

Logic-based information integration

Giuseppe De Giacomo & Riccardo Rosati

**Dipartimento di Informatica e Sistemistica “Antonio Ruberti”
Università di Roma “La Sapienza”**

Joint work with

*Andrea Cali, Diego Calvanese, Maurizio Lenzerini, Domenico Lembo,
Moshe Y. Vardi (Rice Univ.), Guido Vetere (IBM Italia)*

Course at ESSLLI'05

Edinburgh, UK – August 2005

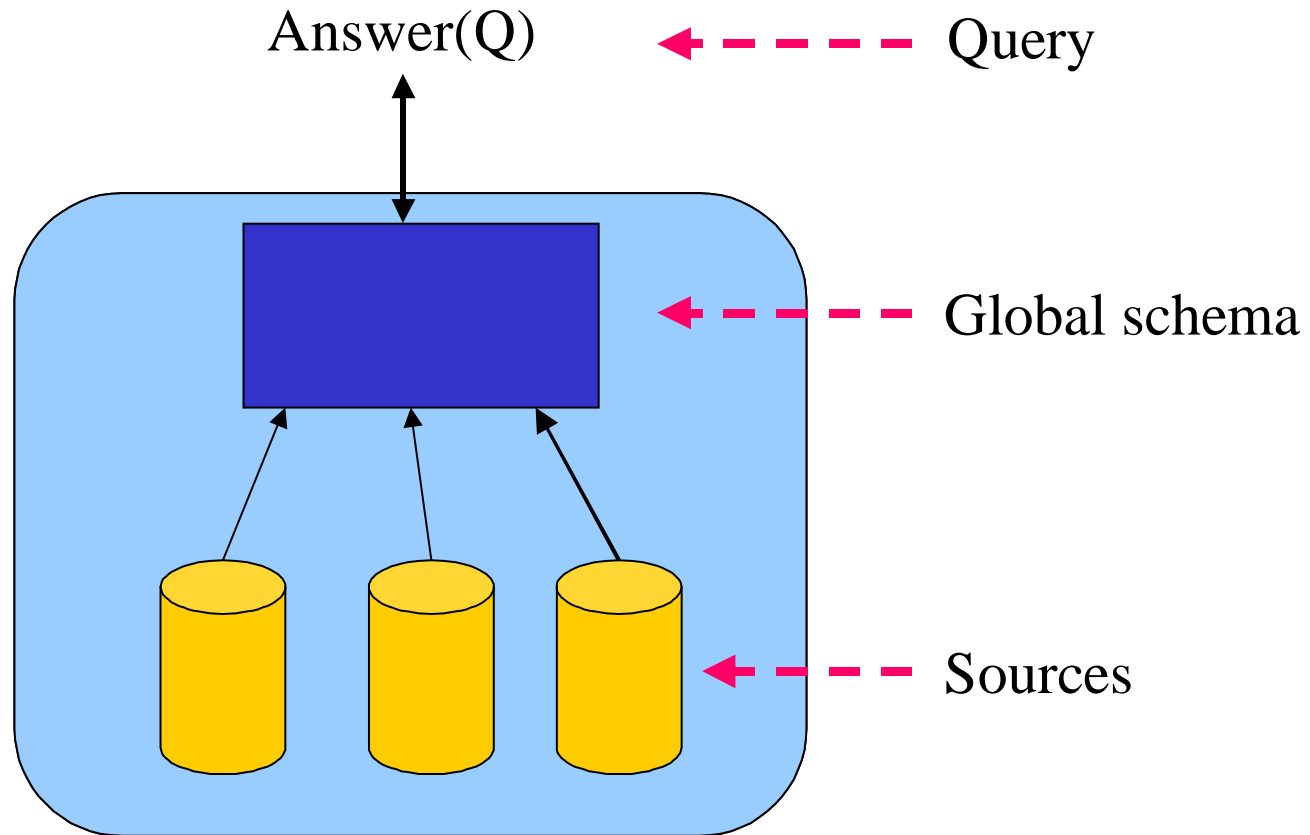
Logic-based information integration: overview

- Introduction to information (data) integration – *De Giacomo*
- Query answering in GAV and LAV information integration systems – *De Giacomo*
- Information integration under constraints: basic techniques – *Rosati*
- Information integration under constraints: results – *Rosati*
- Inconsistency tolerance in information integration – *Rosati*
(see also *Bertossi's ESSLLI'05 course*)

Lecture 2: outline

- Query answering in information integration
- Query answering in GAV information integration systems
- Query answering in LAV information integration systems

Information integration



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** (or, unions thereof) as both user queries and queries in the mapping.
- A conjunctive query has the form

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

- *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

Lecture 2: outline

- Query answering in information integration
- Query answering in GAV information integration systems
- Query answering in LAV information integration systems

Incompleteness and inconsistency

Constraints in \mathcal{G}	Type of mapping	Incompleteness	Inconsistency
<i>no</i>	<i>GAV</i>	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} :

student(*code*, *name*, *city*)

university(*code*, *name*)

enrolled(*Scode*, *Ucode*)

Source schema \mathcal{S} : relations $s_1(X, Y, W, Z)$, $s_2(X, Y)$, $s_3(X, Y)$

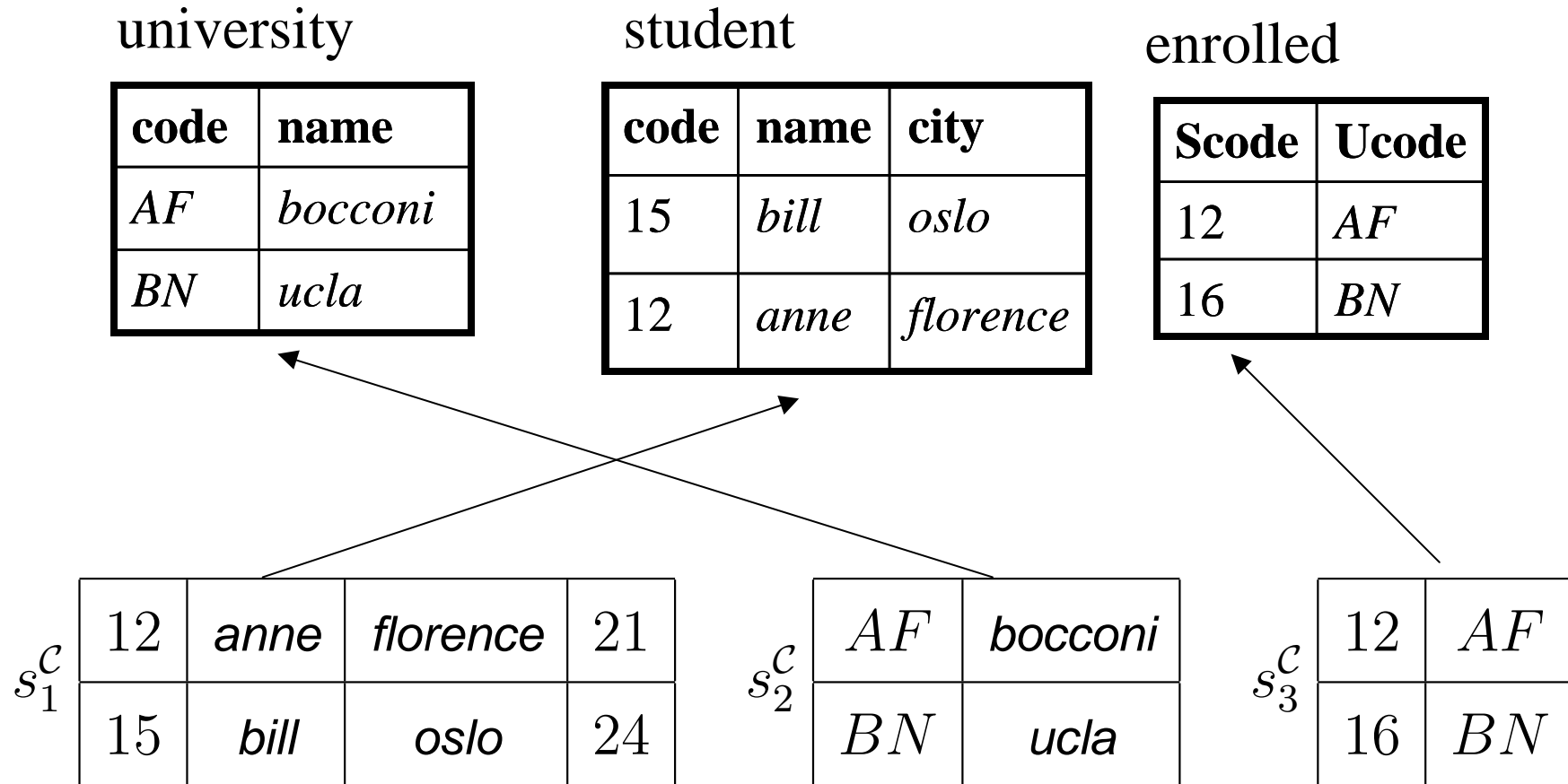
Mapping \mathcal{M} :

student(X, Y, Z) \rightsquigarrow $\{ (X, Y, Z) \mid s_1(X, Y, Z, W) \}$

university(X, Y) \rightsquigarrow $\{ (X, Y) \mid s_2(X, Y) \}$

enrolled(X, W) \rightsquigarrow $\{ (X, W) \mid s_3(X, W) \}$

GAV: example



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

GAV: minimal model

GAV mapping assertions $g \rightsquigarrow \phi_S$ have the logical form:

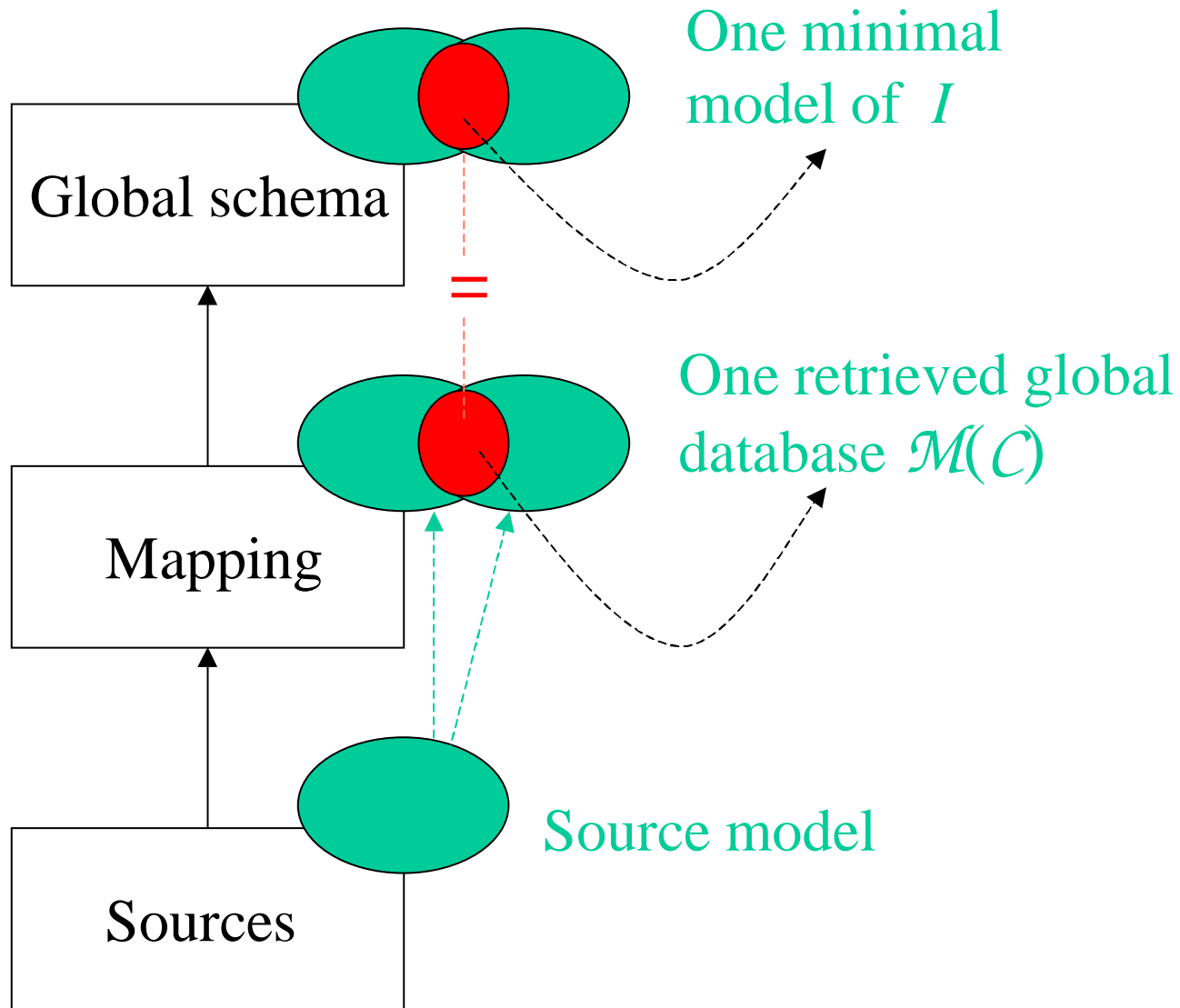
$$\forall \vec{x} \phi_S(\vec{x}) \rightarrow g(\vec{x})$$

where ϕ_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 $\vec{t} \in \text{cert}(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{t} \in q^{\mathcal{M}(\mathcal{C})}$
- If q is query over \mathcal{G} , then the unfolding of q wrt \mathcal{M} , $\text{unf}_{\mathcal{M}}(q)$, 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 $\text{unf}_{\mathcal{M}}(q)$ over \mathcal{C} . It follows that, if q is a conjunctive query, then $\vec{t} \in \text{cert}(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{t} \in \text{unf}_{\mathcal{M}}(q)^{\mathcal{C}}$

Unfolding is therefore sufficient

- **Data complexity** of query answering is **polynomial** (actually **LOGSPACE**): the query $\text{unf}_{\mathcal{M}}(q)$ 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
<i>AF</i>	<i>bocconi</i>
<i>BN</i>	<i>ucla</i>

student

code	name	city
15	<i>bill</i>	<i>oslo</i>
12	<i>anne</i>	<i>florence</i>

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

unfolding

s_1^C

12	<i>anne</i>	<i>florence</i>	21
15	<i>bill</i>	<i>oslo</i>	24

s_2^C

<i>AF</i>	<i>bocconi</i>
<i>BN</i>	<i>ucla</i>

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

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 $\Gamma \cup \text{Var}$.

- A **homomorphism** $h : B_1 \rightarrow B_2$ is a mapping from $(\Gamma \cup \text{Var}(B_1))$ to $(\Gamma \cup \text{Var}(B_2))$ 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 $t = (a_1, \dots, a_n)$, then $h(t) = (h(a_1), \dots, h(a_n))$)
- B_1 is **homomorphically equivalent** to B_2 if there is a homomorphism $h : B_1 \rightarrow B_2$ and a homomorphism $h' : B_2 \rightarrow B_1$

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 $h : J \rightarrow J'$ (see [Fagin&al. ICDT'03]).

GAV: another view

- **Homomorphism preserves satisfaction of conjunctive queries**: if there exists a homomorphism $h : J \rightarrow J'$, and q is a conjunctive query, then $\vec{t} \in q^J$ implies $\vec{t} \in q^{J'}$
- 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 $\vec{t} \in cert(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{t} \in q^{\mathcal{M}(\mathcal{C})}$
 - complexity of query answering is polynomial

Lecture 2: outline

- Query answering in information integration
- Query answering in GAV information integration systems
- Query answering in LAV information integration systems

Incompleteness and inconsistency

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

(G)LAV: example

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

Global schema \mathcal{G} :

student(*code*, *name*, *city*)

enrolled(*Scode*, *Ucode*)

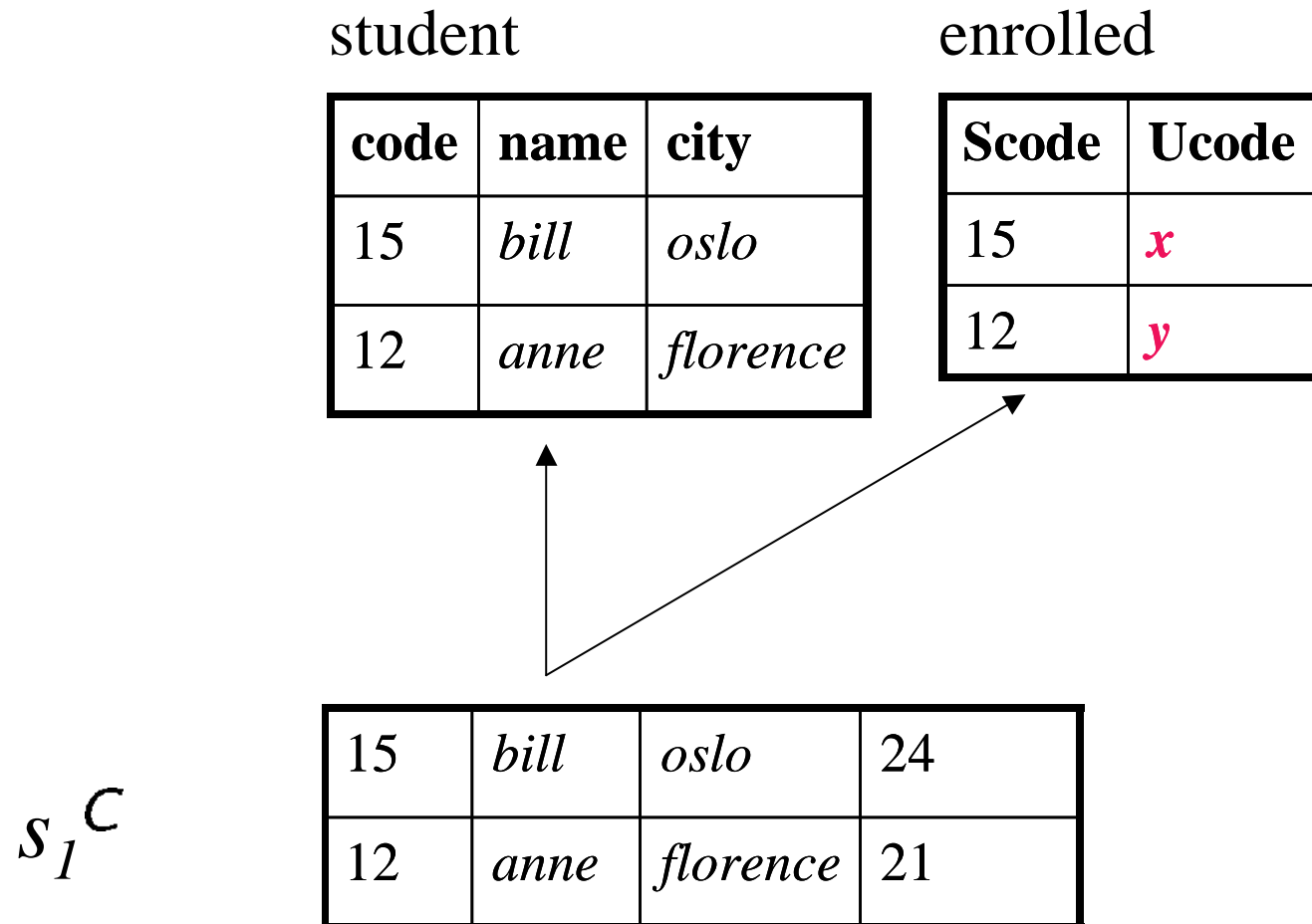
Source schema \mathcal{S} : relation $s_1(X, Y, W, Z)$

Mapping \mathcal{M} :

$$\{ (X, Y, Z) \mid s_1(X, Y, Z, W) \} \rightsquigarrow \{ (X, Y, Z) \mid \text{student}(X, Y, Z) \wedge \text{enrolled}(X, W) \}$$

(G)LAV: example

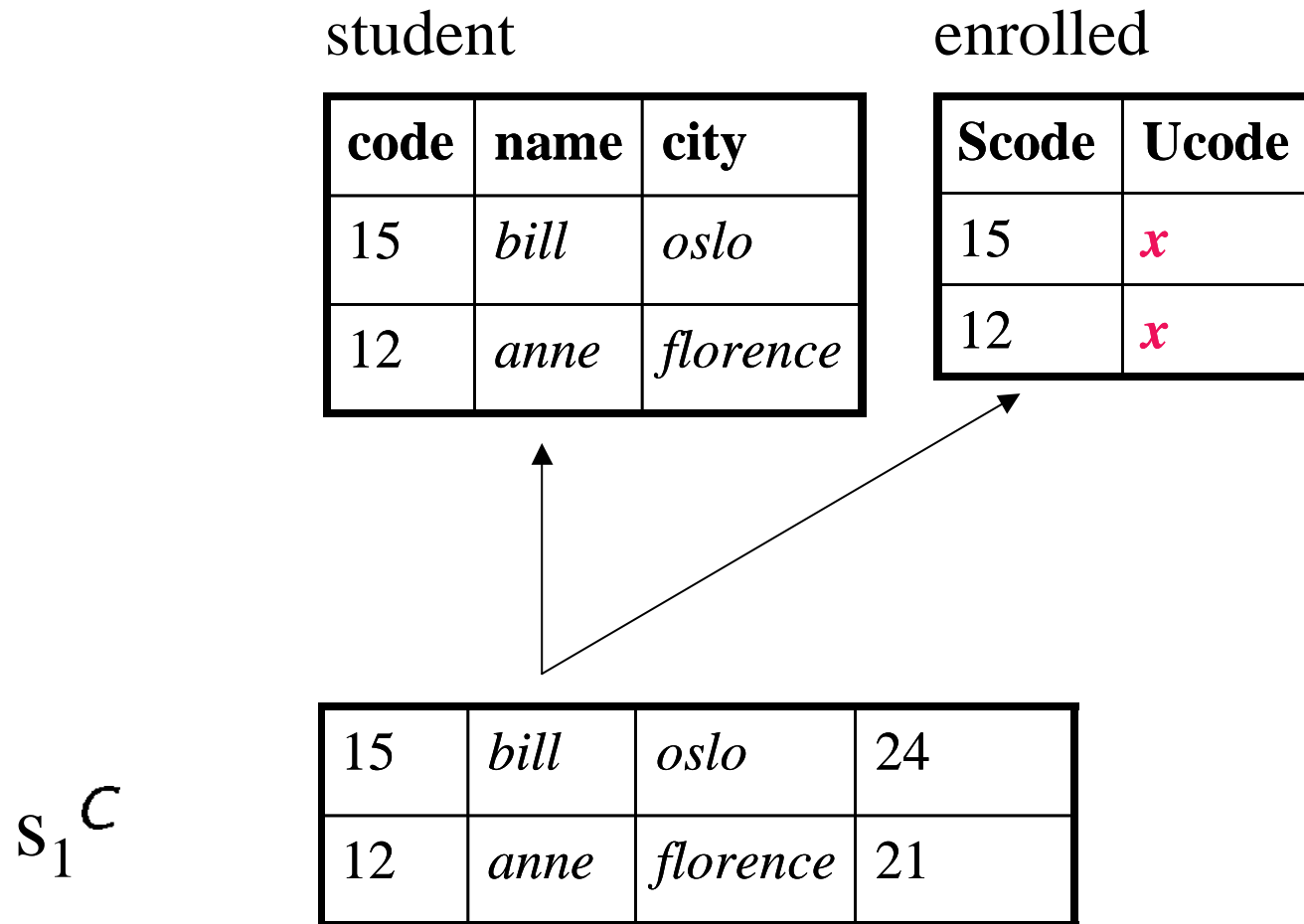
$$\{ (X, Y, Z) \mid s_1(X, Y, Z, W) \} \rightsquigarrow \{ (X, Y, Z) \mid \text{student}(X, Y, Z) \wedge \text{enrolled}(X, W) \}$$



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

(G)LAV: example

$$\{ (X, Y, Z) \mid s_1(X, Y, Z, W) \} \rightsquigarrow \{ (X, Y, Z) \mid \text{student}(X, Y, Z) \wedge \text{enrolled}(X, W) \}$$



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

(G)LAV: incompleteness

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

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

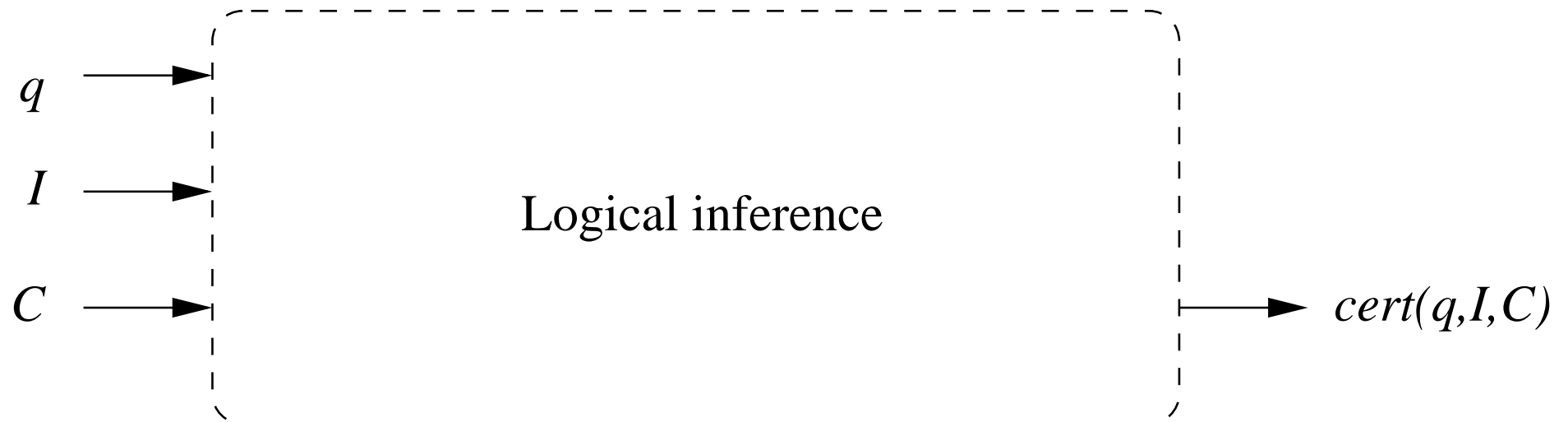
where ϕ_S and ϕ_G are conjunctions of atoms.

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 ϕ_G :

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

Query answering is based on logical inference



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{x} is in $s^{\mathcal{C}}$, then the corresponding literal is $s(\vec{x})$)
- 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 $\vec{t} \in \text{cert}(q, \mathcal{I}, \mathcal{C})$ under sound sources can be reduced to the problem of checking whether $q_{\mathcal{C}}$ **is contained in $q_{\vec{t}}$ 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]:

$$r_1(T) \quad \rightsquigarrow \quad \{ (T) \mid \text{movie}(T, Y, D) \wedge \text{european}(D) \}$$

$$r_2(T, V) \quad \rightsquigarrow \quad \{ (T, V) \mid \text{movie}(T, Y, D) \wedge \text{review}(T, V) \}$$

$$\forall T \ r_1(T) \quad \rightarrow \quad \exists Y \exists D \ \text{movie}(T, Y, D) \wedge \text{european}(D)$$

$$\forall T \ \forall V \ r_2(T, V) \quad \rightarrow \quad \exists Y \exists D \ \text{movie}(T, Y, D) \wedge \text{review}(T, V)$$

$$\text{movie}(T, f_1(T), f_2(T)) \quad \leftarrow \quad r_1(T)$$

$$\text{european}(f_2(T)) \quad \leftarrow \quad r_1(T)$$

$$\text{movie}(T, f_4(T, V), f_5(T, V)) \quad \leftarrow \quad r_2(T, V)$$

$$\text{review}(T, V) \quad \leftarrow \quad r_2(T, V)$$

- 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?

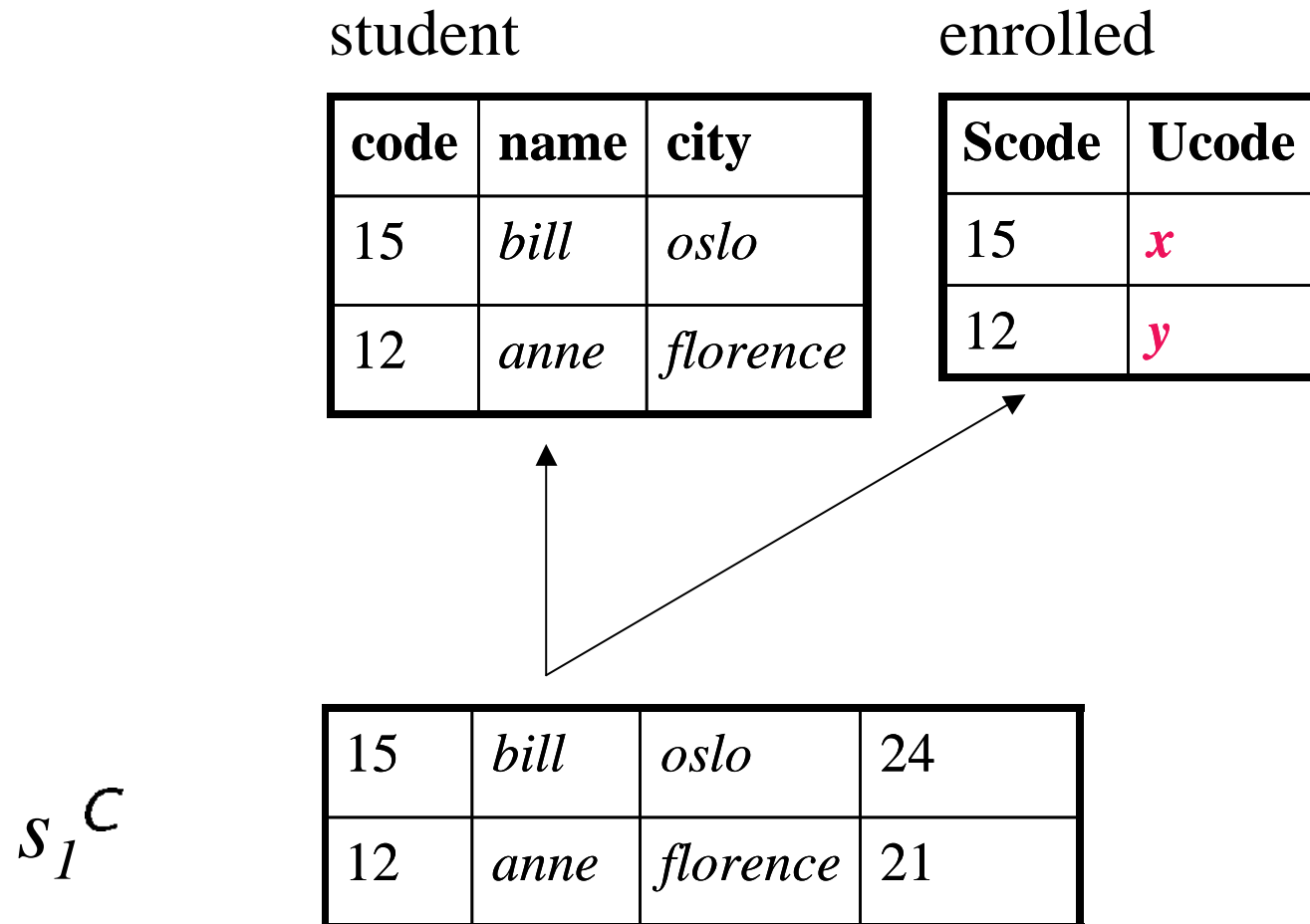
We build what we call the **canonical retrieved global database** for \mathcal{I} relative to \mathcal{C} , denoted $\mathcal{M}(\mathcal{C})\downarrow$, as follows:

- let all predicates be empty in $\mathcal{M}(\mathcal{C})\downarrow$
- for each mapping assertion $\phi_S \rightsquigarrow \phi_G$ in \mathcal{M}
 - for each tuple $\vec{t} \in \phi_S^{\mathcal{C}}$ such that $\vec{t} \notin \phi_G^{\mathcal{M}(\mathcal{C})\downarrow}$, add \vec{t} to $\phi_G^{\mathcal{M}(\mathcal{C})\downarrow}$ by inventing fresh variables (Skolem terms) in order to satisfy the existentially quantified variables in ϕ_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}** .

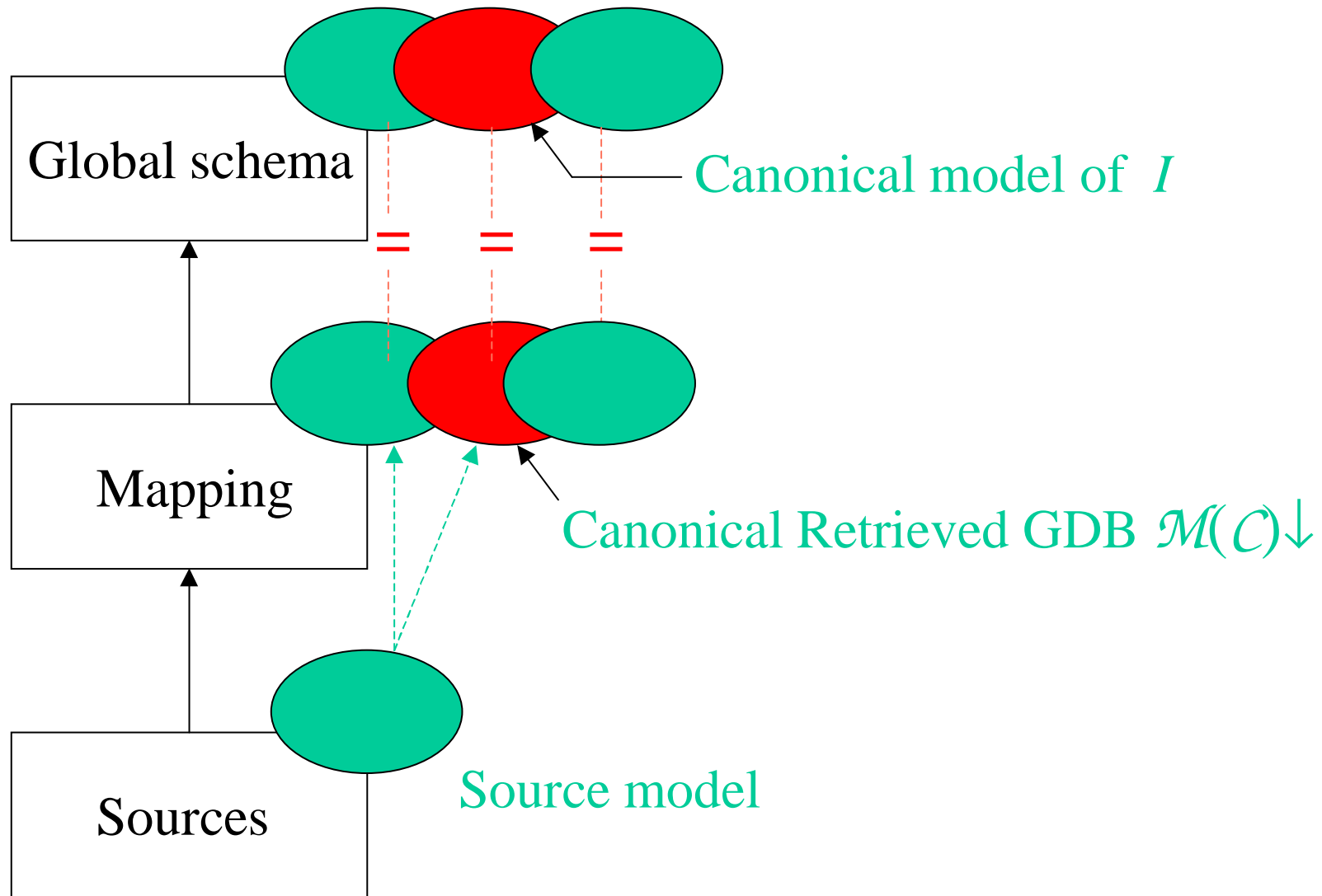
(G)LAV: example of canonical model

$$\{ (X, Y, Z) \mid s_1(X, Y, Z, W) \} \rightsquigarrow \{ (X, Y, Z) \mid \text{student}(X, Y, Z) \wedge \text{enrolled}(X, W) \}$$



Example of source database \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. 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 $\vec{t} \in \text{cert}(q, \mathcal{I}, \mathcal{C})$ if and only if $\vec{t} \in q^{\mathcal{M}(\mathcal{C}) \downarrow}$
- 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 [≠]	PQ	Datalog	FOL
CQ	PTIME	coNP	PTIME	PTIME	undec.
CQ [≠]	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 queries and views

From [Calvanese&al. ICDE'00], given a graph $G = (N, E)$, we define $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and source database \mathcal{C} , with $\mathcal{S} = \{V_b, V_f, V_e\}$, and $\mathcal{G} = \{R_b, R_f, R_{rg}, R_{gr}, R_{rb}, R_{br}, R_{gb}, R_{bg}\}$

\mathcal{M} :

$$V_b \rightsquigarrow R_b$$

$$V_f \rightsquigarrow R_f$$

$$V_e \rightsquigarrow R_{rg} \vee R_{gr} \vee R_{rb} \vee R_{br} \vee R_{gb} \vee R_{bg}$$

\mathcal{C} :

$$V_b^{\mathcal{C}} = \{(c, a) \mid a \in N, c \notin N\}$$

$$V_f^{\mathcal{C}} = \{(a, d) \mid a \in N, d \notin N\}$$

$$V_e^{\mathcal{C}} = \{(a, b), (b, a) \mid (a, b) \in E\}$$

Query q : $\{(X, Z) \mid R_b(X, Y) \wedge M(Y, W) \wedge R_f(W, Z)\}$

where M describes all mismatched edge pairs (e.g., $\{(X, Z) \mid R_{rg}(X, Y) \wedge R_{rb}(Y, Z)\}$).

- If G is 3-colorable, then $\exists \mathcal{B}$ where M (and q) is empty, i.e. $(c, d) \notin \text{cert}(q, \mathcal{I}, \mathcal{C})$
- If G is not 3-colorable, then M is nonempty $\forall \mathcal{B}$, i.e. $(c, d) \in \text{cert}(q, \mathcal{I}, \mathcal{C})$

\implies **coNP-hard data complexity** for positive queries and positive views.

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

In the case of positive queries and positive views:

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

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

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, $\vec{t} \notin q^{\mathcal{B}'}$
- 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}'

\implies **coNP data complexity** for positive queries and positive views.

(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]):

$$\mathcal{M} : s(X, Y) \rightsquigarrow \{ (X, Y) \mid T(X, Z) \wedge T(Z, Y) \}$$

$$\mathcal{C} : \{ s(a, a) \}$$

$$q : \{ () \mid T(X, Y) \wedge X \neq Y \}$$

- $J_1 = \{T(a, a)\}$ 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)

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

\Rightarrow 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

⇒ **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

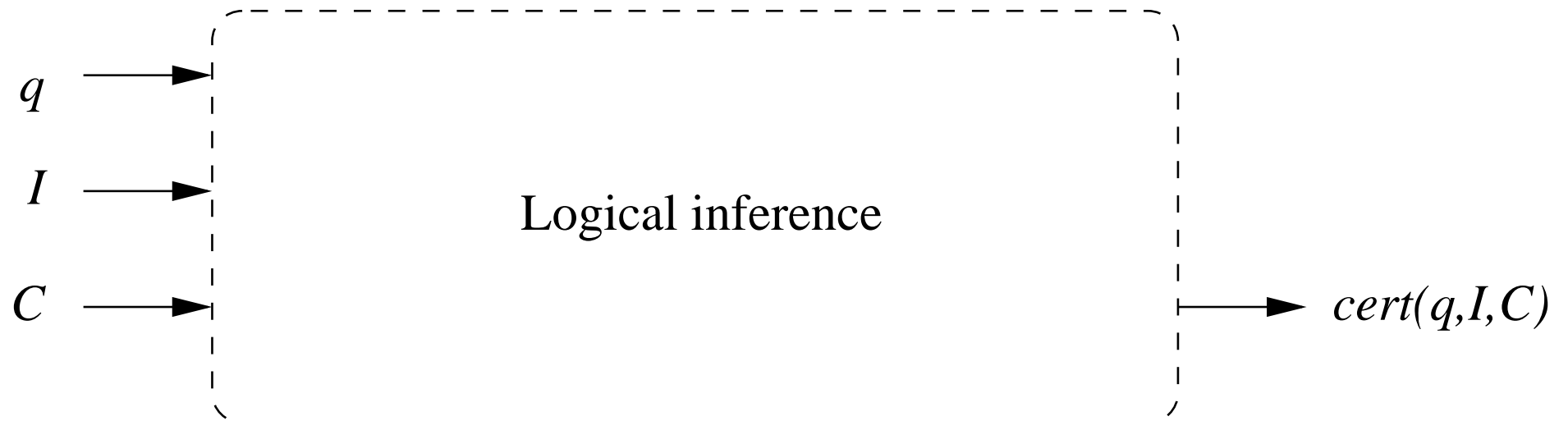
*In (G)LAV data integration the **views are the sources***

(G)LAV: view-based query rewriting

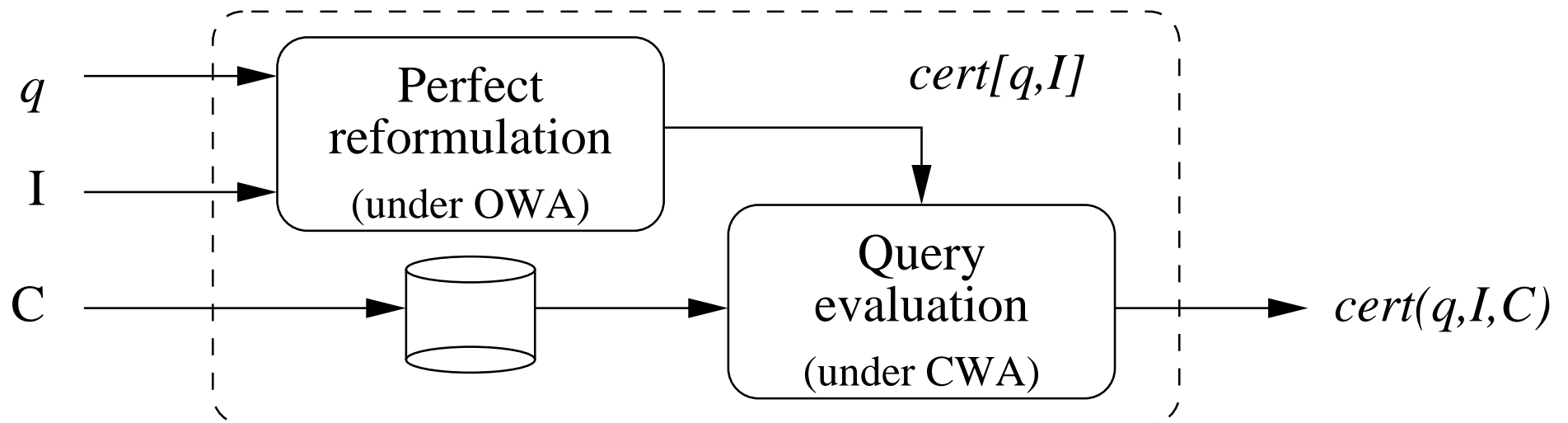
View-based query rewriting: query answering is divided in two steps

1. re-express the query in terms of a **given query language** over the alphabet of \mathcal{A}_S
2. evaluate the rewriting over the source database \mathcal{C}

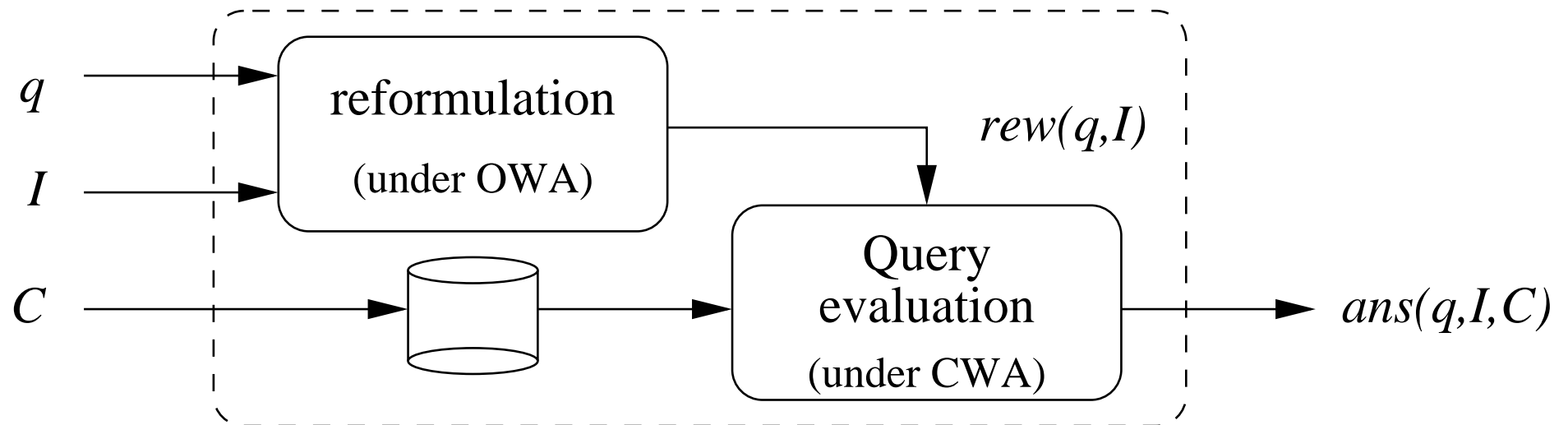
Query answering



Query answering: reformulation+evaluation



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 exist).

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

- *When does the rewriting compute **all** certain answers?*
- *What do we gain or lose 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]