Data Integration

Introduction

Giuseppe De Giacomo

Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
Università di Roma "La Sapienza"

Seminari di Ingegneria Informatica: Integrazione di Dati e Servizi

A.A. 2006/07

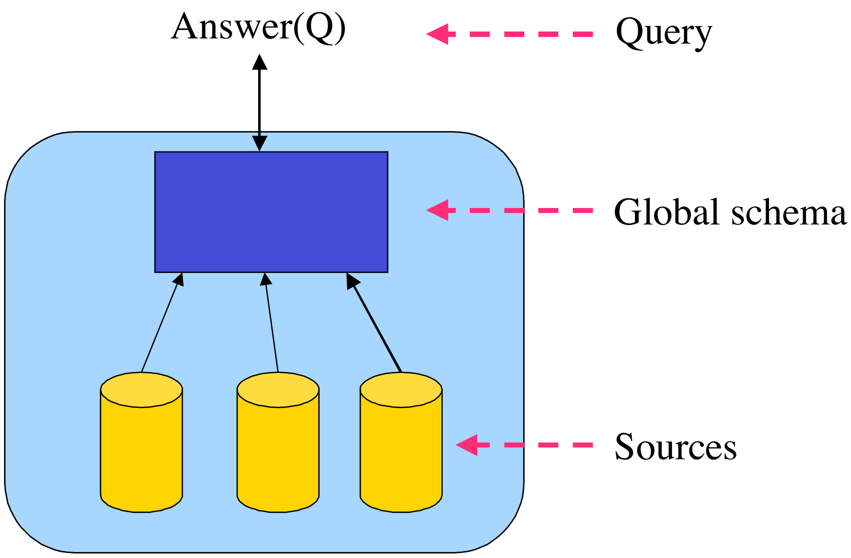
Laurea Specialistica in Ingegneria Informatica

Università di Roma "La Sapienza"

Data integration: outline

- Introduction to data integration
- Data integration: logical formalization
- Query answering in GAV data integration systems
- Query answering in LAV data integration systems

Data integration

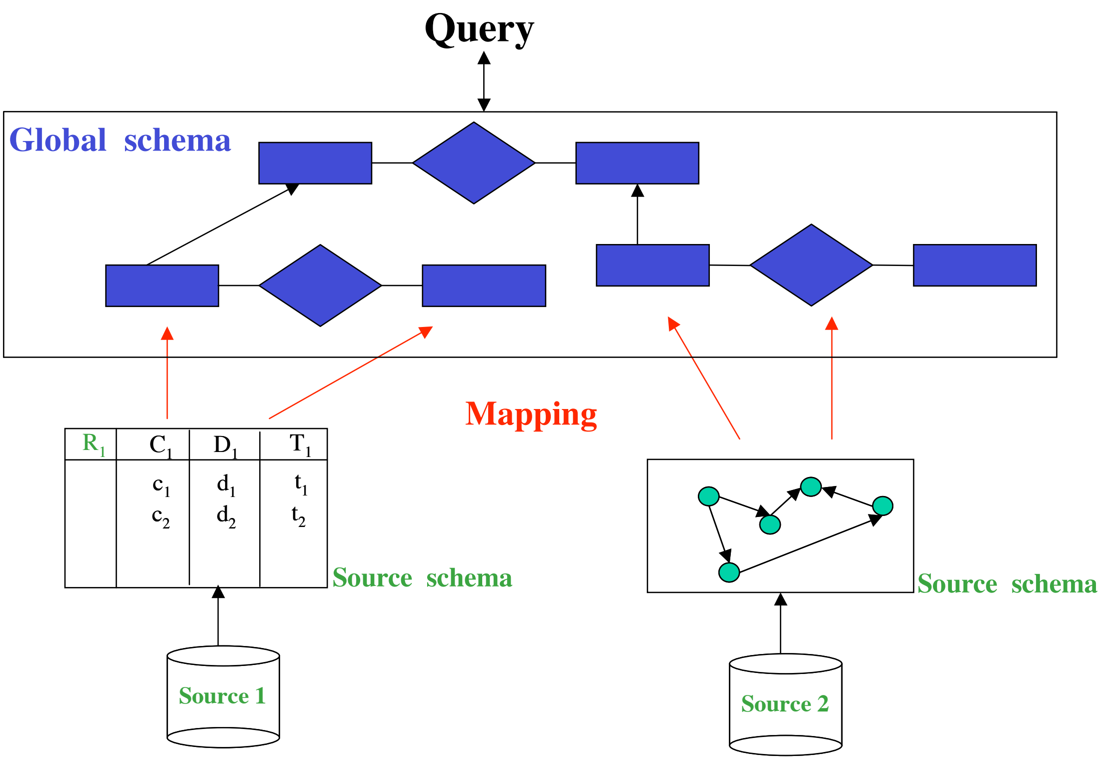

Answer(Q)

Query

Global schema

Sources

Data integration


Query

Global schema

Mapping

R_1	C_1	D_1	T_1
	c_1	d_1	t_1
	c_2	d_2	t_2

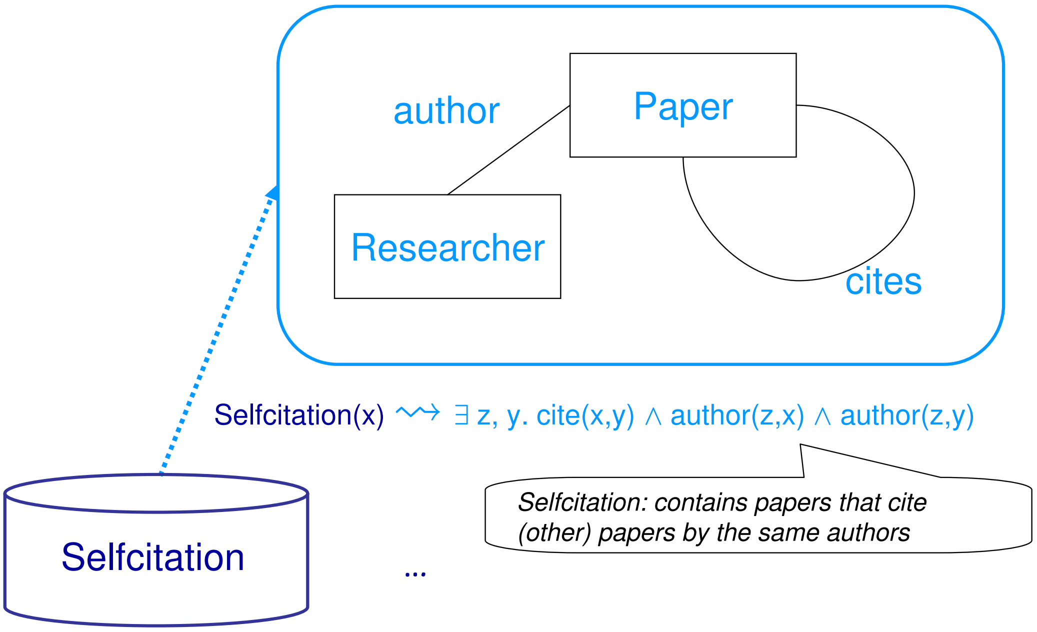
Source schema

Source schema

Source 1

Source 2

An example



$\text{Selfcitation}(x) \rightsquigarrow \exists\, z, y.\ \text{cite}(x,y) \wedge \text{author}(z,x) \wedge \text{author}(z,y)$

Selfcitation: contains papers that cite (other) papers by the same authors

IT hype

The current trend in IT industry is operating in on-demand environments. Operating on-demand is based on three key elements:

- **Integration**: *"Integration creates the necessary business flexibility to optimize operations across and beyond the enterprise".*
- **Automation**: *"Automation reduces the complexity and cost of IT management and improves the availability and the resilience".*
- **Virtualization**: *"Virtualization provides a single consolidated view of (and easy access to) all available resources, which improves working capital and asset utilization".*

Data integration

Data integration is the problem of providing unified and transparent access to a set of autonomous and heterogeneous sources

- 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

Integrated access to distributed data

Different approaches/architectures:

- **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 (masking from the user the physical characteristics of the sources)
- Heterogeinity (federating highly diverse types of sources)
- Extensibility
- Autonomy of data sources
- Performance (distributed query optimization)

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

Logical transparency

Basic ingredients for achieving logical transparency:

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

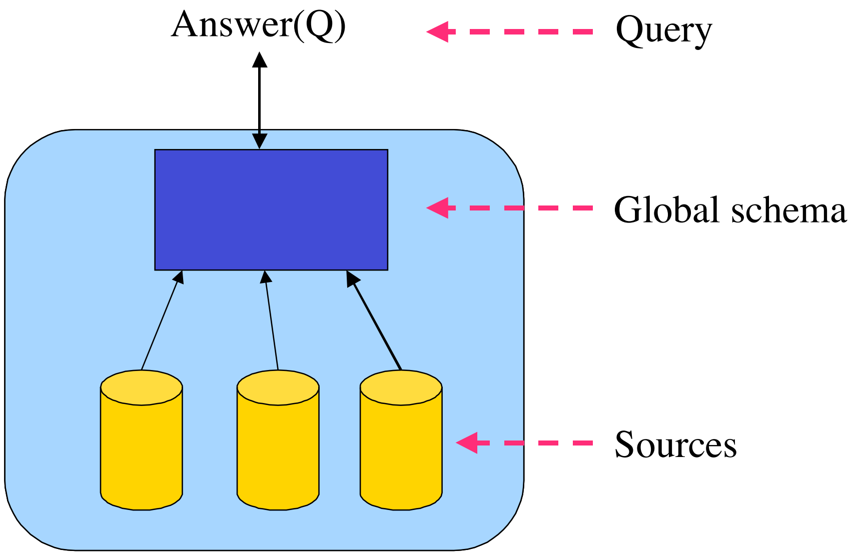
All the above aspects are not appropriately dealt with by current tools. This means that data integration cannot be simply addressed on a tool basis.

Variants of data integration

- (Mediator-based) data integration
  - queries over the global schema
- Data exchange
  - materialization of the global view
- P2P data integration
  - several peers
  - each peer with local and external sources
  - queries over one peer

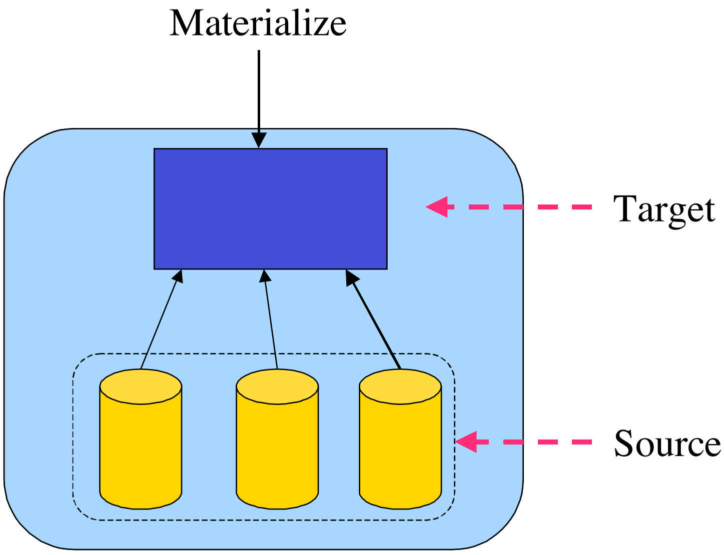
Data integration

- Queries over the global schema

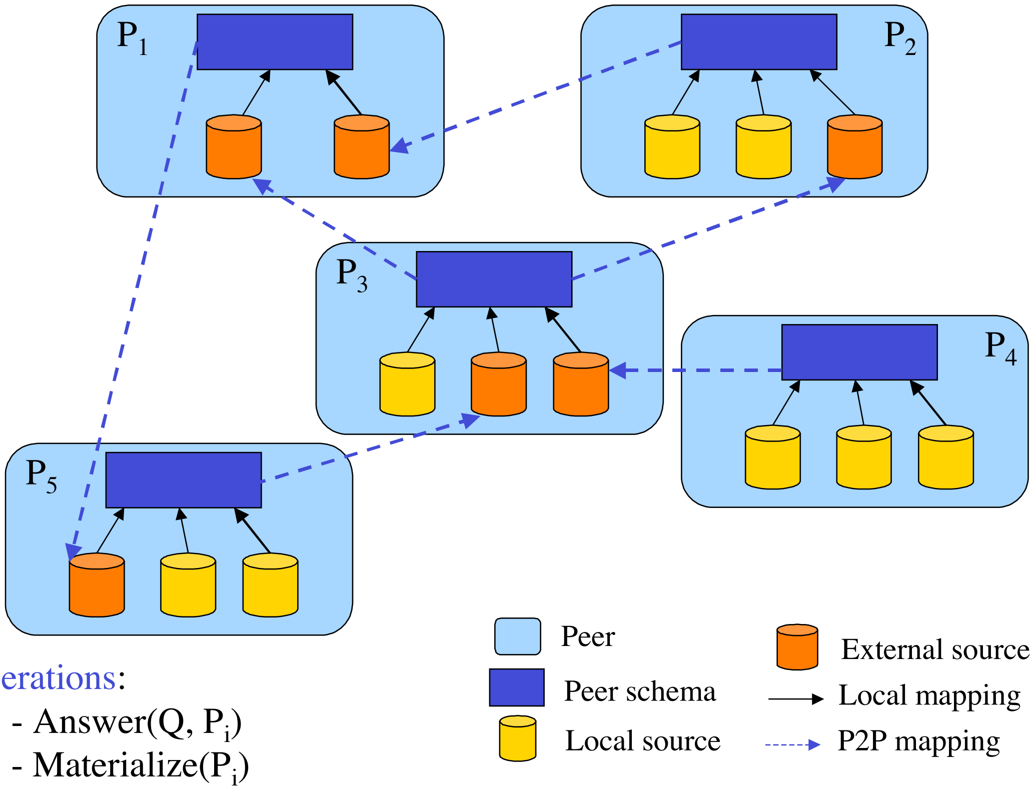


Data exchange

- Materialization of the global view



Peer-to-peer data integration



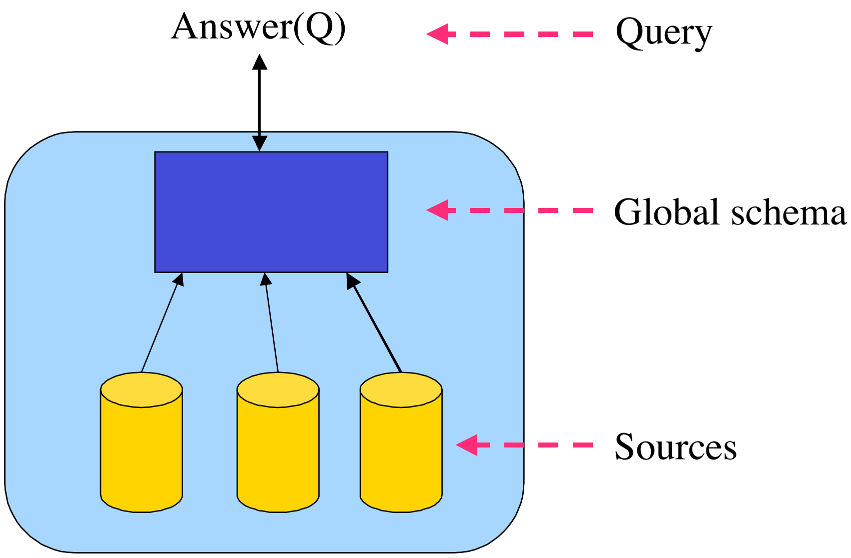
Operations:

- Answer(Q, P_i)
- Materialize(P_i)

Data integration: outline

- Introduction to data integration
- Data integration: logical formalization
- Query answering in GAV data integration systems
- Query answering in LAV data integration systems

Data integration



Main problems in data integration

1. How to construct the global schema
2. (Automatic) source wrapping
3. How to discover mappings between the sources and the global schema
4. Limitations in the mechanisms for accessing the sources
5. Data extraction, cleaning and reconciliation
6. How to process updates expressed on the global schema, and updates expressed on the sources (“read/write” vs “read-only” data integration)
7. The modeling problem: How to model the mappings between the sources and the global schema
8. The querying problem: How to answer queries expressed on the global schema

   This is view-based query answering!
9. Query optimization

Data integration: outline

- Introduction to data integration
- Data integration: logical formalization
- Query answering in GAV data integration systems
- Query answering in LAV data integration systems

Formal framework for data integration

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

- $\mathcal{G}$ is the global schema

  The global schema is a logical theory over an alphabet $\mathcal{A}_{\mathcal{G}}$

- $\mathcal{S}$ is the source schema

  The source schema is constituted simply by an alphabet $\mathcal{A}_{\mathcal{S}}$ *disjoint from* $\mathcal{A}_{\mathcal{G}}$

- $\mathcal{M}$ is the mapping between $\mathcal{S}$ and $\mathcal{G}$

  Different approaches to the specification of mapping

Semantics of a data integration system

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

The databases that satisfy $\mathcal{I}$ are logical interpretations for $\mathcal{A}_\mathcal{G}$ (called global databases). We refer only to databases over a fixed infinite domain Γ of constants.

Let $\mathcal{C}$ be **a source database** over Γ (also called source model), fixing the extension of the predicates of $\mathcal{A}_\mathcal{S}$ (thus modeling the data present in the sources).

The set of models of (i.e., databases for $\mathcal{A}_\mathcal{G}$ that satisfy) $\mathcal{I}$ relative to $\mathcal{C}$ is:

$$\boxed{\begin{array}{l} sem^{\mathcal{C}}(\mathcal{I}) = \{\, \mathcal{B} \;\mid\; \mathcal{B} \text{ is a } \mathcal{G}\text{-model (i.e., a global database that is legal wrt } \mathcal{G}) \\ \qquad\qquad\qquad\qquad \text{and is an } \mathcal{M}\text{-model wrt } \mathcal{C} \text{ (i.e., satisfies } \mathcal{M} \text{ wrt } \mathcal{C}) \;\} \end{array}}$$

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

Semantics of queries to $\mathcal{I}$

A **query** q of arity n is a formula with n free variables.

If $\mathcal{D}$ is a database, then $q^{\mathcal{D}}$ denotes the extension of q in $\mathcal{D}$ (i.e., the set of n-tuples that are valuations in Γ for the free variables of q that make q true in $\mathcal{D}$).

If q is a query of arity n posed to a data integration system $\mathcal{I}$ (i.e., a formula over $\mathcal{A}_\mathcal{G}$ with n free variables), then the set of **certain answers to q wrt $\mathcal{I}$ and $\mathcal{C}$** is

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

Note: query answering is logical implication.

Note: complexity will be mainly measured wrt the size of the source database $\mathcal{C}$, and will refer to the problem of deciding whether $\vec{\mathbf{c}} \in cert(q, \mathcal{I}, \mathcal{C})$, for a given $\vec{\mathbf{c}}$.

Databases with incomplete information, or Knowledge Bases

- Traditional database: one model of a first-order theory

  Query answering means evaluating a formula in the model

- Database with incomplete information, or Knowledge Base: set of models (specified, for example, as a restricted first-order theory)

  Query answering means computing the tuples that satisfy the query in all the models in the set

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

Query answering with incomplete information

- [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

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

$\exists D.$ movie$(T, 1998, D) \wedge$ review(T, R), written

$\{\ (T, R) \mid$ movie$(T, 1998, D) \wedge$ review$(T, R)\ \}$

Formalization of LAV

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

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

one for each source element s in $\mathcal{A}_{\mathcal{S}}$, where $\phi_{\mathcal{G}}$ is a **query** over $\mathcal{G}$ of the arity of s.

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

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

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

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: associated to source relations we have views over the global schema

$$\mathsf{r}_1(T, Y, D) \quad \rightsquigarrow \quad \{\ (T, Y, D) \mid \mathsf{movie}(T, Y, D) \wedge \mathsf{european}(D) \wedge Y \geq 1960\ \}$$

$$\mathsf{r}_2(T, R) \quad \rightsquigarrow \quad \{\ (T, R) \mid \mathsf{movie}(T, Y, D) \wedge \mathsf{review}(T, R) \wedge Y \geq 1990\ \}$$

The query $\{\ (T, R) \mid \mathsf{movie}(T, 1998, D) \wedge \mathsf{review}(T, R)\ \}$ 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:

$$\{\ (T, R) \ \mid\ \mathsf{r}_2(T, R) \wedge \mathsf{r}_1(T, 1998, D)\ \}$$

Formalization of GAV

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

$$g \;\leadsto\; \phi_{\mathcal{S}}$$

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

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

$$g^{\mathcal{B}} \;\supseteq\; {\phi_{\mathcal{S}}}^{\mathcal{C}}$$

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

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: associated to relations in the global schema we have views over the sources

$$\text{movie}(T, Y, D) \;\leadsto\; \{\, (T, Y, D) \mid \text{r}_1(T, Y, D) \,\}$$

$$\text{european}(D) \;\leadsto\; \{\, (D) \mid \text{r}_1(T, Y, D) \,\}$$

$$\text{review}(T, R) \;\leadsto\; \{\, (T, R) \mid \text{r}_2(T, R) \,\}$$

GAV – example (constraints) – *see more later*

Global schema containing constraints:

$\text{movie}(Title, Year, Director)$ $\quad$ $\text{european}(Director)$ $\quad$ $\text{review}(Title, Critique)$

$\text{european\_movie\_}60s(Title, Year, Director)$

$$\forall T, Y, D.\ \text{european\_movie\_}60s(T, Y, D) \ \supset\ \text{movie}(T, Y, D)$$

$$\forall D.\ \exists T, Y.\ \text{european\_movie\_}60s(T, Y, D) \ \supset\ \text{european}(D).$$

GAV mappings:

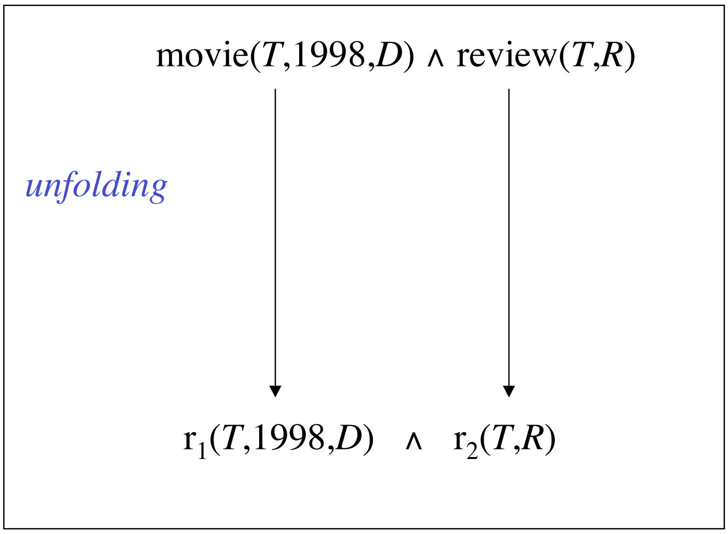
$$\text{european\_movie\_}60s(T, Y, D) \leadsto \{\ (T, Y, D) \mid \text{r}_1(T, Y, D)\ \}$$

$$\text{european}(D) \leadsto \{\ (D) \mid \text{r}_1(T, Y, D)\ \}$$

$$\text{review}(T, R) \leadsto \{\ (T, R) \mid \text{r}_2(T, R)\ \}$$

GAV – example of query processing

The query $\{\ (T, R) \mid \text{movie}(T, 1998, D) \wedge \text{review}(T, R)\ \}$ is processed by means of unfolding, i.e., by expanding each atom according to its associated definition in $\mathcal{M}$, so as to come up with source relations. In this case:



$\text{movie}(T,1998,D) \wedge \text{review}(T,R)$

unfolding

$\text{r}_1(T,1998,D) \ \wedge \ \text{r}_2(T,R)$

GAV and LAV – comparison

LAV: (Information Manifold, DWQ)

- 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)

GAV: (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)

Beyond GAV and LAV: GLAV

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

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

where $\phi_{\mathcal{S}}$ is a **query** over $\mathcal{S}$, and $\phi_{\mathcal{G}}$ is a **query** over $\mathcal{G}$ of the arity $\phi_{\mathcal{S}}$.

Given source database $\mathcal{C}$, a database $\mathcal{B}$ that is legal wrt $\mathcal{G}$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$ if for each assertion in $\mathcal{M}$:

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

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

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 $\mathcal{C}$.

Example of GLAV

Global schema: $Work(Person, Project),\quad Area(Project, Field)$

Source 1: $HasJob(Person, Field)$

Source 2: $Teach(Professor, Course),\ \ In(Course, Field)$

Source 3: $Get(Researcher, Grant),\ \ For(Grant, Project)$

GLAV mapping:

$$\{\,(r,f) \mid HasJob(r,f)\,\} \rightsquigarrow \{\,(r,f) \mid Work(r,p) \land Area(p,f)\,\}$$
$$\{\,(r,f) \mid Teach(r,c) \land In(c,f)\,\} \rightsquigarrow \{\,(r,f) \mid Work(r,p) \land Area(p,f)\,\}$$
$$\{\,(r,p) \mid Get(r,g) \land For(g,p)\,\} \rightsquigarrow \{\,(r,p) \mid Work(r,p)\,\}$$

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 $\forall\vec{\mathbf{x}}\,(\phi_{\mathcal{S}}(\vec{\mathbf{x}}) \rightarrow \phi_{\mathcal{G}}(\vec{\mathbf{x}}))$ with $\forall\vec{\mathbf{x}}\,(\phi_{\mathcal{S}}(\vec{\mathbf{x}}) \rightarrow r(\vec{\mathbf{x}}))$ and $\forall\vec{\mathbf{x}}\,(r(\vec{\mathbf{x}}) \rightarrow \phi_{\mathcal{G}}(\vec{\mathbf{x}}))$.

Data integration: outline

- Introduction to data integration
- Data integration: logical formalization
- Query answering in GAV data integration systems
- Query answering in LAV data integration systems

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

- Unless otherwise specified, we consider **conjunctive queries** as both user queries and queries in the mapping.
- A conjunctive query has the form

$$\{\ (\vec{\mathbf{x}}) \mid \exists \vec{\mathbf{y}}\ \ p_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}) \land \cdots \land p_m(\vec{\mathbf{x}}, \vec{\mathbf{y}})\ \}$$

- *Conjunctive query are also known as Select-Project-Join queries in Databases, and are the most common (and most optimizable) kind of queries.*

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**

GAV data integration systems

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**

Retrieved global database

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 $\mathcal{G}$:

$$\text{student}(code, name, city)$$

$$\text{university}(code, name)$$

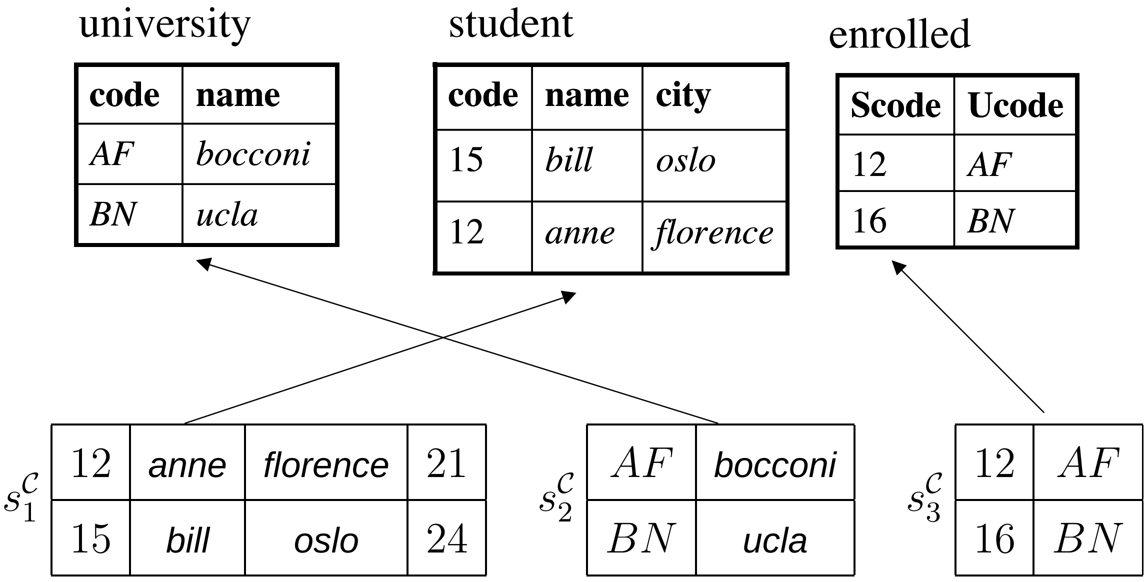
$$\text{enrolled}(Scode, Ucode)$$

Source schema $\mathcal{S}$: relations $\text{s}_1(X, Y, W, Z)$, $\text{s}_2(X, Y)$, $\text{s}_3(X, Y)$

Mapping $\mathcal{M}$:

$$\begin{array}{rcl} \text{student}(X, Y, Z) & \rightsquigarrow & \{ \, (X, Y, Z) \mid \text{s}_1(X, Y, Z, W) \, \} \\ \text{university}(X, Y) & \rightsquigarrow & \{ \, (X, Y) \mid \text{s}_2(X, Y) \, \} \\ \text{enrolled}(X, W) & \rightsquigarrow & \{ \, (X, W) \mid \text{s}_3(X, W) \, \} \end{array}$$

GAV: example



university

code	name
AF	*bocconi*
BN	*ucla*

student

code	name	city
15	*bill*	*oslo*
12	*anne*	*florence*

enrolled

Scode	Ucode
12	*AF*
16	*BN*

$s_1^{\mathcal{C}}$

12	*anne*	*florence*	21
15	*bill*	*oslo*	24

$s_2^{\mathcal{C}}$

AF	*bocconi*
BN	*ucla*

$s_3^{\mathcal{C}}$

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 $g \leadsto \phi_{\mathcal{S}}$ have the logical form:

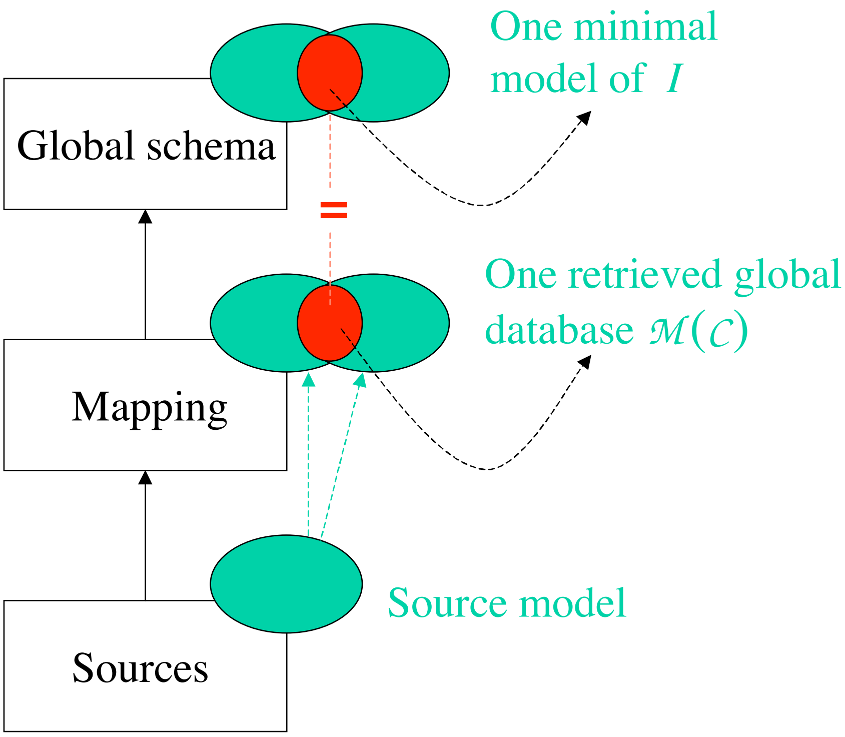
$$\forall \vec{\mathbf{x}} \;\; \phi_{\mathcal{S}}(\vec{\mathbf{x}}) \rightarrow g(\vec{\mathbf{x}})$$

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

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

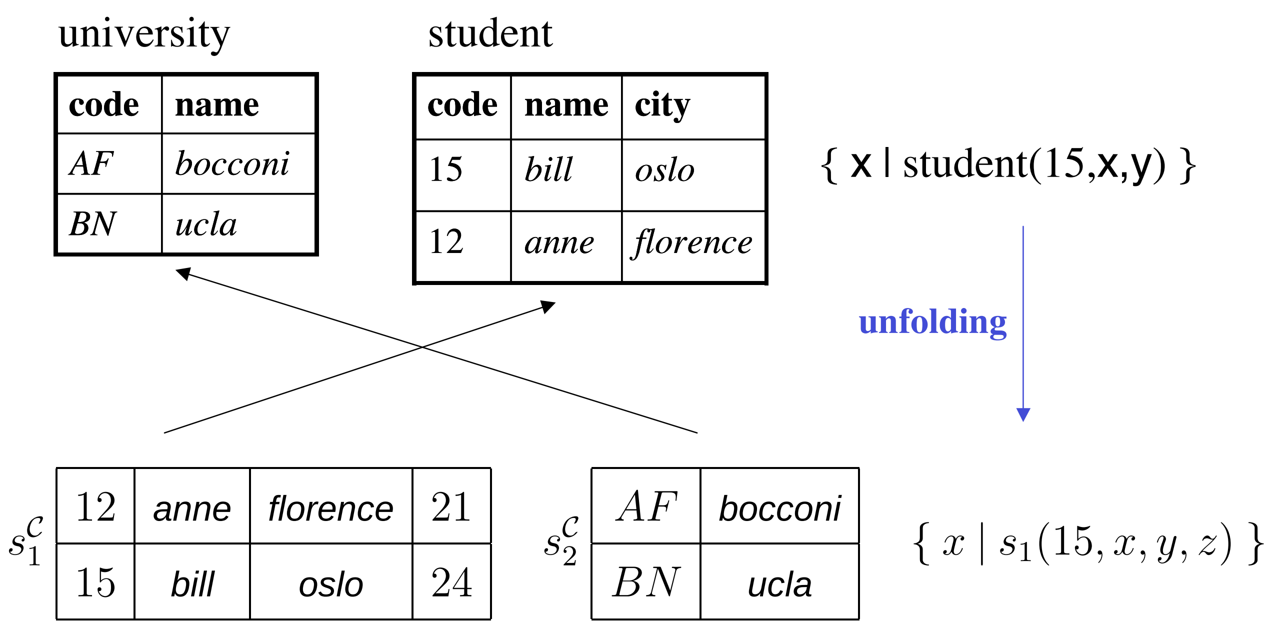


GAV: query answering

- If q is a conjunctive query, then $\vec{\mathbf{t}} \in cert(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{\mathbf{t}} \in q^{\mathcal{M}(\mathcal{C})}$
- If q is query over $\mathcal{G}$, then the unfolding of q wrt $\mathcal{M}$, $\mathit{unf}_{\mathcal{M}}(q)$, is the query over $\mathcal{S}$ obtained from q by substituting every symbol g in q with the query $\phi_{\mathcal{S}}$ that $\mathcal{M}$ associates to g
- It is easy to see that evaluating a query q over $\mathcal{M}(\mathcal{C})$ is equivalent to evaluating $\mathit{unf}_{\mathcal{M}}(q)$ over $\mathcal{C}$. It follows that, if q is a conjunctive query, then $\vec{\mathbf{t}} \in cert(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{\mathbf{t}} \in \mathit{unf}_{\mathcal{M}}(q)^{\mathcal{C}}$

  Unfolding is therefore sufficient
- **Data complexity** of query answering is **polynomial** (actually **LOGSPACE**): the query $\mathit{unf}_{\mathcal{M}}(q)$ is first-order (in fact conjunctive)
- Also, combined complexity is polynomial ($|\mathcal{M}(\mathcal{C})|$ is polynomial wrt $|\mathcal{C}|$)

GAV: example



university

code	name
AF	*bocconi*
BN	*ucla*

student

code	name	city
15	*bill*	*oslo*
12	*anne*	*florence*

{ x | student(15,x,y) }

unfolding

$s_1^{\mathcal{C}}$

12	*anne*	*florence*	21
15	*bill*	*oslo*	24

$s_2^{\mathcal{C}}$

AF	*bocconi*
BN	*ucla*

$\{ \, x \mid s_1(15, x, y, z) \, \}$

GAV: more expressive queries?

- More expressive queries in the mapping?
  - Same results hold if we use any computable query in the mapping
- 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

GAV: another view

Let B_1 and B_2 be two global databases with values in $\Gamma \cup$ Var.

- A homomorphism $h : B_1 \longrightarrow B_2$ is a mapping from ($\Gamma \cup$ Var(B_1)) to ($\Gamma \cup$ Var(B_2)) such that
  1. $h(c) = c$, for every $c \in \Gamma$
  2. for every fact $R_i(t)$ of B_1, we have that $R_i(h(t))$ is a fact in B_2 (where, if $t = (a_1, \ldots, a_n)$, then $h(t) = (h(a_1), \ldots, h(a_n))$
- B_1 is homomorphically equivalent to B_2 if there is a homomorphism $h : B_1 \longrightarrow B_2$ and a homomorphism $h' : B_2 \longrightarrow B_1$

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system. If $\mathcal{C}$ is a source database, then a universal solution for $\mathcal{I}$ relative to $\mathcal{C}$ is a model J of $\mathcal{I}$ relative to $\mathcal{C}$ such that for every model J' of $\mathcal{I}$ relative to $\mathcal{C}$, there exists a homomorphism $h : J \longrightarrow J'$ (see [Fagin&al. ICDT'03]).

GAV: another view

- Homomorphism preserves satisfaction of conjunctive queries: if there exists a homomorphism $h : J \longrightarrow J'$, and q is a conjunctive query, then $\vec{\mathbf{t}} \in q^J$ implies $\vec{\mathbf{t}} \in q^{J'}$
- Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a GAV data integration system without constraints in the global schema. If $\mathcal{C}$ is a source database, then $\mathcal{M}(\mathcal{C})$ is the minimal universal solution for $\mathcal{I}$ relative to $\mathcal{C}$
- We derive again the following results
  - if q is a conjunctive query, then $\vec{\mathbf{t}} \in cert(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{\mathbf{t}} \in q^{\mathcal{M}(\mathcal{C})}$
  - complexity of query answering is polynomial

Data integration: outline

- Introduction to data integration
- Data integration: logical formalization
- Query answering in GAV data integration systems
- Query answering in LAV data integration systems

(G)LAV data integration systems

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

(G)LAV: example

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

Global schema $\mathcal{G}$:

$$\text{student}(code, name, city)$$

$$\text{enrolled}(Scode, Ucode)$$

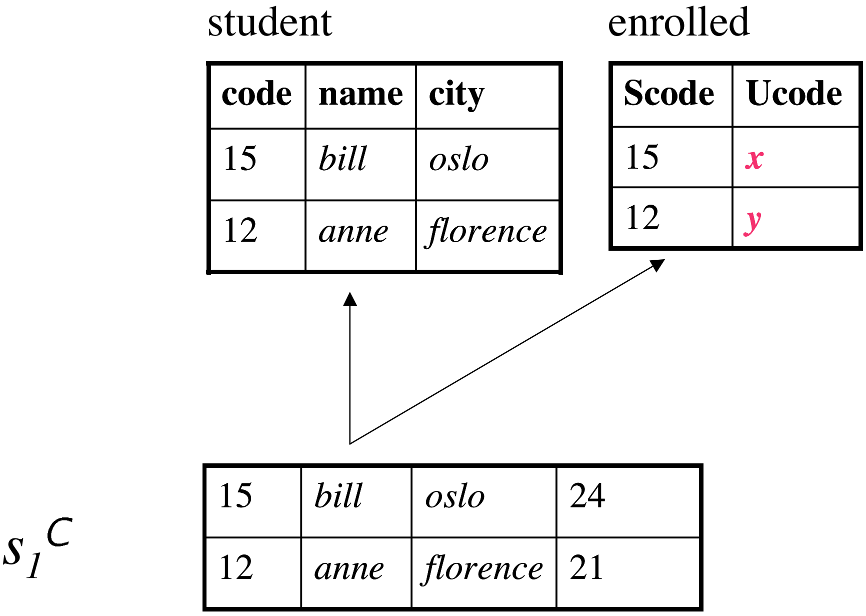
Source schema $\mathcal{S}$: relation $\mathsf{s}_1(X, Y, Z, W)$

Mapping $\mathcal{M}$:

$$\{\ (X, Y, Z) \mid \mathsf{s}_1(X, Y, Z, W)\} \leadsto \{\ (X, Y, Z) \mid \text{student}(X, Y, Z) \\ \wedge\ \text{enrolled}(X, V)\ \}$$

(G)LAV: example

$$\{\ (X,Y,Z) \mid \mathsf{s}_1(X,Y,Z,W)\} \ \leadsto\ \{\ (X,Y,Z) \mid \mathsf{student}(X,Y,Z) \land \mathsf{enrolled}(X,V)\ \}$$



student

code	name	city
15	*bill*	*oslo*
12	*anne*	*florence*

enrolled

Scode	Ucode
15	*x*
12	*y*

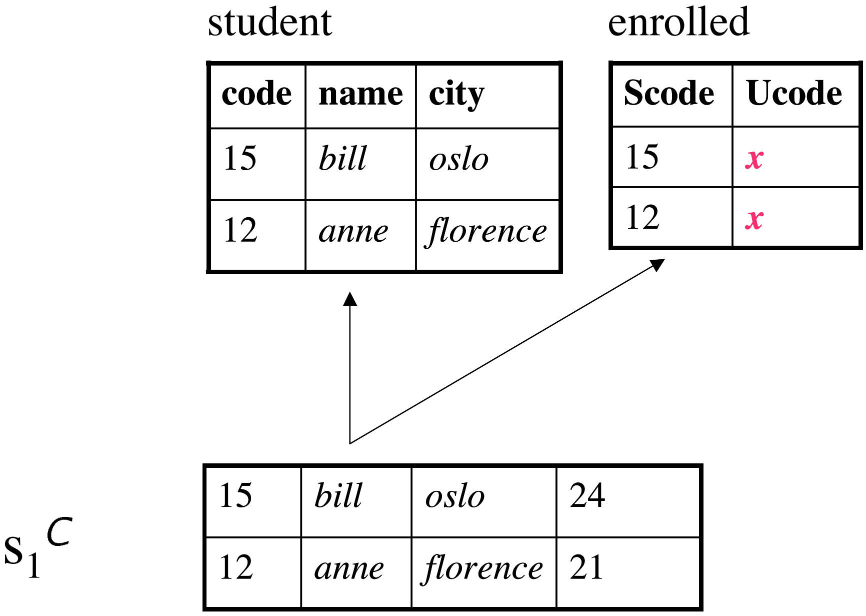
$s_1^{\mathcal{C}}$

15	*bill*	*oslo*	24
12	*anne*	*florence*	21

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

(G)LAV: example

$$\{\ (X,Y,Z) \mid \mathsf{s}_1(X,Y,Z,W)\} \ \leadsto\ \{\ (X,Y,Z) \mid \mathsf{student}(X,Y,Z) \land \mathsf{enrolled}(X,V)\ \}$$



student

code	name	city
15	*bill*	*oslo*
12	*anne*	*florence*

enrolled

Scode	Ucode
15	*x*
12	*x*

$s_1^{\mathcal{C}}$

15	*bill*	*oslo*	24
12	*anne*	*florence*	21

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

(G)LAV: incompleteness

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

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

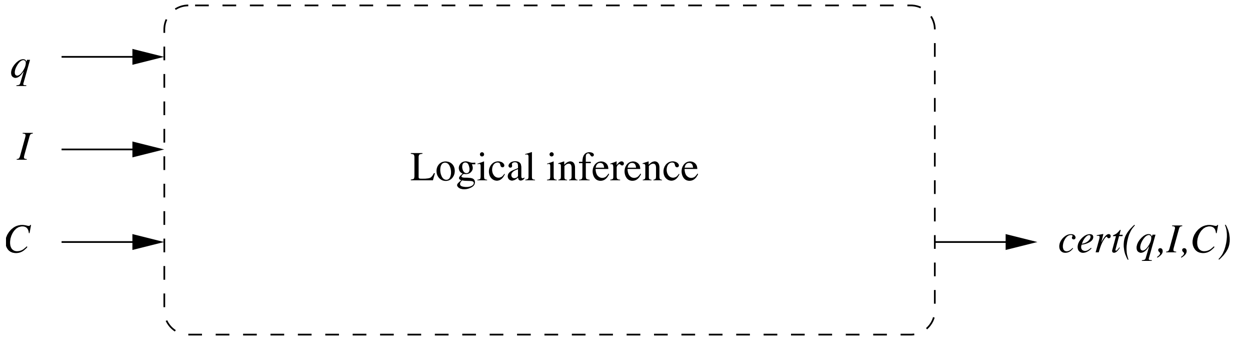
where $\phi_{\mathcal{S}}$ and $\phi_{\mathcal{G}}$ are conjunctions of atoms.

In general, given a source database $\mathcal{C}$ there are several solutions for a set of assertions of the above form (i.e., different databases that are legal wrt $\mathcal{G}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$): **incompleteness comes from the mapping**.

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

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

Query answering is based on logical inference



q

I

C

Logical inference

cert(q,I,C)

Approaches to query answering in (G)LAV systems

- Exploit connection with query containment
- Direct methods (aka view-based query answering)
- By (view-based) query rewriting

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

Connection to query containment

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

- 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 $\vec{\mathbf{x}}$ is in $s^{\mathcal{C}}$, then the corresponding literal is $s(\vec{\mathbf{x}})$)
- If q is a query, and $\vec{\mathbf{t}}$ is a tuple, then we denote by $q_{\vec{\mathbf{t}}}$ the query obtained by substituting the free variables of q with $\vec{\mathbf{t}}$
- The problem of checking whether $\vec{\mathbf{t}} \in cert(q, \mathcal{I}, \mathcal{C})$ under sound sources can be reduced to the problem of checking whether $q_{\mathcal{C}}$ **is contained in** $q_{\vec{\mathbf{t}}}$ **under the constraints** $\mathcal{G} \cup \mathcal{M}$

The **combined complexity** of checking certain answers under sound sources is identical to the complexity of query containment under constraints, and the **data complexity** is at most the complexity of query containment under constraints.

Approaches to query answering in (G)LAV systems

- Exploit connection with query containment
- Direct methods (aka view-based query answering)
- By (view-based) query rewriting

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

(G)LAV: basic technique

From [Duschka&Genesereth PODS'97]:

$$\begin{array}{lcl} \mathsf{r}_1(T) & \rightsquigarrow & \{\ (T) \mid \mathsf{movie}(T,Y,D) \wedge \mathsf{european}(D)\ \} \\ \mathsf{r}_2(T,V) & \rightsquigarrow & \{\ (T,V) \mid \mathsf{movie}(T,Y,D) \wedge \mathsf{review}(T,V)\ \} \end{array}$$

$$\begin{array}{rcl} \forall T\ \mathsf{r}_1(T) & \rightarrow & \exists Y \exists D\ \mathsf{movie}(T,Y,D) \wedge \mathsf{european}(D) \\ \forall T\ \forall V\ \mathsf{r}_2(T,V) & \rightarrow & \exists Y \exists D\ \mathsf{movie}(T,Y,D) \wedge \mathsf{review}(T,V) \end{array}$$

$$\begin{array}{rcl} \mathsf{movie}(T, f_1(T), f_2(T)) & \leftarrow & \mathsf{r}_1(T) \\ \mathsf{european}(f_2(T)) & \leftarrow & \mathsf{r}_1(T) \\ \mathsf{movie}(T, f_4(T,V), f_5(T,V)) & \leftarrow & \mathsf{r}_2(T,V) \\ \mathsf{review}(T,V) & \leftarrow & \mathsf{r}_2(T,V) \end{array}$$

- Answering a query means evaluating a goal wrt to this nonrecursive logic program (that can be transformed into a union of conjunctive query)
- PTIME (actually LOGSPACE) data complexity

(G)LAV: canonical retrieved global database

What is a retrieved global database in this case?

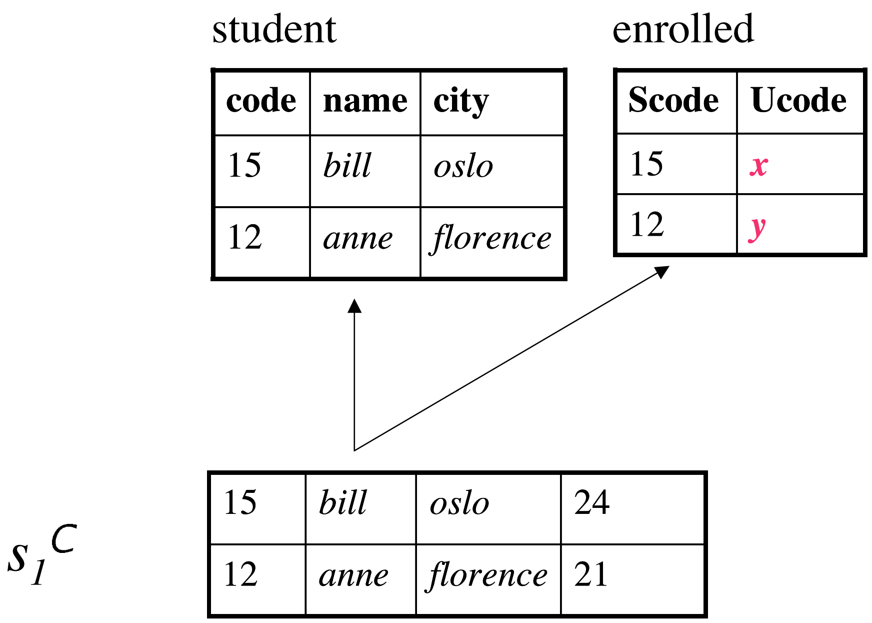
We build what we call the **canonical retrieved global database** for $\mathcal{I}$ relative to $\mathcal{C}$, denoted $\mathcal{M}(\mathcal{C})\!\downarrow$, as follows:

- let all predicates be empty in $\mathcal{M}(\mathcal{C})\!\downarrow$
- for each mapping assertion $\phi_{\mathcal{S}} \leadsto \phi_{\mathcal{G}}$ 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 $\phi_{\mathcal{G}}$

There is a unique (up to variable renaming) canonical retrieved global database for $\mathcal{I}$ relative to $\mathcal{C}$, that can be computed in polynomial time wrt the size of $\mathcal{C}$. $\mathcal{M}(\mathcal{C})\!\downarrow$ obviously satisfies $\mathcal{G}$, and is also called the **canonical model of $\mathcal{I}$ relative to $\mathcal{C}$**.

(G)LAV: example of canonical model

$\{ (X, Y, Z) \mid \mathsf{s}_1(X, Y, Z, V) \} \;\leadsto\; \{ (X, Y, Z) \mid \mathsf{student}(X, Y, Z) \wedge \mathsf{enrolled}(X, W) \}$



student

code	name	city
15	*bill*	*oslo*
12	*anne*	*florence*

enrolled

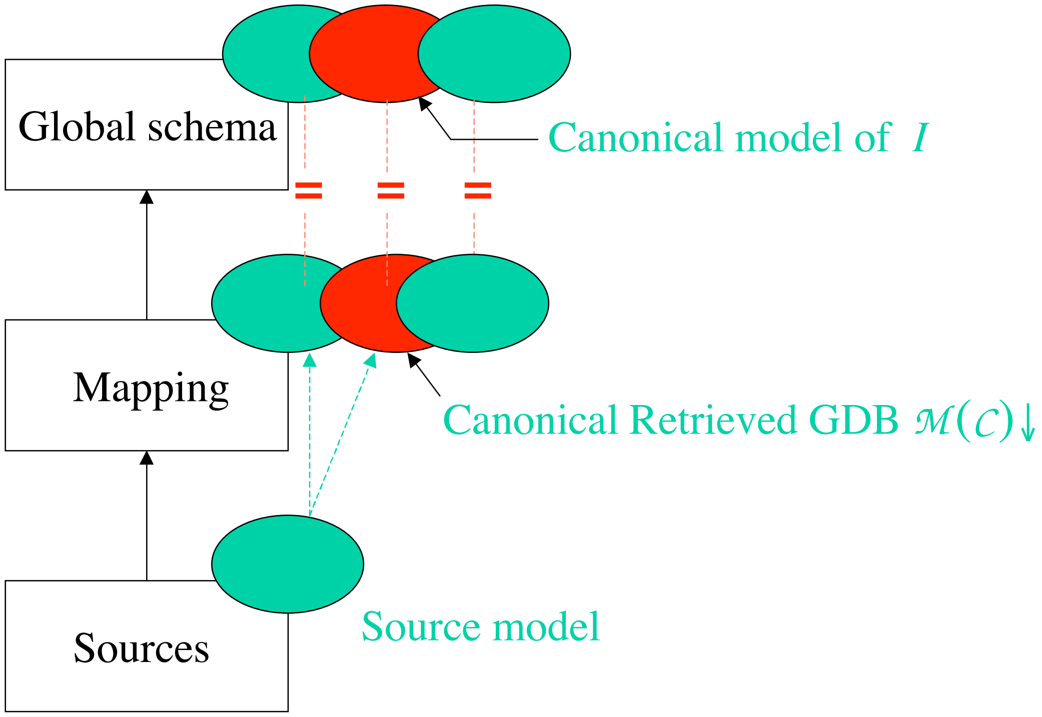
Scode	Ucode
15	*x*
12	*y*

s_1^{C}

15	*bill*	*oslo*	24
12	*anne*	*florence*	21

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

(G)LAV: canonical model



Global schema

Mapping

Sources

Canonical model of I

Canonical Retrieved GDB $\mathcal{M}(\mathcal{C})\!\downarrow$

Source 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. If $\mathcal{C}$ is a source database, 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 $\vec{\mathbf{t}} \in cert(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{\mathbf{t}} \in q^{\mathcal{M}(\mathcal{C})\downarrow}$
- complexity of query answering is **polynomial**

(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: data complexity

From [Abiteboul&Duschka PODS'98]:

Sound sources	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: intractability for positive views

From [van der Meyden TCS'03]
given a graph $G = (N, E)$, we define $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and source database $\mathcal{C}$:

$$\begin{array}{ll} \mathcal{G}: & \{edge(x,y), colour(x,c)\} \\ \mathcal{S}: & \{V_e(x,y), V_c(x)\} \\ \mathcal{M}: & V_E(x,y) \rightsquigarrow edge(x,y) \\ & V_N(x) \rightsquigarrow colour(x, RED) \vee colour(x, BLUE) \vee colour(x, GREEN) \\ \mathcal{C}: & {V_E}^{\mathcal{C}} = \{(a,b),(b,a) \mid (a,b) \in E\} \\ & {V_N}^{\mathcal{C}} = \{(a \mid a \in N\} \end{array}$$

Now consider the query q : $\exists x, y, c. colour(x,c) \wedge edge(x,y) \wedge colour(y,c)$
that describes mismatched edge pairs.

- If G is not 3-colorable, then $cert(q, \mathcal{I}, \mathcal{C}) = true$; otherwise $cert(q, \mathcal{I}, \mathcal{C}) = false$.

$\Longrightarrow$ **coNP-hard data complexity** for positive views and conjunctive queries.

(G)LAV: in coNP for positive views and queries

In the case of positive views and queries:

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

$$\forall \vec{\mathbf{x}}\, (\phi_{\mathcal{S}}(\vec{\mathbf{x}}) \;\rightarrow\; \exists \vec{\mathbf{y_1}} \alpha_1(\vec{\mathbf{x}}, \vec{\mathbf{y_1}}) \;\vee\; \ldots \;\vee\; \exists \vec{\mathbf{y_h}} \alpha_h(\vec{\mathbf{x}}, \vec{\mathbf{y_h}}))$$

  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 conjuncts in $\mathcal{M}$
- If $\mathcal{C}$ has n tuples, then there is a database $\mathcal{B}' \subseteq \mathcal{B}$ for $\mathcal{I}$ that satisfies $\mathcal{M}$ wrt $\mathcal{C}$ with at most $n \cdot k$ tuples. Since q is monotone, $\vec{\mathbf{t}} \notin q^{\mathcal{B}'}$
- Checking whether $\mathcal{B}'$ satisfies $\mathcal{M}$ wrt $\mathcal{C}$, and checking whether $\vec{\mathbf{t}} \notin q^{\mathcal{B}'}$ can be done in PTIME wrt the size of $\mathcal{B}'$

$\Longrightarrow$ **coNP data complexity** for positive views and queries.

(G)LAV: conjunctive user queries with inequalities

Consider the following $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and the following query q (from [Fagin&al. ICDT'03]):

$$\begin{array}{lcl} \mathcal{M} & : & s(X,Y) \rightsquigarrow \{ \, (X,Y) \mid T(X,Z) \wedge T(Z,Y) \, \} \\ \mathcal{C} & : & \{ \, s(a,a) \, \} \\ q & : & \{ \, (\,) \mid T(X,Y) \; \wedge \; X \neq Y) \, \} \end{array}$$

- $J_1 = \{T(a,a)\}$ is a solution, and $q^{J_1} = false$
- if J is a universal solution, then both $T(a,X)$ and $T(X,a)$ are in J, with $X \neq a$ (otherwise $T(a,a)$ would be true in every solution)

$\Longrightarrow$ $cert(q, \mathcal{I}, \mathcal{C}) = false$, but $q^J = true$ for every universal solution J for $\mathcal{I}$ relative to $\mathcal{C}$

$\Longrightarrow$ the notion of universal solution is not the right tool

(G)LAV: conjunctive user queries with inequalities

- still polynomial with one inequalities
- coNP algorithm: guess equalities on variables in the canonical retrieved global database
- coNP-hard with six inequalities (see [Abiteboul&Duschka PODS'98])
- open problem for a number of inequalities between two and five

$\Longrightarrow$ **coNP-complete** for conjunctive user queries with inequalities.

Approaches to query answering in (G)LAV systems

- Exploit connection with query containment
- Direct methods (aka view-based query answering)
- By (view-based) query rewriting

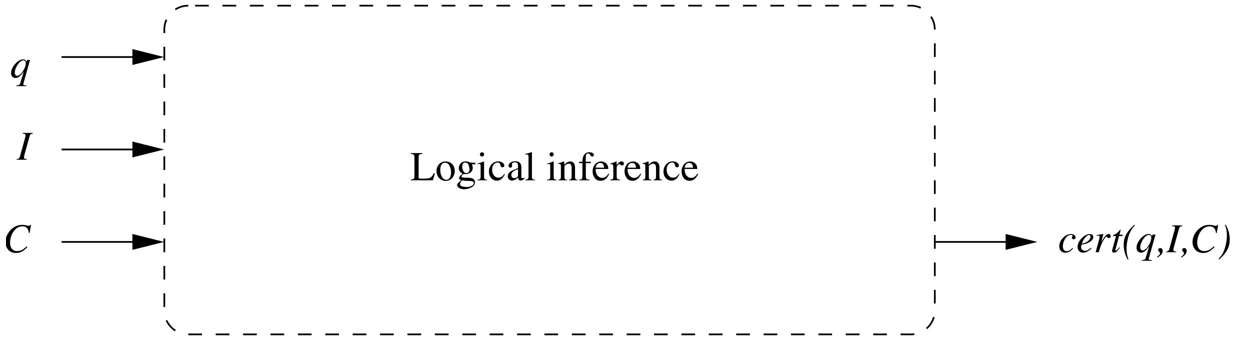
In (G)LAV data integration the views are the sources

(G)LAV: view-based query rewriting

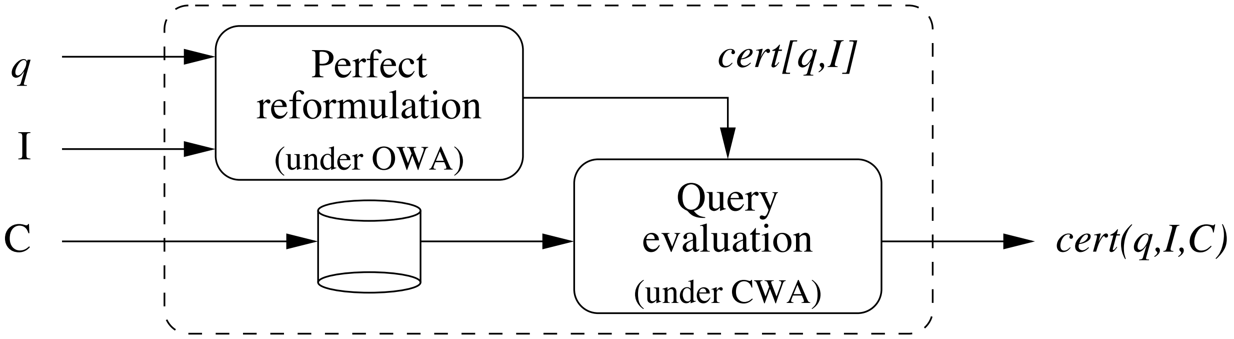
View-based query rewriting: query answering is divided in two steps

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 $\mathcal{C}$

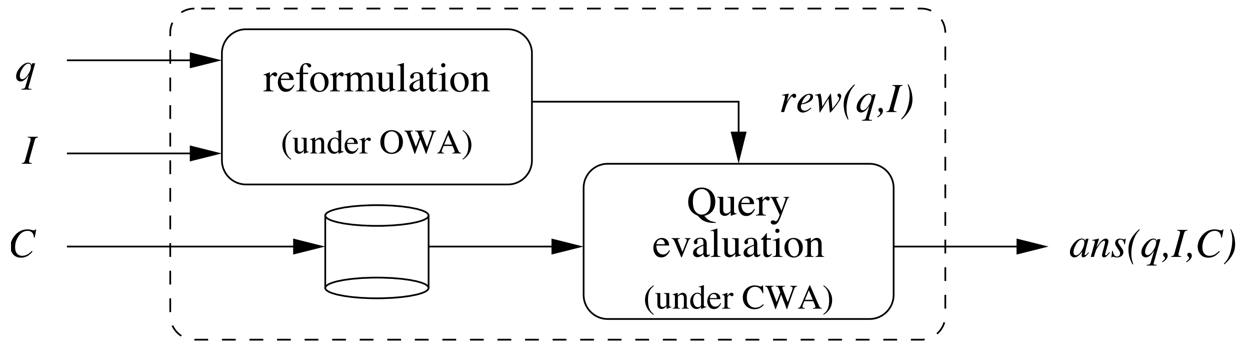
Query answering



Query answering: reformulation+evaluation


q
I
C
Perfect reformulation (under OWA)
cert[q,I]
Query evaluation (under CWA)
cert(q,I,C)

Query rewriting



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

(G)LAV: connection to rewriting

Query answering by rewriting:

- Given $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and given a query q over $\mathcal{G}$, rewrite q into a query, called $rew(q, \mathcal{I})$, in the alphabet $\mathcal{A}_\mathcal{S}$ of the sources
- Evaluate the rewriting $rew(q, \mathcal{I})$ over the source database

We are interested in sound rewritings (i.e., computing only tuples in $cert(q, \mathcal{I}, \mathcal{C})$ for every source database $\mathcal{C}$) that are expressed in a given query language, and that are maximal for the class of queries expressible in such language.

Sometimes, we are interested in exact rewritings, i.e., rewritings that are logically equivalent to the query, modulo $\mathcal{M}$ (observe that such rewritings may not exists).

But (see [Calvanese &al. ICDT'05]):

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

Perfect 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 $cert_{[q,\mathcal{I}]}$
- $cert_{[q,\mathcal{I}]}$ is called the **perfect rewriting** of q wrt $\mathcal{I}$

Properties of the perfect rewriting

- Can we express the perfect rewriting in a certain query language?
- How does a maximal rewriting for a given class of queries compare with the perfect rewriting?
  - From a semantical point of view
  - From a computational point of view
- Which is the computational complexity of (finding, evaluating) the perfect rewriting?

The case of conjunctive queries

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

View-based query processing for UPQs

As we saw before, view-based query answering is coNP-complete in data complexity when we add (a very simple form of) union to the query language used to express queries over the global schema in the mapping [Calvanese&al. ICDE'00].

In other words, in this case $cert(q, \mathcal{I}, \mathcal{C})$, with q and $\mathcal{I}$ fixed, is a coNP-complete function, and therefore **the perfect rewriting $cert_{[q,\mathcal{I}]}$ is a coNP-complete query**.

If in the mapping we use a query language with union, then the perfect rewriting is coNP-hard — we do not have the ideal situation we had for conjunctive queries.

(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]
- Conjunctive queries 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], [Calvanese&al. PODS'00], [Deutsch&Tannen DBPL'01], [Calvanese&al. DBPL'01],
- Relationship between view-based rewriting and answering [Calvanese&al. LICS'00], [Calvanese&al. PODS'03], [Calvanese&al. ICDT'05]