Data Integration

Maurizio Lenzerini

Universitá di Roma "La Sapienza"

DASI '06: Phd School on Data and Service Integration
Bertinoro, December 11–15, 2006

Structure of the course

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
   - Inconsistency tolerance
4. Concluding remarks

Structure of the course

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
   - Inconsistency tolerance
4. Concluding remarks

Structure of the course

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
   - Inconsistency tolerance
4. Concluding remarks

Structure of the course

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
   - Inconsistency tolerance
4. Concluding remarks

Part I

Introduction to data integration

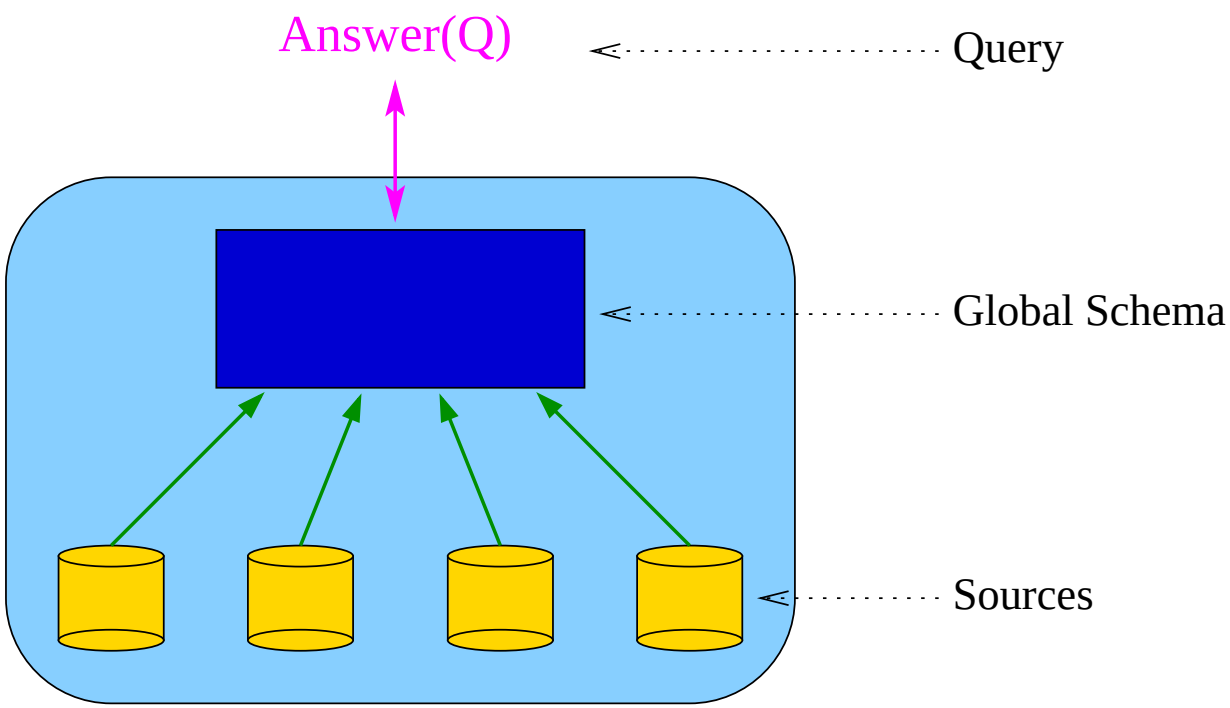
Outline

Outline

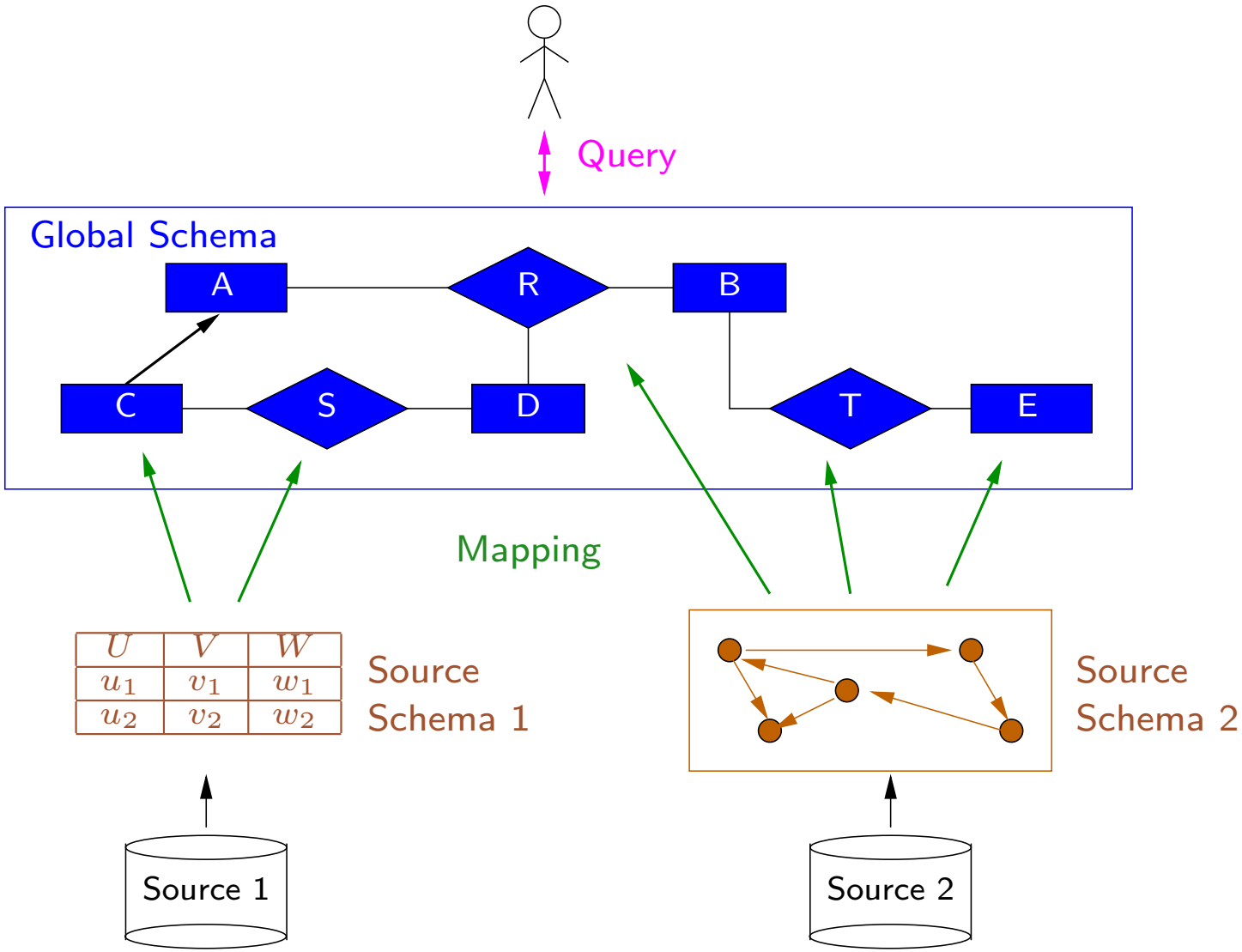
Outline

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

Outline

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

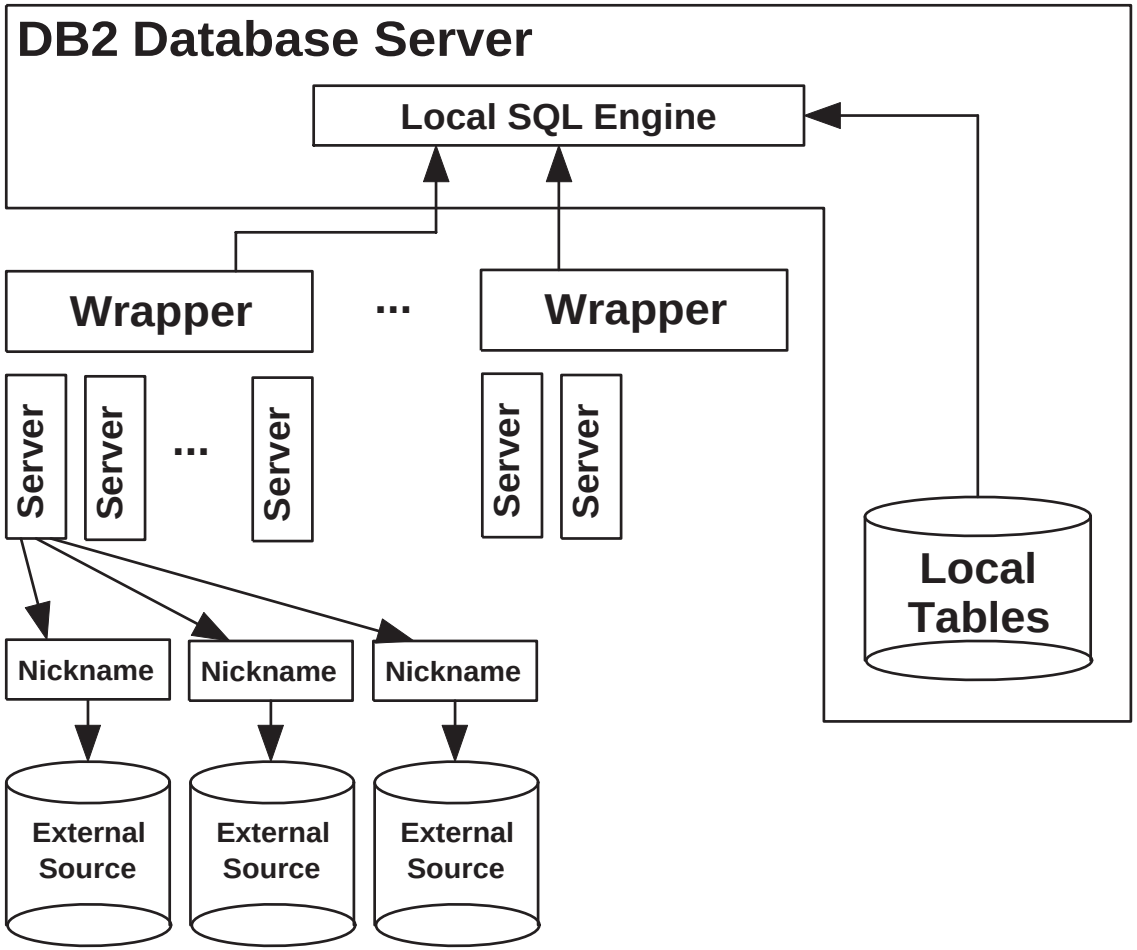
However, current tools do not (directly) support logical (or conceptual) transparency

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

DB2 Information integrator



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

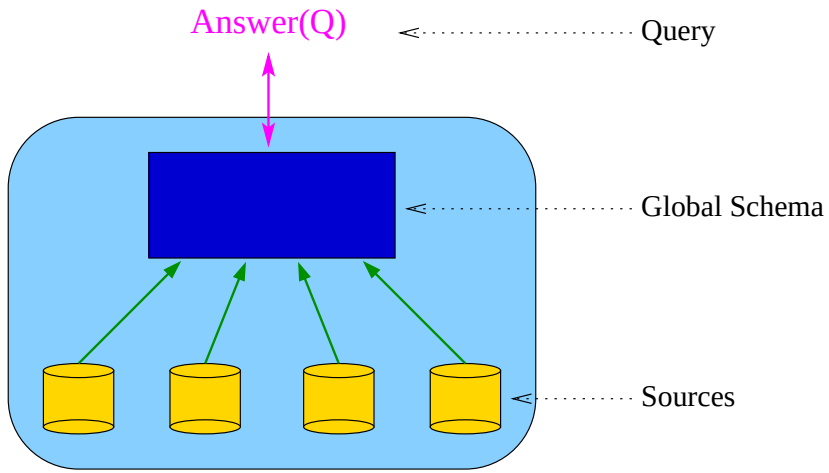
- (Mediator-based) data integration
- Data exchange [Fagin & al. TCS'05, Kolaitis PODS'05]
  - materialization of the global view
  - allows for query answering without accessing the sources
- P2P data integration [Halevy & al. ICDE'03, Calvanese & al. PODS'04, Calvanese & al. DBPL'05]
  - several peers
  - each peer with local and external sources
  - queries over one peer

Approaches to data integration

- (Mediator-based) data integration *. . . is the topic of this course*
- Data exchange [Fagin & al. TCS'05, Kolaitis PODS'05]
  - materialization of the global view
  - allows for query answering without accessing the sources
- P2P data integration [Halevy & al. ICDE'03, Calvanese & al. PODS'04, Calvanese & al. DBPL'05]
  - several peers
  - each peer with local and external sources
  - queries over one peer

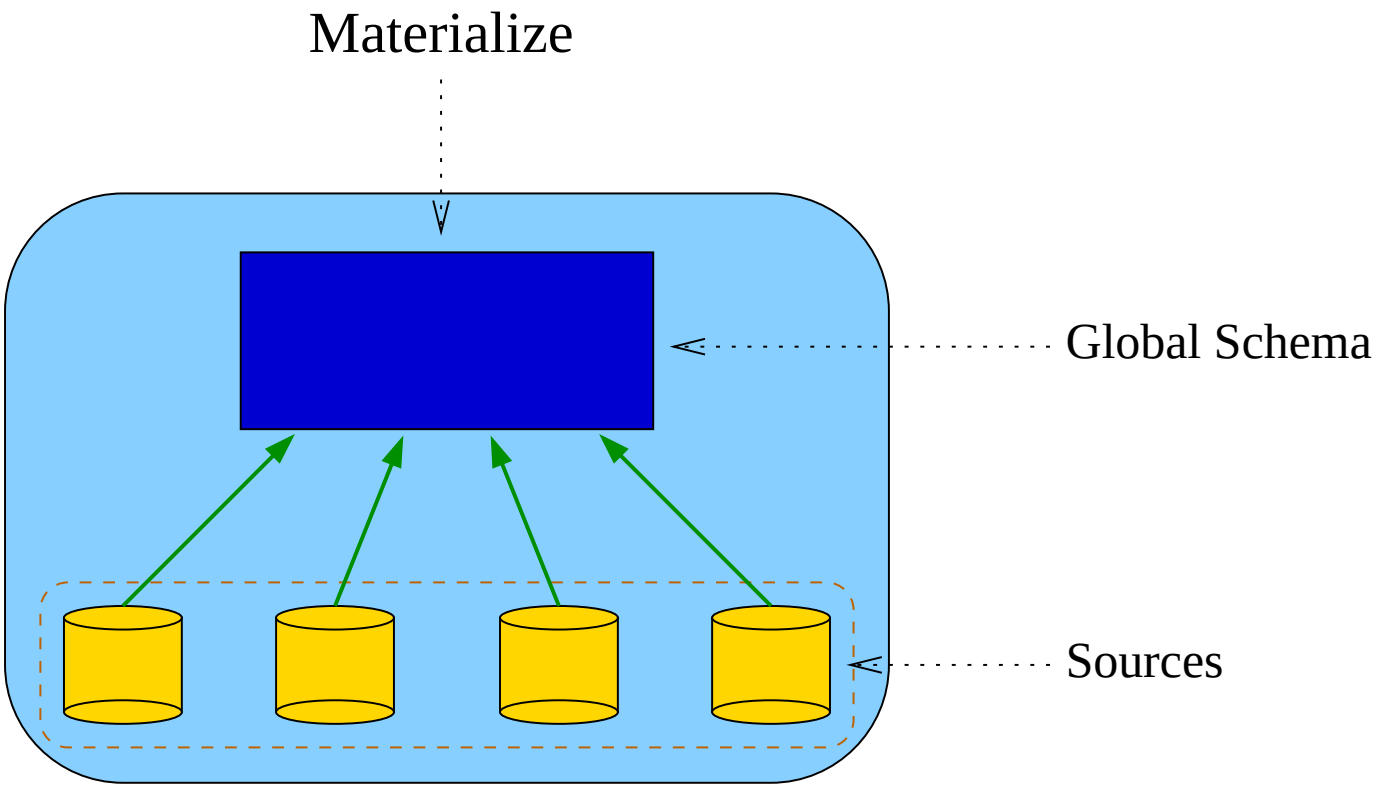
Mediator based data integration

- 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

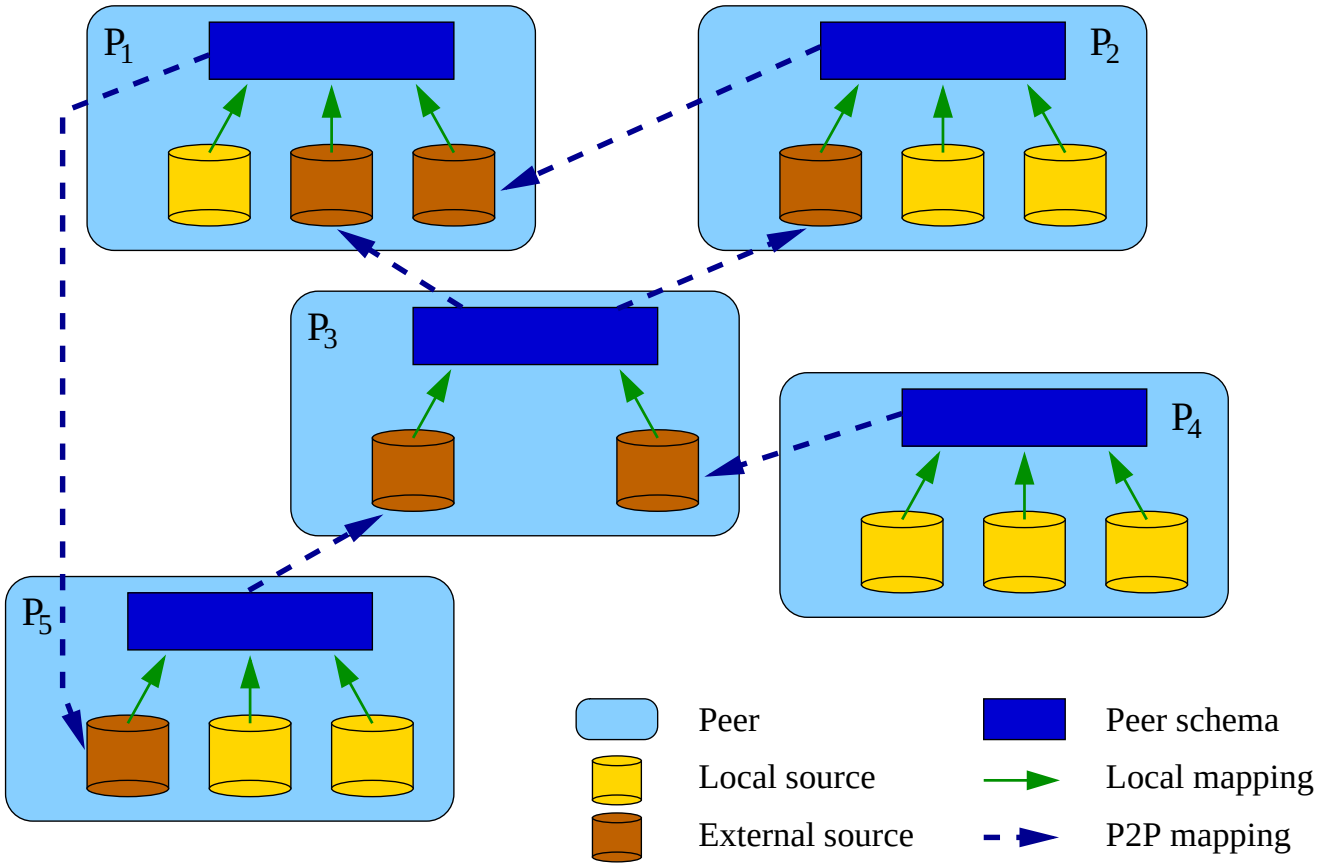


Data exchange

- Materialization of the global schema



Peer-to-peer data integration



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

Outline

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

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 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 course

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 course

Outline

Outline

Formal framework for data integration

Definition

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

Definition

The set of databases for $\mathcal{A}_\mathcal{G}$ that satisfy $\mathcal{I}$ relative to $\mathcal{C}$ is:

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

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

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 $\mathcal{C}$ 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

Definition

The set of databases for $\mathcal{A}_\mathcal{G}$ that satisfy $\mathcal{I}$ relative to $\mathcal{C}$ is:

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

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

Outline

Relational calculus: the basics

Basic idea: we use the language of first-order logic to express which tuples should be in the result to a query

- We assume to have a domain Δ and a set Σ of constants, one for each element of Δ
- Let $\mathcal{A}$ be a relational alphabet, i.e., a set of predicates, each with an associated arity (we assume a positional notation)
- A database $\mathcal{D}$ over $\mathcal{A}$ and Δ is a set of relations, one for each predicate in $\mathcal{A}$, over the constants in Σ (in turn interpreted as elements of Δ)
- Let $\mathcal{L}_{\mathcal{A}}$ be the first-order language over
  - the constants in Σ
  - the predicates of $\mathcal{A}$ plus the built-in predicates of relational algebra (e.g., <, >, ...)
  - no function symbols

Relational calculus: Syntax

Definition

An (domain) relational calculus query over alphabet $\mathcal{A}$ has the form

$$\{ (x_1, \ldots, x_n) \mid \varphi \},$$

where

- $n \geq 0$ is the arity of the query
- $x_1, \ldots, x_n$ are (not necessarily distinct) variables
- φ is the body of the query, i.e., a formula of $\mathcal{L}_\mathcal{A}$ whose free variables are exactly $x_1, \ldots, x_n$
- $(x_1, \ldots, x_n)$ is called the target list of the query

If r is a predicate of arity k, an atom with predicate r has the form $r(y_1, \ldots, y_k)$, where $y_1, \ldots, y_k$ are variables or constants

Relational calculus: Semantics

Relational calculus queries are evaluated on particular interpretations

Definition

A correct interpretation for relational calculus queries over $\mathcal{A}$ is a pair $\mathcal{I} = \langle \Delta, \mathcal{D} \rangle$, where Δ is a domain, and $\mathcal{D}$ is a database over $\mathcal{A}$ and Δ

Definition

The value of a relational calculus query $q = \{(x_1, \ldots, x_n) \mid \varphi\}$ in an interpretation $\mathcal{I} = \langle \Delta, \mathcal{D} \rangle$ is the set of tuples $(c_1, \ldots, c_n)$ of constants in Σ such that $\langle \mathcal{I}, \mathcal{V} \rangle \models \varphi$, where $\mathcal{V}$ is the variable assignment that assigns c_i to x_i

When the domain Δ is clear, we can omit it, and write directly $\langle \mathcal{D}, \mathcal{V} \rangle \models \varphi$, instead of $\langle \langle \Delta, \mathcal{D} \rangle, \mathcal{V} \rangle \models \varphi$

Relational calculus: Semantics

Relational calculus queries are evaluated on particular interpretations

Definition

A correct interpretation for relational calculus queries over $\mathcal{A}$ is a pair $\mathcal{I} = \langle \Delta, \mathcal{D} \rangle$, where Δ is a domain, and $\mathcal{D}$ is a database over $\mathcal{A}$ and Δ

Definition

The value of a relational calculus query $q = \{(x_1, \ldots, x_n) \mid \varphi\}$ in an interpretation $\mathcal{I} = \langle \Delta, \mathcal{D} \rangle$ is the set of tuples $(c_1, \ldots, c_n)$ of constants in Σ such that $\langle \mathcal{I}, \mathcal{V} \rangle \models \varphi$, where $\mathcal{V}$ is the variable assignment that assigns c_i to x_i

When the domain Δ is clear, we can omit it, and write directly $\langle \mathcal{D}, \mathcal{V} \rangle \models \varphi$, instead of $\langle \langle \Delta, \mathcal{D} \rangle, \mathcal{V} \rangle \models \varphi$

Relational calculus: Semantics

Relational calculus queries are evaluated on particular interpretations

Definition

A correct interpretation for relational calculus queries over $\mathcal{A}$ is a pair $\mathcal{I} = \langle \Delta, \mathcal{D} \rangle$, where Δ is a domain, and $\mathcal{D}$ is a database over $\mathcal{A}$ and Δ

Definition

The value of a relational calculus query $q = \{(x_1, \ldots, x_n) \mid \varphi\}$ in an interpretation $\mathcal{I} = \langle \Delta, \mathcal{D} \rangle$ is the set of tuples $(c_1, \ldots, c_n)$ of constants in Σ such that $\langle \mathcal{I}, \mathcal{V} \rangle \models \varphi$, where $\mathcal{V}$ is the variable assignment that assigns c_i to x_i

When the domain Δ is clear, we can omit it, and write directly $\langle \mathcal{D}, \mathcal{V} \rangle \models \varphi$, instead of $\langle \langle \Delta, \mathcal{D} \rangle, \mathcal{V} \rangle \models \varphi$

Result of relational calculus queries

Definition

The result of the evaluation of a relational calculus query $q = \{(x_1, \ldots, x_n) \mid \varphi\}$ on a database $\mathcal{D}$ over $\mathcal{A}$ and Δ is the relation $q^{\mathcal{D}}$ such that

- the arity of $q^{\mathcal{D}}$ is n
- the extension of $q^{\mathcal{D}}$ is the set of constants that constitute the value of the query q in the interpretation $\langle \Delta, \mathcal{D} \rangle$

Conjunctive queries

- are the most common kind of relational calculus queries
- also known as select-project-join SQL queries
- allow for easy optimization in relational DBMSs

Definition

A conjunctive query (CQ) is a relational calculus query of the form

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

where

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

We use the following abbreviation: $\{ (\vec{x}) \mid r_1(\vec{x}_1, \vec{y}_1), \ldots, r_m(\vec{x}_m, \vec{y}_m) \}$

Conjunctive queries

- are the most common kind of relational calculus queries
- also known as select-project-join SQL queries
- allow for easy optimization in relational DBMSs

Definition

A conjunctive query (CQ) is a relational calculus query of the form

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

where

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

We use the following abbreviation: $\{ (\vec{x}) \mid r_1(\vec{x}_1, \vec{y}_1), \ldots, r_m(\vec{x}_m, \vec{y}_m) \}$

Complexity of relational calculus

We consider the complexity of the recognition problem, i.e., checking whether a tuple of constants is in the answer to a query:

- measured wrt the size of the database $\rightsquigarrow$ data complexity
- measured wrt the size of the query and the database $\rightsquigarrow$ combined complexity

Complexity of relational calculus

- data complexity: polynomial, actually in LogSpace
- combined complexity: PSpace-complete

Complexity of conjunctive queries

- data complexity: in LogSpace
- combined complexity: NP-complete

Complexity of relational calculus

We consider the complexity of the recognition problem, i.e., checking whether a tuple of constants is in the answer to a query:

- measured wrt the size of the database $\rightsquigarrow$ data complexity
- measured wrt the size of the query and the database $\rightsquigarrow$ combined complexity

Complexity of relational calculus

- data complexity: polynomial, actually in LogSpace
- combined complexity: PSpace-complete

Complexity of conjunctive queries

- data complexity: in LogSpace
- combined complexity: NP-complete

Complexity of relational calculus

We consider the complexity of the recognition problem, i.e., checking whether a tuple of constants is in the answer to a query:

- measured wrt the size of the database $\rightsquigarrow$ data complexity
- measured wrt the size of the query and the database $\rightsquigarrow$ combined complexity

Complexity of relational calculus

- data complexity: polynomial, actually in LogSpace
- combined complexity: PSpace-complete

Complexity of conjunctive queries

- data complexity: in LogSpace
- combined complexity: NP-complete

Outline

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 $\leadsto$ 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 $\mathcal{C}$, there may be many global databases $\mathcal{B}$ in $sem^{\mathcal{C}}(\mathcal{I})$
- We consider those answers to q that hold for all global databases in $sem^{\mathcal{C}}(\mathcal{I})$
  $\leadsto$ certain answers

Semantics of queries to $\mathcal{I}$

Definition

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

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

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

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

- [Reiter '84]: relational setting, databases with incomplete information modeled as a first order theory
- [Vardi '86]: relational setting, complexity of reasoning in closed world databases with unknown values
- Several approaches both from the DB and the KR community
- [van der Meyden '98]: survey on logical approaches to incomplete information in databases

Outline

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:

$r_1(Title, Year, Director)$ since 1960, european directors

Source 2:

$r_2(Title, Critique)$ since 1990

Query: Title and critique of movies in 1998

$\{ (t, r) \mid \exists d.\ \text{movie}(t, 1998, d) \wedge \text{review}(t, r) \}$, abbreviated
$\{ (t, r) \mid \text{movie}(t, 1998, d),\ \text{review}(t, r) \}$

Outline

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

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

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 database 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:

$$\{ (t, y, d) \mid \mathsf{r}_1(t, y, d) \} \rightsquigarrow \mathsf{movie}(t, y, d)$$
$$\{ (d) \mid \mathsf{r}_1(t, y, d) \} \rightsquigarrow \mathsf{european}(d)$$
$$\{ (t, r) \mid \mathsf{r}_2(t, r) \} \rightsquigarrow \mathsf{review}(t, r)$$

Logical formalization:

$$\forall t, y, d.\ \mathsf{r}_1(t, y, d) \rightarrow \mathsf{movie}(t, y, d)$$
$$\forall d.\ (\exists t, y.\ \mathsf{r}_1(t, y, d)) \rightarrow \mathsf{european}(d)$$
$$\forall t, r.\ \mathsf{r}_2(t, r) \rightarrow \mathsf{review}(t, r)$$

GAV – Example of query processing

The query

$$\{ (t, r) \mid \mathsf{movie}(t, 1998, d), \ \mathsf{review}(t, r) \}$$

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:

$$\begin{array}{lccc} \{ (t, r) \mid & \mathsf{movie}(t, 1998, d), & \mathsf{review}(t, r) \} \\ \text{unfolding} & \downarrow & \downarrow \\ \{ (t, r) \mid & \mathsf{r}_1(t, 1998, d), & \mathsf{r}_2(t, r) \} \end{array}$$

GAV – Example of constraints

Global schema containing constraints:

movie($Title$, $Year$, $Director$)
european($Director$)
review($Title$, $Critique$)
european_movie_60s($Title$, $Year$, $Director$)

$\forall t, y, d.$ european_movie_60s$(t, y, d) \rightarrow$ movie(t, y, d)
$\forall d.\ \exists t, y.$ european_movie_60s$(t, y, d) \rightarrow$ european(d)

GAV mappings:

$\{ (t, y, d) \mid \mathsf{r}_1(t, y, d) \} \rightsquigarrow$ european_movie_60s(t, y, d)
$\{ (d) \mid \mathsf{r}_1(t, y, d) \} \rightsquigarrow$ european(d)
$\{ (t, r) \mid \mathsf{r}_2(t, r) \} \rightsquigarrow$ review(t, r)

Outline

Formalization of LAV

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

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

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

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

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

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 $\mathcal{C}$ 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:

$$\mathsf{r}_1(t, y, d) \rightsquigarrow \{ (t, y, d) \mid \mathsf{movie}(t, y, d),\ \mathsf{european}(d),\ y \geq 1960 \}$$
$$\mathsf{r}_2(t, r) \rightsquigarrow \{ (t, r) \mid \mathsf{movie}(t, y, d),\ \mathsf{review}(t, r),\ y \geq 1990 \}$$

The query $\{ (t, r) \mid \mathsf{movie}(t, 1998, d),\ \mathsf{review}(t, r) \}$ 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:

$$\{ (t, r) \mid \mathsf{r}_2(t, r),\ \mathsf{r}_1(t, 1998, d) \}$$

GAV and LAV – Comparison

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

- 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:

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

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

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

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

As for 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 $\mathcal{C}$

GLAV – Example

Global schema: $\mathsf{work}(Person, Project)$, $\mathsf{area}(Project, Field)$

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

GLAV mapping:

$$\{(r, f) \mid \mathsf{hasjob}(r, f)\} \rightsquigarrow \{(r, f) \mid \mathsf{work}(r, p),\ \mathsf{area}(p, f)\}$$
$$\{(r, f) \mid \mathsf{teaches}(r, c),\ \mathsf{in}(c, f)\} \rightsquigarrow \{(r, f) \mid \mathsf{work}(r, p),\ \mathsf{area}(p, f)\}$$
$$\{(r, p) \mid \mathsf{get}(r, g),\ \mathsf{for}(g, p)\} \rightsquigarrow \{(r, f) \mid \mathsf{work}(r, p)\}$$

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 (not to be used in the queries) of the same arity as the two queries and replace the assertion with the following two:

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

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

Part II

Query answering in the absence of constraints

Outline

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 definition of a conjunctive query:

Definition

A conjunctive query (CQ) is a relational calculus query of the form

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

where

- $\vec{x}$ is the union of the $\vec{x}_i$'s, and $\vec{y}$ is the union of the $\vec{y}_i$'s
- $r_1, \ldots, r_m$ 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

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

Outline

GAV – Retrieved global database

Definition

Given a source database $\mathcal{C}$, we call retrieved global database, denoted $\mathcal{M}(\mathcal{C})$, 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 $\mathsf{s}_1(Scode, Sname, City, Age)$,
$\mathsf{s}_2(Ucode, Uname)$, $\mathsf{s}_3(Scode, Ucode)$

Mapping $\mathcal{M}$:

$$\{ (c, n, ci) \mid \mathsf{s}_1(c, n, ci, a) \} \rightsquigarrow \mathsf{student}(c, n, ci)$$
$$\{ (c, n) \mid \mathsf{s}_2(c, n) \} \rightsquigarrow \mathsf{university}(c, n)$$
$$\{ (s, u) \mid \mathsf{s}_3(s, u) \} \rightsquigarrow \mathsf{enrolled}(s, u)$$

GAV – Example of retrieved global database

university

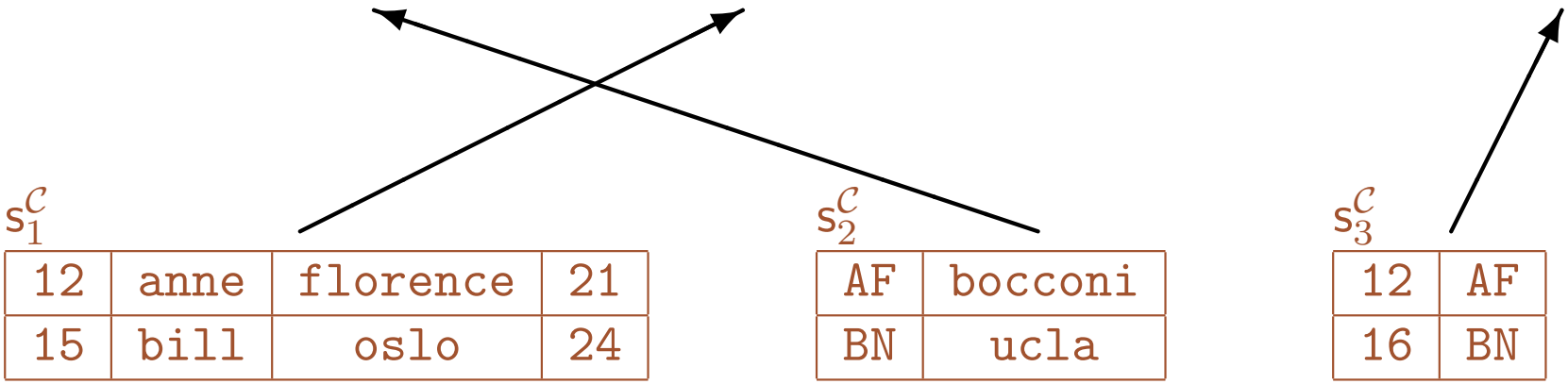
Code	*Name*
AF	bocconi
BN	ucla

student

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

enrolled

Scode	*Ucode*
12	AF
16	BN



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

GAV – Minimal model

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

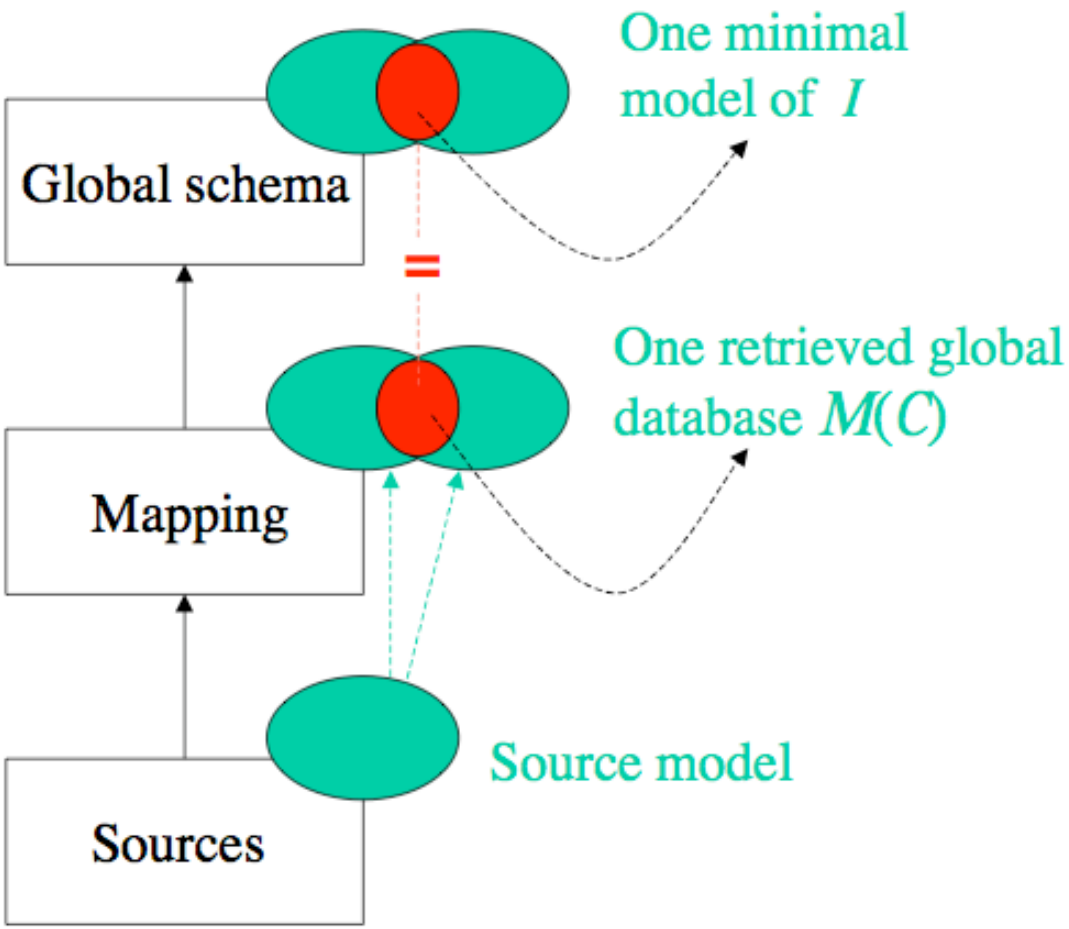
$$\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow g(\vec{x})$$

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

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

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

GAV without constraints



Outline

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:

- $unf_{\mathcal{M}}(q)$ is a query over $\mathcal{S}$
- Evaluating q over $\mathcal{M}(\mathcal{C})$ is equivalent to evaluating $unf_{\mathcal{M}}(q)$ over $\mathcal{C}$. i.e., $\vec{t} \in q^{\mathcal{M}(\mathcal{C})}$ iff $\vec{t} \in unf_{\mathcal{M}}(q)^{\mathcal{C}}$
- If q is a conjunctive query, then $\vec{t} \in cert(q, \mathcal{I}, \mathcal{C})$ iff $\vec{t} \in q^{\mathcal{M}(\mathcal{C})}$
- Hence, $\vec{t} \in cert(q, \mathcal{I}, \mathcal{C})$ iff $\vec{t} \in q^{\mathcal{M}(\mathcal{C})}$ iff $\vec{t} \in unf_{\mathcal{M}}(q)^{\mathcal{C}}$
  $\rightsquigarrow$ Unfolding suffices for query answering in GAV without constraints
- Data complexity of query answering is polynomial (actually LogSpace): the query $unf_{\mathcal{M}}(q)$ is first-order (in fact conjunctive)

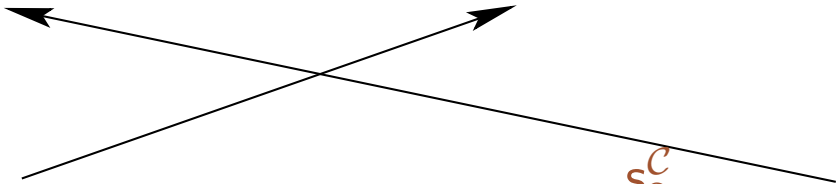
GAV – Example of unfolding

university

Code	*Name*
AF	bocconi
BN	ucla

student

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



$s_1^{\mathcal{C}}$

12	anne	florence	21
15	bill	oslo	24

$s_2^{\mathcal{C}}$

AF	bocconi
BN	ucla

GAV – Example of unfolding

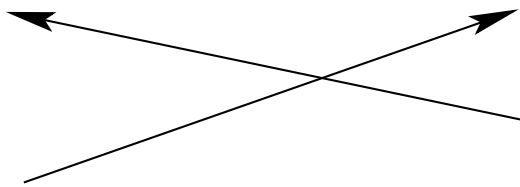
university

Code	*Name*
AF	bocconi
BN	ucla

student

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

$\{ x \mid \text{student}(15, x, y) \}$



$s_1^{\mathcal{C}}$

12	anne	florence	21
15	bill	oslo	24

$s_2^{\mathcal{C}}$

AF	bocconi
BN	ucla

GAV – Example of unfolding

university

Code	*Name*
AF	bocconi
BN	ucla

student

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

$\{ x \mid \mathsf{student}(15, x, y) \}$

unfolding

$\{ x \mid \mathsf{s}_1(15, x, y, z) \}$

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

12	anne	florence	21
15	bill	oslo	24

$\mathsf{s}_2^{\mathcal{C}}$

AF	bocconi
BN	ucla

GAV – More expressive queries?

Do the 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 [van der Meyden TCS'93]

GAV – More expressive queries?

Do the 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 [van der Meyden TCS'93]

GAV – More expressive queries?

Do the 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 [van der Meyden TCS'93]

Outline

Homomorphisms

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two databases with values in $\Delta \cup \mathsf{Var}$

Definition

A homomorphism $h : \mathcal{D}_1 \rightarrow \mathcal{D}_2$ is a mapping from $(\Delta \cup \mathsf{Var}(\mathcal{D}_1))$ to $(\Delta \cup \mathsf{Var}(\mathcal{D}_2))$ such that

1. $h(c) = c$, for every constant $c \in \Delta$
2. each variable can be mapped to a constant or a variable
3. for every fact $r_i(\vec{t})$ of $\mathcal{D}_1$, we have that $r_i(h(\vec{t}))$ is a fact in $\mathcal{D}_2$

Definition

$\mathcal{D}_1$ is homomorphically equivalent to $\mathcal{D}_2$ if there is a homomorphism $h : \mathcal{D}_1 \rightarrow \mathcal{D}_2$ and a homomorphism $h' : \mathcal{D}_2 \rightarrow \mathcal{D}_1$

Conjunctive query answering and homomorphisms

Consider a conjunctive query

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

For a tuple $\vec{c}$ of constants, let $\mathcal{D}_q^{\vec{c}}$ be the set of facts (over constants and variables in $\vec{y}$) obtained by replacing in $r_1(\vec{x}_1, \vec{y}_1), \ldots, r_m(\vec{x}_m, \vec{y}_m)$ each x_i with c_i.

Note that $\mathcal{D}_q^{\vec{c}}$ can be viewed as a database with values in $\Delta \cup \mathsf{Var}$

Theorem (Chandra & Merlin '77)

$\vec{c} \in q^{\mathcal{D}}$ if and only if there exists a homomorphism $h : \mathcal{D}_q^{\vec{c}} \to \mathcal{D}$

Hence, homomorphisms preserve satisfaction of conjunctive queries: if there exists a homomorphism $h : \mathcal{D} \to \mathcal{D}'$, then $\vec{t} \in q^{\mathcal{D}}$ implies $\vec{t} \in q^{\mathcal{D}'}$, for each conjunctive query q

Conjunctive query answering and homomorphisms

Consider a conjunctive query

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

For a tuple $\vec{c}$ of constants, let $\mathcal{D}_q^{\vec{c}}$ be the set of facts (over constants and variables in $\vec{y}$) obtained by replacing in $r_1(\vec{x}_1, \vec{y}_1), \ldots, r_m(\vec{x}_m, \vec{y}_m)$ each x_i with c_i.

Note that $\mathcal{D}_q^{\vec{c}}$ can be viewed as a database with values in $\Delta \cup \mathrm{Var}$

Theorem (Chandra & Merlin '77)

$\vec{c} \in q^{\mathcal{D}}$ if and only if there exists a homomorphism $h : \mathcal{D}_q^{\vec{c}} \to \mathcal{D}$

Hence, homomorphisms preserve satisfaction of conjunctive queries: if there exists a homomorphism $h : \mathcal{D} \to \mathcal{D}'$, then $\vec{t} \in q^{\mathcal{D}}$ implies $\vec{t} \in q^{\mathcal{D}'}$, for each conjunctive query q

Conjunctive query answering and homomorphisms

Consider a conjunctive query

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

For a tuple $\vec{c}$ of constants, let $\mathcal{D}_q^{\vec{c}}$ be the set of facts (over constants and variables in $\vec{y}$) obtained by replacing in $r_1(\vec{x}_1, \vec{y}_1), \ldots, r_m(\vec{x}_m, \vec{y}_m)$ each x_i with c_i.

Note that $\mathcal{D}_q^{\vec{c}}$ can be viewed as a database with values in $\Delta \cup \mathsf{Var}$

Theorem (Chandra & Merlin '77)

$\vec{c} \in q^{\mathcal{D}}$ if and only if there exists a homomorphism $h : \mathcal{D}_q^{\vec{c}} \to \mathcal{D}$

Hence, homomorphisms preserve satisfaction of conjunctive queries: if there exists a homomorphism $h : \mathcal{D} \to \mathcal{D}'$, then $\vec{t} \in q^{\mathcal{D}}$ implies $\vec{t} \in q^{\mathcal{D}'}$, for each conjunctive query q

GAV – Universal solutions

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

Definition

A universal solution for $\mathcal{I}$ relative to $\mathcal{C}$ is a global db $\mathcal{B}$ that satisfies $\mathcal{I}$ relative to $\mathcal{C}$ such that, for every global db $\mathcal{B}'$ that satisfies $\mathcal{I}$ relative to $\mathcal{C}$, there exists a homomorphism $h : \mathcal{B} \rightarrow \mathcal{B}'$ (see [Fagin & al. TCS'05])

Theorem

If $\mathcal{I}$ is GAV without constraints in the global schema, then $\mathcal{M}(\mathcal{C})$ is the minimal universal solution for $\mathcal{I}$ relative to $\mathcal{C}$

We derive again the following results (see Phokion's course):

Theorem

- *If q is a conjunctive query, then $\vec{t} \in cert(q, \mathcal{I}, \mathcal{C})$ iff $\vec{t} \in q^{\mathcal{M}(\mathcal{C})}$*
- *Complexity of query answering is polynomial*

Outline

(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

Outline

(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 $\mathsf{s}_1(Scode, Sname, City, Age)$

Mapping $\mathcal{M}$:

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

(G)LAV – Example

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

student

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

enrolled

Scode	*Ucode*
12	x
15	y

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

12	anne	florence	21
15	bill	oslo	24

A source db $\mathcal{C}$ and a corresponding possible retrieved global db $\mathcal{M}(\mathcal{C})$

(G)LAV – Example

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

student

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

enrolled

Scode	*Ucode*
12	x
15	x

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

12	anne	florence	21
15	bill	oslo	24

A source db $\mathcal{C}$ and another possible retrieved global db $\mathcal{M}(\mathcal{C})$

(G)LAV – Incompleteness

(G)LAV mapping assertions $\phi_S \rightsquigarrow \phi_G$ have the logical form:

$$\forall \vec{x}.\ \phi_S(\vec{x}) \rightarrow \exists \vec{y}.\ \phi_G(\vec{x}, \vec{y})$$

where ϕ_S and ϕ_G are conjunctions of atoms

Given a source database $\mathcal{C}$, 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 $\mathcal{C}$)
$\rightsquigarrow$ Incompleteness comes from the mapping

This holds even for the case of very simple queries ϕ_G:

$$s_1(x) \rightsquigarrow \{ (x) \mid \exists y.\ g(x, y) \}$$

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

Outline

(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 $\mathcal{C}$
- 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 $\mathcal{C}$

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

(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 $\mathcal{C}$
- 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 $\mathcal{C}$

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

(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 $\mathcal{C}$
- 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 $\mathcal{C}$

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

(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 $\mathcal{C}$
- 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 $\mathcal{C}$

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

Connection between query answering and containment

Definition

Query containment (under constraints) is the problem of checking whether $q_1^{\mathcal{P}}$ is contained in $q_2^{\mathcal{P}}$ for every database $\mathcal{D}$ (satisfying the constraints), where q_1, q_2 are queries of the same arity

Query answering can be rephrased in terms of query containment:

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

Connection between query answering and containment

Definition

Query containment (under constraints) is the problem of checking whether $q_1^{\mathcal{D}}$ is contained in $q_2^{\mathcal{D}}$ for every database $\mathcal{D}$ (satisfying the constraints), where q_1, q_2 are queries of the same arity

Query answering can be rephrased in terms of query containment:

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

$\rightsquigarrow$ *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 unify, i.e., to denote the same object.

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

$\rightsquigarrow$ *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 unify, i.e., to denote the same object.

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

$\rightsquigarrow$ *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 unify, i.e., to denote the same object.

Outline

(G)LAV – Basic technique

From [Duschka & Genesereth PODS'97]:

$$\begin{aligned} \mathsf{r}_1(t) &\rightsquigarrow \{\ (t) \mid \mathsf{movie}(t,y,d) \wedge \mathsf{european}(d)\ \} \\ \mathsf{r}_2(t,v) &\rightsquigarrow \{\ (t,v) \mid \mathsf{movie}(t,y,d) \wedge \mathsf{review}(t,v)\ \} \end{aligned}$$

$$\begin{aligned} \forall t.\ \mathsf{r}_1(t) &\rightarrow \exists y,d.\ \mathsf{movie}(t,y,d) \wedge \mathsf{european}(d) \\ \forall t,v.\ \mathsf{r}_2(t,v) &\rightarrow \exists y,d.\ \mathsf{movie}(t,y,d) \wedge \mathsf{review}(t,v) \end{aligned}$$

$$\begin{aligned} \mathsf{movie}(t, f_1(t), f_2(t)) &\leftarrow \mathsf{r}_1(t) \\ \mathsf{european}(f_2(t)) &\leftarrow \mathsf{r}_1(t) \\ \mathsf{movie}(t, f_4(t,v), f_5(t,v)) &\leftarrow \mathsf{r}_2(t,v) \\ \mathsf{review}(t,v) &\leftarrow \mathsf{r}_2(t,v) \end{aligned}$$

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

$\rightsquigarrow$ Data complexity is polynomial (actually LogSpace)

(G)LAV – Canonical retrieved global database

What is a retrieved global database in this case?

Definition

We build what we call the canonical retrieved global database for $\mathcal{I}$ relative to $\mathcal{C}$, denoted $\mathcal{M}(\mathcal{C})\downarrow$, as follows:

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

Properties of $\mathcal{M}(\mathcal{C})\downarrow$ (also called canonical model of $\mathcal{I}$ relative to $\mathcal{C}$)

- It is unique up to variable renaming
- It can be computed in polynomial time wrt the size of $\mathcal{C}$
- Since there are no constraints in $\mathcal{G}$, it obviously satisfies $\mathcal{G}$

(G)LAV – Example of canonical model

$$\{ (c, n, ci) \mid \mathsf{s}_1(c, n, ci, a) \} \rightsquigarrow \{ (c, n, ci) \mid \mathsf{student}(c, n, ci) \wedge \mathsf{enrolled}(c, u) \}$$

student

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

enrolled

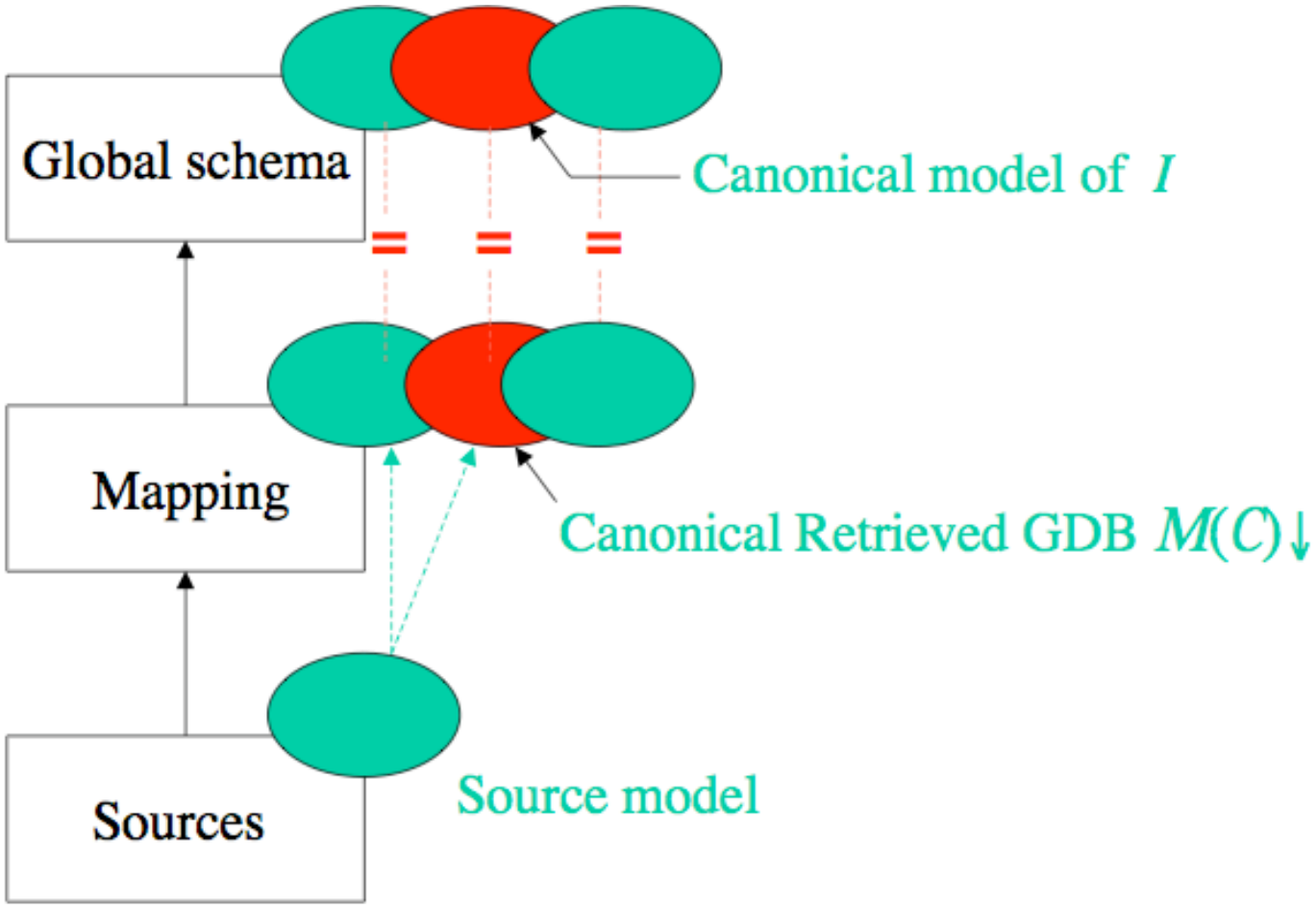
Scode	*Ucode*
12	x
15	y

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

12	anne	florence	21
15	bill	oslo	24

Example of source db $\mathcal{C}$ and corresponding canonical model $\mathcal{M}(\mathcal{C})\downarrow$

(G)LAV – Canonical model



(G)LAV – Universal solution

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 $\mathcal{C}$ a source database

Theorem

Then $\mathcal{M}(\mathcal{C})\downarrow$ is a universal solution for $\mathcal{I}$ relative to $\mathcal{C}$ (follows from [Fagin&al. ICDT'03])

It follows that:

- If q is a conjunctive query, then $\vec{t} \in cert(q, \mathcal{I}, \mathcal{C})$ iff $\vec{t} \in q^{\mathcal{M}(\mathcal{C})\downarrow}$
- Complexity of query answering is polynomial, actually LogSpace

(G)LAV – Universal solution

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 $\mathcal{C}$ a source database

Theorem

Then $\mathcal{M}(\mathcal{C})\downarrow$ is a universal solution for $\mathcal{I}$ relative to $\mathcal{C}$ (follows from [Fagin&al. ICDT'03])

It follows that:

- If q is a conjunctive query, then $\vec{t} \in cert(q, \mathcal{I}, \mathcal{C})$ iff $\vec{t} \in q^{\mathcal{M}(\mathcal{C})\downarrow}$
- Complexity of query answering is polynomial, actually LogSpace

(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 positive 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 – 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 positive 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 – 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 positive 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 positive views

From [van der Meyden TCS'93], by reduction from 3-colorability

We define the following LAV data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$:

$\mathcal{G}$: $\mathsf{edge}(x, y)$, $\mathsf{color}(x, c)$ $\quad$ $\mathcal{S}$: $\mathsf{s}_E(x, y)$, $\mathsf{s}_N(x)$

$\mathcal{M}$: $\mathsf{s}_E(x, y) \rightsquigarrow \mathsf{edge}(x, y)$

$\mathsf{s}_N(x) \rightsquigarrow \mathsf{color}(x, \mathtt{RED}) \vee \mathsf{color}(x, \mathtt{BLUE}) \vee \mathsf{color}(x, \mathtt{GREEN})$

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

$\mathsf{s}_E{}^{\mathcal{C}} = \{ (a, b), (b, a) \mid (a, b) \in E \}$ $\quad$ $\mathsf{s}_N{}^{\mathcal{C}} = \{ (a) \mid a \in N \}$

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

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

Theorem

Data complexity is coNP-hard for positive views and conjunctive queries

(G)LAV – Intractability for positive views

From [van der Meyden TCS'93], by reduction from 3-colorability

We define the following LAV data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$:

$\mathcal{G}$: $\mathsf{edge}(x, y)$, $\mathsf{color}(x, c)$ $\quad$ $\mathcal{S}$: $\mathsf{s}_E(x, y)$, $\mathsf{s}_N(x)$

$\mathcal{M}$: $\mathsf{s}_E(x, y) \rightsquigarrow \mathsf{edge}(x, y)$

$\mathsf{s}_N(x) \rightsquigarrow \mathsf{color}(x, \mathtt{RED}) \vee \mathsf{color}(x, \mathtt{BLUE}) \vee \mathsf{color}(x, \mathtt{GREEN})$

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

$\mathsf{s}_E{}^{\mathcal{C}} = \{ (a, b), (b, a) \mid (a, b) \in E \}$ $\quad$ $\mathsf{s}_N{}^{\mathcal{C}} = \{ (a) \mid a \in N \}$

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

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

Theorem

Data complexity is coNP-hard for positive views and conjunctive queries

(G)LAV – Intractability for positive views

From [van der Meyden TCS'93], by reduction from 3-colorability

We define the following LAV data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$:

$\mathcal{G}$: $\mathsf{edge}(x, y)$, $\mathsf{color}(x, c)$ $\quad$ $\mathcal{S}$: $\mathsf{s}_E(x, y)$, $\mathsf{s}_N(x)$

$\mathcal{M}$: $\mathsf{s}_E(x, y) \rightsquigarrow \mathsf{edge}(x, y)$

$\mathsf{s}_N(x) \rightsquigarrow \mathsf{color}(x, \mathtt{RED}) \vee \mathsf{color}(x, \mathtt{BLUE}) \vee \mathsf{color}(x, \mathtt{GREEN})$

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

$\mathsf{s}_E{}^{\mathcal{C}} = \{ (a, b), (b, a) \mid (a, b) \in E \}$ $\quad$ $\mathsf{s}_N{}^{\mathcal{C}} = \{ (a) \mid a \in N \}$

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

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

Theorem

Data complexity is coNP-hard for positive views and conjunctive queries

(G)LAV – Intractability for positive views

From [van der Meyden TCS'93], by reduction from 3-colorability

We define the following LAV data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$:

$\mathcal{G}$: $\mathsf{edge}(x, y)$, $\mathsf{color}(x, c)$ $\quad$ $\mathcal{S}$: $\mathsf{s}_E(x, y)$, $\mathsf{s}_N(x)$

$\mathcal{M}$: $\mathsf{s}_E(x, y) \rightsquigarrow \mathsf{edge}(x, y)$

$\mathsf{s}_N(x) \rightsquigarrow \mathsf{color}(x, \mathtt{RED}) \vee \mathsf{color}(x, \mathtt{BLUE}) \vee \mathsf{color}(x, \mathtt{GREEN})$

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

$\mathsf{s}_E{}^{\mathcal{C}} = \{ (a, b), (b, a) \mid (a, b) \in E \}$ $\quad$ $\mathsf{s}_N{}^{\mathcal{C}} = \{ (a) \mid a \in N \}$

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

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

Theorem

Data complexity is coNP-hard for positive views and conjunctive queries

(G)LAV – Intractability for positive views

From [van der Meyden TCS'93], by reduction from 3-colorability

We define the following LAV data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$:

$\mathcal{G}$: $\mathsf{edge}(x, y)$, $\mathsf{color}(x, c)$ $\quad$ $\mathcal{S}$: $\mathsf{s}_E(x, y)$, $\mathsf{s}_N(x)$

$\mathcal{M}$: $\mathsf{s}_E(x, y) \rightsquigarrow \mathsf{edge}(x, y)$

$\mathsf{s}_N(x) \rightsquigarrow \mathsf{color}(x, \mathtt{RED}) \vee \mathsf{color}(x, \mathtt{BLUE}) \vee \mathsf{color}(x, \mathtt{GREEN})$

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

$\mathsf{s}_E{}^{\mathcal{C}} = \{ (a, b), (b, a) \mid (a, b) \in E \}$ $\quad$ $\mathsf{s}_N{}^{\mathcal{C}} = \{ (a) \mid a \in N \}$

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

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

Theorem

Data complexity is coNP-hard for positive views and conjunctive queries

(G)LAV – In coNP for positive views and queries

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

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

  Hence, each tuple in $\mathcal{C}$ forces the existence of k tuples in any database that satisfies $\mathcal{M}$ wrt $\mathcal{C}$, where k is the maximal length of conjunctions $\alpha_i(\vec{x}, \vec{y}_i)$ in $\mathcal{M}$
- If $\mathcal{C}$ has n tuples, then there is a db $\mathcal{B}' \subseteq \mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$ with at most $n \cdot k$ tuples. Since q is monotone, $\vec{t} \notin q^{\mathcal{B}'}$
- Checking whether $\mathcal{B}'$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$, and checking whether $\vec{t} \notin q^{\mathcal{B}'}$ can be done in PTIME wrt the size of $\mathcal{B}'$

Theorem

For positive views and queries, query answ. is coNP in data complexity

(G)LAV – In coNP for positive views and queries

- $\vec{t} \notin cert(q, \mathcal{I}, \mathcal{C})$ if and only if there is a database $\mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$, and such that $\vec{t} \notin q^{\mathcal{B}}$
- 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 $\mathcal{C}$ forces the existence of k tuples in any database that satisfies $\mathcal{M}$ wrt $\mathcal{C}$, where k is the maximal length of conjunctions $\alpha_i(\vec{x}, \vec{y}_i)$ in $\mathcal{M}$
- If $\mathcal{C}$ has n tuples, then there is a db $\mathcal{B}' \subseteq \mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$ with at most $n \cdot k$ tuples. Since q is monotone, $\vec{t} \notin q^{\mathcal{B}'}$
- Checking whether $\mathcal{B}'$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$, and checking whether $\vec{t} \notin q^{\mathcal{B}'}$ can be done in PTIME wrt the size of $\mathcal{B}'$

Theorem

For positive views and queries, query answ. is coNP in data complexity

(G)LAV – In coNP for positive views and queries

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

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

  Hence, each tuple in $\mathcal{C}$ forces the existence of k tuples in any database that satisfies $\mathcal{M}$ wrt $\mathcal{C}$, where k is the maximal length of conjunctions $\alpha_i(\vec{x}, \vec{y}_i)$ in $\mathcal{M}$
- If $\mathcal{C}$ has n tuples, then there is a db $\mathcal{B}' \subseteq \mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$ with at most $n \cdot k$ tuples. Since q is monotone, $\vec{t} \notin q^{\mathcal{B}'}$
- Checking whether $\mathcal{B}'$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$, and checking whether $\vec{t} \notin q^{\mathcal{B}'}$ can be done in PTIME wrt the size of $\mathcal{B}'$

Theorem

For positive views and queries, query answ. is coNP in data complexity

(G)LAV – In coNP for positive views and queries

- $\vec{t} \notin cert(q, \mathcal{I}, \mathcal{C})$ if and only if there is a database $\mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$, and such that $\vec{t} \notin q^{\mathcal{B}}$
- 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 $\mathcal{C}$ forces the existence of k tuples in any database that satisfies $\mathcal{M}$ wrt $\mathcal{C}$, where k is the maximal length of conjunctions $\alpha_i(\vec{x}, \vec{y}_i)$ in $\mathcal{M}$
- If $\mathcal{C}$ has n tuples, then there is a db $\mathcal{B}' \subseteq \mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$ with at most $n \cdot k$ tuples. Since q is monotone, $\vec{t} \notin q^{\mathcal{B}'}$
- Checking whether $\mathcal{B}'$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$, and checking whether $\vec{t} \notin q^{\mathcal{B}'}$ can be done in PTIME wrt the size of $\mathcal{B}'$

Theorem

For positive views and queries, query answ. is coNP in data complexity

(G)LAV – In coNP for positive views and queries

- $\vec{t} \notin cert(q, \mathcal{I}, \mathcal{C})$ if and only if there is a database $\mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$, and such that $\vec{t} \notin q^{\mathcal{B}}$
- 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 $\mathcal{C}$ forces the existence of k tuples in any database that satisfies $\mathcal{M}$ wrt $\mathcal{C}$, where k is the maximal length of conjunctions $\alpha_i(\vec{x}, \vec{y}_i)$ in $\mathcal{M}$
- If $\mathcal{C}$ has n tuples, then there is a db $\mathcal{B}' \subseteq \mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$ with at most $n \cdot k$ tuples. Since q is monotone, $\vec{t} \notin q^{\mathcal{B}'}$
- Checking whether $\mathcal{B}'$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$, and checking whether $\vec{t} \notin q^{\mathcal{B}'}$ can be done in PTIME wrt the size of $\mathcal{B}'$

Theorem

For positive views and queries, query answ. is coNP in data complexity

(G)LAV – Conjunctive user queries with inequalities

Consider $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and source db $\mathcal{C}$ (see [Fagin & al. ICDT'03]):

$$\begin{array}{lll} \mathcal{G}: & g(x,y) & \mathcal{S}: \; s(x,y) \\ \mathcal{M}: & s(x,y) \rightsquigarrow \{ \; (x,y) \; \mid \; g(x,z) \wedge g(z,y) \; \} & \\ \mathcal{C}: & \{ \; s(\mathtt{a},\mathtt{a}) \; \} & \end{array}$$

- $\mathcal{B}_1 = \{ \; g(\mathtt{a},\mathtt{a}) \; \}$ is a solution
- If $\mathcal{B}$ is a universal solution, then both $g(\mathtt{a}, x)$ and $g(x, \mathtt{a})$ are in $\mathcal{B}$, with $x \neq \mathtt{a}$ (otherwise $g(\mathtt{a},\mathtt{a})$ would be $true$ in every solution)

Let $q = \{ \; () \; \mid \; g(x,y) \wedge x \neq y \; \}$

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

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

(G)LAV – Conjunctive user queries with inequalities

Consider $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and source db $\mathcal{C}$ (see [Fagin & al. ICDT'03]):

$$\begin{array}{lll} \mathcal{G}: & g(x,y) & \mathcal{S}: \ s(x,y) \\ \mathcal{M}: & s(x,y) \rightsquigarrow \{ \ (x,y) \ \mid \ g(x,z) \wedge g(z,y) \ \} & \\ \mathcal{C}: & \{ \ s(\mathtt{a},\mathtt{a}) \ \} & \end{array}$$

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

Let $q = \{ \ () \ \mid \ g(x,y) \wedge x \neq y \ \}$

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

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 two inequalities (and conjunctive view definitions) [Fagin & al. ICDT'03], see Phokion's course

Theorem

For conjunctive user queries with inequalities, (G)LAV query answering 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)

(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 two inequalities (and conjunctive view definitions) [Fagin & al. ICDT'03], see Phokion's course

Theorem

For conjunctive user queries with inequalities, (G)LAV query answering 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)

Outline

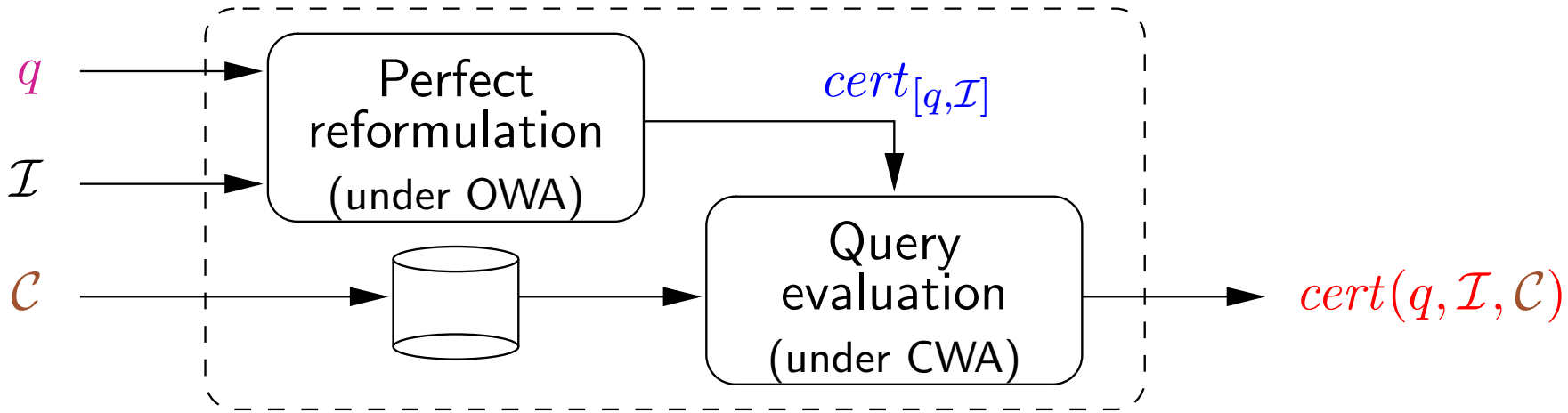
(G)LAV – View-based query rewriting

Query answering is divided in two steps:

1. Re-express the query in terms of a given query language over the alphabet of $\mathcal{A}_\mathcal{S}$
2. Evaluate the rewriting over the source database $\mathcal{C}$

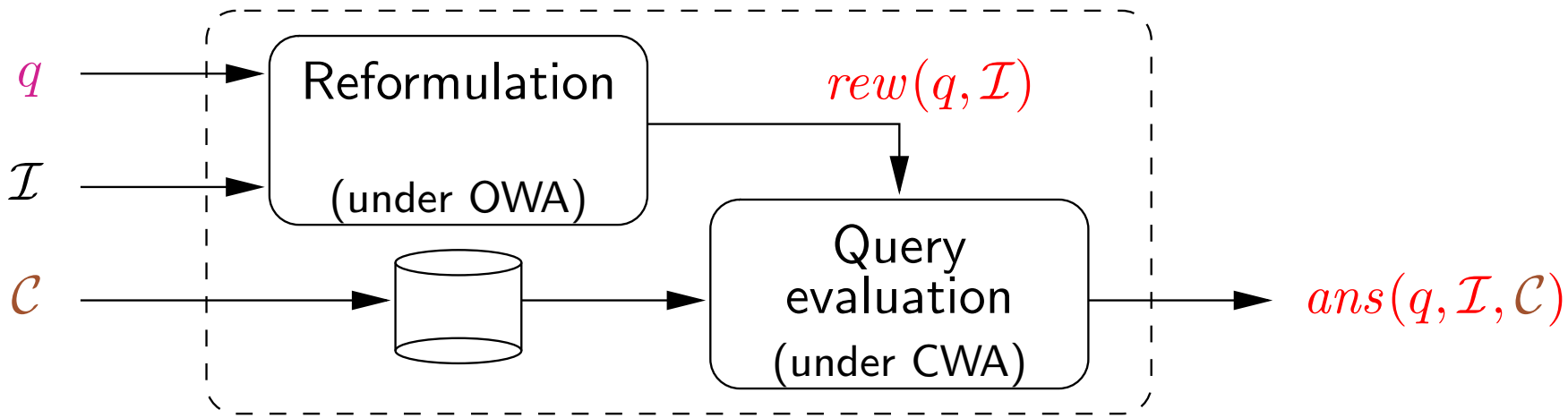
Query answering

Query answering: reformulation + evaluation



The query $cert_{[q,\mathcal{I}]}$ could be expressed in an arbitrary query language

Query rewriting



The language of $rew(q, \mathcal{I})$ is chosen a priori!

(G)LAV – Connection to rewriting

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 $\mathcal{C}$

We are interested in rewritings that are:

- sound, i.e., compute only tuples in $cert(q, \mathcal{I}, \mathcal{C})$ for every $\mathcal{C}$
- expressed in a given query language $\mathcal{L}$
- maximal for the class of queries expressible in $\mathcal{L}$

We may be interested in exact rewritings, i.e., rewritings that are logically equivalent to the query, modulo $\mathcal{M}$

Exact rewritings may not exist

(G)LAV – Connection to rewriting

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 $\mathcal{C}$

We are interested in rewritings that are:

- sound, i.e., compute only tuples in $cert(q, \mathcal{I}, \mathcal{C})$ for every $\mathcal{C}$
- expressed in a given query language $\mathcal{L}$
- maximal for the class of queries expressible in $\mathcal{L}$

We may be interested in exact rewritings, i.e., rewritings that are logically equivalent to the query, modulo $\mathcal{M}$

Exact rewritings may not exist

(G)LAV – Connection to rewriting

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 $\mathcal{C}$

We are interested in rewritings that are:

- sound, i.e., compute only tuples in $cert(q, \mathcal{I}, \mathcal{C})$ for every $\mathcal{C}$
- expressed in a given query language $\mathcal{L}$
- maximal for the class of queries expressible in $\mathcal{L}$

We may be interested in exact rewritings, i.e., rewritings that are logically equivalent to the query, modulo $\mathcal{M}$

Exact rewritings may not exist

(G)LAV – Example of maximal rewriting

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

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

$\mathcal{M}$: flightsByUnited$(From, To) \rightsquigarrow$
nonstop$(\texttt{UA}, Num, From, To)$
flightsFromSFO$(Airline, Num, To) \rightsquigarrow$
nonstop$(Airline, Num, \texttt{SFO}, To)$

q: $\{ (airline, num) \mid \text{nonstop}(airline, num, \texttt{LAX}, \texttt{PHX}) \}$

A maximal (wrt positive queries) rewriting of q is:

$$\{ (\texttt{UA}, num) \mid \text{flightsByUnited}(num, \texttt{LAX}, \texttt{PHX}) \}$$

(G)LAV – Example of maximal rewriting

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

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

$\mathcal{M}$: flightsByUnited$(From, To) \rightsquigarrow$
nonstop$(\texttt{UA}, Num, From, To)$
flightsFromSFO$(Airline, Num, To) \rightsquigarrow$
nonstop$(Airline, Num, \texttt{SFO}, To)$

q: $\{ (airline, num) \mid \text{nonstop}(airline, num, \texttt{LAX}, \texttt{PHX}) \}$

A maximal (wrt positive queries) rewriting of q is:

$$\{ (\texttt{UA}, num) \mid \text{flightsByUnited}(num, \texttt{LAX}, \texttt{PHX}) \}$$

Perfect rewriting

What is the relationship between answering by rewriting and certain answers? [Calvanese & al. ICDT'05]:

- 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 $\mathcal{C}$, computes the certain answers $cert(q, \mathcal{I}, \mathcal{C})$.

- $cert_{[q,\mathcal{I}]}$ 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 $\mathcal{I}$

Perfect rewriting

What is the relationship between answering by rewriting and certain answers? [Calvanese & al. ICDT'05]:

- 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 $\mathcal{C}$, computes the certain answers $cert(q, \mathcal{I}, \mathcal{C})$.

- $cert_{[q,\mathcal{I}]}$ 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 $\mathcal{I}$

Perfect rewriting

What is the relationship between answering by rewriting and certain answers? [Calvanese & al. ICDT'05]:

- 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 $\mathcal{C}$, computes the certain answers $cert(q, \mathcal{I}, \mathcal{C})$.

- $cert_{[q,\mathcal{I}]}$ 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

Let q and the queries in $\mathcal{M}$ be conjunctive queries (CQs)
Let q' be the union of all maximal rewritings of q for the class of CQs

Theorem (Levy & al. PODS'95, Abiteboul & Duschka PODS'98)

- *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 on to cases where q and $\mathcal{M}$ allow for union?

(G)LAV – The case of conjunctive queries

Let q and the queries in $\mathcal{M}$ be conjunctive queries (CQs)
Let q' be the union of all maximal rewritings of q for the class of CQs

Theorem (Levy & al. PODS'95, Abiteboul & Duschka PODS'98)

- *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 on to cases where q and $\mathcal{M}$ allow for union?

(G)LAV – View-based query processing for UPQs

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 [van der Meyden TCS'93]
- hence, $cert(q, \mathcal{I}, \mathcal{C})$, 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 UPQs q and $\mathcal{I}$ for which the perfect rewriting $cert_{[q,\mathcal{I}]}$ is a PTIME function (assuming P≠NP) [Calvanese & al. LICS'00].

(G)LAV – View-based query processing for UPQs

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 [van der Meyden TCS'93]
- hence, $cert(q, \mathcal{I}, \mathcal{C})$, 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 UPQs q and $\mathcal{I}$ for which the perfect rewriting $cert_{[q,\mathcal{I}]}$ is a PTIME function (assuming P≠NP) [Calvanese & al. LICS'00].

(G)LAV – Data complexity of query answering

From [Abiteboul & Duschka PODS'98], 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 [Duschka & Genesereth PODS'97]
- Bucket algorithm for query rewriting [Levy & al. AAAI'96]
- MiniCon algorithm for query rewriting [Pottinger & Levy VLDB'00]
- Conjunctive queries using conjunctive views [Levy & al. PODS'95]
- Recursive queries (Datalog programs) using conjunctive views [Duschka & Genesereth PODS'97; Afrati & al. ICDT'99]
- CQs with arithmetic comparison [Afrati & al. PODS'01]
- Complexity analysis [Abiteboul & Duschka PODS'98; Grahne & Mendelzon ICDT'99]
- Variants of Regular Path Queries [Calvanese & al. ICDE'00, PODS'00, DBPL'01; Deutsch & Tannen DBPL'01],
- Relationship between view-based rewriting and answering [Calvanese & al. LICS'00, PODS'03, ICDT'05]

Part III

Query answering in the presence of constraints

Outline

Outline

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
- Many open issues

Integrity constraints for relational schemas

Most important types of ICs for the relational model:

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

Integrity constraints for relational schemas

Most important types of ICs for the relational model:

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

Integrity constraints for relational schemas

Most important types of ICs 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 $\vec{t}_1$

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

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 $\vec{t}_1$

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

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 $\vec{t}_1$

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

Syntax of key dependencies

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

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

Example

For r of arity 3, the KD $key(r) = \{1\}$ 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

Key dependencies (KDs)

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

Syntax of key dependencies

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

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

Example

For r of arity 3, the KD $key(r) = \{1\}$ 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

Key dependencies (KDs)

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

Syntax of key dependencies

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

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

Example

For r of arity 3, the KD $key(r) = \{1\}$ 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 $\vec{t}_1$

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

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 $\vec{t}_1$

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

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 $\vec{t}_1$

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

Outline

Outline

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 $\phi_{\mathcal{S}} \rightsquigarrow g$ 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 $\mathcal{M}(\mathcal{C})$ may not be legal for $\mathcal{G}$

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 $\phi_{\mathcal{S}} \rightsquigarrow g$ 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 $\mathcal{M}(\mathcal{C})$ may not be legal for $\mathcal{G}$

Semantics of GAV systems with integrity constraints

Given a source db $\mathcal{C}$, a global db $\mathcal{B}$ (over Δ) satisfies $\mathcal{I}$ relative to $\mathcal{C}$ 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 $\mathcal{M}(\mathcal{C})$ (sound mappings)

Recall:

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

Semantics of GAV systems with integrity constraints

Given a source db $\mathcal{C}$, a global db $\mathcal{B}$ (over Δ) satisfies $\mathcal{I}$ relative to $\mathcal{C}$ 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 $\mathcal{M}(\mathcal{C})$ (sound mappings)

Recall:

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

GAV with constraints – Example

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

$\mathcal{G}$: student$(Code, Name, City)$ $\quad key(\text{student}) = \{Code\}$
university$(Code, Name)$ $\quad key(\text{university}) = \{Code\}$
enrolled$(Scode, Ucode)$

$\text{enrolled}[Scode] \subseteq \text{student}[Code]$
$\text{enrolled}[Ucode] \subseteq \text{university}[Code]$

Source schema $\mathcal{S}$: $\mathsf{s}_1(Scode, Sname, City, Age)$,
$\mathsf{s}_2(Ucode, Uname)$, $\quad \mathsf{s}_3(Scode, Ucode)$

Mapping $\mathcal{M}$:
$$\{ (c, n, ci) \mid \mathsf{s}_1(c, n, ci, a) \} \rightsquigarrow \text{student}(c, n, ci)$$
$$\{ (c, n) \mid \mathsf{s}_2(c, n) \} \rightsquigarrow \text{university}(c, n)$$
$$\{ (s, u) \mid \mathsf{s}_3(s, u) \} \rightsquigarrow \text{enrolled}(s, u)$$

GAV with constraints – Example

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

$\mathcal{G}$: student$(Code, Name, City)$ $\quad$ *key*(student) $= \{Code\}$
university$(Code, Name)$ $\quad$ *key*(university) $= \{Code\}$
enrolled$(Scode, Ucode)$

enrolled$[Scode] \subseteq$ student$[Code]$
enrolled$[Ucode] \subseteq$ university$[Code]$

Source schema $\mathcal{S}$: $\quad$ $\mathsf{s}_1(Scode, Sname, City, Age)$,
$\mathsf{s}_2(Ucode, Uname)$, $\quad$ $\mathsf{s}_3(Scode, Ucode)$

Mapping $\mathcal{M}$:
$$\{ (c, n, ci) \mid \mathsf{s}_1(c, n, ci, a) \} \rightsquigarrow \text{student}(c, n, ci)$$
$$\{ (c, n) \mid \mathsf{s}_2(c, n) \} \rightsquigarrow \text{university}(c, n)$$
$$\{ (s, u) \mid \mathsf{s}_3(s, u) \} \rightsquigarrow \text{enrolled}(s, u)$$

GAV with constraints – Example of retrieved global db

university

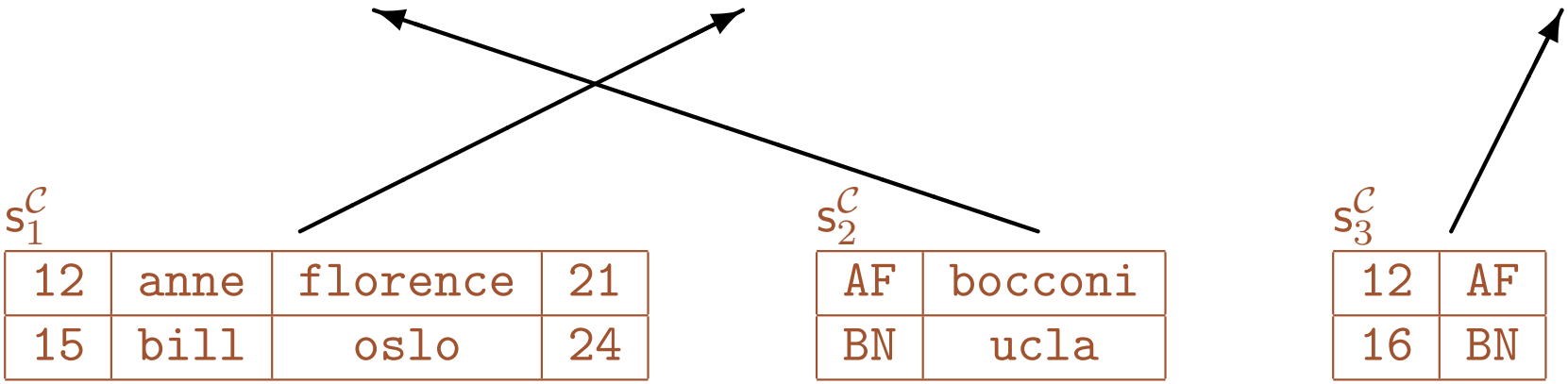
Code	*Name*
AF	bocconi
BN	ucla

student

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

enrolled

Scode	*Ucode*
12	AF
16	BN



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

GAV with constraints – Example of incompleteness

$s_3^{\mathcal{C}}$

12	AF
16	BN

enrolled$^{\mathcal{B}}$

Scode	*Ucode*
12	AF
16	BN

student$^{\mathcal{B}}$

Code	*Name*	*City*
12	anne	florence
15	bill	oslo
16	x	y

$s_3^{\mathcal{C}}(16, \mathtt{BN})$ and the mapping imply enrolled$^{\mathcal{B}}(16, \mathtt{BN})$ for all $\mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I})$

Due to the inclusion dependency enrolled$[Scode] \subseteq$ student$[Code]$ in $\mathcal{G}$, 16 is the code of some student in all $\mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I})$

Since $\mathcal{C}$ does not provide information about name and city of the student with code 16, a global database that is legal for $\mathcal{I}$ wrt $\mathcal{C}$ may contain arbitrary values for these

GAV with constraints – Example of incompleteness

$\mathsf{s}_3^{\mathcal{C}}$

12	AF
16	BN

enrolled$^{\mathcal{B}}$

Scode	*Ucode*
12	AF
16	BN

student$^{\mathcal{B}}$

Code	*Name*	*City*
12	anne	florence
15	bill	oslo
16	x	y

$\mathsf{s}_3^{\mathcal{C}}(16, \mathtt{BN})$ and the mapping imply enrolled$^{\mathcal{B}}(16, \mathtt{BN})$ for all $\mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I})$

Due to the inclusion dependency enrolled$[Scode] \subseteq$ student$[Code]$ in $\mathcal{G}$, 16 is the code of some student in all $\mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I})$

Since $\mathcal{C}$ does not provide information about name and city of the student with code 16, a global database that is legal for $\mathcal{I}$ wrt $\mathcal{C}$ may contain arbitrary values for these

GAV with constraints – Example of incompleteness

$s_3^{\mathcal{C}}$

12	AF
16	BN

enrolled$^{\mathcal{B}}$

Scode	*Ucode*
12	AF
16	BN

student$^{\mathcal{B}}$

Code	*Name*	*City*
12	anne	florence
15	bill	oslo
16	x	y

$s_3^{\mathcal{C}}(16, \mathtt{BN})$ and the mapping imply enrolled$^{\mathcal{B}}(16, \mathtt{BN})$ for all $\mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I})$

Due to the inclusion dependency enrolled$[Scode] \subseteq$ student$[Code]$ in $\mathcal{G}$, 16 is the code of some student in all $\mathcal{B} \in sem^{\mathcal{C}}(\mathcal{I})$

Since $\mathcal{C}$ does not provide information about name and city of the student with code 16, a global database that is legal for $\mathcal{I}$ wrt $\mathcal{C}$ may contain arbitrary values for these

GAV with constraints – Unfolding is not sufficient

Mapping $\mathcal{M}$:

$$\{ (c, n, ci) \mid \mathsf{s}_1(c, n, ci, a) \} \rightsquigarrow \mathsf{student}(c, n, ci)$$
$$\{ (c, n) \mid \mathsf{s}_2(c, n) \} \rightsquigarrow \mathsf{university}(c, n)$$
$$\{ (s, u) \mid \mathsf{s}_3(s, u) \} \rightsquigarrow \mathsf{enrolled}(s, u)$$

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

12	anne	florence	21
15	bill	oslo	24

$\mathsf{s}_2^{\mathcal{C}}$

AF	bocconi
BN	ucla

$\mathsf{s}_3^{\mathcal{C}}$

12	AF
16	BN

Consider the query: $q = \{ (c) \mid \mathsf{student}(c, n, ci) \}$

Unfolding of q wrt $\mathcal{M}$: $unf_{\mathcal{M}}(q) = \{ (c) \mid \mathsf{s}_1(c, n, ci, a) \}$

The query $unf_{\mathcal{M}}(q)$ retrieves from $\mathcal{C}$ only the answer $\{12, 15\}$, while the correct answer would be $\{12, 15, 16\}$

The simple unfolding strategy is not sufficient for GAV with constraints

GAV with constraints – Unfolding is not sufficient

Mapping $\mathcal{M}$:

$$\{ (c, n, ci) \mid \mathsf{s}_1(c, n, ci, a) \} \rightsquigarrow \mathsf{student}(c, n, ci)$$
$$\{ (c, n) \mid \mathsf{s}_2(c, n) \} \rightsquigarrow \mathsf{university}(c, n)$$
$$\{ (s, u) \mid \mathsf{s}_3(s, u) \} \rightsquigarrow \mathsf{enrolled}(s, u)$$

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

12	anne	florence	21
15	bill	oslo	24

$\mathsf{s}_2^{\mathcal{C}}$

AF	bocconi
BN	ucla

$\mathsf{s}_3^{\mathcal{C}}$

12	AF
16	BN

Consider the query: $q = \{ (c) \mid \mathsf{student}(c, n, ci) \}$

Unfolding of q wrt $\mathcal{M}$: $unf_{\mathcal{M}}(q) = \{ (c) \mid \mathsf{s}_1(c, n, ci, a) \}$

The query $unf_{\mathcal{M}}(q)$ retrieves from $\mathcal{C}$ only the answer $\{12, 15\}$, while the correct answer would be $\{12, 15, 16\}$

The simple unfolding strategy is not sufficient for GAV with constraints

GAV with constraints – Unfolding is not sufficient

Mapping $\mathcal{M}$:

$$\{ (c, n, ci) \mid \mathsf{s}_1(c, n, ci, a) \} \rightsquigarrow \mathsf{student}(c, n, ci)$$
$$\{ (c, n) \mid \mathsf{s}_2(c, n) \} \rightsquigarrow \mathsf{university}(c, n)$$
$$\{ (s, u) \mid \mathsf{s}_3(s, u) \} \rightsquigarrow \mathsf{enrolled}(s, u)$$

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

12	anne	florence	21
15	bill	oslo	24

$\mathsf{s}_2^{\mathcal{C}}$

AF	bocconi
BN	ucla

$\mathsf{s}_3^{\mathcal{C}}$

12	AF
16	BN

Consider the query: $q = \{ (c) \mid \mathsf{student}(c, n, ci) \}$

Unfolding of q wrt $\mathcal{M}$: $unf_{\mathcal{M}}(q) = \{ (c) \mid \mathsf{s}_1(c, n, ci, a) \}$

The query $unf_{\mathcal{M}}(q)$ retrieves from $\mathcal{C}$ only the answer $\{12, 15\}$, while the correct answer would be $\{12, 15, 16\}$

The simple unfolding strategy is not sufficient for GAV with constraints

GAV with constraints – Example of inconsistency

$s_1^{\mathcal{C}}$

12	anne	florence	21
12	bill	oslo	24

$\text{student}^{\mathcal{B}}$

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

The tuples in $s_1^{\mathcal{C}}$ and the mapping imply $\text{student}^{\mathcal{B}}(12, \texttt{anne}, \texttt{florence})$ and $\text{student}^{\mathcal{B}}(12, \texttt{bill}, \texttt{oslo})$, for all $\mathcal{B}$ that satisfy the mapping

Due to the key dependency $key(\text{student}) = \{Code\}$ in $\mathcal{G}$, there is no global database that satisfies the mapping and is legal wrt the global schema, i.e., $sem^{\mathcal{I}}(\mathcal{C}) = \emptyset$

GAV with constraints – Example of inconsistency

$s_1^{\mathcal{C}}$

12	anne	florence	21
12	bill	oslo	24

$\text{student}^{\mathcal{B}}$

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

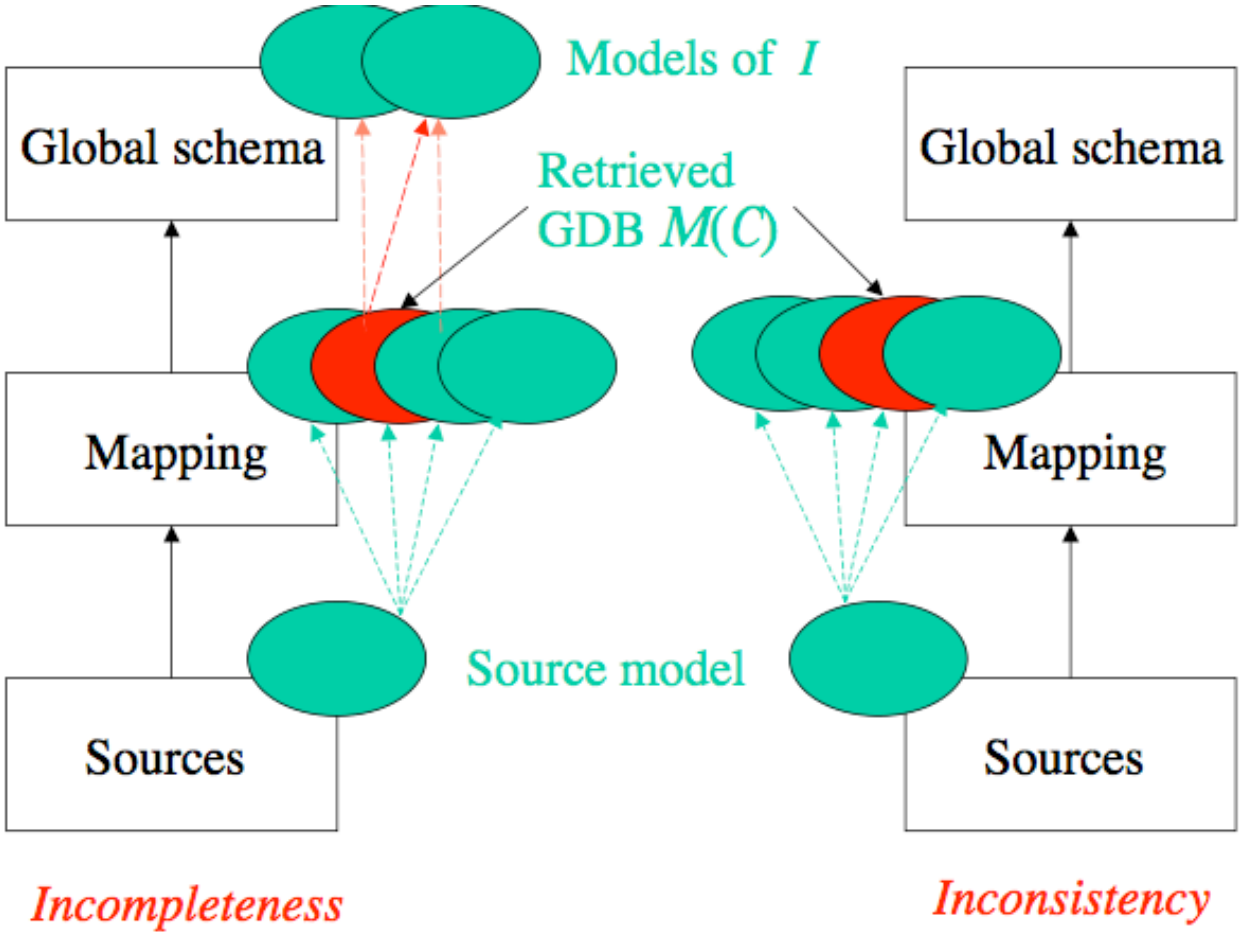
The tuples in $s_1^{\mathcal{C}}$ and the mapping imply $\text{student}^{\mathcal{B}}(12, \texttt{anne}, \texttt{florence})$ and $\text{student}^{\mathcal{B}}(12, \texttt{bill}, \texttt{oslo})$, for all $\mathcal{B}$ that satisfy the mapping

Due to the key dependency $key(\text{student}) = \{Code\}$ in $\mathcal{G}$, there is no global database that satisfies the mapping and is legal wrt the global schema, i.e., $sem^{\mathcal{I}}(\mathcal{C}) = \emptyset$

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



Outline

Inclusion dependencies – Example

Global schema $\mathcal{G}$: player$(Pname, YOB, Pteam)$
team$(Tname, Tcity, Tleader)$

Constraints: team$[Tleader, Tname] \subseteq$ player$[Pname, Pteam]$

Sources $\mathcal{S}$: s_1 and s_3 store players
s_2 stores teams

Mapping $\mathcal{M}$: $\{ (x, y, z) \mid s_1(x, y, z) \vee s_3(x, y, z) \} \rightsquigarrow$ player(x, y, z)
$\{ (x, y, z) \mid s_2(x, y, z) \} \rightsquigarrow$ team(x, y, z)

Inclusion dependencies – Example retrieved global db

Source database $\mathcal{C}$:

s_1:

Totti	1971	Roma

s_2:

Juve	Torino	Del Piero

s_3:

Buffon	1978	Juve

Retrieved global database $\mathcal{M}(\mathcal{C})$:

player:

Totti	1971	Roma
Buffon	1978	Juve

team:

Juve	Torino	Del Piero

Inclusion dependencies – Example retrieved global db

Source database $\mathcal{C}$:

s_1:

Totti	1971	Roma

s_2:

Juve	Torino	Del Piero

s_3:

Buffon	1978	Juve

Retrieved global database $\mathcal{M}(\mathcal{C})$:

player:

Totti	1971	Roma
Buffon	1978	Juve

team:

Juve	Torino	Del Piero

Inclusion dependencies – Example retrieved global db

player:

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team:

Juve	Torino	Del Piero

The ID on the global schema tells us that Del Piero is a player of Juve

All global databases satisfying $\mathcal{I}$ have at least the tuples shown above, where α is some value of the domain Δ

Warnings

1. There may be an infinite number of databases satisfying $\mathcal{I}$
2. In case of cyclic IDs, databases satisfying $\mathcal{I}$ may be of infinite size

Inclusion dependencies – Example retrieved global db

player:

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team:

Juve	Torino	Del Piero

The ID on the global schema tells us that `Del Piero` is a player of `Juve`

All global databases satisfying $\mathcal{I}$ have at least the tuples shown above, where α is some value of the domain Δ

Warnings

1. There may be an infinite number of databases satisfying $\mathcal{I}$
2. In case of cyclic IDs, databases satisfying $\mathcal{I}$ may be of infinite size

Inclusion dependencies – Example retrieved global db

player:

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team:

Juve	Torino	Del Piero

The ID on the global schema tells us that `Del Piero` is a player of `Juve`

All global databases satisfying $\mathcal{I}$ have at least the tuples shown above, where α is some value of the domain Δ

Warnings

1. There may be an infinite number of databases satisfying $\mathcal{I}$
2. In case of cyclic IDs, databases satisfying $\mathcal{I}$ may be of infinite size

Inclusion dependencies – Example retrieved global db

player:

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team:

Juve	Torino	Del Piero

The ID on the global schema tells us that Del Piero is a player of Juve

All global databases satisfying $\mathcal{I}$ have at least the tuples shown above, where α is some value of the domain Δ

Consider the query $q = \{ (x, z) \mid \text{player}(x, y, z) \}$

$cert(q, \mathcal{I}, \mathcal{C}) = \{ (\text{Totti}, \text{Roma}), (\text{Buffon}, \text{Juve}), (\text{Del Piero}, \text{Juve}) \}$

Inclusion dependencies – Example retrieved global db

player:

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team:

Juve	Torino	Del Piero

The ID on the global schema tells us that Del Piero is a player of Juve

All global databases satisfying $\mathcal{I}$ have at least the tuples shown above, where α is some value of the domain Δ

Consider the query $q = \{ (x, z) \mid \text{player}(x, y, z) \}$

$cert(q, \mathcal{I}, \mathcal{C}) = \{ (\texttt{Totti}, \texttt{Roma}), (\texttt{Buffon}, \texttt{Juve}), (\texttt{Del Piero}, \texttt{Juve}) \}$

Chasing inclusion dependencies – Infinite construction

Intuitive strategy: Add new facts until IDs are satisfied

Problem: Infinite construction in the presence of cyclic IDs

Example

Let r be binary with
$r[2] \subseteq r[1]$

Suppose $\mathcal{M}(\mathcal{C}) = \{ r(a, b) \}$

1. add $r(b, c_1)$
2. add $r(c_1, c_2)$
3. add $r(c_2, c_3)$
4. ... (ad infinitum)

Example

Let r, s be binary with
$r[1] \subseteq s[1], \quad s[2] \subseteq r[1]$

Suppose $\mathcal{M}(\mathcal{C}) = \{ r(a, b) \}$

1. add $s(a, c_1)$
2. add $r(c_1, c_2)$
3. add $s(c_1, c_3)$
4. add $r(c_3, c_4)$
5. ... (ad infinitum)

Chasing inclusion dependencies – Infinite construction

Intuitive strategy: Add new facts until IDs are satisfied

Problem: Infinite construction in the presence of cyclic IDs

Example

Let r be binary with
$r[2] \subseteq r[1]$

Suppose $\mathcal{M}(\mathcal{C}) = \{ r(a, b) \}$

1. add $r(b, c_1)$
2. add $r(c_1, c_2)$
3. add $r(c_2, c_3)$
4. ... (ad infinitum)

Example

Let r, s be binary with
$r[1] \subseteq s[1], \quad s[2] \subseteq r[1]$

Suppose $\mathcal{M}(\mathcal{C}) = \{ r(a, b) \}$

1. add $s(a, c_1)$
2. add $r(c_1, c_2)$
3. add $s(c_1, c_3)$
4. add $r(c_3, c_4)$
5. ... (ad infinitum)

Chasing inclusion dependencies – Infinite construction

Intuitive strategy: Add new facts until IDs are satisfied

Problem: Infinite construction in the presence of cyclic IDs

Example

Let r be binary with
$r[2] \subseteq r[1]$

Suppose $\mathcal{M}(\mathcal{C}) = \{ r(a, b) \}$

1. add $r(b, c_1)$
2. add $r(c_1, c_2)$
3. add $r(c_2, c_3)$
4. ... (ad infinitum)

Example

Let r, s be binary with
$r[1] \subseteq s[1], \quad s[2] \subseteq r[1]$

Suppose $\mathcal{M}(\mathcal{C}) = \{ r(a, b) \}$

1. add $s(a, c_1)$
2. add $r(c_1, c_2)$
3. add $s(c_1, c_3)$
4. add $r(c_3, c_4)$
5. ... (ad infinitum)

The chase of a database

Definition

The chase of a database is the exhaustive application of a set of rules that transform the database, in order to make it consistent with a set of integrity constraints

Typically, there will be one or more chase rules for each different type of constraint

The ID-chase rule

The chase for IDs has only one rule, the ID-chase rule

Let $\mathcal{D}$ be a database:

if the schema contains the ID $r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k]$
and there is a fact in $\mathcal{D}$ of the form $r(a_1, \ldots, a_n)$
and there are no facts in $\mathcal{D}$ of the form $s(b_1, \ldots, b_m)$
such that $a_{i_\ell} = b_{j_\ell}$ for each $\ell \in \{1, \ldots, k\}$,
then add to $\mathcal{D}$ the fact $s(c_1, \ldots, c_m)$,
where for each $h \in \{1, \ldots, m\}$,
if $h = j_\ell$ for some ℓ then $c_h = a_{i_\ell}$
otherwise c_h is a new constant symbol (not in $\mathcal{D}$ yet)

Notice: New existential symbols are introduced (skolem terms)

Properties of the chase

- Bad news: the chase is in general infinite
- Good news: the chase identifies a canonical model
  A canonical model is a database that "represents" all the models of the system
- We can use the chase to prove soundness and completeness of a query processing method
- . . . but only for positive queries!

Limiting the chase

Why don't we use a finite number of existential constants in the chase?

Example

Consider $r[1] \subseteq s[1]$, and $s[2] \subseteq r[1]$ and suppose $\mathcal{M}(\mathcal{C}) = \{ r(a, b) \}$

Compute chase$(\mathcal{M}(\mathcal{C}))$ with only one new constant c_1:
0) $r(a, b)$; 1) add $s(a, c_1)$ 2) add $r(c_1, c_1)$ 3) add $s(c_1, c_1)$

This database is not a canonical model for $\mathcal{I}$ wrt $\mathcal{C}$
E.g., for query $q = \{ (x) \mid r(x, y), s(y, y) \}$, we have $a \in q^{\text{chase}(\mathcal{M}(\mathcal{C}))}$
while $a \notin cert(q, \mathcal{I}, \mathcal{C})$

Arbitrarily limiting the chase is unsound, for any finite number of new constants

Outline

Chasing the query

When chasing the data the termination condition would need to take into account the query

We consider an alternative approach, based on the idea of a query chase

- Instead of chasing the data, we chase the query
- Is the dual notion of the database chase
- IDs are applied from right to left to the query atoms
- Advantage: much easier termination conditions, which imply:
  - decidability properties
  - efficiency

This technique provides an algorithm for rewriting UCQs under IDs

Chasing the query

When chasing the data the termination condition would need to take into account the query

We consider an alternative approach, based on the idea of a query chase

- Instead of chasing the data, we chase the query
- Is the dual notion of the database chase
- IDs are applied from right to left to the query atoms
- Advantage: much easier termination conditions, which imply:
  - decidability properties
  - efficiency

This technique provides an algorithm for rewriting UCQs under IDs

Query rewriting under inclusion dependencies

- Given a query q over the global schema $\mathcal{G}$, we look for a rewriting rew of q expressed over $\mathcal{S}$
- A rewriting rew is perfect if $rew^{\mathcal{C}} = cert(q, \mathcal{I}, \mathcal{C})$, for every source database $\mathcal{C}$
- With a perfect rewriting, we can do query answering by rewriting
  $\rightsquigarrow$ We avoid actually constructing the retrieved global database $\mathcal{M}(\mathcal{C})$

Rewriting rule for inclusion dependencies

Intuition: Use the IDs as basic rewriting rules

Example

Consider a query $q = \{ (x, z) \mid \mathsf{player}(x, y, z) \}$

and the constraint $\mathsf{team}[Tleader, Tname] \subseteq \mathsf{player}[Pname, Pteam]$

as a logic rule: $\mathsf{player}(w_3, w_4, w_1) \leftarrow \mathsf{team}(w_1, w_2, w_3)$

We add to the rewriting the query $q' = \{ (x, z) \mid \mathsf{team}(x, y, z) \}$

Definition

Basic rewriting step:

when an atom unifies with the head of the rule

substitute the atom with the body of the rule

Rewriting rule for inclusion dependencies

Intuition: Use the IDs as basic rewriting rules

Example

Consider a query $q = \{ (x, z) \mid \mathsf{player}(x, y, z) \}$

and the constraint $\mathsf{team}[Tleader, Tname] \subseteq \mathsf{player}[Pname, Pteam]$

as a logic rule: $\mathsf{player}(w_3, w_4, w_1) \leftarrow \mathsf{team}(w_1, w_2, w_3)$

We add to the rewriting the query $q' = \{ (x, z) \mid \mathsf{team}(x, y, z) \}$

Definition

Basic rewriting step:

when an atom unifies with the head of the rule

substitute the atom with the body of the rule

Rewriting rule for inclusion dependencies

Intuition: Use the IDs as basic rewriting rules

Example

Consider a query $q = \{ (x, z) \mid \mathsf{player}(x, y, z) \}$

and the constraint $\mathsf{team}[Tleader, Tname] \subseteq \mathsf{player}[Pname, Pteam]$

as a logic rule: $\mathsf{player}(w_3, w_4, w_1) \leftarrow \mathsf{team}(w_1, w_2, w_3)$

We add to the rewriting the query $q' = \{ (x, z) \mid \mathsf{team}(x, y, z) \}$

Definition

Basic rewriting step:

when an atom unifies with the head of the rule

substitute the atom with the body of the rule

Query Rewriting for IDs – Algorithm *ID-rewrite*

Iterative execution of:

1. Reduction:
   - Atoms that unify with other atoms are eliminated and the unification is applied
   - Variables that appear only once are marked
2. Basic rewriting step
   - A rewriting step is applicable to an atom if it does not eliminate variables that appear somewhere else
   - May introduce fresh variables

Note: The algorithm works directly for unions of conjunctive queries (UCQs), and produces an UCQ as result

Query Rewriting for IDs – Algorithm *ID-rewrite*

Iterative execution of:

1. Reduction:
   - Atoms that unify with other atoms are eliminated and the unification is applied
   - Variables that appear only once are marked
2. Basic rewriting step
   - A rewriting step is applicable to an atom if it does not eliminate variables that appear somewhere else
   - May introduce fresh variables

Note: The algorithm works directly for unions of conjunctive queries (UCQs), and produces an UCQ as result

The algorithm *ID-rewrite*

Input: relational schema $\mathcal{G}$, set Ψ_{ID} of IDs, UCQ Q
Output: perfect rewriting of Q
$Q' := Q$;
repeat
  $Q_{aux} := Q'$;
  for each $q \in Q_{aux}$ **do**
  (a) **for each** $g_1, g_2 \in \mathit{body}(q)$ **do**
    if g_1 and g_2 unify **then** $Q' := Q' \cup \{\tau(\mathit{reduce}(q, g_1, g_2))\}$;
  (b) **for each** $g \in \mathit{body}(q)$ **do**
    for each $ID \in \Psi_{ID}$ **do**
      if ID is applicable to g
        then $Q' := Q' \cup \{\, q[g/\mathit{rewrite}(g, ID)] \,\}$
until $Q_{aux} = Q'$;
return Q'

Query answering in GAV under IDs

Properties of *ID-rewrite*

- *ID-rewrite* terminates
- *ID-rewrite* produces a perfect rewriting of the input query

More precisely, let $unf_{\mathcal{M}}(q)$ be the unfolding of the query q wrt the GAV mapping $\mathcal{M}$

Theorem

$unf_{\mathcal{M}}(\text{ID-rewrite}(q))$ is a perfect rewriting of the query q

Theorem

Query answering in GAV systems under IDs is in PTIME *in data complexity (actually in* LOGSPACE*)*

Query answering in GAV under IDs

Properties of *ID-rewrite*

- *ID-rewrite* terminates
- *ID-rewrite* produces a perfect rewriting of the input query

More precisely, let $unf_{\mathcal{M}}(q)$ be the unfolding of the query q wrt the GAV mapping $\mathcal{M}$

Theorem

$unf_{\mathcal{M}}$(ID-rewrite(q)) is a perfect rewriting of the query q

Theorem

Query answering in GAV systems under IDs is in PTIME *in data complexity (actually in* LOGSPACE*)*

Outline

Query answering under IDs and KDs

We have already seen that in GAV systems under sound mappings

- Key dependencies may give rise to inconsistencies
- When $\mathcal{M}(\mathcal{C})$ violates the KDs, no legal database exists and query answering becomes trivial

How do KDs interact with IDs?

Theorem

Query answering under IDs and KDs is undecidable

Proof: By reduction from implication of IDs and FDs

We need to look for syntactic restrictions on the form of the dependencies that ensures decidability

Query answering under IDs and KDs

We have already seen that in GAV systems under sound mappings

- Key dependencies may give rise to inconsistencies
- When $\mathcal{M}(\mathcal{C})$ violates the KDs, no legal database exists and query answering becomes trivial

How do KDs interact with IDs?

Theorem

Query answering under IDs and KDs is undecidable

Proof: By reduction from implication of IDs and FDs

We need to look for syntactic restrictions on the form of the dependencies that ensures decidability

Query answering under IDs and KDs

We have already seen that in GAV systems under sound mappings

- Key dependencies may give rise to inconsistencies
- When $\mathcal{M}(\mathcal{C})$ violates the KDs, no legal database exists and query answering becomes trivial

How do KDs interact with IDs?

Theorem

Query answering under IDs and KDs is undecidable

Proof: By reduction from implication of IDs and FDs

We need to look for syntactic restrictions on the form of the dependencies that ensures decidability

Query answering under IDs and KDs

We have already seen that in GAV systems under sound mappings

- Key dependencies may give rise to inconsistencies
- When $\mathcal{M}(\mathcal{C})$ violates the KDs, no legal database exists and query answering becomes trivial

How do KDs interact with IDs?

Theorem

Query answering under IDs and KDs is undecidable

Proof: By reduction from implication of IDs and FDs

We need to look for syntactic restrictions on the form of the dependencies that ensures decidability

Non-key-conflicting IDs

Definition

Non-key-conflicting IDs (NKCIDs) are of the form $r_1[\vec{x}_1] \subseteq r_2[\vec{x}_2]$ where $\vec{x}_2$ is not a strict superset of $key(r_2)$

Example

Let r be of arity 3 and s of arity 4 with $key(s) = \{1, 2\}$

- The following are NKCIDs
  - $r[2] \subseteq s[2]$, since $\{2\}$ is a strict subset of $key(s)$
  - $r[2, 3] \subseteq s[1, 2]$, since $\{1, 2\}$ coincides with $key(s)$
  - $r[1, 2] \subseteq s[2, 3]$, since $1 \in key(s)$ but $1 \notin \{2, 3\}$
- The following is not a NKCID: $r[1, 2, 3] \subseteq s[1, 2, 4]$

Note: Foreign keys (FKs) are a special case of NKCIDs

Non-key-conflicting IDs

Definition

Non-key-conflicting IDs (NKCIDs) are of the form $r_1[\vec{x}_1] \subseteq r_2[\vec{x}_2]$ where $\vec{x}_2$ is not a strict superset of $key(r_2)$

Example

Let r be of arity 3 and s of arity 4 with $key(s) = \{1, 2\}$

- The following are NKCIDs
  - $r[2] \subseteq s[2]$, since $\{2\}$ is a strict subset of $key(s)$
  - $r[2, 3] \subseteq s[1, 2]$, since $\{1, 2\}$ coincides with $key(s)$
  - $r[1, 2] \subseteq s[2, 3]$, since $1 \in key(s)$ but $1 \notin \{2, 3\}$
- The following is not a NKCID: $r[1, 2, 3] \subseteq s[1, 2, 4]$

Note: Foreign keys (FKs) are a special case of NKCIDs

Non-key-conflicting IDs

Definition

Non-key-conflicting IDs (NKCIDs) are of the form $r_1[\vec{x}_1] \subseteq r_2[\vec{x}_2]$ where $\vec{x}_2$ is not a strict superset of $key(r_2)$

Example

Let r be of arity 3 and s of arity 4 with $key(s) = \{1, 2\}$

- The following are NKCIDs
  - $r[2] \subseteq s[2]$, since $\{2\}$ is a strict subset of $key(s)$
  - $r[2, 3] \subseteq s[1, 2]$, since $\{1, 2\}$ coincides with $key(s)$
  - $r[1, 2] \subseteq s[2, 3]$, since $1 \in key(s)$ but $1 \notin \{2, 3\}$
- The following is not a NKCID: $r[1, 2, 3] \subseteq s[1, 2, 4]$

Note: Foreign keys (FKs) are a special case of NKCIDs

Separation for IDs and KDs

Theorem (IDs-KDs separation)

Under KDs and NKCIDs, if $\mathcal{M}(\mathcal{C})$ satisfies the KDs, then the KDs can be ignored wrt certain answers of a user query q

Intuition: For NKCIDs, when applying the ID-chase rule to a tuple $\vec{t}_1 \in r_1^{\mathcal{B}}$, we can choose the tuple $\vec{t}_2$ to introduce in $r_2^{\mathcal{B}}$ so that it does not violate $key(r_2)$:

- When $key(r_2) \not\subseteq \vec{x}_2$, fresh constants in $\vec{t}_2$ are chosen for key attributes, and so there is no other tuple in $r_2^{\mathcal{B}}$ coinciding with $\vec{t}_2$ on all key attributes
- When $key(r_2) = \vec{x}_2$, if there is already a tuple $\vec{t}$ in $r_2^{\mathcal{B}}$ such that $\vec{t}_1[\vec{x}_1] = \vec{t}[\vec{x}_2]$, we choose $\vec{t}$ for $\vec{t}_2$

Query answering becomes undecidable as soon as we extend the language of the IDs

Separation for IDs and KDs

Theorem (IDs-KDs separation)

Under KDs and NKCIDs, if $\mathcal{M}(\mathcal{C})$ satisfies the KDs, then the KDs can be ignored wrt certain answers of a user query q

Intuition: For NKCIDs, when applying the ID-chase rule to a tuple $\vec{t}_1 \in r_1^{\mathcal{B}}$, we can choose the tuple $\vec{t}_2$ to introduce in $r_2^{\mathcal{B}}$ so that it does not violate $key(r_2)$:

- When $key(r_2) \not\subseteq \vec{x}_2$, fresh constants in $\vec{t}_2$ are chosen for key attributes, and so there is no other tuple in $r_2^{\mathcal{B}}$ coinciding with $\vec{t}_2$ on all key attributes
- When $key(r_2) = \vec{x}_2$, if there is already a tuple $\vec{t}$ in $r_2^{\mathcal{B}}$ such that $\vec{t}_1[\vec{x}_1] = \vec{t}[\vec{x}_2]$, we choose $\vec{t}$ for $\vec{t}_2$

Query answering becomes undecidable as soon as we extend the language of the IDs

Separation for IDs and KDs

Theorem (IDs-KDs separation)

Under KDs and NKCIDs, if $\mathcal{M}(\mathcal{C})$ satisfies the KDs, then the KDs can be ignored wrt certain answers of a user query q

Intuition: For NKCIDs, when applying the ID-chase rule to a tuple $\vec{t}_1 \in r_1^{\mathcal{B}}$, we can choose the tuple $\vec{t}_2$ to introduce in $r_2^{\mathcal{B}}$ so that it does not violate $key(r_2)$:

- When $key(r_2) \not\subseteq \vec{x}_2$, fresh constants in $\vec{t}_2$ are chosen for key attributes, and so there is no other tuple in $r_2^{\mathcal{B}}$ coinciding with $\vec{t}_2$ on all key attributes
- When $key(r_2) = \vec{x}_2$, if there is already a tuple $\vec{t}$ in $r_2^{\mathcal{B}}$ such that $\vec{t}_1[\vec{x}_1] = \vec{t}[\vec{x}_2]$, we choose $\vec{t}$ for $\vec{t}_2$

Query answering becomes undecidable as soon as we extend the language of the IDs

Query processing under separable KDs and IDs

Global algorithm:

1. Verify consistency of $\mathcal{M}(\mathcal{C})$ with respect to KDs
2. Compute *ID-rewrite* of the input query
3. Unfold wrt $\mathcal{M}$ the query computed at previous step
4. Evaluate the unfolded query over the sources

Note:

- The KD consistency check can be done by suitable CQs with inequality
- The computation of $\mathcal{M}(\mathcal{C})$ can be avoided (by unfolding the queries for the KD consistency check)

Query processing under separable KDs and IDs

Global algorithm:

1. Verify consistency of $\mathcal{M}(\mathcal{C})$ with respect to KDs
2. Compute *ID-rewrite* of the input query
3. Unfold wrt $\mathcal{M}$ the query computed at previous step
4. Evaluate the unfolded query over the sources

Note:

- The KD consistency check can be done by suitable CQs with inequality
- The computation of $\mathcal{M}(\mathcal{C})$ can be avoided (by unfolding the queries for the KD consistency check)

Checking KD consistency – Example

Relation: $\mathsf{player}[Pname, Pteam]$
Key dependency: $\mathit{key}(\mathsf{player}) = \{Pname\}$

Query to check (in)consistency of the KD:

$$q = \{ () \mid \mathsf{player}(x, y), \mathsf{player}(x, z), y \neq z \}$$

is $true$ iff the instance of player violates the KD

Mapping $\mathcal{M}$: $\{ (x, y) \mid \mathsf{s}_1(x, y) \vee \mathsf{s}_2(x, y) \} \rightsquigarrow \mathsf{player}(x, y)$

Unfolding of q wrt $\mathcal{M}$:

$$\{ () \mid \mathsf{s}_1(x, y), \mathsf{s}_1(x, z), y \neq z \vee \mathsf{s}_1(x, y), \mathsf{s}_2(x, z), y \neq z \vee \mathsf{s}_2(x, y), \mathsf{s}_1(x, z), y \neq z \vee \mathsf{s}_2(x, y), \mathsf{s}_2(x, z), y \neq z \}$$

Checking KD consistency – Example

Relation: $\mathsf{player}[Pname, Pteam]$
Key dependency: $\mathit{key}(\mathsf{player}) = \{Pname\}$

Query to check (in)consistency of the KD:

$$q \;=\; \{\; () \mid \mathsf{player}(x, y), \mathsf{player}(x, z), y \neq z \;\}$$

is true iff the instance of player violates the KD

Mapping $\mathcal{M}$: $\{\; (x, y) \mid \mathsf{s}_1(x, y) \vee \mathsf{s}_2(x, y) \;\} \rightsquigarrow \mathsf{player}(x, y)$

Unfolding of q wrt $\mathcal{M}$:

$$\begin{aligned}\{\; () \mid\; & \mathsf{s}_1(x, y), \mathsf{s}_1(x, z), y \neq z \vee \\ & \mathsf{s}_1(x, y), \mathsf{s}_2(x, z), y \neq z \vee \\ & \mathsf{s}_2(x, y), \mathsf{s}_1(x, z), y \neq z \vee \\ & \mathsf{s}_2(x, y), \mathsf{s}_2(x, z), y \neq z \;\}\end{aligned}$$

Query answering in GAV under separable IDs+KDs

Theorem (Calì, Lembo & Rosati, PODS'03)

Answering conjunctive queries in GAV systems under KDs and NKCIDs is in PTIME *in data complexity (actually in* LOGSPACE *)*

Can we extend these results to more expressive user queries?

- The rewriting technique extends immediately to unions of CQs
  $$\textit{ID-rewrite}(q_1 \vee \cdots \vee q_n) = \textit{ID-rewrite}(q_1) \vee \cdots \vee \textit{ID-rewrite}(q_n)$$
- This is not the case for recursive queries

Theorem (Calvanese & Rosati KRDB'03)

Answering recursive queries under KDs and FKs is undecidable
Answering recursive queries under IDs is undecidable

Query answering in GAV under separable IDs+KDs

Theorem (Calì, Lembo & Rosati, PODS'03)

Answering conjunctive queries in GAV systems under KDs and NKCIDs is in PTIME *in data complexity (actually in* LOGSPACE *)*

Can we extend these results to more expressive user queries?

- The rewriting technique extends immediately to unions of CQs
  $$\textit{ID-rewrite}(q_1 \vee \cdots \vee q_n) = \textit{ID-rewrite}(q_1) \vee \cdots \vee \textit{ID-rewrite}(q_n)$$
- This is not the case for recursive queries

Theorem (Calvanese & Rosati KRDB'03)

Answering recursive queries under KDs and FKs is undecidable
Answering recursive queries under IDs is undecidable

Query answering in GAV under separable IDs+KDs

Theorem (Calì, Lembo & Rosati, PODS'03)

Answering conjunctive queries in GAV systems under KDs and NKCIDs is in PTIME *in data complexity (actually in* LOGSPACE *)*

Can we extend these results to more expressive user queries?

- The rewriting technique extends immediately to unions of CQs
  $$\textit{ID-rewrite}(q_1 \vee \cdots \vee q_n) = \textit{ID-rewrite}(q_1) \vee \cdots \vee \textit{ID-rewrite}(q_n)$$
- This is not the case for recursive queries

Theorem (Calvanese & Rosati KRDB'03)

Answering recursive queries under KDs and FKs is undecidable
Answering recursive queries under IDs is undecidable

Outline

Query answering under IDs and EDs

Under EDs:

- Possibility of inconsistencies
- When $\mathcal{M}(\mathcal{C})$ violates the EDs, no legal database exists and query answering becomes trivial

Under IDs and EDs:

- How do EDs and IDs interact?
- Is query answering separable?
- Is query answering decidable?

Query answering under IDs and EDs

Under EDs:

- Possibility of inconsistencies
- When $\mathcal{M}(\mathcal{C})$ violates the EDs, no legal database exists and query answering becomes trivial

Under IDs and EDs:

- How do EDs and IDs interact?
- Is query answering separable?
- Is query answering decidable?

Exclusion dependencies – Example

Global schema $\mathcal{G}$: player$(Pname, YOB, Pteam)$
team$(Tname, Tcity, Tleader)$
coach$(Cname, Cteam)$

Constraints: team$[Tleader, Tname] \subseteq$ player$[Pname, Pteam]$
coach$[Cname] \cap$ player$[Pname] = \emptyset$

Sources $\mathcal{S}$: s_1 and s_3 store players
s_2 stores teams
s_4 stores coaches

Mapping $\mathcal{M}$: $\{ (x, y, z) \mid s_1(x, y, z) \vee s_3(x, y, z) \} \rightsquigarrow$ player(x, y, z)
$\{ (x, y, z) \mid s_2(x, y, z) \} \rightsquigarrow$ team(x, y, z)
$\{ (x, y) \mid s_4(x, y, z) \} \rightsquigarrow$ coach(x, y)

Retrieved global db under EDs – Example

Source database $\mathcal{C}$:

s_1:

Totti	1971	Roma

s_2:

Juve	Torino	Del Piero

s_3:

Buffon	1978	Juve

s_4:

Del Piero	Viterbese

Retrieved global database $\mathcal{M}(\mathcal{C})$:

player :

Totti	1971	Roma
Buffon	1978	Juve

team :

Juve	Torino	Del Piero

coach :

Del Piero	Viterbese

Retrieved global db under EDs – Example

Source database $\mathcal{C}$:

s_1:

Totti	1971	Roma

s_2:

Juve	Torino	Del Piero

s_3:

Buffon	1978	Juve

s_4:

Del Piero	Viterbese

Retrieved global database $\mathcal{M}(\mathcal{C})$:

player :

Totti	1971	Roma
Buffon	1978	Juve

team :

Juve	Torino	Del Piero

coach :

Del Piero	Viterbese

"Repair" of retrieved global db under EDs – Example

Retrieved global database $\mathcal{M}(\mathcal{C})$:

player :

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

coach :

Del Piero	Viterbese

$\text{team}[Tleader, Tname] \subseteq \text{player}[Pname, Pteam]$

Violation of $\text{coach}[Cname] \cap \text{player}[Pname] = \emptyset$

Can we detect such situations without actually constructing $\mathcal{M}(\mathcal{C})$?

"Repair" of retrieved global db under EDs – Example

Retrieved global database $\mathcal{M}(\mathcal{C})$:

player :

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

coach :

Del Piero	Viterbese

Violation of team$[Tleader, Tname] \subseteq$ player$[Pname, Pteam]$

Violation of coach$[Cname] \cap$ player$[Pname] = \emptyset$

Can we detect such situations without actually constructing $\mathcal{M}(\mathcal{C})$?

"Repair" of retrieved global db under EDs – Example

Retrieved global database $\mathcal{M}(\mathcal{C})$:

player :

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

coach :

Del Piero	Viterbese

"Repair" of team$[Tleader, Tname] \subseteq$ player$[Pname, Pteam]$

Violation of coach$[Cname] \cap$ player$[Pname] = \emptyset$

Can we detect such situations without actually constructing $\mathcal{M}(\mathcal{C})$?

"Repair" of retrieved global db under EDs – Example

Retrieved global database $\mathcal{M}(\mathcal{C})$:

player :

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

coach :

Del Piero	Viterbese

"Repair" of team$[Tleader, Tname] \subseteq$ player$[Pname, Pteam]$

Violation of coach$[Cname] \cap$ player$[Pname] = \emptyset$

Can we detect such situations without actually constructing $\mathcal{M}(\mathcal{C})$?

"Repair" of retrieved global db under EDs – Example

Retrieved global database $\mathcal{M}(\mathcal{C})$:

player :

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

coach :

Del Piero	Viterbese

"Repair" of team$[Tleader, Tname] \subseteq$ player$[Pname, Pteam]$

Violation of coach$[Cname] \cap$ player$[Pname] = \emptyset$

Can we detect such situations without actually constructing $\mathcal{M}(\mathcal{C})$?

Deductive closure of EDs under IDs – Example

Can we saturate (close) the EDs by adding all the EDs that are logical consequences of the EDs and IDs?

Example

From

$$\text{team}[Tleader, Tname] \subseteq \text{player}[Pname, Pteam]$$
$$\text{coach}[Cname] \cap \text{player}[Pname] = \emptyset$$

it follows that

$$\text{coach}[Cname] \cap \text{team}[Tleader] = \emptyset$$

This constraint is violated by the retrieved global database $\mathcal{M}(\mathcal{C})$

Deductive closure of EDs under IDs – Example

Can we saturate (close) the EDs by adding all the EDs that are logical consequences of the EDs and IDs?

Example

From

$$\mathsf{team}[Tleader, Tname] \subseteq \mathsf{player}[Pname, Pteam]$$
$$\mathsf{coach}[Cname] \cap \mathsf{player}[Pname] = \emptyset$$

it follows that

$$\mathsf{coach}[Cname] \cap \mathsf{team}[Tleader] = \emptyset$$

This constraint is violated by the retrieved global database $\mathcal{M}(\mathcal{C})$

Deductive closure of EDs under IDs

Definition

Derivation rule of EDs under EDs and IDs:

From the ED $r[i_1, \ldots, i_k] \cap s[j_1, \ldots, j_k] = \emptyset$
and the ID $t[\ell_1, \ldots, \ell_k] \subseteq s[j_1, \ldots, j_k]$
derive the ED $r[i_1, \ldots, i_k] \cap t[\ell_1, \ldots, \ell_k] = \emptyset$

Corresponds to a simple application of resolution on the FOL sentences corresponding to EDs and IDs

Theorem

If the set of EDs is closed with respect to the above rule, it contains all EDs that are logical consequences of the initial EDs and IDs

Deductive closure of EDs under IDs

Definition

Derivation rule of EDs under EDs and IDs:

From the ED $r[i_1, \ldots, i_k] \cap s[j_1, \ldots, j_k] = \emptyset$
and the ID $t[\ell_1, \ldots, \ell_k] \subseteq s[j_1, \ldots, j_k]$
derive the ED $r[i_1, \ldots, i_k] \cap t[\ell_1, \ldots, \ell_k] = \emptyset$

Corresponds to a simple application of resolution on the FOL sentences corresponding to EDs and IDs

Theorem

If the set of EDs is closed with respect to the above rule, it contains all EDs that are logical consequences of the initial EDs and IDs

Query answering in GAV under IDs and EDs

Theorem (ID-ED Separation)

Under IDs and EDs,
if $\mathcal{M}(\mathcal{C})$ satisfies all EDs derived from the IDs and the original EDs then the EDs can be ignored wrt certain answers of a query

We obtain a method for query answering in GAV under EDs and IDs:

1. Close the set of EDs with respect to the IDs
2. Verify consistency of $\mathcal{M}(\mathcal{C})$ with respect to EDs
3. Compute ID-rewrite of the input query
4. Unfold the query computed at the previous step
5. Evaluate the query over the sources

The ED consistency check can be done by suitable CQs

Query answering in GAV under IDs and EDs

Theorem (ID-ED Separation)

Under IDs and EDs,
if $\mathcal{M}(\mathcal{C})$ satisfies all EDs derived from the IDs and the original EDs then the EDs can be ignored wrt certain answers of a query

We obtain a method for query answering in GAV under EDs and IDs:

1. Close the set of EDs with respect to the IDs
2. Verify consistency of $\mathcal{M}(\mathcal{C})$ with respect to EDs
3. Compute ID-rewrite of the input query
4. Unfold the query computed at the previous step
5. Evaluate the query over the sources

The ED consistency check can be done by suitable CQs

Query answering in GAV under IDs, KDs and EDs

Theorem (ID-KD-ED Separation)

Under KDs, NKCIDs, and EDs,
if $\mathcal{M}(\mathcal{C})$ satisfies all the KDs
and satisfies all EDs derived from the IDs and the original EDs
then the KDs and EDs can be ignored wrt certain answers of a query

We obtain a method for query answering in GAV under KDs, NKCIDs, and EDs:

1. Close the set of EDs with respect to the IDs
2. Verify consistency of $\mathcal{M}(\mathcal{C})$ with respect to KDs and EDs
3. Compute ID-rewrite of the input query
4. Unfold the query computed at the previous step
5. Evaluate the query over the sources

Query answering in GAV under IDs, KDs and EDs

Theorem (ID-KD-ED Separation)

Under KDs, NKCIDs, and EDs,
if $\mathcal{M}(\mathcal{C})$ satisfies all the KDs
and satisfies all EDs derived from the IDs and the original EDs
then the KDs and EDs can be ignored wrt certain answers of a query

We obtain a method for query answering in GAV under KDs, NKCIDs, and EDs:

1. Close the set of EDs with respect to the IDs
2. Verify consistency of $\mathcal{M}(\mathcal{C})$ with respect to KDs and EDs
3. Compute ID-rewrite of the input query
4. Unfold the query computed at the previous step
5. Evaluate the query over the sources

Query answ. in GAV under IDs, KDs and EDs – Complexity

Note:

1. Closing the set of EDs wrt the IDs is independent of the data
2. Consistency of $\mathcal{M}(\mathcal{C})$ wrt KDs and EDs can be verified through suitable queries over the source database $\mathcal{C}$

Theorem (Lembo & Rosati, 2004)

Answering conjunctive queries in GAV systems under KDs, NKCIDs and EDs is in PTIME *in data complexity (actually in* LOGSPACE *)*

Query answ. in GAV under IDs, KDs and EDs – Complexity

Note:

1. Closing the set of EDs wrt the IDs is independent of the data
2. Consistency of $\mathcal{M}(\mathcal{C})$ wrt KDs and EDs can be verified through suitable queries over the source database $\mathcal{C}$

Theorem (Lembo & Rosati, 2004)

Answering conjunctive queries in GAV systems under KDs, NKCIDs and EDs is in PTIME *in data complexity (actually in* LOGSPACE *)*

Outline

Outline

(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 $\phi_{\mathcal{S}} \rightsquigarrow \phi_{\mathcal{G}}$ 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 $\mathcal{M}(\mathcal{C})\downarrow$ may not be legal for $\mathcal{G}$

(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 $\phi_{\mathcal{S}} \rightsquigarrow \phi_{\mathcal{G}}$ 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 $\mathcal{M}(\mathcal{C})\downarrow$ may not be legal for $\mathcal{G}$

Semantics of (G)LAV systems with integrity constraints

Given a source db $\mathcal{C}$, a global db $\mathcal{B}$ (over Δ) satisfies $\mathcal{I}$ relative to $\mathcal{C}$ 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 $\mathcal{M}(\mathcal{C})\downarrow$ (sound mappings)

Recall:

- $\mathcal{M}(\mathcal{C})$ is obtained by evaluating, for each mapping assertion $\phi_S \rightsquigarrow \phi_G$, the query ϕ_S over $\mathcal{C}$, and using the obtained tuples to populate the global relations according to ϕ_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 $\mathcal{C}$

Semantics of (G)LAV systems with integrity constraints

Given a source db $\mathcal{C}$, a global db $\mathcal{B}$ (over Δ) satisfies $\mathcal{I}$ relative to $\mathcal{C}$ 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 $\mathcal{M}(\mathcal{C})\downarrow$ (sound mappings)

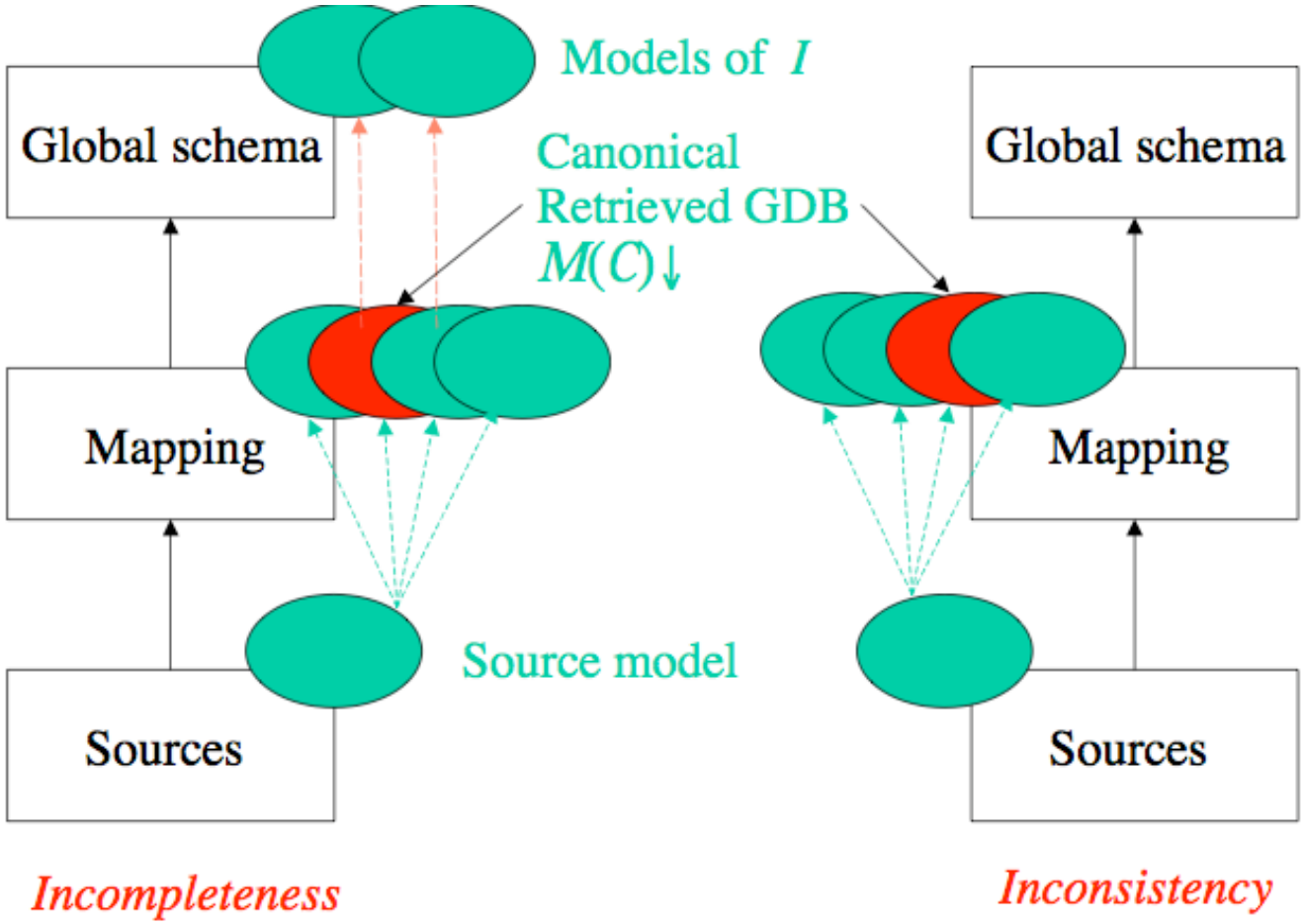
Recall:

- $\mathcal{M}(\mathcal{C})$ is obtained by evaluating, for each mapping assertion $\phi_S \rightsquigarrow \phi_G$, the query ϕ_S over $\mathcal{C}$, and using the obtained tuples to populate the global relations according to ϕ_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 $\mathcal{C}$

(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



Outline

(G)LAV systems under IDs

Under IDs only, we can exploit the previous results for GAV also for (G)LAV, by turning the (G)LAV mappings into GAV mappings:

- We transform a (G)LAV integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with IDs only into a GAV system $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$
- With respect to $\mathcal{I}$, the transformed system $\mathcal{I}'$ contains auxiliary IDs and auxiliary global relation symbols
- The transformation is query-preserving:

  For every conjunctive query q and for every source database $\mathcal{C}$, the certain answers to q wrt $\mathcal{I}$ and $\mathcal{C}$ are equal to the certain answers to q wrt $\mathcal{I}'$ and $\mathcal{C}$

Transforming LAV into GAV – Example

Initial LAV mappings:

$$s(x,y) \rightsquigarrow \{ (x,y) \mid r_1(x,z), r_2(y,w) \}$$
$$t(x,y) \rightsquigarrow \{ (x,y) \mid r_1(x,z), r_3(y,x) \}$$

We introduce two new global relations for each mapping assertion:
$s_i/2$, $s_e/4$, and $t_i/2$, $t_e/3$

Transformed GAV mappings:

$$\{ (x,y) \mid s(x,y) \} \rightsquigarrow s_i(x,y)$$
$$\{ (x,y) \mid t(x,y) \} \rightsquigarrow t_i(x,y)$$

Additional IDs generated by the transformation:

$$s_i[1,2] \subseteq s_e[1,2] \quad s_e[1,3] \subseteq r_1[1,2] \quad s_e[2,4] \subseteq r_2[1,2]$$
$$t_i[1,2] \subseteq t_e[1,2] \quad t_e[1,3] \subseteq r_1[1,2] \quad t_e[2,1] \subseteq r_3[1,2]$$

Transforming LAV into GAV – Example

Initial LAV mappings:

$$s(x,y) \rightsquigarrow \{ (x,y) \mid r_1(x,z), r_2(y,w) \}$$
$$t(x,y) \rightsquigarrow \{ (x,y) \mid r_1(x,z), r_3(y,x) \}$$

We introduce two new global relations for each mapping assertion:
$s_i/2$, $s_e/4$, and $t_i/2$, $t_e/3$

Transformed GAV mappings:

$$\{ (x,y) \mid s(x,y) \} \rightsquigarrow s_i(x,y)$$
$$\{ (x,y) \mid t(x,y) \} \rightsquigarrow t_i(x,y)$$

Additional IDs generated by the transformation:

$$s_i[1,2] \subseteq s_e[1,2] \qquad s_e[1,3] \subseteq r_1[1,2] \qquad s_e[2,4] \subseteq r_2[1,2]$$
$$t_i[1,2] \subseteq t_e[1,2] \qquad t_e[1,3] \subseteq r_1[1,2] \qquad t_e[2,1] \subseteq r_3[1,2]$$

Query answering in (G)LAV systems under IDs

Method for query answering in a (G)LAV system $\mathcal{I}$ with IDs:

1. Transform $\mathcal{I}$ into a GAV system $\mathcal{I}'$
2. Apply the query answering method for GAV systems under IDs (The unfolding step must take into account the presence of auxiliary global symbols)

Theorem

Answering conjunctive queries in (G)LAV systems under IDs is in PTIME *in data complexity (actually in* LOGSPACE *)*

Query answering in (G)LAV systems under IDs

Method for query answering in a (G)LAV system $\mathcal{I}$ with IDs:

1. Transform $\mathcal{I}$ into a GAV system $\mathcal{I}'$
2. Apply the query answering method for GAV systems under IDs (The unfolding step must take into account the presence of auxiliary global symbols)

Theorem

Answering conjunctive queries in (G)LAV systems under IDs is in PTIME *in data complexity (actually in* LOGSPACE *)*

Outline

(G)LAV systems under IDs and EDs

What happens if we have also EDs in the global schema?

- The above transformation of (G)LAV into GAV is still correct in the presence of EDs
- It is thus possible to first turn the (G)LAV system into a GAV one and then compute query answering in the transformed system
- The addition of EDs is completely modular (we just need to add auxiliary steps in the query answering technique)

(G)LAV systems under IDs and EDs

What happens if we have also EDs in the global schema?

- The above transformation of (G)LAV into GAV is still correct in the presence of EDs
- It is thus possible to first turn the (G)LAV system into a GAV one and then compute query answering in the transformed system
- The addition of EDs is completely modular (we just need to add auxiliary steps in the query answering technique)

Query answering in (G)LAV systems under IDs and EDs

Method for query answering in a (G)LAV system $\mathcal{I}$ with IDs and EDs:

1. Transform $\mathcal{I}$ into a GAV system $\mathcal{I}'$
2. Apply the query answering method for GAV systems under IDs and EDs
   (The unfolding step must take into account the presence of auxiliary global symbols)

Theorem

Answering conjunctive queries in (G)LAV systems under IDs end Eds is in PTIME *in data complexity (actually in* LOGSPACE *)*

Query answering in (G)LAV systems under IDs and EDs

Method for query answering in a (G)LAV system $\mathcal{I}$ with IDs and EDs:

1. Transform $\mathcal{I}$ into a GAV system $\mathcal{I}'$
2. Apply the query answering method for GAV systems under IDs and EDs
   (The unfolding step must take into account the presence of auxiliary global symbols)

Theorem

Answering conjunctive queries in (G)LAV systems under IDs end Eds is in PTIME *in data complexity (actually in* LOGSPACE *)*

Outline

(G)LAV systems under KDs

We consider a (G)LAV system with only KDs in the global schema:

- The transformation of (G)LAV into GAV is still correct in the presence of KDs
- More precisely, starting from a (G)LAV system $\mathcal{I}$ with KDs, we obtain a GAV system $\mathcal{I}'$ with KDs and IDs
- But in general, $\mathcal{I}'$ is such that the IDs added by the transformation are key-conflicting IDs (i.e., these IDs are not NKCIDs), and hence the KDs are in general not separable

Therefore, it is not possible to apply the query answering method for (G)LAV systems under separable KDs and IDs

Question: Can we find some analogous query answering method based on query rewriting?

(G)LAV systems under KDs

We consider a (G)LAV system with only KDs in the global schema:

- The transformation of (G)LAV into GAV is still correct in the presence of KDs
- More precisely, starting from a (G)LAV system $\mathcal{I}$ with KDs, we obtain a GAV system $\mathcal{I}'$ with KDs and IDs
- But in general, $\mathcal{I}'$ is such that the IDs added by the transformation are key-conflicting IDs (i.e., these IDs are not NKCIDs), and hence the KDs are in general not separable

Therefore, it is not possible to apply the query answering method for (G)LAV systems under separable KDs and IDs

Question: Can we find some analogous query answering method based on query rewriting?

(G)LAV systems under KDs – A negative result

Problem: KDs and LAV mappings derive new equality-generating dependencies (not simple KDs)

Theorem (Duschka & al., 1998)

Given a LAV data integration system $\mathcal{I}$ with KDs in the global schema and a conjunctive query q, in general there does not exist a first-order query rew such that $rew^{\mathcal{C}} = cert(q, \mathcal{I}, \mathcal{C})$ for every source database $\mathcal{C}$

In other words, in LAV with KDs, conjunctive queries are not first-order rewritable, and one would need to resort to more powerful relational query languages (e.g., Datalog)

Data integration with constraints – First-order rewritability

Can query answering in integration systems 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

Data integration with constraints – Complexity results

EDs	KDs	IDs	Data/Combined complexity
no	no	general	PTIME/PSPACE
yes-no	yes	no	PTIME/NP
yes	yes-no	no	PTIME/NP
yes-no	yes	NKC	PTIME/PSPACE
yes	no	general	PTIME/PSPACE
yes-no	yes	1KC	undecidable
yes-no	yes	general	undecidable

Outline

Inconsistency tolerance

Under the "classical" (i.e., first-order) semantics considered so far:

- if data at the sources violate (through the mapping) a single KD or ED, the integration system has no legal databases (i.e., no models)
- consequently, the certain answers to any query of arity n are all the n-tuples of constants of Γ (ex falso quodlibet)
- non-interesting case for query answering

Inconsistency tolerance: example

$\mathcal{G} = \{r/2,\ key(r) = \{1\}\}$
$\mathcal{S} = \{s/2, t/2\}$
$\mathcal{M} = \{s(X,Y) \rightsquigarrow r(X,Y), \quad t(X,Y) \rightsquigarrow r(X,Y)\}$
$\mathcal{C} = \{s(a,b), t(a,c)\}$

Query: $q = \{(X) \mid r(X,Y)\}$

There is no database in $sem^{\mathcal{C}}(\mathcal{I})$ ($\mathcal{M}(\mathcal{C})$ violates the KD on r)
$\Rightarrow$ $cert(q, \mathcal{I}, \mathcal{C}) = \{c \mid c \in \Delta\}$

However: we would like the only certain answer to q to be a

Consistent query answering (CQA)

- Study of methods and techniques for repairing a database instance that is inconsistent with the integrity constraints declared on its schema [Arenas et al., 2000]
- Peculiarity of CQA: repairing is virtual and based on a logical/declarative semantics
- Therefore, data are not changed, not a material repairing (as in data cleaning)
- The CQA principles and methods can be extended to data integration scenarios
- In the following, we only consider GAV mapping

The loosely-sound semantics

We introduce one particular semantics for inconsistency tolerance in GAV integration systems, the loosely-sound semantics [Calì et al., 2005]

- the loosely-sound semantics principle: add as much as you like (as with sound semantics), and throw away only a minimal set of tuples

Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two global databases that satisfy constraints on the global schema. Then, $\mathcal{B}_1$ is better than $\mathcal{B}_2$, denoted $\mathcal{B}_1 \succ_{(\mathcal{I},\mathcal{D})} \mathcal{B}_2$, iff

$$\mathcal{B}_1 \cap \mathcal{M}(\mathcal{C}) \supset \mathcal{B}_2 \cap \mathcal{M}(\mathcal{C})$$

The answers $cert_\ell(Q, \mathcal{I}, \mathcal{D})$ to a query are those that are true on all "best" legal global databases w.r.t. $\succ_{(\mathcal{I},\mathcal{D})}$

Example

Global schema: $\mathsf{player}(Pname, YOB, Pteam)$
$\mathsf{team}(Tname, Tcity, Tleader)$

Constraints: $\mathsf{team}[Tleader, Tname] \subseteq \mathsf{player}[Pname, Pteam]$
$key(\texttt{player}) = \{Pname\}$

Mapping:

$$\begin{array}{rcl} \text{player} & : & \begin{cases} \mathsf{s}_1(X,Y,Z) & \rightsquigarrow & \mathsf{player}(X,Y,Z) \\ \mathsf{s}_3(X,Y,Z) & \rightsquigarrow & \mathsf{player}(X,Y,Z) \end{cases} \\ \text{team} & : & \mathsf{s}_2(X,Y,Z) \rightsquigarrow \mathsf{team}(X,Y,Z) \end{array}$$

Example (continued)

Source database $\mathcal{C}$

s_1:

Totti	1971	Roma
Vieri	1950	Inter

s_2:

Juve	Torino	Del Piero

s_3:

Vieri	1970	Inter

Example (IDs – KDs)

Retrieved global database $\mathcal{M}(\mathcal{C})$

player:

Totti	1971	Roma
Vieri	1970	Inter
Vieri	1950	Inter

team:

Juve	Torino	Del Piero

In $\mathcal{M}(\mathcal{C})$ there is a violation of the KD and a violation of the ID

There are two possible ways of repairing the violation of the KD with a minimum deletion of tuples.

Example (continued)

First form

player:

Totti	1971	Roma
Vieri	1970	Inter
Del Piero	α	Juve

team:

Juve	Torino	Del Piero

Second form

player:

Totti	1971	Roma
Vieri	1950	Inter
Del Piero	α	Juve

team:

Juve	Torino	Del Piero

For query $\mathsf{q} = \{(X, Z) \mid \mathsf{player}(X, Y, Z)\}$
we have
$cert_\ell(\mathsf{q}, \mathcal{I}, \mathcal{C}) = \{\langle \mathtt{Totti}, \mathtt{Roma}\rangle, \langle \mathtt{Vieri}, \mathtt{Inter}\rangle, \langle \mathtt{Del\ Piero}, \mathtt{Juve}\rangle\}$

Query rewriting under the loosely-sound semantics

Query language: Datalog$^\neg$ under stable model semantics

Rewriting under KDs: set of rules Π_{KD} that take KDs into account for each KD $key(r) = \{X_1, \ldots X_n\}$ in $\mathcal{G}$:

$$
\begin{aligned}
& r(\vec{x}, \vec{y}) \leftarrow r_{\mathcal{D}}(\vec{x}, \vec{y}) \ , \ not\, \overline{r}(\vec{x}, \vec{y}) \\
& \overline{r}(\vec{x}, \vec{y}) \leftarrow r_{\mathcal{D}}(\vec{x}, \vec{y}) \ , \ r(\vec{x}, \vec{z}) \ , \ Y_1 \neq Z_1 \\
& \ldots \\
& \overline{r}(\vec{x}, \vec{y}) \leftarrow r_{\mathcal{D}}(\vec{x}, \vec{y}) \ , \ r(\vec{x}, \vec{z}) \ , \ Y_m \neq Z_m
\end{aligned}
$$

where $\vec{x} = X_1, \ldots, X_n$, $\vec{y} = Y_1, \ldots, Y_m$ and $\vec{z} = Z_1, \ldots, Z_m$

Query rewriting under the loosely-sound semantics

Theorem

$\Pi_{ID} \cup \Pi_{KD} \cup \Pi_{\mathcal{MC}}$ *is a perfect rewriting of* q*, where:*

- $\Pi_{\mathcal{MC}}$ = *rules obtained from the mapping rules* $\Pi_{\mathcal{M}}$ *by replacing each* r *with* $r^{\mathcal{C}}$
- Π_{ID} = *rewriting of the query* q *obtained by the algorithm ID-rewrite*

Remark: $\Pi_{ID} \cup \Pi_{KD} \cup \Pi_{\mathcal{MC}}$ is a Datalog$^{\neg}$ program (and is interpreted under stable model semantics)

Note: a stable model of a Datalog$^{\neg}$ program Π is any set σ of ground atoms that coincides with the unique minimal Herbrand model of the Datalog$^{\neg}$ progam Π_σ, where Π_σ is obtained from Π by deleting every rule that has a negative literal $\neg B$ with $B \in \sigma$, and all negative literals in the bodies of the remaining rules.

What about Exclusion Dependencies (EDs)?

We extend the previous example with an ED:

Global schema: $\mathsf{player}(Pname, YOB, Pteam)$
$\mathsf{team}(Tname, Tcity, Tleader)$
$\mathsf{coach}(Cname, Cteam)$

Constraints: $\mathsf{team}[Tleader,\ Tname] \subseteq \mathsf{player}[Pname,\ Pteam]$
$\mathsf{coach}[Cname] \cap \mathsf{player}[Pname] = \emptyset$
$\mathit{key}(player) = \{Pname, Pteam\}$
$\mathit{key}(team) = \{Tname\}$ $\mathit{key}(coach) = \{Cname\}$

Mapping: $\mathsf{player}(X, Y, Z) \leftarrow \mathsf{s}_1(X, Y, Z)$
$\mathsf{player}(X, Y, Z) \leftarrow \mathsf{s}_3(X, Y, Z)$
$\mathsf{team}(X, Y, Z) \leftarrow \mathsf{s}_2(X, Y, Z)$
$\mathsf{coach}(X, Y) \leftarrow \mathsf{s}_4(X, Y)$

Example (continued)

Source database $\mathcal{C}$

s_1:

Totti	1971	Roma

s_2:

Juve	Torino	Del Piero

s_3:

Vieri	1970	Inter

s_4:

Del Piero	Viterbese

Example IDs – EDs

Retrieved global database $\mathcal{M}(\mathcal{C})$

player:

Totti	1971	Roma
Vieri	1970	Inter

team:

Juve	Torino	Del Piero

coach:

Del Piero	Viterbese

There are two possible ways of repairing the violation with a minimum deletion of tuples.

Example (continued)

First form

player:

Totti	1971	Roma
Vieri	1950	Inter
Del Piero	α	Juve

team:

Juve	Torino	Del Piero

Second form

player:

Totti	1971	Roma
Vieri	1950	Inter

coach:

Del Piero	Viterbese

For the query $\mathsf{q} = \{(X, Z) \mid \mathsf{player}(X, Y, Z)\}$
we have
$cert_\ell(\mathsf{q}, \mathcal{I}, \mathcal{D}) = \{\langle \mathtt{Totti}, \mathtt{Roma}\rangle, \langle \mathtt{Vieri}, \mathtt{Inter}\rangle\}$

Query rewriting for EDs

Set of rules Π_{ED} that take EDs into account

For each exclusion dependency $r[\mathbf{A}] \cap s[\mathbf{B}] = \emptyset$ in the closure of EDs wrt logical implication by IDs and EDs:

$$
\begin{aligned}
r(\vec{x}, \vec{y}) &\leftarrow r_{\mathcal{D}}(\vec{x}, \vec{y}) \ , \ not\ \overline{r}(\vec{x}, \vec{y}) \\
s(\vec{x}, \vec{y}) &\leftarrow s_{\mathcal{D}}(\vec{x}, \vec{y}) \ , \ not\ \overline{s}(\vec{x}, \vec{y}) \\
\overline{r}(\vec{x}, \vec{y}) &\leftarrow r_{\mathcal{D}}(\vec{x}, \vec{y}) \ , \ s(\vec{x}, \vec{z}) \\
\overline{s}(\vec{x}, \vec{y}) &\leftarrow s_{\mathcal{D}}(\vec{x}, \vec{y}) \ , \ r(\vec{x}, \vec{z})
\end{aligned}
$$

where in $r(\vec{x}, \vec{z})$ the variables in $\vec{x}$ correspond to the sequence of attributes $\mathbf{A}$ of r, and in $s(\vec{x}, \vec{z})$ the variables in $\vec{x}$ correspond to the sequence of attributes $\mathbf{B}$ of s.

Query rewriting under the loosely-sound semantics: IDs, KDs and EDs

Theorem

$\Pi_{ID} \cup \Pi_{KD} \cup \Pi_{ED} \cup \Pi_{\mathcal{MC}}$ *is a perfect rewriting of* Q.

Example

Global schema: $\mathsf{player}(Pname, YOB, Pteam)$
$\mathsf{team}(Tname, Tcity, Tleader)$
$\mathsf{coach}(Cname, Cteam)$

Constraints: $\mathsf{team}[Tleader,\ Tname] \subseteq \mathsf{player}[Pname,\ Pteam]$
$\mathsf{coach}[Cname] \cap \mathsf{player}[Pname] = \emptyset$
$\mathit{key}(player) = \{Pname, Pteam\}$
$\mathit{key}(team) = \{Tname\} \quad \mathit{key}(coach) = \{Cname\}$

Mapping: $\mathsf{player}(X, Y, Z) \leftarrow \mathsf{s}_1(X, Y, Z)$
$\mathsf{player}(X, Y, Z) \leftarrow \mathsf{s}_3(X, Y, Z)$
$\mathsf{team}(X, Y, Z) \leftarrow \mathsf{s}_2(X, Y, Z)$
$\mathsf{coach}(X, Y) \leftarrow \mathsf{s}_4(X, Y)$

Query: $\mathsf{q} = \{(X, Z) \mid \mathsf{player}(X, Y, Z)\}$

Example (continued)

Rewriting of the query q:

$$
\begin{aligned}
\mathsf{q}(X,Z) &\leftarrow \mathsf{player}(X,Y,Z) \\
\mathsf{q}(X,Z) &\leftarrow \mathsf{team}(Z,Y,X) \\
\mathsf{player}(X,Y,Z) &\leftarrow \mathsf{player}_{\mathcal{D}}(X,Y,Z)\ ,\ \mathit{not}\ \overline{\mathsf{player}}(X,Y,Z) \\
\overline{\mathsf{player}}(X,Y,Z) &\leftarrow \mathsf{player}_{D}(X,Y,Z)\ ,\ \mathsf{player}(X,W,Z)\ ,\ Y \neq W \\
\mathsf{team}(X,Y,Z) &\leftarrow \mathsf{team}_{\mathcal{D}}(X,Y,Z)\ ,\ \mathit{not}\ \overline{\mathsf{team}}(X,Y,Z) \\
\overline{\mathsf{team}}(X,Y,Z) &\leftarrow \mathsf{team}_{\mathcal{D}}(X,Y,Z)\ ,\ \mathsf{team}(X,V,W)\ ,\ Y \neq V \\
\overline{\mathsf{team}}(X,Y,Z) &\leftarrow \mathsf{team}_{\mathcal{D}}(X,Y,Z)\ ,\ \mathsf{team}(X,V,W)\ ,\ Z \neq W \\
\mathsf{coach}(X,Y) &\leftarrow \mathsf{coach}_{\mathcal{D}}(X,Y)\ ,\ \mathit{not}\ \overline{\mathsf{coach}}(X,Y) \\
\overline{\mathsf{coach}}(X,Y) &\leftarrow \mathsf{coach}_{D}(X,Y)\ ,\ \mathsf{coach}(X,Z)\ ,\ Y \neq Z
\end{aligned}
$$

Example (continued)

Rewriting of the query q (continued):

$$
\begin{aligned}
\overline{\mathsf{player}}(X,Y,Z) &\leftarrow \mathsf{player}_{\mathcal{D}}(X,Y,Z)\ ,\ \mathsf{coach}(X,V)\\
\overline{\mathsf{coach}}(X,Y) &\leftarrow \mathsf{coach}_{\mathcal{D}}(X,Y)\ ,\ \mathsf{player}(X,Z,V)\\
\overline{\mathsf{coach}}(X,Y) &\leftarrow \mathsf{coach}_{\mathcal{D}}(X,Y)\ ,\ \mathsf{team}(Z,V,X)\\
\overline{\mathsf{team}}(X,Y,Z) &\leftarrow \mathsf{team}_{\mathcal{D}}(X,Y,Z)\ ,\ \mathsf{coach}(Z,V)\\
\mathsf{player}_{\mathcal{D}}(X,Y,Z) &\leftarrow \mathsf{s}_1(X,Y,Z)\\
\mathsf{player}_{\mathcal{D}}(X,Y,Z) &\leftarrow \mathsf{s}_3(X,Y,Z)\\
\mathsf{team}_{\mathcal{D}}(X,Y,Z) &\leftarrow \mathsf{s}_2(X,Y,Z)\\
\mathsf{coach}_{\mathcal{D}}(X,Y) &\leftarrow \mathsf{s}_4(X,Y)
\end{aligned}
$$

Complexity of consistent query answering

- With respect to standard strictly-sound semantics, the loosely-sound semantics adds complexity to query answering
- Intuitive explanation: the number of repairs of a system with even a single KD may be exponential in the size of the source database
- Consequence: query answering is not tractable in data complexity (while it is tractable under strictly-sound semantics)
- Such increase of complexity is general (shared by all approaches to consistent query answering)

Summary of complexity results

EDs	KDs	IDs	strictly-sound	loosely-sound
no	no	GEN	PTIME/PSPACE	PTIME/PSPACE
yes-no	yes	no	PTIME/NP	coNP/Π_2^p
yes	yes-no	no	PTIME/NP	coNP/Π_2^p
yes-no	yes	NKC	PTIME/PSPACE	coNP/PSPACE
yes	no	GEN	PTIME/PSPACE	coNP/PSPACE
yes-no	yes	1KC	undecidable	undecidable
yes-no	yes	GEN	undecidable	undecidable

Data/Combined complexity

Part IV

Concluding remarks

Outline

Outline

Further issues and open problems

- Further forms of constraints, e.g.,
  - KDs with restricted forms of key-conflicting IDs
  - ontology languages, description logics, RDF
    [Calvanese & al. PODS'98, Calvanese & al., KR'06]
- Semistructured data and XML
  - constraints (DTDs, XML Schema, ...)
  - query languages (transitive closure)
- Finite models vs. unrestricted models [Rosati, PODS'06]
- Data exchange and materialization

Acknowledgements

- Andrea Calì
- Diego Calvanese
- Giuseppe De Giacomo
- Domenico Lembo
- Riccardo Rosati
- Moshe Y. Vardi

References I


S. Abiteboul and O. Duschka.
Complexity of answering queries using materialized views.
In *Proc. of PODS'98*, pages 254–265, 1998.


A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini.
On the expressive power of data integration systems.
In *Proc. of ER 2002*, 2002.


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


A. Calì, D. Lembo, and R. Rosati.
Query rewriting and answering under constraints in data integration systems.
In *Proc. of IJCAI 2003*, pages 16–21, 2003.

References II


D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.
Data complexity of query answering in description logics.
In *Proc. of KR 2006*, 2006.


D. Calvanese, G. De Giacomo, and M. Lenzerini.
On the decidability of query containment under constraints.
In *Proc. of PODS'98*, pages 149–158, 1998.


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 ICDT 2005*, volume 3363 of *LNCS*, pages 321–336. Springer, 2005.


D. Calvanese and R. Rosati.
Answering recursive queries under keys and foreign keys is undecidable.
In *Proc. of KRDB 2003*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-79/`, 2003.

References III


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.


A. Fuxman and R. J. Miller.
First-order query rewriting for inconsistent databases.
In *Proc. of ICDT 2005*, volume 3363 of *LNCS*, pages 337–351. Springer, 2005.


G. Grahne and A. O. Mendelzon.
Tableau techniques for querying information sources through global schemas.
In *Proc. of ICDT'99*, volume 1540 of *LNCS*, pages 332–347. Springer, 1999.


A. Y. Halevy.
Answering queries using views: A survey.
VLDB Journal, 10(4):270–294, 2001.

References IV

M. Lenzerini.

Data integration: A theoretical perspective.

In *Proc. of PODS 2002*, pages 233–246, 2002.

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 SEBD 2005*, pages 55–66, 2005.

A. Y. Levy, A. Rajaraman, and J. J. Ordille.

Query answering algorithms for information agents.

In *Proc. of AAAI'96*, pages 40–47, 1996.

References V

R. Rosati.

On the decidability and finite controllability of query processing in databases with incomplete information.

In *Proc. of PODS 2006*, 2006.