Query Processing in Data Integration Systems

Diego Calvanese

Free University of Bozen-Bolzano

BIT PhD Summer School – Bressanone
July 3–7, 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
4. Concluding remarks

Part I

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

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 $\mathcal{M}$:

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

GAV – Example of retrieved global database

university

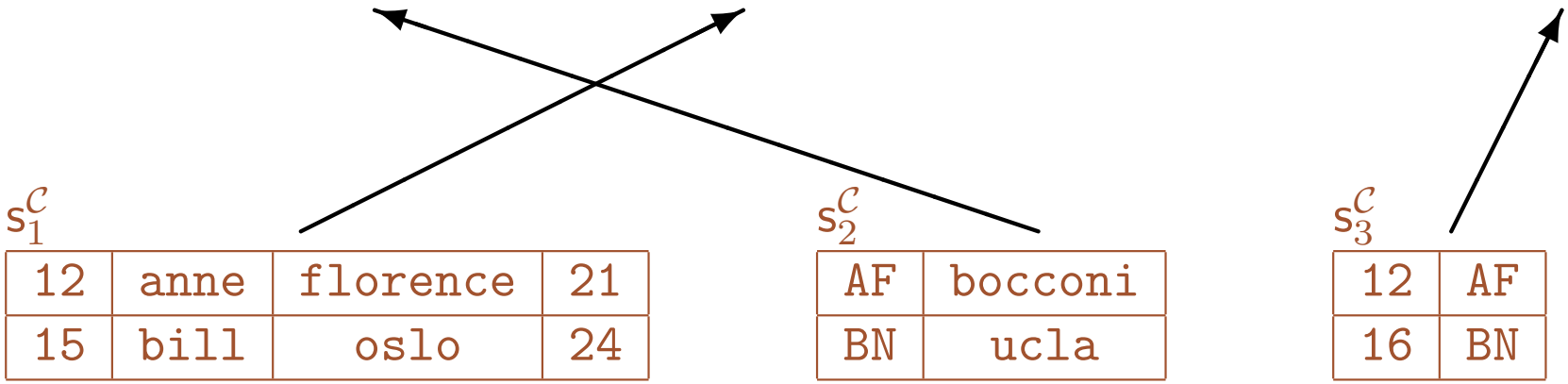
Code	*Name*
AF	bocconi
BN	ucla

student

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

enrolled

Scode	*Ucode*
12	AF
16	BN



$s_1^{\mathcal{C}}$

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

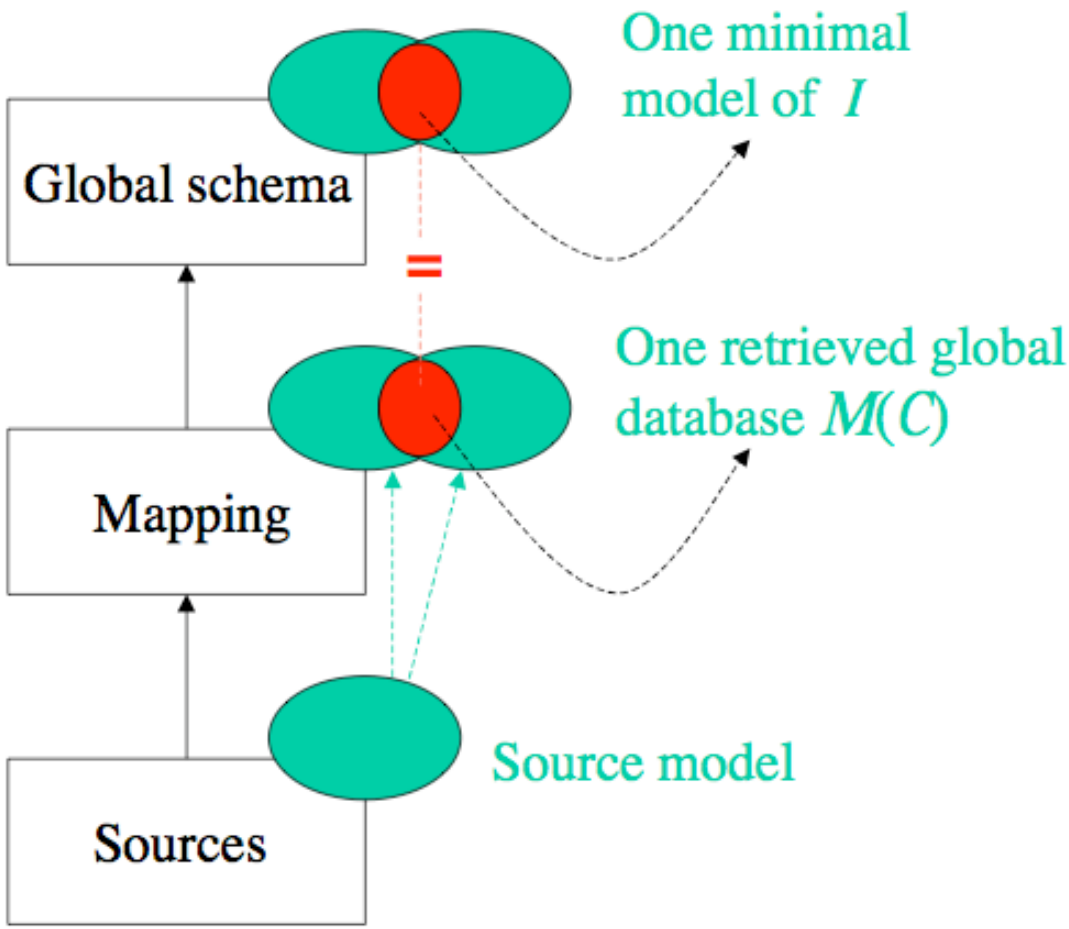
$$\forall \vec{x}.\ \phi_{\mathcal{S}}(\vec{x}) \rightarrow g(\vec{x})$$

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



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)

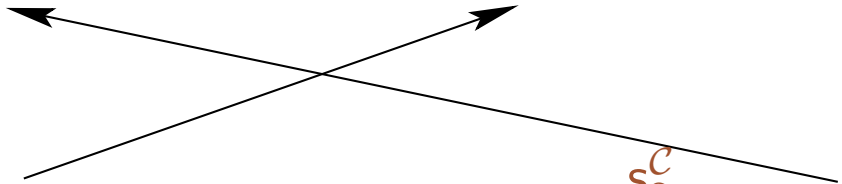
GAV – Example of unfolding

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 – 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]

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

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

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

(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 – Incompleteness

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

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

where ϕ_S and $\phi_{\mathcal{G}}$ 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 $\phi_{\mathcal{G}}$:

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

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

(G)LAV – Approaches to query answering

- Exploit connection with query containment
- Direct methods (aka view-based query answering):
  Try to answer directly the query by means of an algorithm that takes as input the user query q, the specification of $\mathcal{I}$, and the source database $\mathcal{C}$
- 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 $q_1^{\mathcal{D}}$ 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.

(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

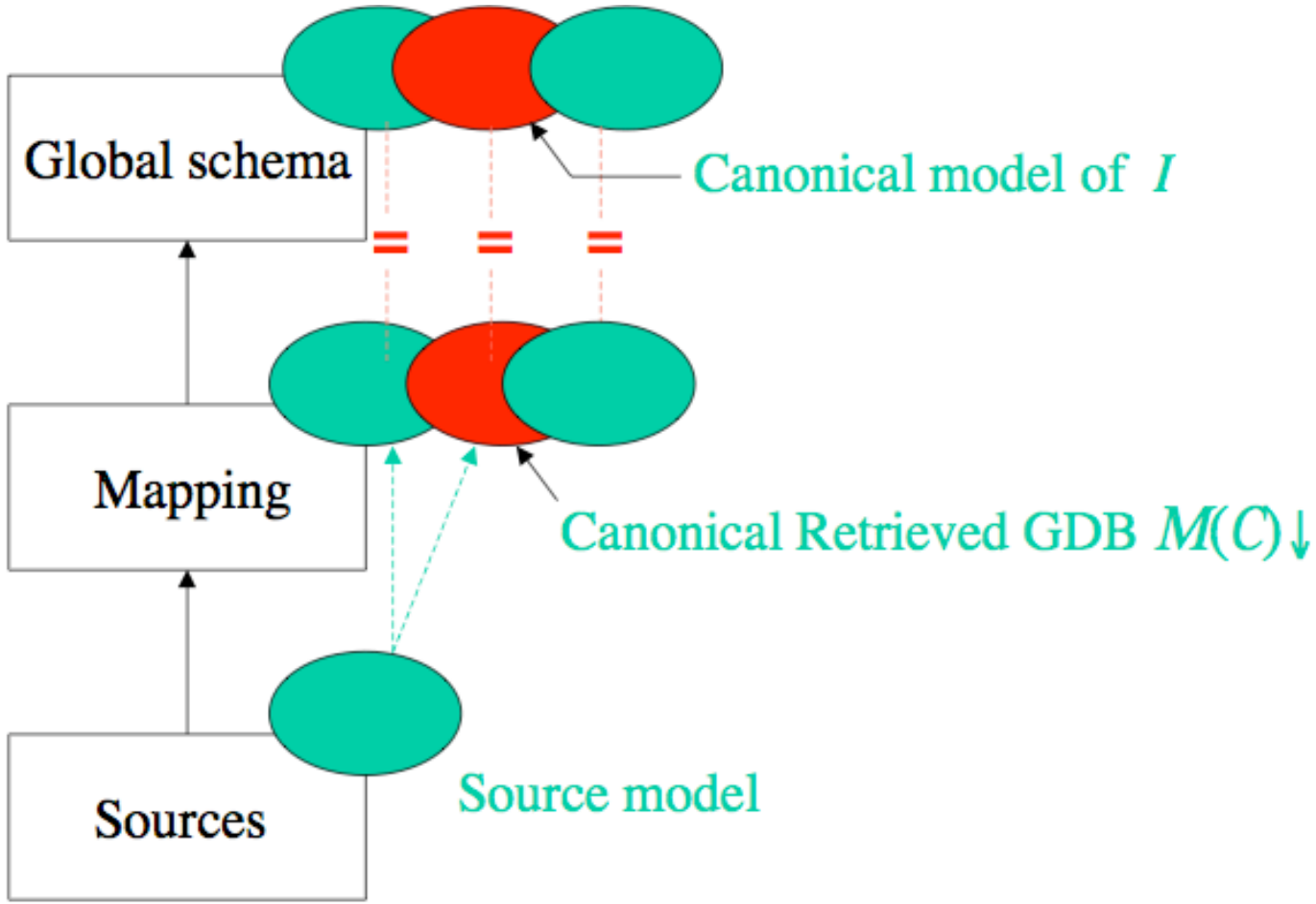
Scode	*Ucode*
12	x
15	y

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

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 – 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 – 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

- coNP algorithm: guess equalities on variables in the canonical retrieved global database
- coNP-hard already for a conjunctive user query with one inequality (and conjunctive view definitions) [Abiteboul & Duschka PODS'98]

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)

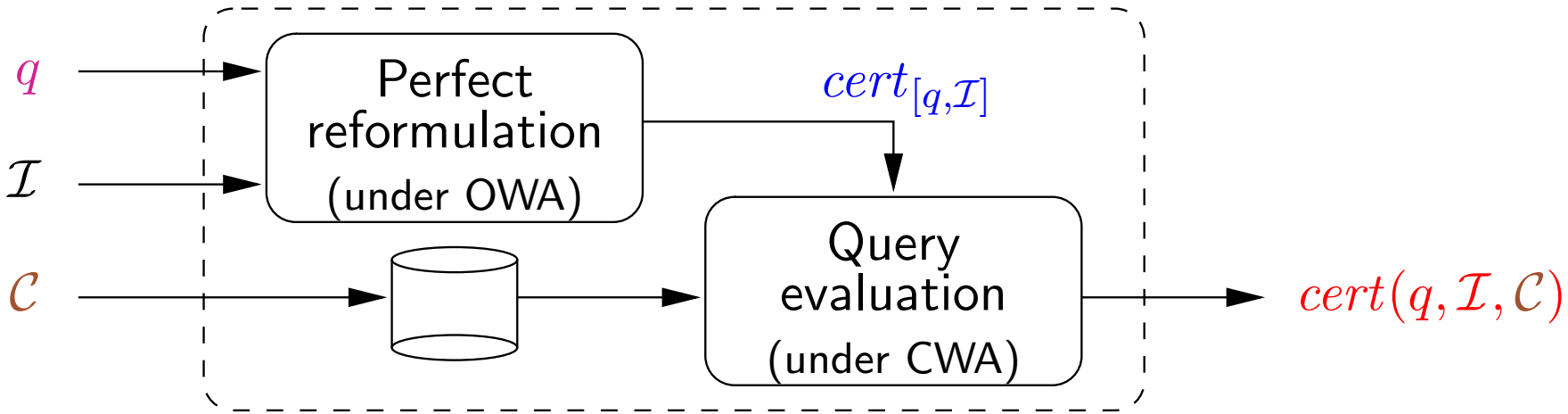
(G)LAV – View-based query rewriting

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 $\mathcal{C}$

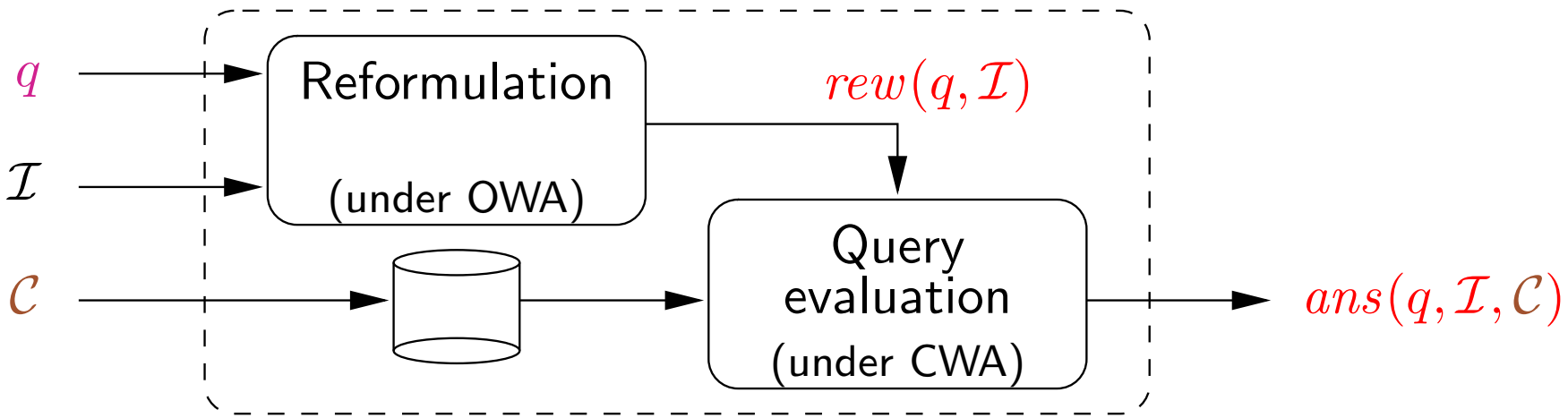
Query answering

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 – 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? [— & 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 – 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) [— & 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 [— & al. ICDE'00, PODS'00, DBPL'01; Deutsch & Tannen DBPL'01],
- Relationship between view-based rewriting and answering [— & al. LICS'00, PODS'03, ICDT'05]