Knowledge Bases and Databases

Part 3: Information Integration

Diego Calvanese

Faculty of Computer Science
Master of Science in Computer Science

A.Y. 2007/2008

Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Overview of Part 3: Information integration

1. Introduction to data integration
   - Basic issues in data integration
   - Logical formalization
2. Query answering in the absence of constraints
   - Global-as-view (GAV) setting
   - Local-as-view (LAV) and GLAV setting
3. Query answering in the presence of constraints
   - The role of integrity constraints
   - Global-as-view (GAV) setting
   - Local-as-view (LAV) and GLAV setting
4. Concluding remarks

Chapter I

Introduction to data integration

Outline

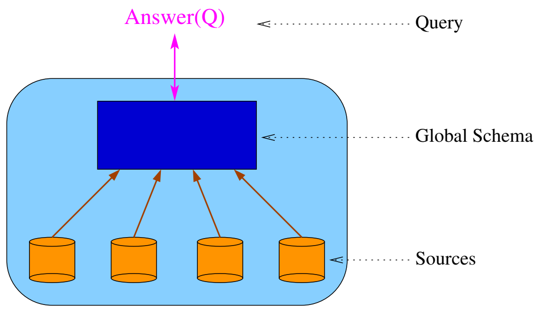
1. Basic issues in data integration
2. Data integration: Logical formalization

Outline

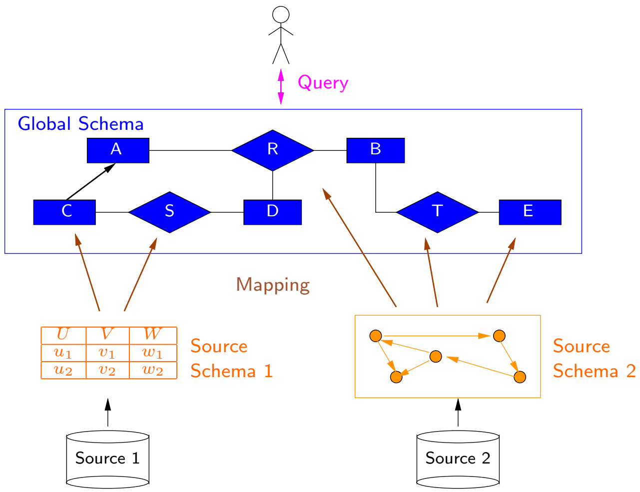
1. Basic issues in data integration
   - The problem of data integration
   - Variants of data integration
   - Problems in data integration
2. Data integration: Logical formalization

What is data integration?

Data integration is the problem of providing unified and transparent access to a collection of data stored in multiple, autonomous, and heterogeneous data sources.



Conceptual architecture of a data integration system



Relevance of data integration

- Growing market
- One of the major challenges for the future of IT
- At least two contexts
  - Intra-organization data integration (e.g., EIS)
  - Inter-organization data integration (e.g., integration on the Web)

Data integration: Available industrial efforts

- Distributed database systems
- Information on demand
- Tools for source wrapping
- Tools based on database federation, e.g., DB2 Information Integrator
- Distributed query optimization

Architectures for integrated access to distributed data

- Distributed databases
  Data sources are homogeneous databases under the control of the distributed database management system.
- Multidatabase or federated databases
  Data sources are autonomous, heterogeneous databases; procedural specification.
- (Mediator-based) data integration
  Access through a global schema mapped to autonomous and heterogeneous data sources; declarative specification.
- Peer-to-peer data integration
  Network of autonomous systems mapped one to each other, without a global schema; declarative specification.

Database federation tools: Characteristics

- Physical transparency, i.e., masking from the user the physical characteristics of the sources
- Heterogeinity, i.e., federating highly diverse types of sources
- Extensibility
- Autonomy of data sources
- Performance, through distributed query optimization

However, current tools do not (directly) support logical (or conceptual) transparency.

Logical transparency

Basic ingredients for achieving logical transparency:

- The global schema (ontology) provides a conceptual view that is independent from the sources.
- The global schema is described with a semantically rich formalism.
- The mappings are the crucial tools for realizing the independence of the global schema from the sources.
- Obviously, the formalism for specifying the mapping is also a crucial point.

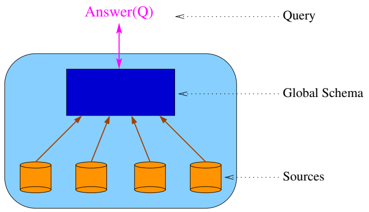
All the above aspects are not appropriately dealt with by current tools. This means that data integration cannot be simply addressed on a tool basis.

Approaches to data integration

- (Mediator-based) data integration . . . *is the topic of this course*
- Data exchange [FKMP05, FKP05]
  - materialization of the global view
  - allows for query answering without accessing the sources
- P2P data integration [HIST03, CDGLR04, CDGL+05]
  - several peers
  - each peer with local and external sources
  - queries over one peer

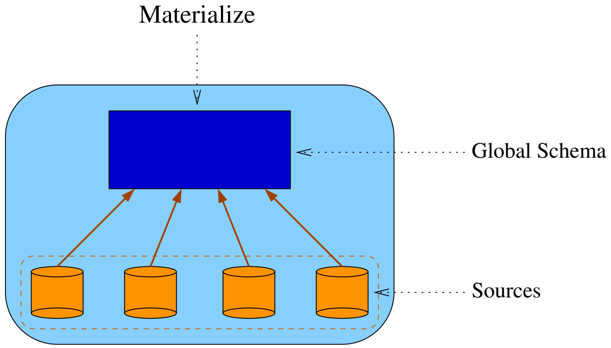
Mediator based data integration

- Queries are expressed over a global schema (a.k.a. mediated schema, enterprise model, . . .).
- Data are stored in a set of sources.
- Wrappers access the sources (provide a view in a uniform data model of the data stored in the sources).
- Mediators combine answers coming from wrappers and/or other mediators.

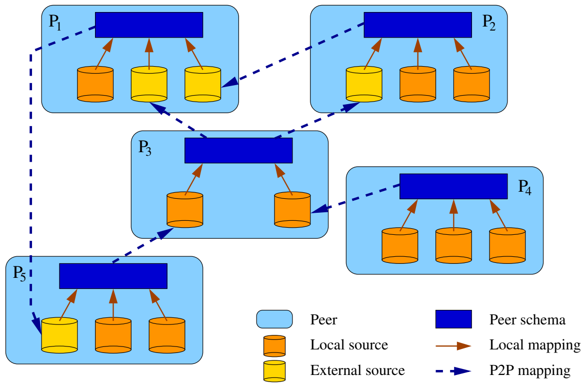


Data exchange

- Materialization of the global schema



Peer-to-peer data integration



Operations: – Answer(Q,P_i) – Materialize(P_i)

Main problems in data integration

1. How to construct the global schema.
2. (Automatic) source wrapping.
3. How to discover mappings between sources and global schema.
4. Limitations in mechanisms for accessing sources.
5. Data extraction, cleaning, and reconciliation.
6. How to process updates expressed on the global schema and/or the sources ("read/write" vs. "read-only" data integration).
7. How to model the global schema, the sources, and the mappings between the two.
8. How to answer queries expressed on the global schema.
9. How to optimize query answering.

The modeling problem

Basic questions:

- How to model the global schema:
  - data model
  - constraints
- How to model the sources:
  - data model (conceptual and logical level)
  - access limitations
  - data values (common vs. different domains)
- How to model the mapping between global schemas and sources.
- How to verify the quality of the modeling process.

A word of caution: Data modeling (in data integration) is an art. Theoretical frameworks can help humans, not replace them.

The querying problem

- A query expressed in terms of the global schema must be reformulated in terms of (a set of) queries over the sources and/or materialized views.
- The computed sub-queries are shipped to the sources, and the results are collected and assembled into the final answer.
- The computed query plan should guarantee:
  - completeness of the obtained answers wrt the semantics;
  - efficiency of the whole query answering process;
  - efficiency in accessing sources.
- This process heavily depends on the approach adopted for modeling the data integration system.

This is the problem that we want to address in this part of the course.

Outline

1. Basic issues in data integration
2. Data integration: Logical formalization
   - Semantics of a data integration system
   - Queries to a data integration system
   - Formalizing the mapping
   - Formalizing GAV data integration systems
   - Formalizing LAV data integration systems
   - Formalizing GLAV data integration systems

Formal framework for data integration

Def.: Data integration system $\mathcal{I}$

A data integration system is a triple $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where:

- $\mathcal{G}$ is the global schema
  i.e., a logical theory over a relational alphabet $\mathcal{A}_\mathcal{G}$.
- $\mathcal{S}$ is the source schema
  i.e., simply a relational alphabet $\mathcal{A}_\mathcal{S}$ disjoint from $\mathcal{A}_\mathcal{G}$.
- $\mathcal{M}$ is the mapping between $\mathcal{S}$ and $\mathcal{G}$.
  We consider different approaches to the specification of mappings.

Semantics of a data integration system

Which are the dbs that satisfy $\mathcal{I}$, i.e., the logical models of $\mathcal{I}$?

- We refer only to dbs over a fixed infinite domain Δ of elements.
- We start from the data present in the sources: these are modeled through a source database $\mathcal{D}$ over Δ (also called source model), fixing the extension of the predicates of $\mathcal{A}_\mathcal{S}$.
- The dbs for $\mathcal{I}$ are logical interpretations for $\mathcal{A}_\mathcal{G}$, called global dbs.

Def.: Semantics of a data integration system

The set of databases for $\mathcal{A}_\mathcal{G}$ that satisfy $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ relative to $\mathcal{D}$ is:

$$Sem_\mathcal{I}(\mathcal{D}) = \{ \mathcal{B} \mid \mathcal{B} \text{ is a global database that is legal wrt } \mathcal{G} \text{ and that satisfies } \mathcal{M} \text{ wrt } \mathcal{D} \}$$

What it means to satisfy $\mathcal{M}$ wrt $\mathcal{D}$ depends on the nature of $\mathcal{M}$.

Queries to a data integration system $\mathcal{I}$

- The domain Δ is fixed, and we do not distinguish an element of Δ from the constant denoting it $\rightsquigarrow$ standard names.
- Queries to $\mathcal{I}$ are relational calculus queries over the alphabet $\mathcal{A}_\mathcal{G}$ of the global schema.
- When "evaluating" q over $\mathcal{I}$, we have to consider that for a given source database $\mathcal{D}$, there may be many global databases $\mathcal{B}$ in $Sem_\mathcal{I}(\mathcal{D})$.
- We consider those answers to q that hold for all global databases in $Sem_\mathcal{I}(\mathcal{D})$ $\rightsquigarrow$ certain answers.

Semantics of queries to $\mathcal{I}$

Def.: Certain answers in a data integration system

Given q, $\mathcal{I}$, and $\mathcal{D}$, the set of certain answers to q wrt $\mathcal{I}$ and $\mathcal{D}$ is

$$cert(q, \mathcal{I}, \mathcal{D}) = \{ (c_1, \ldots, c_n) \in q^\mathcal{B} \mid \text{for all } \mathcal{B} \in Sem_\mathcal{I}(\mathcal{D}) \}$$

- Query answering is logical implication.
- Complexity is measured mainly *wrt the size of the source db $\mathcal{D}$*, i.e., we consider data complexity.
- We consider the problem of deciding whether $\vec{c} \in cert(q, \mathcal{I}, \mathcal{D})$, for a given $\vec{c}$.

Databases with incomplete information, or knowledge bases

- Traditional database: one model of a first-order theory.
  Query answering means evaluating a formula in the model.
- Database with incomplete information, or knowledge base: set of models (specified, for example, as a restricted first-order theory).
  Query answering means computing the tuples that satisfy the query in all the models in the set.

There is a strong connection between query answering in data integration and query answering in databases with incomplete information under constraints (or, query answering in knowledge bases).

Query answering with incomplete information

- [Rei84]: relational setting, databases with incomplete information modeled as a first order theory
- [Var86]: relational setting, complexity of reasoning in closed world databases with unknown values
- Several approaches both from the DB and the KR community
- [vdM98]: survey on logical approaches to incomplete information in databases

The mapping

How is the mapping $\mathcal{M}$ between $\mathcal{S}$ and $\mathcal{G}$ specified?

- Are the sources defined in terms of the global schema?
  Approach called source-centric, or local-as-view, or LAV.
- Is the global schema defined in terms of the sources?
  Approach called global-schema-centric, or global-as-view, or GAV.
- A mixed approach?
  Approach called GLAV.

GAV vs. LAV – Example

Global schema:
movie$(Title, Year, Director)$
european$(Director)$
review$(Title, Critique)$

Source 1:
$\mathsf{r}_1(Title, Year, Director)$ since 1960, european directors

Source 2:
$\mathsf{r}_2(Title, Critique)$ since 1990

Query: Title and critique of movies in 1998
$q(t, r) \leftarrow \exists d.\, \mathsf{movie}(t, 1998, d) \wedge \mathsf{review}(t, r)$, in Datalog notation
$q(t, r) \leftarrow \mathsf{movie}(t, 1998, d),\ \mathsf{review}(t, r)$

Formalization of GAV

In GAV (with sound sources), the mapping $\mathcal{M}$ is a set of assertions:

$$\phi_\mathcal{S} \rightsquigarrow g$$

one for each element g in $\mathcal{A}_\mathcal{G}$, with $\phi_\mathcal{S}$ a query over $\mathcal{S}$ of the arity of g.

Given a source db $\mathcal{D}$, a db $\mathcal{B}$ for $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{D}$ if for each $g \in \mathcal{G}$:

$$\phi_\mathcal{S}^\mathcal{D} \subseteq g^\mathcal{B}$$

In other words, the assertion means: $\forall \vec{x}.\ \phi_\mathcal{S}(\vec{x}) \rightarrow g(\vec{x})$.

Given a source database, $\mathcal{M}$ provides direct information about which data satisfy the elements of the global schema.

Relations in $\mathcal{G}$ are views, and queries are expressed over the views.
Thus, it seems that we can simply evaluate the query over the data satisfying the global relations (as if we had a single db at hand).

GAV – Example

Global schema: movie(*Title*, *Year*, *Director*)
european(*Director*)
review(*Title*, *Critique*)

GAV: to each relation in the global schema, $\mathcal{M}$ associates a view over the sources:

$$q_1(t,y,d) \leftarrow r_1(t,y,d) \rightsquigarrow \text{movie}(t,y,d)$$
$$q_2(d) \leftarrow r_1(t,y,d) \rightsquigarrow \text{european}(d)$$
$$q_3(t,r) \leftarrow r_2(t,r) \rightsquigarrow \text{review}(t,r)$$

Logical formalization:

$$\forall t,y,d.\ r_1(t,y,d) \rightarrow \text{movie}(t,y,d)$$
$$\forall d.\ (\exists t,y.\ r_1(t,y,d)) \rightarrow \text{european}(d)$$
$$\forall t,r.\ r_2(t,r) \rightarrow \text{review}(t,r)$$

GAV – Example of query processing

The query

$$q(t,r) \leftarrow \text{movie}(t,1998,d),\ \text{review}(t,r)$$

is processed by means of unfolding, i.e., by expanding each atom according to its associated definition in $\mathcal{M}$, so as to come up with source relations.

In this case:

$$q(t,r) \leftarrow \text{movie}(t,1998,d),\quad \text{review}(t,r)$$

unfolding $\qquad\downarrow\qquad\qquad\downarrow$

$$q(t,r) \leftarrow r_1(t,1998,d),\quad r_2(t,r)$$

GAV – Example of constraints

Global schema containing constraints:
movie(*Title*, *Year*, *Director*)
european(*Director*)
review(*Title*, *Critique*)
european_movie_60s(*Title*, *Year*, *Director*)

$$\forall t,y,d.\ \text{european\_movie\_60s}(t,y,d) \rightarrow \text{movie}(t,y,d)$$
$$\forall d.\ \exists t,y.\ \text{european\_movie\_60s}(t,y,d) \rightarrow \text{european}(d)$$

GAV mappings:

$$q_1(t,y,d) \leftarrow r_1(t,y,d) \rightsquigarrow \text{european\_movie\_60s}(t,y,d)$$
$$q_2(d) \leftarrow r_1(t,y,d) \rightsquigarrow \text{european}(d)$$
$$q_3(t,r) \leftarrow r_2(t,r) \rightsquigarrow \text{review}(t,r)$$

Formalization of LAV

In LAV (with sound sources), the mapping $\mathcal{M}$ is a set of assertions:

$$s \leadsto \phi_{\mathcal{G}}$$

one for each source element s in $\mathcal{A}_{\mathcal{S}}$, with $\phi_{\mathcal{G}}$ a query over $\mathcal{G}$.

Given a source db $\mathcal{D}$, a db $\mathcal{B}$ for $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{D}$ if for each $s \in \mathcal{S}$:

$$s^{\mathcal{D}} \subseteq \phi_{\mathcal{G}}^{\mathcal{B}}$$

In other words, the assertion means: $\forall \vec{x}.\ s(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$.

The mapping $\mathcal{M}$ and the source database $\mathcal{D}$ do not provide direct information about which data satisfy the global schema.

Sources are views, and we have to answer queries on the basis of the available data in the views.

LAV – Example

Global schema: movie(*Title*, *Year*, *Director*)
european(*Director*)
review(*Title*, *Critique*)

LAV: to each source relation, $\mathcal{M}$ associates a view over the global schema:

$$r_1(t, y, d) \leadsto q_1(t, y, d) \leftarrow \text{movie}(t, y, d),\ \text{european}(d),\ y \geq 1960$$
$$r_2(t, r) \leadsto q_2(t, r) \leftarrow \text{movie}(t, y, d),\ \text{review}(t, r),\ y \geq 1990$$

The query $q(t, r) \leftarrow \text{movie}(t, 1998, d),\ \text{review}(t, r)$ is processed by means of an inference mechanism that aims at re-expressing the atoms of the global schema in terms of atoms at the sources.
In this case:

$$q(t, r) \leftarrow r_2(t, r),\ r_1(t, 1998, d)$$

GAV and LAV – Comparison

GAV: (e.g., Carnot, SIMS, Tsimmis, IBIS, Momis, Mastro, ...)

- Quality depends on how well we have compiled the sources into the global schema through the mapping.
- Whenever a source changes or a new one is added, the global schema needs to be reconsidered.
- Query processing can be based on some sort of unfolding (query answering looks easier – without constraints).

LAV: (e.g., Information Manifold, DWQ, Picsel)

- Quality depends on how well we have characterized the sources.
- High modularity and extensibility (if the global schema is well designed, when a source changes, only its definition is affected).
- Query processing needs reasoning (query answering complex).

Beyond GAV and LAV: GLAV

In GLAV (with sound sources), the mapping $\mathcal{M}$ is a set of assertions:

$$\phi_{\mathcal{S}} \leadsto \phi_{\mathcal{G}}$$

with $\phi_{\mathcal{S}}$ a query over $\mathcal{S}$, and $\phi_{\mathcal{G}}$ a query over $\mathcal{G}$ of the same arity as $\phi_{\mathcal{S}}$.

Given a source db $\mathcal{D}$, a db $\mathcal{B}$ for $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{D}$ if for each $\phi_{\mathcal{S}} \leadsto \phi_{\mathcal{G}}$ in $\mathcal{M}$:

$$\phi_{\mathcal{S}}^{\mathcal{D}} \subseteq \phi_{\mathcal{G}}^{\mathcal{B}}$$

In other words, the assertion means: $\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$.

As in LAV, the mapping $\mathcal{M}$ does not provide direct information about which data satisfy the global schema.

To answer a query q over $\mathcal{G}$, we have to infer how to use $\mathcal{M}$ in order to access the source database $\mathcal{D}$.

GLAV – Example

Global schema: work$(Person, Project)$, area$(Project, Field)$

Source 1: hasjob$(Person, Field)$
Source 2: teaches$(Professor, Course)$, in$(Course, Field)$
Source 3: get$(Researcher, Grant)$, for$(Grant, Project)$

GLAV mapping:

$q^s_1(r, f) \leftarrow$ hasjob(r, f) $\rightsquigarrow$ $q^g_1(r, f) \leftarrow$ work(r, p), area(p, f)
$q^s_2(r, f) \leftarrow$ teaches(r, c), in(c, f) $\rightsquigarrow$ $q^g_2(r, f) \leftarrow$ work(r, p), area(p, f)
$q^s_3(r, p) \leftarrow$ get(r, g), for(g, p) $\rightsquigarrow$ $q^g_3(r, f) \leftarrow$ work$(r, p)\}$

GLAV – A technical observation

In GLAV (with sound sources), the mapping $\mathcal{M}$ is constituted by a set of assertions:

$$\phi_{\mathcal{S}} \rightsquigarrow \phi_{\mathcal{G}}$$

Each such assertion can be rewritten wlog by introducing a new predicate r of the same arity as the two queries and replace the assertion with the following two:

$$\phi_{\mathcal{S}} \rightsquigarrow r \qquad r \rightsquigarrow \phi_{\mathcal{G}}$$

In other words, we replace $\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$
with $\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow r(\vec{x})$ and $\forall \vec{x}.\ r(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$

Note: The new relations r can considered to be part of $\mathcal{G}$ (but should not appear in user queries). Hence, $\phi_{\mathcal{S}} \rightsquigarrow r$ is like a GAV mapping assertion, while $r \rightsquigarrow \phi_{\mathcal{G}}$ is a form of constraint on $\mathcal{G}$.

Chapter II

Query answering in the absence of constraints

Outline

3 Query answering in GAV without constraints
4 Query answering in (G)LAV without constraints

Query answering in different approaches

The problem of query answering comes in different forms, depending on several parameters:

- Global schema
  - without constraints (i.e., empty theory)
  - with constraints
- Mapping
  - GAV
  - LAV (or GLAV)
- Queries
  - user queries
  - queries in the mapping

Conjunctive queries

We recall the following definition:

Def.: A conjunctive query (CQ) is a query of the form

$$q(\vec{x}) \leftarrow \exists \vec{y}.\, r_1(\vec{x}_1, \vec{y}_1) \wedge \cdots \wedge r_m(\vec{x}_m, \vec{y}_m)$$

where

- $\vec{x}$ is the union of the $\vec{x}_i$'s, called the distinguished variables;
- $\vec{y}$ is the union of the $\vec{y}_i$'s, called the non-distinguished variables;
- $r_1, \ldots, r_m$ are relation symbols (not built-in predicates).

Unless otherwise specified, we consider conjunctive queries, both as user queries and as queries in the mapping.

Incompleteness and inconsistency

Query answering heavily depends upon whether incompleteness/inconsistency shows up:

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
no	GAV	yes / no	no
no	(G)LAV	yes	no
yes	GAV	yes	yes
yes	(G)LAV	yes	yes

Outline

3 Query answering in GAV without constraints
- Retrieved global database
- Query answering via unfolding

4 Query answering in (G)LAV without constraints

GAV data integration systems without constraints

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
no	**GAV**	yes / no	no
no	(G)LAV	yes	no
yes	GAV	yes	yes
yes	(G)LAV	yes	yes

GAV – Retrieved global database

Def.: Retrieved global database

Given a source database $\mathcal{D}$, we call retrieved global database, denoted $\mathcal{M}(\mathcal{D})$, the global database obtained by "applying" the queries in the mapping, and "transferring" to the elements of $\mathcal{G}$ the corresponding retrieved tuples.

GAV – Example

Consider $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with

Global schema $\mathcal{G}$: student$(Code, Name, City)$
university$(Code, Name)$
enrolled$(Scode, Ucode)$

Source schema $\mathcal{S}$: relations $s_1(Scode, Sname, City, Age)$, $s_2(Ucode, Uname)$, $s_3(Scode, Ucode)$

Mapping $\mathcal{M}$:

$$q_1(c, n, ci) \leftarrow s_1(c, n, ci, a) \rightsquigarrow \text{student}(c, n, ci)$$
$$q_2(c, n) \leftarrow s_2(c, n) \rightsquigarrow \text{university}(c, n)$$
$$q_3(s, u) \leftarrow s_3(s, u) \rightsquigarrow \text{enrolled}(s, u)$$

GAV – Example of retrieved global database

university

Code	*Name*
AF	bocconi
BN	ucla

student

Code	*Name*	*City*
12	anne	florence
15	bill	oslo

enrolled

Scode	*Ucode*
12	AF
16	BN

$s_1^{\mathcal{D}}$

12	anne	florence	21
15	bill	oslo	24

$s_2^{\mathcal{D}}$

AF	bocconi
BN	ucla

$s_3^{\mathcal{D}}$

12	AF
16	BN

Example of source database $\mathcal{D}$ and corresponding retrieved global database $\mathcal{M}(\mathcal{D})$.

GAV – Minimal model

GAV mapping assertions $\phi_{\mathcal{S}} \rightsquigarrow g$ have the logical form:

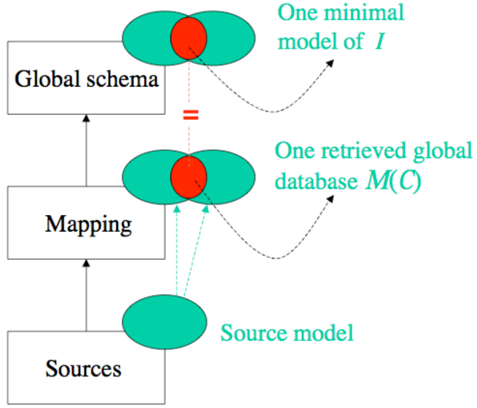
$$\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \to g(\vec{x})$$

where $\phi_{\mathcal{S}}$ is a conjunctive query over the source relations, and g is an element of $\mathcal{G}$.

In general, given a source database $\mathcal{D}$, there are several databases legal wrt $\mathcal{G}$ that satisfy $\mathcal{M}$ wrt $\mathcal{D}$.

However, it is easy to see that $\mathcal{M}(\mathcal{D})$ is the intersection of all such databases, and therefore, is the unique "minimal" model of $\mathcal{I}$.

GAV without constraints



GAV – Query answering via unfolding

The unfolding wrt $\mathcal{M}$ of a query q over $\mathcal{G}$: is the query obtained from q by substituting every symbol g in q with the query $\phi_{\mathcal{S}}$ that $\mathcal{M}$ associates to g. We denote the unfolding of q wrt $\mathcal{M}$ with $unf_{\mathcal{M}}(q)$.

Observations:

- Since $\mathcal{M}(\mathcal{D})$ is the unique minimal model of $\mathcal{I}$, if q is a CQ or an UCQ, then $\vec{c} \in cert(q, \mathcal{I}, \mathcal{D})$ iff $\vec{c} \in q^{\mathcal{M}(\mathcal{D})}$.
- $unf_{\mathcal{M}}(q)$ is a query expressed over the source schema $\mathcal{S}$.
- Evaluating q over $\mathcal{M}(\mathcal{D})$ is equiv. to evaluating $unf_{\mathcal{M}}(q)$ over $\mathcal{D}$, i.e., $\vec{c} \in q^{\mathcal{M}(\mathcal{D})}$ iff $\vec{c} \in unf_{\mathcal{M}}(q)^{\mathcal{D}}$.
- Hence, $\vec{c} \in cert(q, \mathcal{I}, \mathcal{D})$ iff $\vec{c} \in q^{\mathcal{M}(\mathcal{D})}$ iff $\vec{c} \in unf_{\mathcal{M}}(q)^{\mathcal{D}}$.
  $\rightsquigarrow$ Unfolding suffices for query answering in GAV without constraints.

GAV – Example of unfolding

university

Code	*Name*
AF	bocconi
BN	ucla

student

Code	*Name*	*City*
12	anne	florence
15	bill	oslo

$s_1^{\mathcal{D}}$

12	anne	florence	21
15	bill	oslo	24

$s_2^{\mathcal{D}}$

AF	bocconi
BN	ucla

GAV – Complexity of query answering

Observations:

- If q is a CQ or a UCQ, then $unf_{\mathcal{M}}(q)$ is a first-order query (in fact, a CQ or UCQ).
- $|\mathcal{M}(\mathcal{D})|$ is polynomial wrt $|\mathcal{D}|$.

Hence, we obtain the following results.

Theorem

In a GAV data integration system without constraints, answering unions of conjunctive queries is LogSpace in data complexity and polynomial in combined complexity.

GAV – More expressive queries?

Do these results extend to the case of more expressive queries?

- With more expressive queries in the mapping?
  - Same results hold if we use any computable query in the mapping.
- With more expressive user queries?
  - Same results hold if we use Datalog queries as user queries.
  - Same results hold if we use union of conjunctive queries with inequalities as user queries [vdM93].
  - *Note:* The results do not extend to user queries that contain forms of negation (since it is not true anymore that $\vec{c} \in cert(q, \mathcal{I}, \mathcal{D})$ iff $\vec{c} \in q^{\mathcal{M}(\mathcal{D})}$).

Outline

3 Query answering in GAV without constraints

4 Query answering in (G)LAV without constraints
- (G)LAV and incompleteness
- Approaches to query answering in (G)LAV
- (G)LAV: Direct methods (aka view-based query answering)
- (G)LAV: Query answering by (view-based) query rewriting

(G)LAV data integration systems without constraints

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
no	GAV	yes / no	no
no	**(G)LAV**	yes	no
yes	GAV	yes	yes
yes	(G)LAV	yes	yes

(G)LAV – Example

Consider $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with

Global schema $\mathcal{G}$: student($Code, Name, City$)
enrolled($Scode, Ucode$)

Source schema $\mathcal{S}$: relation s$_1$($Scode, Sname, City, Age$)

Mapping $\mathcal{M}$:

$$q_s(c, n, ci) \leftarrow \mathsf{s}_1(c, n, ci, a) \rightsquigarrow q_g(c, n, ci) \leftarrow \mathsf{student}(c, n, ci), \mathsf{enrolled}(c, u)$$

(G)LAV – Example

$$q_s(c, n, ci) \leftarrow \mathsf{s}_1(c, n, ci, a) \rightsquigarrow q_g(c, n, ci) \leftarrow \mathsf{student}(c, n, ci), \mathsf{enrolled}(c, u)$$

student

Code	*Name*	*City*
12	anne	florence
15	bill	oslo

enrolled

Scode	*Ucode*
12	x
15	y

$\mathsf{s}_1^{\mathcal{D}}$

12	anne	florence	21
15	bill	oslo	24

A source db $\mathcal{D}$ and a corresponding possible global db.

(G)LAV – Incompleteness

(G)LAV mapping assertions $\phi_{\mathcal{S}} \rightsquigarrow \phi_{\mathcal{G}}$ have the logical form:

$$\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow \exists \vec{y}.\ \phi_{\mathcal{G}}(\vec{x}, \vec{y})$$

where $\phi_{\mathcal{S}}$ and $\phi_{\mathcal{G}}$ are conjunctions of atoms.

Given a source database $\mathcal{D}$, in general there are several solutions for a set of (G)LAV assertions (i.e., different databases that are legal wrt $\mathcal{G}$ that satisfy $\mathcal{M}$ wrt $\mathcal{D}$).
$\rightsquigarrow$ Incompleteness comes from the mapping.

This holds even for the case of very simple queries $\phi_{\mathcal{G}}$:

$$s_1(x) \rightsquigarrow q(x) \leftarrow \exists y.\ g(x, y)$$

(G)LAV – Query answering is based on logical inference

(G)LAV – Approaches to query answering

- Exploit connection with query containment.
- Direct methods (aka view-based query answering):
  Try to answer directly the query by means of an algorithm that takes as input the user query q, the specification of $\mathcal{I}$, and the source database $\mathcal{D}$.
- By (view-based) query rewriting:
  1. Taking into account $\mathcal{I}$, reformulate the user query q as a new query (called a rewriting of q) over the source relations.
  2. Evaluate the rewriting over the source database $\mathcal{D}$.

Note: In (G)LAV data integration the views are the sources.

Connection between query answering and containment

Def.: Query containment (under a set of constraints Σ)

is the problem of checking, given two queries q_1, q_2 of the same arity, whether $q_1^{\mathcal{D}}$ is contained in $q_2^{\mathcal{D}}$ for every database $\mathcal{D}$ (satisfying the constraints Σ).

Query answering can be rephrased in terms of query containment:

- A source database $\mathcal{D}$ can be represented as a conjunction $q_{\mathcal{D}}$ of ground literals over $\mathcal{A}_{\mathcal{S}}$ (e.g., if $\vec{c} \in s^{\mathcal{D}}$, there is a literal $s(\vec{c})$).
- If q is a query, and $\vec{c}$ is a tuple, then we denote by $q_{\vec{c}}$ the query obtained by substituting the free variables of q with $\vec{c}$.
- The problem of checking whether $\vec{c} \in cert(q, \mathcal{I}, \mathcal{D})$ under sound sources can be reduced to the problem of checking whether the conjunctive query $q_{\mathcal{D}}$ is contained in $q_{\vec{c}}$ under the constraints expressed by $\mathcal{G} \cup \mathcal{M}$.

Query answering via query containment

Complexity of checking certain answers under sound sources:

- The combined complexity is identical to the complexity of query containment under constraints.
- The data complexity is the complexity of query containment under constraints when the right-hand side query is considered fixed. Hence, it is at most the complexity of query containment under constraints.

It follows that most results and techniques for query containment (under constraints) are relevant also for query answering (under constraints).

Note: Also, query containment can be reduced to query answering. However, (in the presence of constraints) we need to allow for constants of the database to denote the same object (unique name assumption does not hold).

(G)LAV – Canonical model

Def.: Canonical retrieved global database for $\mathcal{I}$ relative to $\mathcal{D}$

Such a database, denoted $Can_{\mathcal{I}}(\mathcal{D})$ (also called canonical model of $\mathcal{I}$ relative to $\mathcal{D}$), is constructed as follows:

- Let all predicates initially be empty in $Can_{\mathcal{I}}(\mathcal{D})$.
- For each mapping assertion $\phi_{\mathcal{S}} \leadsto \phi_{\mathcal{G}}$ in $\mathcal{M}$
  - for each tuple $\vec{c} \in \phi_{\mathcal{S}}^{\mathcal{D}}$ such that $\vec{c} \notin \phi_{\mathcal{G}}^{Can_{\mathcal{I}}(\mathcal{D})}$, add $\vec{c}$ to $\phi_{\mathcal{G}}^{Can_{\mathcal{I}}(\mathcal{D})}$ by inventing fresh variables (Skolem terms) in order to satisfy the existentially quantified variables in $\phi_{\mathcal{G}}$.

Properties of $Can_{\mathcal{I}}(\mathcal{D})$:

- Unique up to variable renaming.
- Can be computed in polynomial time wrt the size of $\mathcal{D}$.
- Satisfies $\mathcal{M}$ by construction, and obviously satisfies $\mathcal{G}$ (since there are no constraints). Hence, $Can_{\mathcal{I}}(\mathcal{D}) \in Sem_{\mathcal{I}}(\mathcal{D})$.

(G)LAV – Example of canonical model

$q_s(c,n,ci) \leftarrow s_1(c,n,ci,a) \quad \rightsquigarrow \quad q_g(c,n,ci) \leftarrow \mathsf{student}(c,n,ci) \wedge \mathsf{enrolled}(c,u)$

student

Code	*Name*	*City*
12	anne	florence
15	bill	oslo

enrolled

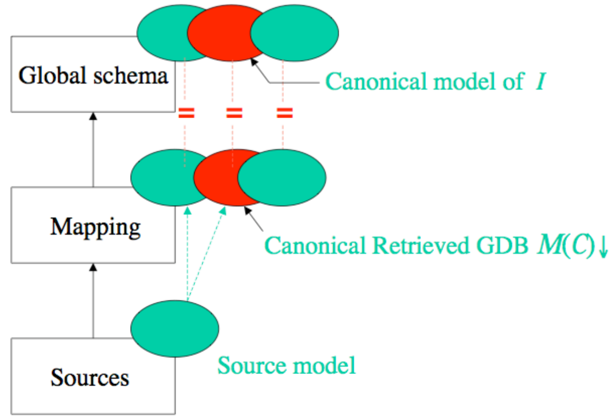
Scode	*Ucode*
12	x
15	y

$s_1^{\mathcal{D}}$

12	anne	florence	21
15	bill	oslo	24

Example of source db $\mathcal{D}$ and corresponding canonical model $Can_{\mathcal{I}}(\mathcal{D})$.

(G)LAV – Canonical model



(G)LAV – Universal solution

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, and $\mathcal{D}$ a source db.

Def.: Universal solution for $\mathcal{I}$ relative to $\mathcal{D}$

Is a global db $\mathcal{B}$ that satisfies $\mathcal{I}$ relative to $\mathcal{D}$ and such that, for every global db $\mathcal{B}'$ that satisfies $\mathcal{I}$ relative to $\mathcal{D}$, there exists a homomorphism $h : \mathcal{B} \rightarrow \mathcal{B}'$ (see [FKMP05]).

Theorem

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a (G)LAV data integration system without constraints in the global schema, and $\mathcal{D}$ a source database. Then $Can_{\mathcal{I}}(\mathcal{D})$ is a universal solution for $\mathcal{I}$ relative to $\mathcal{D}$ (follows from [FKMP05]).

(G)LAV – Query answering

Theorem

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a (G)LAV data integration system without constraints in the global schema, $\mathcal{D}$ a source database, and q a conjunctive query. Then $\vec{c} \in cert(q, \mathcal{I}, \mathcal{D})$ iff $\vec{c} \in q^{Can_{\mathcal{I}}(\mathcal{D})}$.

Proof.

"$\Rightarrow$" Trivial, since $Can_{\mathcal{I}}(\mathcal{D}) \in Sem_{\mathcal{I}}(\mathcal{D})$.

"$\Leftarrow$" Consider a global db $\mathcal{B} \in Sem_{\mathcal{I}}(\mathcal{D})$.

- Since $\vec{c} \in q^{Can_{\mathcal{I}}(\mathcal{D})}$, there exists a homomorphism $h_1 : q(\vec{c}) \rightarrow Can_{\mathcal{I}}(\mathcal{D})$.
- Since $Can_{\mathcal{I}}(\mathcal{D})$ is a universal solution, there exists a homomorphism $h_2 : Can_{\mathcal{I}}(\mathcal{D}) \rightarrow \mathcal{B}$.

Hence, $h_1 \circ h_2$ is a homomorphism from $q(\vec{c})$ to $\mathcal{B}$, and $\vec{c} \in q^{\mathcal{B}}$. □

(G)LAV – Complexity of query answering

From the above results, we obtain that for a CQ q, we can compute $cert(q, \mathcal{I}, \mathcal{D})$ as follows:

1. Compute $Can_{\mathcal{I}}(\mathcal{D})$ from $\mathcal{D}$ — polynomial in $|\mathcal{D}|$.
2. Evaluate q over $Can_{\mathcal{I}}(\mathcal{D})$ — LogSpace in $|\mathcal{D}|$.

The above applies also to UCQs. Hence, we obtain the following result.

Theorem

In a (G)LAV data integration system without constraints, answering unions of conjunctive queries is polynomial in data and combined complexity.

The data complexity upper bound can actually be improved.

(G)LAV – "Inverse rules" technique

From [DG97]: consider mappings as "inverse" rules:

$$r_1(t) \rightsquigarrow q_1(t) \leftarrow \text{movie}(t,y,d) \land \text{european}(d)$$
$$r_2(t,v) \rightsquigarrow q_2(t,v) \leftarrow \text{movie}(t,y,d) \land \text{review}(t,v)$$

$$\forall t.\ r_1(t) \rightarrow \exists y,d.\ \text{movie}(t,y,d) \land \text{european}(d)$$
$$\forall t,v.\ r_2(t,v) \rightarrow \exists y,d.\ \text{movie}(t,y,d) \land \text{review}(t,v)$$

$$\text{movie}(t, f_1(t), f_2(t)) \leftarrow r_1(t)$$
$$\text{european}(f_2(t)) \leftarrow r_1(t)$$
$$\text{movie}(t, f_4(t,v), f_5(t,v)) \leftarrow r_2(t,v)$$
$$\text{review}(t,v) \leftarrow r_2(t,v)$$

Answering a query means evaluating a goal wrt to this nonrecursive logic program (which can be transformed into a union of CQs).

Theorem

In a (G)LAV data integration system without constraints, answering unions of conjunctive queries is LogSpace in data complexity.

(G)LAV – More expressive queries?

- More expressive source queries in the mapping?
  - Same results hold if we use any computable query as source query in the mapping assertions.
- More expressive queries over the global schema in the mapping?
  - Already unions of conjunctive queries lead to intractability.
- More expressive user queries?
  - Same results hold if we use Datalog queries as user queries.
  - Even the simplest form of negation (inequalities) leads to intractability.

(G)LAV – Intractability for views that contain union

From [vdM93], by reduction from 3-colorability.
We define the following LAV data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$:

$\mathcal{G}$: $\text{edge}(x,y)$, $\text{color}(x,c)$ $\mathcal{S}$: $s_E(x,y)$, $s_N(x)$

$\mathcal{M}$: $s_E(x,y) \rightsquigarrow q_E(x,y) \leftarrow \text{edge}(x,y)$
$s_N(x) \rightsquigarrow q_N(x) \leftarrow \text{color}(x,\texttt{RED}) \lor \text{color}(x,\texttt{BLUE}) \lor \text{color}(x,\texttt{GREEN})$

Given a graph $G = (N, E)$, we define the following source database $\mathcal{D}$:

$s_E^{\mathcal{D}} = \{\ (a,b),(b,a) \mid (a,b) \in E\ \}$ $s_N^{\mathcal{D}} = \{\ (a) \mid a \in N\ \}$

Consider the boolean query: $q() \leftarrow \exists x,y,c.\ \text{edge}(x,y) \land \text{color}(x,c) \land \text{color}(y,c)$ describing mismatched edge pairs:

- If G is 3-colorable, then $\exists \mathcal{B}$ s.t. $q^{\mathcal{B}} = \mathit{false}$, hence $cert(q, \mathcal{I}, \mathcal{D}) = \mathit{false}$.
- If G is not 3-colorable, then $cert(q, \mathcal{I}, \mathcal{D}) = \mathit{true}$.

Theorem

In a LAV data integration system without constraints and with UCQs as views, answering CQs is coNP-hard in data complexity.

(G)LAV – In coNP for views and queries that are UCQs

- $\vec{c} \notin cert(q, \mathcal{I}, \mathcal{D})$ if and only if there is a database $\mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{D}$, and such that $\vec{c} \notin q^{\mathcal{B}}$.
- The mapping $\mathcal{M}$ has the form:

  $$\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow \exists \vec{y}_1.\ \alpha_1(\vec{x}, \vec{y}_1) \vee \cdots \vee \exists \vec{y}_h\ \alpha_h(\vec{x}, \vec{y}_h))$$

  Hence, each tuple in $\mathcal{D}$ forces the existence of k tuples in any database that satisfies $\mathcal{M}$ wrt $\mathcal{D}$, where k is the maximal length of conjunctions $\alpha_i(\vec{x}, \vec{y}_i)$ in $\mathcal{M}$.
- If $\mathcal{D}$ has n tuples, then there is a db $\mathcal{B}' \subseteq \mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{D}$ with at most $n \cdot k$ tuples. Since q is monotone, $\vec{c} \notin q^{\mathcal{B}'}$.
- Checking whether $\mathcal{B}'$ satisfies $\mathcal{M}$ wrt $\mathcal{D}$, and checking whether $\vec{c} \notin q^{\mathcal{B}'}$ can be done in PTIME wrt the size of $\mathcal{B}'$.

Theorem

In a LAV data integration system without constraints and with UCQs as views, answering UCQs is coNP-complete in data complexity.

(G)LAV – Conjunctive user queries with inequalities

Consider $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and source db $\mathcal{D}$ (see [FKMP05]):

$$\mathcal{G}: \quad g(x,y) \qquad\qquad \mathcal{S}: \quad s(x,y)$$
$$\mathcal{M}: \quad s(x,y) \rightsquigarrow q(x,y) \leftarrow g(x,z) \wedge g(z,y)$$
$$\mathcal{D}: \quad \{\ s(\mathsf{a},\mathsf{a})\ \}$$

- Both $\mathcal{B}_1 = \{g(\mathsf{a},\mathsf{a})\}$ and $\mathcal{B}_2 = \{g(\mathsf{a},\mathsf{b}),\ g(\mathsf{b},\mathsf{a})\}$ are solutions.
- If $\mathcal{B}$ is a universal solution, then both $g(\mathsf{a},x)$ and $g(x,\mathsf{a})$ are in $\mathcal{B}$, with $x \neq \mathsf{a}$ (otherwise $g(\mathsf{a},\mathsf{a})$ would be $true$ in every solution).

Let $q() \leftarrow g(x,y) \wedge x \neq y$

- $q^{\mathcal{B}_1} = false$, hence $cert(q, \mathcal{I}, \mathcal{D}) = false$.
- But $q^{\mathcal{B}} = true$ for every universal solution $\mathcal{B}$ for $\mathcal{I}$ relative to $\mathcal{D}$.

Hence, the notion of universal solution is not the right tool.

(G)LAV – Conjunctive user queries with inequalities

- coNP algorithm: guess equalities on variables in the canonical retrieved global database.
- coNP-hard already for a conjunctive user query with one inequality (and conjunctive view definitions) [AD98]

Theorem

In a (G)LAV data integration system without constraints and with CQs as views, answering CQs with inequalities is coNP-complete in data complexity.

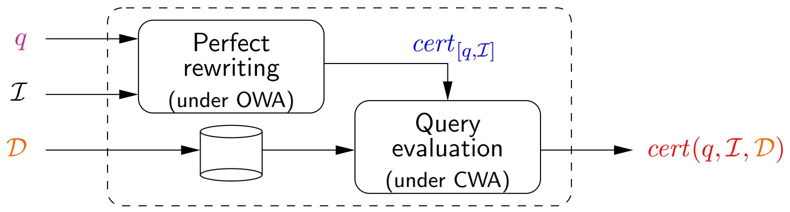
Note: inequalities in the view definitions do not affect expressive power and complexity (in fact, they can be removed).

Query answering

In the presence of incomplete information, as is the case in (G)LAV data integration, query answering is a form of logical inference.

Query answering: perfect rewriting + evaluation

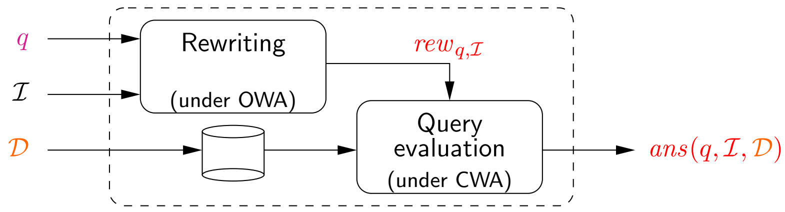
We can (at least conceptually) separate the contribution of the query, global schema, and mappings from the contribution of the data.



The query $cert_{[q,\mathcal{I}]}$ that is the result of the perfect rewriting could be expressed in an arbitrary query language.

Query answering: rewriting + evaluation

In practice, we can divide query answering in two steps by chosing a priori the language of the rewriting $rew_{q,\mathcal{I}}$:



1. Rewrite the query in terms of the chosen query language over the alphabet of $\mathcal{A}_\mathcal{S}$.
2. Evaluate the rewriting over the source database $\mathcal{D}$.

(G)LAV – Maximal rewritings

Query answering by rewriting:

1. Given $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and a query q over $\mathcal{G}$, rewrite q into a query, called $rew_{q,\mathcal{I}}$, over the alphabet $\mathcal{A}_\mathcal{S}$ of the sources.
2. Evaluate the rewriting $rew_{q,\mathcal{I}}$ over the source database $\mathcal{D}$.

Def.: Maximal $\mathcal{L}$-rewriting of q wrt $\mathcal{I}$

Given $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, a query q over $\mathcal{G}$, and a query language $\mathcal{L}$, a maximal $\mathcal{L}$-rewriting of q wrt $\mathcal{I}$ is a query that:

- is expressed in $\mathcal{L}$;
- is sound, i.e., for every db $\mathcal{D}$ computes only tuples in $cert(q,\mathcal{I},\mathcal{D})$;
- is the maximal such query among those expressible in $\mathcal{L}$.

We are interested in computing maximal $\mathcal{L}$-rewritings.

(G)LAV – Example of maximal rewriting

$\mathcal{G}$: nonstop$(Airline, Num, From, To)$

$\mathcal{S}$: flightsByUnited$(Num, From, To)$
flightsFromSFO$(Airline, Num, To)$

$\mathcal{M}$: flightsByUnited$(num, from, to)$ $\rightsquigarrow$
$g_1(num, from, to) \leftarrow$ nonstop$(\mathtt{UA}, num, from, to)$
flightsFromSFO$(airline, num, to)$ $\rightsquigarrow$
$g_2(airline, num, to) \leftarrow$ nonstop$(airline, num, \mathtt{SFO}, to)$

Queries: $q_1(al, num) \leftarrow$ nonstop$(al, num, \mathtt{LAX}, \mathtt{PHX})$
$q_2(al, num) \leftarrow$ nonstop$(al, num, \mathtt{SFO}, to)$

Maximal (wrt positive queries) rewritings of q_1 and q_2 are:

$rew_{q_1,\mathcal{I}}(al, num) \leftarrow$ flightsByUnited$(num, \mathtt{LAX}, \mathtt{PHX})$, $al = \mathtt{UA}$
$rew_{q_2,\mathcal{I}}(al, num) \leftarrow$ flightsByUnited$(num, \mathtt{SFO}, to)$, $al = \mathtt{UA}$ $\vee$
flightsFromSFO(al, num, to)

(G)LAV – Exact rewritings

The (mappings in) a data integration system and the choice of $\mathcal{L}$ may be such that even a maximal $\mathcal{L}$-rewriting does not provide all answers that the query evaluated over a global db would provide.

Def.: Exact rewriting

An exact rewriting of a query q wrt a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ is a rewriting that is logically equivalent to q, modulo the mappings $\mathcal{M}$.

Note: exact rewritings may not exist for a given query.

Example (from the previous slide)

- $rew_{q_1,\mathcal{I}}$ is not an exact rewriting of q_1 wrt $\mathcal{I}$.
- $rew_{q_2,\mathcal{I}}$ is an exact rewriting of q_2 wrt $\mathcal{I}$.

Perfect rewriting

What is the relationship between answering by rewriting and certain answers? [CDGLV05]:

- When does the (maximal) rewriting compute all certain answers?
- What do we gain or loose by focusing on a given class of queries?

Let's try to consider the "best possible" rewriting.

Define $cert_{[q,\mathcal{I}]}(\cdot)$ to be the function that, with q and $\mathcal{I}$ fixed, given source database $\mathcal{D}$, computes the certain answers $cert(q, \mathcal{I}, \mathcal{D})$.

- $cert_{[q,\mathcal{I}]}$ can be seen as a query on the alphabet $\mathcal{A}_{\mathcal{S}}$.
- $cert_{[q,\mathcal{I}]}$ is a (sound) rewriting of q wrt $\mathcal{I}$.
- No sound rewriting exists that is better than $cert_{[q,\mathcal{I}]}$.

Hence, $cert_{[q,\mathcal{I}]}$ is called the perfect rewriting of q wrt $\mathcal{I}$.

Properties of the perfect rewriting

- Can the perfect rewriting be expressed in a certain query language?
- For a given class of queries, what is the relationship between a maximal rewriting and the perfect rewriting?
  - From a semantical point of view
  - From a computational point of view
- Which is the computational complexity of finding the perfect rewriting, and how big is it?
- Which is the computational complexity of evaluating the perfect rewriting?

(G)LAV – The case of conjunctive queries

Theorem ([LMSS95, AD98])

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a (G)LAV data integration system where the queries in $\mathcal{M}$ are CQs. Let q be a CQ and let q' be the union of all maximal rewritings of q for the class of CQs. Then:

- q' is the maximal rewriting for the class of unions of conjunctive queries (UCQs).
- q' is the perfect rewriting of q wrt $\mathcal{I}$.
- q' is a PTIME query.
- q' is an exact rewriting (equivalent to q for each database $\mathcal{B}$ of $\mathcal{I}$), if an exact rewriting exists.

Does this "ideal situation" carry over to cases where q and $\mathcal{M}$ allow for union?

(G)LAV – The case of mappings with union

When queries over the global schema in the mapping contain union:

- We have seen that view-based query answering is coNP-complete in data complexity [vdM93].
- Hence, $cert(q, \mathcal{I}, \mathcal{D})$, with q, $\mathcal{I}$ fixed, is a coNP-complete function.
- Hence, the perfect rewriting $cert_{[q,\mathcal{I}]}$ is a coNP-complete query.

We do not have the ideal situation we had for conjunctive queries.

Problem:

Isolate those cases of view based query rewriting for data integration systems $\mathcal{I}$ where mappings contain unions for which the perfect rewriting $cert_{[q,\mathcal{I}]}$ is a PTIME function (assuming $\mathrm{P} \neq \mathrm{NP}$) [CDGLV00c].

(G)LAV – Data complexity of query answering

From [AD98], for sound sources:

Global schema mapping query	User queries				
	CQ	$CQ^{\neq}$	PQ	Datalog	FOL
CQ	PTIME	coNP	PTIME	PTIME	undec.
$CQ^{\neq}$	PTIME	coNP	PTIME	PTIME	undec.
PQ	coNP	coNP	coNP	coNP	undec.
Datalog	coNP	undec.	coNP	undec.	undec.
FOL	undec.	undec.	undec.	undec.	undec.

(G)LAV – Further references

- Inverse rules [DG97]
- Bucket algorithm for query rewriting [LRO96]
- MiniCon algorithm for query rewriting [PL00]
- Conjunctive queries using conjunctive views [LMSS95]
- Recursive queries (Datalog programs) using conjunctive views [DG97, AGK99]
- CQs with arithmetic comparison [ALM02]
- Complexity analysis [AD98, GM99]
- Variants of Regular Path Queries [CDGLV00a, CDGLV00b, CDGLV01, DT01]
- Relationship between view-based rewriting and answering [CDGLV00c, CDGLV03, CDGLV05]

Chapter III

Query answering in the presence of constraints

Outline

5 The role of global integrity constraints

Outline

5 The role of global integrity constraints
- Types of integrity constraints
- GAV systems with integrity constraints
- (G)LAV systems with integrity constraints
- Query answering with integrity constraints

Global integrity constraints

- Integrity constraints (ICs) are posed over the global schema.
- Specify intensional knowledge about the domain of interest.
- Add semantics to the information.
- But data in the sources can conflict with global ICs.
- The presence of global ICs raises semantic and computational problems.

Note: global integrity constraints play the same role as an ontology in Ontology-Based Data Access.

Integrity constraints for relational schemas

Most important types of ICs that have been considered for the relational model:

- key dependencies (KDs)
- functional dependencies (FDs)
- foreign keys (FKs)
- inclusion dependencies (IDs)
- exclusion dependencies (EDs)

Inclusion dependencies (IDs)

An inclusion dependency (ID) states that the presence of a tuple $\vec{t}_1$ in a relation implies the presence of a tuple $\vec{t}_2$ in another relation, where $\vec{t}_2$ contains a projection of the values contained in $\vec{t}_1$.

Def.: Syntax of inclusion dependencies:

$$r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k]$$

with $i_1, \ldots, i_k$ components of r, and $j_1, \ldots, j_k$ components of s.

Example

For r of arity 3 and s of arity 2, the ID $r[1] \subseteq s[2]$ corresponds to the FOL sentence:

$$\forall x, y, w.\ r(x, y, w) \rightarrow \exists z.\ s(z, x)$$

Note: IDs are a special form of tuple-generating dependencies.

Key dependencies (KDs)

A key dependency (KD) states that a set of attributes functionally determines all the attributes of a relation.

Def.: Syntax of key dependencies:

$$\mathit{key}(r) = \{i_1, \ldots, i_k\}$$

with $i_1, \ldots, i_k$ components of r.

Example

For r of arity 3, the KD $\mathit{key}(r) = \{1\}$ corresponds to the FOL sentence

$$\forall x, y, y', z, z'.\ r(x, y, z) \wedge r(x, y', z') \rightarrow y = y' \wedge z = z'$$

Note: KDs are a special form of equality-generating dependencies.

Exclusion dependencies (EDs)

An exclusion dependency (ED) states that the presence of a tuple $\vec{t}_1$ in a relation implies the absence of a tuple $\vec{t}_2$ in another relation, where $\vec{t}_2$ contains a projection of the values contained in $\vec{t}_1$.

Def.: Syntax of exclusion dependencies:

$$r[i_1, \ldots, i_k] \cap s[j_1, \ldots, j_k] = \emptyset$$

with $i_1, \ldots, i_k$ components of r, and $j_1, \ldots, j_k$ components of s.

Example

For r of arity 3 and s of arity 2, the ED $r[1] \cap s[2] = \emptyset$ corresponds to the FOL sentence

$$\forall x, y, w, z.\ r(x, y, w) \rightarrow \neg s(z, x)$$

Note: EDs are a special form of denial constraints.

GAV system with integrity constraints

We consider a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ where

- $\mathcal{G}$ is a global schema with constraints.
- $\mathcal{M}$ is a set of GAV mappings, whose assertions have the form $\phi_{\mathcal{S}} \leadsto g$ and are interpreted as

$$\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow g(\vec{x})$$

where $\phi_{\mathcal{S}}$ is a conjunctive query over $\mathcal{S}$, and g is an element of $\mathcal{G}$.

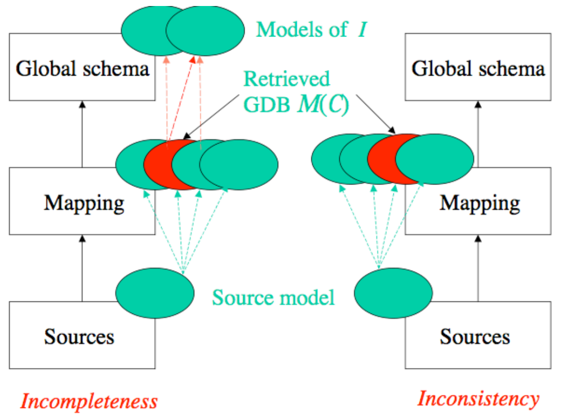
Basic observation

Since $\mathcal{G}$ does have constraints, the retrieved global database $\mathcal{M}(\mathcal{D})$ may not be legal for $\mathcal{G}$.

GAV data integration systems with constraints

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
no	GAV	yes / no	no
no	(G)LAV	yes	no
IDs	**GAV**	yes	no
KDs	**GAV**	yes / no	yes
IDs + KDs	**GAV**	yes	yes
yes	(G)LAV	yes	yes

GAV with constraints – Incompleteness and inconsistency



Semantics of GAV systems with integrity constraints

Given a source db $\mathcal{D}$, a global db $\mathcal{B}$ (over Δ) satisfies $\mathcal{I}$ relative to $\mathcal{D}$ if:

1. It is legal wrt the global schema, i.e., it satisfies the ICs.
2. It satisfies the mapping, i.e., $\mathcal{B}$ is a superset of the retrieved global database $\mathcal{M}(\mathcal{D})$ (sound mappings).

Recall:

- $\mathcal{M}(\mathcal{D})$ is obtained by evaluating, for each relation in $\mathcal{A}_\mathcal{G}$, the corresponding mapping query over the source database $\mathcal{D}$.
- We are interested in certain answers to a query, i.e., those that hold for all global databases that satisfy $\mathcal{I}$ relative to $\mathcal{D}$.

(G)LAV system with integrity constraints

We consider a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ where

- $\mathcal{G}$ is a global schema with constraints.
- $\mathcal{M}$ is a set of LAV mappings, whose assertions have the form $\phi_\mathcal{S} \leadsto \phi_\mathcal{G}$ and are interpreted as

$$\forall \vec{x}.\ \phi_\mathcal{S}(\vec{x}) \rightarrow \phi_\mathcal{G}(\vec{x})$$

where $\phi_\mathcal{S}$ is a conjunctive query over $\mathcal{S}$, and $\phi_\mathcal{G}$ is a conjunctive query over $\mathcal{G}$.

Basic observation

Since $\mathcal{G}$ does have constraints, the canonical retrieved global database $Can_\mathcal{I}(\mathcal{D})$ may not be legal for $\mathcal{G}$.

Semantics of (G)LAV systems with integrity constraints

Given a source db $\mathcal{D}$, a global db $\mathcal{B}$ (over Δ) satisfies $\mathcal{I}$ relative to $\mathcal{D}$ if:

1. It is legal wrt the global schema, i.e., it satisfies the ICs.
2. It satisfies the mapping, i.e., $\mathcal{B}$ is a superset of the canonical retrieved global database $Can_{\mathcal{I}}(\mathcal{D})$ (sound mappings).

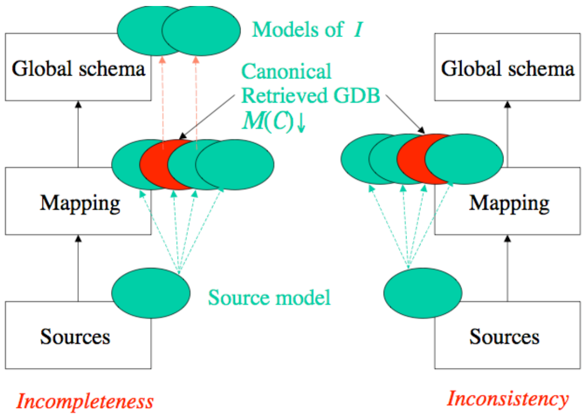
Recall:

- $\mathcal{M}(\mathcal{D})$ is obtained by evaluating, for each mapping assertion $\phi_{\mathcal{S}} \rightsquigarrow \phi_{\mathcal{G}}$, the query $\phi_{\mathcal{S}}$ over $\mathcal{D}$, and using the obtained tuples to populate the global relations according to $\phi_{\mathcal{G}}$, using fresh constants for existentially quantified elements.
- We are interested in certain answers to a query, i.e., those that hold for all global databases that satisfy $\mathcal{I}$ relative to $\mathcal{D}$.

(G)LAV data integration systems with constraints

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
no	GAV	yes / no	no
no	(G)LAV	yes	no
IDs	GAV	yes	no
KDs	GAV	yes / no	yes
IDs + KDs	GAV	yes	yes
IDs	**(G)LAV**	yes	no
KDs	**(G)LAV**	yes	yes
IDs + KDs	**GAV**	yes	yes

(G)LAV with constr. – Incompleteness and inconsistency



Data integration with constraints – Query answering

In integration systems, in the presence of constraints, we can resort to techniques that are analogous to those used in OBDS:

- Look for the possibility of separating IDs from KDs and EDs.
- Look for the possibility of rewriting the query into one that can be evaluated ignoring the constraints.

Can query answering be performed by first-order (UCQ) rewriting?

- GAV with IDs + EDs: yes
- GAV with IDs + KDs + EDs: only if KDs and IDs are separable
- LAV with IDs + EDs: yes
- LAV with KDs: no

Chapter IV

Concluding remarks

Outline

6 Concluding remarks

7 References

Outline

6 Concluding remarks

7 References

Further issues and open problems

- Further forms of constraints, e.g.,
  - KDs with restricted forms of key-conflicting IDs
  - ontology languages, description logics, RDF (cf. OBDA)
- Semistructured data and XML
  - constraints (DTDs, XML Schema, ...)
  - query languages (transitive closure)
- Finite models vs. unrestricted models [Ros06]
- Data exchange and materialization

Outline

6 Concluding remarks

7 References

References I

[AD98] S. Abiteboul and O. Duschka.
Complexity of answering queries using materialized views.
In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998.

[AGK99] F. N. Afrati, M. Gergatsoulis, and T. Kavalieros.
Answering queries using materialized views with disjunction.
In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 435–452. Springer, 1999.

[ALM02] F. N. Afrati, C. Li, and P. Mitra.
Answering queries using views with arithmetic comparisons.
In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 209–220, 2002.

References II

[CCDGL02a] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini.
On the expressive power of data integration systems.
In *Proc. of the 21st Int. Conf. on Conceptual Modeling (ER 2002)*, 2002.

[CCDGL02b] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini.
On the role of integrity constraints in data integration.
Bull. of the IEEE Computer Society Technical Committee on Data Engineering, 25(3):39–45, 2002.

[CDGL98] D. Calvanese, G. De Giacomo, and M. Lenzerini.
On the decidability of query containment under constraints.
In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

References III

[CDGL+05] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.
Inconsistency tolerance in P2P data integration: an epistemic logic approach.
In *Proc. of the 10th Int. Sym. on Database Programming Languages (DBPL 2005)*, pages 90–105, 2005.

[CDGL+06] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.
Data complexity of query answering in description logics.
In *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 260–270, 2006.

[CDGLR04] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati.
Logical foundations of peer-to-peer data integration.
In *Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004)*, pages 241–251, 2004.

References IV

[CDGLV00a] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi.
Answering regular path queries using views.
In *Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000)*, pages 389–398, 2000.

[CDGLV00b] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi.
Query processing using views for regular path queries with inverse.
In *Proc. of the 19th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2000)*, pages 58–66, 2000.

[CDGLV00c] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi.
View-based query processing and constraint satisfaction.
In *Proc. of the 15th IEEE Symp. on Logic in Computer Science (LICS 2000)*, pages 361–371, 2000.

References V

[CDGLV01] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi.
View-based query answering and query containment over semistructured data.
In *Proc. of the 8th Int. Workshop on Database Programming Languages (DBPL 2001)*, 2001.

[CDGLV03] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi.
View-based query containment.
In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, pages 56–67, 2003.

[CDGLV05] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi.
View-based query processing: On the relationship between rewriting, answering and losslessness.
In *Proc. of the 10th Int. Conf. on Database Theory (ICDT 2005)*, volume 3363 of *Lecture Notes in Computer Science*, pages 321–336. Springer, 2005.

References VI

[CLR03] A. Calì, D. Lembo, and R. Rosati.
Query rewriting and answering under constraints in data integration systems.
In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 16–21, 2003.

[CR03] D. Calvanese and R. Rosati.
Answering recursive queries under keys and foreign keys is undecidable.
In *Proc. of the 10th Int. Workshop on Knowledge Representation meets Databases (KRDB 2003)*, volume 79 of *CEUR Electronic Workshop Proceedings*, `http://ceur-ws.org/Vol-79/`, 2003.

[DG97] O. M. Duschka and M. R. Genesereth.
Answering recursive queries using views.
In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 109–116, 1997.

References VII

[DGL00] O. M. Duschka, M. R. Genesereth, and A. Y. Levy.
Recursive query plans for data integration.
J. of Logic Programming, 43(1):49–73, 2000.

[DT01] A. Deutsch and V. Tannen.
Optimization properties for classes of conjunctive regular path queries.
In *Proc. of the 8th Int. Workshop on Database Programming Languages (DBPL 2001)*, 2001.

[FKMP05] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa.
Data exchange: Semantics and query answering.
Theoretical Computer Science, 336(1):89–124, 2005.

[FKP05] R. Fagin, P. G. Kolaitis, and L. Popa.
Data exchange: Getting to the core.
ACM Trans. on Database Systems, 30(1):174–210, 2005.

References VIII

[FM05] A. Fuxman and R. J. Miller.
First-order query rewriting for inconsistent databases.
In *Proc. of the 10th Int. Conf. on Database Theory (ICDT 2005)*, volume 3363 of *LNCS*, pages 337–351. Springer, 2005.

[GM99] G. Grahne and A. O. Mendelzon.
Tableau techniques for querying information sources through global schemas.
In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 1999.

[Hal01] A. Y. Halevy.
Answering queries using views: A survey.
Very Large Database J., 10(4):270–294, 2001.

References IX

[HIST03] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov.
Schema mediation in peer data management systems.
In *Proc. of the 19th IEEE Int. Conf. on Data Engineering (ICDE 2003)*, pages 505–516, 2003.

[LEF+05] N. Leone, T. Eiter, W. Faber, M. Fink, G. Gottlob, G. Greco, E. Kalka, G. Ianni, D. Lembo, V. Lio, B. Nowicki, R. Rosati, M. Ruzzi, W. Staniszkis, and G. Terracina.
Boosting information integration: The INFOMIX system.
In *Proc. of the 13th Ital. Conf. on Database Systems (SEBD 2005)*, pages 55–66, 2005.

[Len02] M. Lenzerini.
Data integration: A theoretical perspective.
In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

References X

[LMSS95] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava.
Answering queries using views.
In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'95)*, pages 95–104, 1995.

[LRO96] A. Y. Levy, A. Rajaraman, and J. J. Ordille.
Query answering algorithms for information agents.
In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, pages 40–47, 1996.

[PL00] R. Pottinger and A. Y. Levy.
A scalable algorithm for answering queries using views.
In *Proc. of the 26th Int. Conf. on Very Large Data Bases (VLDB 2000)*, pages 484–495, 2000.

References XI

[Rei84] R. Reiter.
Towards a logical reconstruction of relational database theory.
In M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors, *On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*. Springer, 1984.

[Ros06] R. Rosati.
On the decidability and finite controllability of query processing in databases with incomplete information.
In *Proc. of the 25th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2006)*, pages 356–365, 2006.

[Var86] M. Y. Vardi.
Querying logical databases.
J. of Computer and System Sciences, 33:142–160, 1986.

References XII

[vdM93] R. van der Meyden.
Recursively indefinite databases.
Theoretical Computer Science, 116(1–2):151–194, 1993.

[vdM98] R. van der Meyden.
Logical approaches to incomplete information.
In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer Academic Publishers, 1998.