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 presence of constraints

Outline

1. Query answering in GAV without constraints
   - Retrieved global database
   - Query answering via unfolding
   - Universal solutions

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

Outline

Global integrity constraints

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

Integrity constraints for relational schemas

Most important types of ICs for the relational model:

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

Inclusion dependencies (IDs)

An inclusion dependency (ID) states that the presence of a tuple $\vec{t_1}$ in a relation implies the presence of a tuple $\vec{t_2}$ in another relation, where $\vec{t_2}$ contains a projection of the values contained in $\vec{t_1}$

Syntax of inclusion dependencies

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

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

Example

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

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

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

Key dependencies (KDs)

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

Syntax of key dependencies

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

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

Example

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

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

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

Exclusion dependencies (EDs)

An exclusion dependency (ED) states that the presence of a tuple $\vec{t_1}$ in a relation implies the absence of a tuple $\vec{t_2}$ in another relation, where $\vec{t_2}$ contains a projection of the values contained in $\vec{t_1}$

Syntax of exclusion dependencies

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

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

Example

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

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

Note: EDs are a special form of denial constraints

Outline

1. Query answering in GAV without constraints
   - Retrieved global database
   - Query answering via unfolding
   - Universal solutions

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

GAV system with integrity constraints

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

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

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

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

Basic observation: Since $\mathcal{G}$ does have constraints, the retrieved global database $\mathcal{M}(\mathcal{C})$ may not be legal for $\mathcal{G}$

Semantics of GAV systems with integrity constraints

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

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

Recall:

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

GAV with constraints – Example

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

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

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

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

Mapping $\mathcal{M}$:

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

GAV with constraints – Example of retrieved global db

university

Code	*Name*
AF	bocconi
BN	ucla

student

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

enrolled

Scode	*Ucode*
12	AF
16	BN

$s_1^{\mathcal{C}}$

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 with constraints – Example of incompleteness

$s_3^{\mathcal{C}}$

12	AF
16	BN

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

Scode	*Ucode*
12	AF
16	BN

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

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

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

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

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

GAV with constraints – Unfolding is not sufficient

Mapping $\mathcal{M}$:

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

$s_1^{\mathcal{C}}$

12	anne	florence	21
15	bill	oslo	24

$s_2^{\mathcal{C}}$

AF	bocconi
BN	ucla

$s_3^{\mathcal{C}}$

12	AF
16	BN

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

Unfolding of q wrt $\mathcal{M}$: $unf_{\mathcal{M}}(q) = \{ (c) \mid s_1(c, n, ci, a) \}$

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

The simple unfolding strategy is not sufficient for GAV with constraints

GAV with constraints – Example of inconsistency

$s_1^{\mathcal{C}}$

12	anne	florence	21
12	bill	oslo	24

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

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

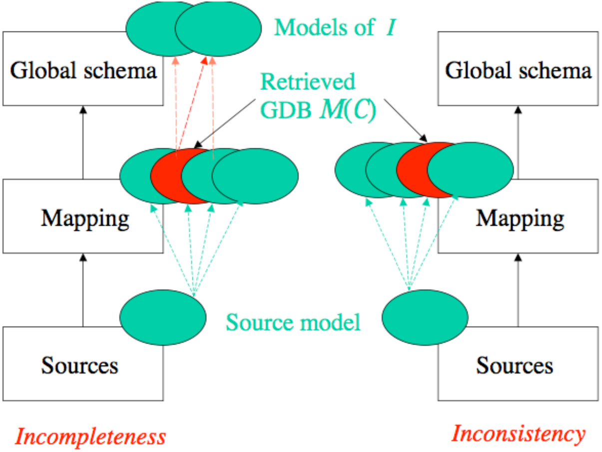
The tuples in $s_1^{\mathcal{C}}$ and the mapping imply $\mathsf{student}^{\mathcal{B}}(12, \mathtt{anne}, \mathtt{florence})$ and $\mathsf{student}^{\mathcal{B}}(12, \mathtt{bill}, \mathtt{oslo})$, for all $\mathcal{B}$ that satisfy the mapping

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

GAV data integration systems with constraints

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

GAV with constraints – Incompleteness and inconsistency



Inclusion dependencies – Example

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

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

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

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

Inclusion dependencies – Example retrieved global db

Source database $\mathcal{C}$:

s_1:

Totti	1971	Roma

s_2:

Juve	Torino	Del Piero

s_3:

Buffon	1978	Juve

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

player:

Totti	1971	Roma
Buffon	1978	Juve

team:

Juve	Torino	Del Piero

Inclusion dependencies – Example retrieved global db

player:

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team:

Juve	Torino	Del Piero

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

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

Warnings

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

Inclusion dependencies – Example retrieved global db

player:

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team:

Juve	Torino	Del Piero

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

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

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

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

Chasing inclusion dependencies – Infinite construction

Intuitive strategy: Add new facts until IDs are satisfied

Problem: Infinite construction in the presence of cyclic IDs

Example

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

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

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

Example

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

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

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

The chase of a database

Definition

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

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

The ID-chase rule

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

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

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

Notice: New existential symbols are introduced (skolem terms)

Properties of the chase

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

Limiting the chase

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

Example

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

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

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

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

Chasing the query

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

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

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

This technique provides an algorithm for rewriting UCQs under IDs

Query rewriting under inclusion dependencies

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

Rewriting rule for inclusion dependencies

Intuition: Use the IDs as basic rewriting rules

Example

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

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

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

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

Definition

Basic rewriting step:

when an atom unifies with the head of the rule

substitute the atom with the body of the rule

Query Rewriting for IDs – Algorithm *ID-rewrite*

Iterative execution of:

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

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

The algorithm *ID-rewrite*

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

Query answering in GAV under IDs

Properties of *ID-rewrite*

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

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

Theorem

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

Theorem

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

Query answering under IDs and KDs

We have already seen that in GAV systems under sound mappings

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

How do KDs interact with IDs?

Theorem

Query answering under IDs and KDs is undecidable

Proof: By reduction from implication of IDs and KDs

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

Non-key-conflicting IDs

Definition

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

Example

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

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

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

Separation for IDs and KDs

Theorem (IDs-KDs separation)

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

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

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

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

Query processing under separable KDs and IDs

Global algorithm:

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

Note:

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

Checking KD consistency – Example

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

Query to check (in)consistency of the KD:

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

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

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

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

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

Query answering in GAV under separable IDs+KDs

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

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

Can we extend these results to more expressive user queries?

- The rewriting technique extends immediately to unions of CQs

$$\textit{ID-rewrite}(q_1 \vee \cdots \vee q_n) = \textit{ID-rewrite}(q_1) \vee \cdots \vee \textit{ID-rewrite}(q_n)$$

- This is not the case for recursive queries

Theorem (— & Rosati KRDB'03)

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

Query answering under IDs and EDs

Under EDs:

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

Under IDs and EDs:

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

Exclusion dependencies – Example

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

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

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

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

Retrieved global db under EDs – Example

Source database $\mathcal{C}$:

s_1:

Totti	1971	Roma

s_2:

Juve	Torino	Del Piero

s_3:

Buffon	1978	Juve

s_4:

Del Piero	Viterbese

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

player :

Totti	1971	Roma
Buffon	1978	Juve

team :

Juve	Torino	Del Piero

coach :

Del Piero	Viterbese

"Repair" of retrieved global db under EDs – Example

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

player :

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

coach :

Del Piero	Viterbese

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

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

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

Deductive closure of EDs under IDs – Example

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

Example

From

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

it follows that

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

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

Deductive closure of EDs under IDs

Definition

Derivation rule of EDs under EDs and IDs:

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

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

Theorem

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

Query answering in GAV under IDs and EDs

Theorem (ID-ED Separation)

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

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

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

The ED consistency check can be done by suitable CQs

Query answering in GAV under IDs, KDs and EDs

Theorem (ID-KD-ED Separation)

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

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

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

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

Note:

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

Theorem (Lembo & Rosati, 2004)

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

Outline

1. Query answering in GAV without constraints
   - Retrieved global database
   - Query answering via unfolding
   - Universal solutions

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

(G)LAV system with integrity constraints

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

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

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

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

Basic observation: Since $\mathcal{G}$ does have constraints, the canonical retrieved global database $\mathcal{M}(\mathcal{C})\downarrow$ may not be legal for $\mathcal{G}$

Semantics of (G)LAV systems with integrity constraints

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

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

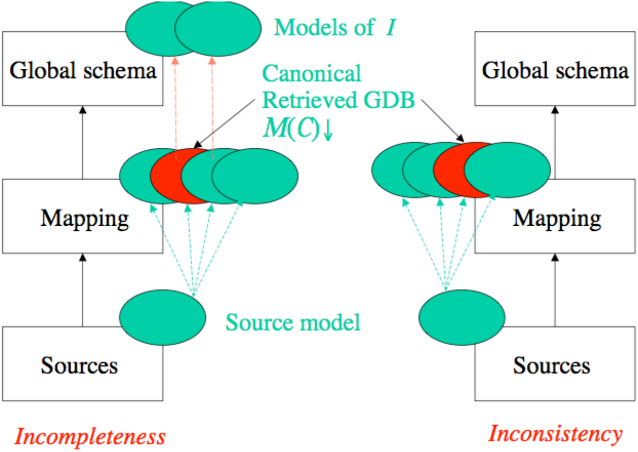
Recall:

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

(G)LAV data integration systems with constraints

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

(G)LAV with constr. – Incompleteness and inconsistency



(G)LAV systems under IDs

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

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

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

Transforming LAV into GAV – Example

Initial LAV mappings:

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

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

Transformed GAV mappings:

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

Additional IDs generated by the transformation:

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

Query answering in (G)LAV systems under IDs

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

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

Theorem

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

(G)LAV systems under IDs and EDs

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

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

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

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

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

Theorem

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

(G)LAV systems under KDs

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

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

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

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

(G)LAV systems under KDs – A negative result

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

Theorem (Duschka & al., 1998)

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

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

Data integration with constraints – First-order rewritability

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

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

Data integration with constraints – Complexity results

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