M}_{S,T_1}$ mappings and one cannot perform schema composition and use it for query answering in this kind of data integration.

# 5. SOURCES WITH SQL NULLS

SQL nulls are a form of incompleteness that often occurs in relational DBs. In this section we address integration of sources with SQL nulls. Consider an example of such a source with two tuples:

$$I^\omega = \{\texttt{Couple}(\mathit{John}, \mathit{Mary}),\ \texttt{Couple}(\mathit{Bob}, \omega)\},$$

where $\cdot^\omega$ in $I^\omega$ denotes the fact that an instance $I$ may contain $\omega$.

## 5.1 Integration With Sound LAV Mappings

In this section we consider sound conjunctive LAV mappings only. Consider the mapping $e$:

$$e\colon\ \texttt{Couple}(x, y) \rightarrow \texttt{Phone}(x, z),\ \texttt{Phone}(y, z),\ \texttt{Female}(y).$$

### 5.1.1 Representing and Querying Global Databases

It turns out that in the presence of SQL nulls an integration setting $\langle I^\omega, \mathcal{M}\rangle$ may be *inconsistent*, that is, $\mathcal{G}(I)$ may be empty. Formally, $\langle I^\omega, \mathcal{M}\rangle$ is inconsistent if there is no global instance $J^\omega$ that is compatible with $I^\omega$ wrt $\mathcal{M}$. Our example setting with the source

$$I^\omega = \{\texttt{Couple}(\mathit{John}, \mathit{Mary}),\ \texttt{Couple}(\mathit{Bob}, \omega)\},$$

and the mapping $e$ is inconsistent. The reason is that, due to the soundness of $e$, *Bob*'s unknown wife $\omega$ should appear in the relations `Phone` and `Female` of any compatible global instance $J^\omega$. Since the semantics of SQL nulls does not allow to join on $\omega$, no such $J^\omega$ exists and the setting is inconsistent.

The following theorem shows that consistency is easy to check.

THEOREM 5.1. *For conjunctive integration settings $\langle I^\omega, \mathcal{M}\rangle$ the consistency problem is* LOGSPACE *in $|I^\omega|$ and* LINEAR *in $|\mathcal{M}|$.*

In the following we assume that all settings are conjunctive and consistent.

For consistent integration settings $\langle I^\omega, \mathcal{M}\rangle$, we define the *global DB* as the set of all global instances compatible with the setting. The difference between global DBs for $\langle I^\omega, \mathcal{M}\rangle$ settings and global DBs from Section 3 for $\langle I, \mathcal{M}\rangle$ settings is the notion of compatibility. Since $I^\omega$ may contain $\omega$, one should allow $\omega$ to be in any global instance $J^\omega$ compatible with $I^\omega$ wrt to $\mathcal{M}$. One can allow $J^\omega$ to contain either

- arbitrary occurrences of $\omega$, or
- only occurrences that are justified by the mappings, that is, $\omega$ can be in $J^\omega$ only if there is a mapping in $\mathcal{M}$ that "ships" $\omega$ from $I^\omega$ to $J^\omega$.

We call the first approach *open-world-* or *OW-compatibility* and the second one *closed-world-* or *CW-compatibility*. The names for the approaches are inspired by OWA and CWA solutions for data exchage settings in [19].

For example, let

$$I^\omega = \{\texttt{Couple}(\omega, \mathit{Mary})\},$$

then a CW-compatible instance $J_1^\omega$ wrt $e$ is

$$J_1^\omega = \{\texttt{Phone}(\omega, c_1),\ \texttt{Phone}(\mathit{Mary}, c_1),\ \texttt{Female}(\mathit{Mary}),\ \texttt{Phone}(\mathit{Bob}, c_2)\},$$

where $c_1$ and $c_2$ are constants, and the fact $\texttt{Phone}(\mathit{Bob}, c_2)$ has nothing to do with $I^\omega$ but it is in $J_1^\omega$ to illustrate that arbitrary tuples are allowed to be in the solutions. An OW-compatible instance $J_2^\omega$ is

$$J_2^\omega = \{\texttt{Phone}(\omega, c_1),\ \texttt{Phone}(\mathit{Mary}, c_1),\ \texttt{Female}(\mathit{Mary}),\ \texttt{Phone}(\mathit{Bob}, c_2)\},\ \texttt{Female}(\omega)\},$$

where $\texttt{Female}(\omega)$ is a fact in $J_2^\omega$ that has $\omega$ that is not justified by the mapping $e$.

We say that the set of all instances CW-compatible with $I^\omega$ is the *CW semantics* of $\langle I^\omega, \mathcal{M}\rangle$ and denote it as $\mathcal{G}^{CW}(I^\omega)$. Analogously, we define the *OW semantics* $\mathcal{G}^{OW}(I^\omega)$.

Obviously, the CW semantics is not weaker than the OW one, that is, $\mathcal{G}^{CW}(I^\omega) \subseteq \mathcal{G}^{OW}(I^\omega)$.

The following Theorem 5.2 shows that one can represent the global DB of $\langle I^\omega, \mathcal{M}\rangle$ using v-multi-tables that may contain $\omega$. We first describe two semantics $\mathit{Rep}(T^\omega)$ and $\mathit{Rep}^\omega(T^\omega)$ for v-multi-tables $T^\omega$ that may contain $\omega$. Instances in $\mathit{Rep}(T^\omega)$ are obtained from $T^\omega$ by substituting labeled nulls with constants distinct from $\omega$ and adding tuples without $\omega$. To define $\mathit{Rep}^\omega(T^\omega)$, we refer to labeled nulls occurring only ones in $T^\omega$ as *singleton nulls* and as *joined nulls* otherwise. The $\mathit{Rep}^\omega(T^\omega)$ consists of instances obtained from $T^\omega$ by substituting singleton nulls with constants or $\omega$ and, substituting join nulls with constants distinct from $\omega$, and adding tuples possibly containing $\omega$.

THEOREM 5.2. *Let $\langle I^\omega, \mathcal{M}\rangle$ be a conjunctive sound* LAV *integration setting and $q$ be a conjunctive query. Then there is a v-multi-table $T^\omega$ such that*

$$\mathcal{G}^{CW}(I^\omega) = \mathit{Rep}(T^\omega),\ \textit{and}\ \ \mathcal{G}^{OW}(I^\omega) = \mathit{Rep}^\omega(T^\omega).$$

For the evaluation of conjunctive queries $q$ over v-multi-tables $T^\omega$ we first generalize the definition of $q(T^\omega)$: this evaluation is different from the one in [16] in that a variable occurring more than once in $q$ cannot be mapped to $\omega$. We introduce *safe assignments* as mappings from variables of $q$ to terms in $T^\omega$ that never map variables occuring more than once in $q$ to singleton variables or $\omega$. Now we define $q^s(T^\omega)$ as the subset of $q(T^\omega)$ obtained by admitting only safe assignments. We notice that safe assignments are variants of $J$-homomorphisms in [11].

Our next theorem shows how to compute certain answers.

THEOREM 5.3. *Let $\langle I^\omega, \mathcal{M}\rangle$ be a conjunctive sound* LAV *integration setting and $q$ be a conjunctive query. Then there is a v-multi-table $T^\omega$ such that*

*(i)* $\mathit{Cert}(q, \mathcal{G}^{CW}(I^\omega)) = \{\vec{t} \mid \vec{t} \in q(T^\omega),\ \vec{t}\ \textit{has no nulls}\}$,

*(ii)* $\mathit{Cert}(q, \mathcal{G}^{OW}(I^\omega)) \subseteq \{\vec{t} \mid \vec{t} \in q^s(T^\omega),\ \vec{t}\ \textit{has no nulls}\}$.

We notice that certain answers may contain $\omega$, hence, incompleteness is in a way present in the answers.

Statement (i) of the theorem says that under closed world semantics certain answers for settings with sources that contain $\omega$ *can be computed in the same way* as for the sources without $\omega$. But Statement (ii) says that this is not possible anymore under open world semantics, that is, one can no longer evaluate conjunctive queries over v-tables in a naive way. One should use safe assignments, because naive evaluation may retrieve tuples that are not certain answers.

In [11] the problem of containment of conjunctive queries over DBs with $\omega$ was studied. Examples in that paper can be adopted to show that some cases the inclusion in Statement (ii) of Theorem 5.3 is proper.

It turns out that the v-multi-table $T^\omega$ from Theorem 5.2 can be computed by a chase and its size is polynomial in $|I^\omega| + |\mathcal{M}|$.

For example, using the mapping $e$ from above and

$$I_0^\omega = \{\texttt{Couple}(\omega, \mathit{Mary})\},$$

the chase $\mathtt{T}_e(I_0^\omega)$ is the v-multi-table:

$$\{\texttt{Phone}(\omega, \perp_1),\ \texttt{Phone}(\mathit{Mary}, \perp_1),\ \texttt{Female}(\mathit{Mary})\}.$$

## 5.2 Integration With GAV Mappings

We now consider integration settings with GAV mappings. Note that in the case without SQL nulls, $\mathcal{G}(I)$ can be represented by a v-multi-table without labelled nulls, which can be identified with a relational instance $J_0$ (all compatible databases are supersets of $J_0$). These results can be generalized for the case when $I^\omega$ may contain $\omega$. Below we only list our results without specifying the details:

- Any integration setting with GAV mappings is consistent.
- The global DB $\mathcal{G}(I^\omega)$ can be represented by a relational instance $J^\omega$ that may contain $\omega$ and certain answers can be computed on the representation. The size of $J^\omega$ is polynomial in $|\mathcal{M}|$.
- In order to compute certain answers for monotone queries without constructing the global DB, one can use a slightly modified unfolding.

# 6. MODEL OF INFORMATION INTEGRATION FOR INCOMPLETE SOURCES

In this section we propose a model that generalizes the classical one for complete sources considered in Section 3.

In our case an information integration setting is a pair $\langle \mathcal{I}, \mathcal{M} \rangle$, where $\mathcal{I}$ is a set of instances of $\mathcal{L}$, and $\mathcal{M}$ is a set of mappings. The semantics of the setting is defined as the set $\mathcal{G}_\mathcal{M}(\mathcal{I})$ or $\mathcal{G}(\mathcal{I})$ of instances obtained as follows:

$$\mathcal{G}(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} \mathcal{G}(I).$$

We notice that the semantics for integration of both complete and incomplete sources is an incomplete DB. This assures backward compatibility of the the semantics, when incomplete sources are singleton sets.

# 7. SOURCES WITH V-TABLES

As we saw in Section 3, LAV integration of complete sources admits the representation of global databases by v-multi-tables. This motivates our work on integration of sources that are v-multi-tables, assuming that they are results of other integrations.

In this section an integration setting is a pair $\langle \mathcal{I}, \mathcal{M} \rangle$, where $\mathcal{I}$ can be represented by a v-multi-table $T^\mathcal{I}$, that is, $\mathit{Rep}(T^\mathcal{I}) = \mathcal{I}$. We refer to these integration settings as *v-integration* settings.

## 7.1 Representing and Querying Global DBs

We observe that v-integration settings under the semantics in Section 6 are always consistent for LAV mappings.

It turns out that global databases for v-integration settings with sound conjunctive LAV mappings can be represented by v-multi-tables and one can compute a representation of $\mathcal{G}(\mathcal{I})$ by computing the chase $\mathtt{T}_\mathcal{M}(T^\mathcal{I})$. The chase of v-multi-tables with LAV mappings can be defined analogously to the chase of relational instances as in [9].

THEOREM 7.1. *Let $\langle \mathcal{I}, \mathcal{M} \rangle$ be a v-integration setting with sound conjunctive* LAV *mappings $\mathcal{M}$. Then there is a v-multi-table $T$, such that*

- *$\mathcal{G}(\mathcal{I})$=Rep(T),*
- *$\mathit{Cert}(q, \mathcal{G}(\mathcal{I})) = \{\vec{t} \in q(T) \mid \vec{t}$ has no nulls$\}$, for any conjunctive global query $q$.*

*The chase $\mathtt{T}_\mathcal{M}(T^\mathcal{I})$ is such a $T$ and the size $|\mathtt{T}_\mathcal{M}(T^\mathcal{I})|$ is polynomial in $|\mathcal{I}| + |\mathcal{M}|$.*

Our next observation is that the set of all answers for a positive global query can be represented by querying the representation of the global DB.

PROPOSITION 7.1. *For a conjunctive* LAV *v-integration setting $\langle \mathcal{I}, \mathcal{M} \rangle$ and a positive global query $q$ it holds that*

$$q(\mathcal{G}_\mathcal{I}) = \mathit{Rep}(q(\mathtt{T}_\mathcal{M}(T^\mathcal{I}))).$$

It turns out that techniques, like in [18] for rewriting queries using views, can be used to compute the set of rewritings that return certain answers of conjunctive queries for v-integration settings with conjunctive LAV mappings. A set $Q$ of rewritings of $q$ is complete if any rewriting is contained in a rewriting in $Q$ (see [15]).

THEOREM 7.2. *Let $\langle \mathcal{I}, \mathcal{M} \rangle$ be a v-integration setting with sound conjunctive* LAV *mappings $\mathcal{M}$, $q$ be a positive global query, and $Q$ be a complete set of rewritings of $q$ with the views from $\mathcal{M}$. Then:*

$$\begin{aligned} \mathit{Cert}(q, \mathcal{G}(\mathcal{I})) &= \{\vec{t} \mid \vec{t} \in Q(\mathtt{T}_\mathcal{M}(\mathcal{I})),\ \vec{t} \text{ has no nulls}\} \\ &= \mathit{Cert}(Q, \mathcal{I}). \end{aligned}$$

We notice that similar results to the ones listed in this section hold for integration of Codd tables. The only essential difference in the techniques is that the chase of Codd tables requires first to label each null in the table with a distinct label and then to chase it.

# 8. INCOMPLETE ANSWERS FOR V-TABLES

We claim that incomplete answers are essential for information integration.

Consider for example an integration scenario, that extends the one in Section 5. An instance $\mathcal{I}$ is the following:

$$\{\texttt{Couple}(John, Mary),\ \texttt{Couple}(Bob, \perp_1)\}.$$

The set $\mathcal{M}$ consists of the one mapping:

$$\{\texttt{Couple}(x, y) \rightarrow \texttt{Phone}(x, z),\ \texttt{Phone}(y, z),\ \texttt{Female}(y)\}.$$

The representation $T = \mathtt{T}_{\mathcal{M}}(\mathcal{I})$ is the following:

$$\{\texttt{Phone}(John, \perp_2),\ \texttt{Phone}(Mary, \perp_2),\ \texttt{Female}(Mary),$$
$$\texttt{Phone}(Bob, \perp_3),\ \texttt{Phone}(\perp_1, \perp_3),\ \texttt{Female}(\perp_1)\}.$$

One can pose a query to the integration: "Return the names $A$ of all people that share phone numbers with their wives and the names of the wives $B$". Obviously, from $\mathtt{T}_{\mathcal{M}}(\mathcal{I})$ one obtains that these pairs $(A, B)$ are $(John, Mary)$ and $(Bob, \perp_3)$. It is easy to see that certain answers do not capture this type of answer, since they never contain labeled nulls.

We define the set of *incomplete answers* for a conjunctive global $q$ over a v-integration setting $\langle \mathcal{I}, \mathcal{M} \rangle$ as

$$Inc(q, \mathcal{I}, \mathcal{M}) = q(\mathtt{T}_{\mathcal{M}}(\mathcal{I})).$$

Incomplete answers for conjunctive queries $q(\vec{x})$ can be computed via rewriting techniques, like in [18], if one rewrites the set of queries

$$S_q = \{q(\vec{y}) \mid \vec{y} \subseteq \vec{x}\},$$

that is, by rewriting all the queries obtained from $q$ by dropping some of the output variables. Of course, this leads to exponentially many rewriting problems[1]. We are investigating more efficient approaches.

# 9. CONCLUSIONS

We provided a semantics for integration of incomplete sources. We studied some problems of integration for sources with SQL nulls and v-tables. We also have started to work on incomplete answers. Currently we are working on integration of c-tables. Next steps aimed at extending our results to other types of mappings (that are not sound conjunctive) and to other representation systems, such as WSDs and in the long term probabilistic DBs. We also plan to develop implementation techniques to efficiently compute different types of answers.

# 10. REFERENCES


[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *PODS*, pages 254–265, 1998.

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

[3] S. Abiteboul, P. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. *Theoretical Comput. Sci.*, 78(1):159–187, 1991.

[4] L. Antova, C. Koch, and D. Olteanu. World-set decompositions: Expressiveness and efficient algorithms. In *ICDT*, pages 194–208, 2007.


[1]Similar ideas were investigated in [12] for the integration of complete sources.


[5] A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *PODS*, pages 260–271, 2003.

[6] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *AAAI*, pages 386–391, 2000.

[7] E. F. Codd. *The Relational Model for Database Management, Version 2*. Addison-Wesley, 1990.

[8] O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *PODS*, pages 109–116, 1997.

[9] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *ICDT*, pages 207–224, 2003.

[10] R. Fagin, P. G. Kolaitis, and L. Popa. Composing Schema Mappings: Second-Order Dependencies to the Rescue. In *ACM Transactions on Database Systems*, Vol. 30, No. 4, pages 207–224, 2005.

[11] C. Farré, W. Nutt, E. Teniente, and T. Urpí. Containment of conjunctive queries over databases with null values. In *ICDT*, pages 389–403, 2007.

[12] G. Grahne and V. Kiricenko. Partial answers in information integration systems. In *WIDM*, pages 98–101, 2003.

[13] G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *ICDT*, pages 332–347, 1999.

[14] L. M. Haas. Beauty and the beast: The theory and practice of information integration. In *ICDT*, pages 28–43, 2007.

[15] A. Y. Halevy Answering queries using views: A survey. *J. VLDB*, 10(4):270–294, 2001.

[16] T. Imielinski and W. L. Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[17] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[18] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *PODS*, pages 95–104, 1995.

[19] A. Libkin. Data exchange and incomplete information. In *PODS*, pages 60–69, 2006.

[20] D. Suciu. Managing imprecisions with probabilistic databases. In *TDM*, page 1, 2006.

[21] J. D. Ullman. Information integration using logical views. In *ICDT*, volume 1186, pages 19–40, 1997.

[22] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.