Query Answering in Data Integration

Piotr Wieczorek

Institute of Computer Science
University of Wrocław

Dagstuhl, November 2010

Outline

Bibliography

1. S. Abiteboul and O.M. Duschka: *Complexity of Answering Queries Using Materialized Views*. In Proc. PODS'98 (Symposium on Principles of Database Systems), pp. 254-263, 1998.
2. O.M. Duschka, M.R. Genesereth, and A.Y. Levy: *Recursive Query Plans for Data Integration.* J. Log. Program. 43(1), pp. 49-73, 2000.
3. R. Pottinger and A.Y. Halevy: *MiniCon: A scalable algorithm for answering queries using views.* VLDB J. 10(2-3), pp. 182-198, 2001.
4. A. Deutsch, B. Ludäscher, and A. Nash: *Rewriting queries using views with access patterns under integrity constraints.* Theor. Comput. Sci. 371(3), pp. 200-226, 2007.

Outline

Quick reminder

Data integration

- global relations (mediated schema)—used in queries
- source relations—store actual data,
- mapping: LAV—each source relation described as a result of a query over the global relations,

Quick reminder

Data integration

- global relations (mediated schema)—used in queries
- source relations—store actual data, view instance I,
- mapping: LAV—each source relation described as a result of a query over the global relations, view definitions $\mathcal{V} = (V_1, \ldots V_n)$,

Certain answers

- certain answers for $\mathcal{Q}$—a set of tuples $\mathcal{Q}(D)$ for each database D consistent with a given instance of source relations,

Quick reminder

Data integration

- global relations (mediated schema)—used in queries
- source relations—store actual data, view instance I,
- mapping: LAV—each source relation described as a result of a query over the global relations, view definitions $\mathcal{V} = (V_1, \ldots V_n)$,

Certain answers

- certain answers for $\mathcal{Q}$—a set of tuples $\mathcal{Q}(D)$ for each database D consistent with a given instance of source relations,
- t is a certain answer
  - under OWA (views are sound) if t is an element of $\mathcal{Q}(D)$ for each database D such that $I \subseteq \mathcal{V}(D)$
  - under CWA (views are exact) if t is an element of $\mathcal{Q}(D)$ for each database D such that $I = \mathcal{V}(D)$

Quick reminder

Data integration

- global relations (mediated schema)—used in queries
- source relations—store actual data, view instance I,
- mapping: LAV—each source relation described as a result of a query over the global relations, view definitions $\mathcal{V} = (V_1, \ldots V_n)$,

Query rewriting

- query rewriting using views—mentions the source relations only, can be equivalent or maximally-contained (possibly relative to a set of constraints).

Outline

Query Answering vs. Incomplete Databases

Idea

- Views (=source data) represent many possible (global) databases
- Idea: use techniques in incomplete databases

Example

View definitions:

$$\begin{array}{lll} \mathrm{v}(0, Y) & :- & \mathrm{p}(0, Y) \\ \mathrm{v}(X, Y) & :- & \mathrm{p}(X, Z), \mathrm{p}(Z, Y) \end{array}$$

View instance:

$$\{v(0,1), v(1,1)\}$$

Query Answering vs. Incomplete Databases

Idea

- Views (=source data) represent many possible (global) databases
- Idea: use techniques in incomplete databases

Example

View definitions:

$$\begin{array}{lll} \mathrm{v}(0, Y) & :- & \mathrm{p}(0, Y) \\ \mathrm{v}(X, Y) & :- & \mathrm{p}(X, Z), \mathrm{p}(Z, Y) \end{array}$$

View instance:

$$\{v(0,1), v(1,1)\}$$

Conditional table (OWA):

$$\begin{array}{lccc} \mathrm{p}: & 0 & 1 & w = 1 \\ & 0 & x & w \neq 1 \\ & x & 1 & w \neq 1 \\ & 1 & u & \mathrm{true} \\ & u & 1 & \mathrm{true} \end{array}$$

Query Answering vs. Incomplete Databases

Idea

- Views (=source data) represent many possible (global) databases
- Idea: use techniques in incomplete databases

Example

View definitions:

$$\begin{array}{lll} \mathrm{v}(0, Y) & :- & \mathrm{p}(0, Y) \\ \mathrm{v}(X, Y) & :- & \mathrm{p}(X, Z), \mathrm{p}(Z, Y) \end{array}$$

View instance:

$$\{v(0,1), v(1,1)\}$$

Conditional table (OWA):

p:			
	0	1	$w = 1$
	0	x	$w \neq 1$
	x	1	$w \neq 1$
	1	u	true
	u	1	true

Conditional table (CWA):

p:			
	0	1	true
	1	1	true

Query Answering under OWA vs. Query Containment

Simple reductions between the two problems in both directions exist (for views and queries in CQ, $CQ^{\neq}$, PQ, datalog)

Reduction to query containment

Input: $\mathcal{V} = (v_1, \ldots, v_k)$, $\mathcal{Q}$, I and a tuple t.
Let $\mathcal{Q}'$ be the query consisting of all the definitions $\mathcal{V}$ together with:

$$\begin{aligned} \mathrm{q}'(t) \quad :- \quad & \mathrm{v}_1(t_{11}), \ldots, \mathrm{v}_1(t_{1n_1}), \ldots \\ & \mathrm{v}_1(t_{k1}), \ldots, \mathrm{v}_k(t_{kn_1}) \end{aligned}$$

where $I(v_i) = \{t_{i1}, \ldots, t_{in_i}\}$
Then t is a certain answer iff $\mathcal{Q}' \subseteq \mathcal{Q}$.

Query Answering under OWA vs. Query Containment

Simple reductions between the two problems in both directions exist (for views and queries in CQ, $CQ^{\neq}$, PQ, datalog)

Reduction to computing certain answers

Input: $\mathcal{Q}_1$ and $\mathcal{Q}_2$.
Let the view definition be the rules of $\mathcal{Q}_1$ together with

$$\mathrm{v}(c) \quad :- \quad \mathrm{q}_1(X), p(X)$$

Let the instance $I = \{v(c)\}$ and let $\mathcal{Q}$ consists of all the rules of $\mathcal{Q}_2$ together with

$$\mathrm{q}(c) \quad :- \quad \mathrm{q}_2(X), p(X)$$

Then $\mathcal{Q}_1 \subseteq \mathcal{Q}_2$ iff (c) is a certain answer.

Query Answering under OWA vs. Query Containment

Simple reductions between the two problems in both directions exist (for views and queries in CQ, $CQ^{\neq}$, PQ, datalog)

Consequences

Decidability and undecidability results carry over in both directions.
If the problems are decidable then
the **combined complexity** of computing certain answers is the same as the **query complexity** of query containment.

Data complexity of computing certain answers under OWA

views \ query	CQ	$CQ^{\neq}$	PQ	datalog	FO
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.
FO	undec.	undec.	undec.	undec.	undec.

Data complexity of computing certain answers under CWA

views \ query	CQ	CQ$^{\neq}$	PQ	datalog	FO
CQ	coNP	coNP	coNP	coNP	undec.
CQ$^{\neq}$	coNP	coNP	coNP	coNP	undec.
PQ	coNP	coNP	coNP	coNP	undec.
datalog	undec.	undec.	undec.	undec.	undec.
FO	undec.	undec.	undec.	undec.	undec.

Maximally contained rewriting vs. certain answers

- A datalog query $\mathcal{P}$ is a **query plan** if all EDB predicates in $\mathcal{P}$ are views literals.
- The **expansion** $\mathcal{P}^{\text{exp}}$ of a query plan $\mathcal{P}$ is $\mathcal{P}$ with all views literals replaced with their definitions.
- A query plan $\mathcal{P}$ is **maximally-contained** in a datalog query $\mathcal{Q}$ w.r.t. view definitions $\mathcal{V}$ if
  - $\mathcal{P}^{\text{exp}} \subseteq \mathcal{Q}$, and
  - for each query plan $\mathcal{P}'$ with $(\mathcal{P}')^{\text{exp}} \subseteq \mathcal{Q}$ we have $(\mathcal{P}')^{\text{exp}} \subseteq \mathcal{P}^{\text{exp}}$.

Maximally contained rewriting vs. certain answers

Theorem

Let $\mathcal{V} \subseteq$ CQ, $\mathcal{Q} \in$ datalog, let $\mathcal{P}$ be maximally-contained in $\mathcal{Q}$ w.r.t. $\mathcal{V}$. Then for each view instance $\mathcal{I}$ the query plan $\mathcal{P}$ computes exactly the certain answers of $\mathcal{Q}$ under OWA.

Proof.

I - view instance such that $\mathcal{P}$ fails to compute a certain answer t.
$\mathcal{P}'$ - the query plan $\mathcal{P}$ with two additional rules:

$$\begin{array}{lllll} r_1: & \mathrm{q}'(X) & :- & \mathrm{q}(X) & \\ r_2: & \mathrm{q}'(t) & :- & \mathrm{v}_1(t_{11}), \ldots, \mathrm{v}_1(t_{1n_1}), \ldots & \\ & & & \mathrm{v}_1(t_{k1}), \ldots, \mathrm{v}_k(t_{kn_1}) & \end{array}$$

where $I(v_i) = \{t_{i1}, \ldots, t_{in_i}\}$ and q is the answer predicate of $\mathcal{P}$.

$(\mathcal{P}')^{\exp}$ is contained in $\mathcal{Q}$ but it is not contained in $(\mathcal{P})^{\exp}$.
That contradicts the maximal containment of $\mathcal{P}$ in $\mathcal{Q}$. □

Outline

Inverse rules

Example

Data sources

$$
\begin{aligned}
s_1(X, Y) &:- \text{edge}(X, Z), \text{edge}(Z, W), \text{edge}(W, Y) \\
s_2(X) &:- \text{edge}(X, Z)
\end{aligned}
$$

Inverse rules

Example

Data sources

$$\begin{aligned} s_1(X, Y) &:- \text{edge}(X, Z), \text{edge}(Z, W), \text{edge}(W, Y) \\ s_2(X) &:- \text{edge}(X, Z) \end{aligned}$$

Inverse rules

$$\text{edge}(X, f_1(X, Y)) :- s_1(X, Y)$$

The fresh function symbol $f_{r,i}$
for each rule r and each existential variable X_i in r

Inverse rules

Example

Data sources

$$\begin{array}{lll} s_1(X, Y) & :- & \text{edge}(X, Z), \text{edge}(Z, W), \text{edge}(W, Y) \\ s_2(X) & :- & \text{edge}(X, Z) \end{array}$$

Inverse rules

$$\begin{array}{lll} \text{edge}(X, f_1(X, Y)) & :- & s_1(X, Y) \\ \text{edge}(f_1(X, Y), f_2(X, Y)) & :- & s_1(X, Y) \end{array}$$

The fresh function symbol $f_{r,i}$
for each rule r and each existential variable X_i in r

Inverse rules

Example

Data sources

$$\begin{array}{lll} s_1(X, Y) & :- & \text{edge}(X, Z), \text{edge}(Z, W), \text{edge}(W, Y) \\ s_2(X) & :- & \text{edge}(X, Z) \end{array}$$

Inverse rules

$$\begin{array}{lll} \text{edge}(X, f_1(X, Y)) & :- & s_1(X, Y) \\ \text{edge}(f_1(X, Y), f_2(X, Y)) & :- & s_1(X, Y) \\ \text{edge}(f_1(X, Y), Y) & :- & s_1(X, Y) \\ \text{edge}(X, f_3(X)) & :- & s_2(X) \end{array}$$

The fresh function symbol $f_{r,i}$
for each rule r and each existential variable X_i in r

Inverse rules algorithm (1)

Example

Query $\mathcal{Q}$:

$$\begin{aligned} \mathrm{q}(X, Y) &:- \mathrm{edge}(X, Y) \\ \mathrm{q}(X, Y) &:- \mathrm{edge}(X, Z), \mathrm{edge}(Z, Y) \end{aligned}$$

Inverse rules algorithm (1)

Example

Query $\mathcal{Q}$:

$$\begin{aligned} \mathrm{q}(X, Y) &:- \mathrm{edge}(X, Y) \\ \mathrm{q}(X, Y) &:- \mathrm{edge}(X, Z), \mathrm{edge}(Z, Y) \end{aligned}$$

Data source:

$$\mathrm{s}(\mathrm{X}, \mathrm{Y}) :- \mathrm{edge}(X, Z), \mathrm{edge}(Z, Y)$$

Inverse rules algorithm (1)

Example

Query $\mathcal{Q}$:

$$\begin{aligned} \mathrm{q}(X, Y) &:- \mathrm{edge}(X, Y) \\ \mathrm{q}(X, Y) &:- \mathrm{edge}(X, Z), \mathrm{edge}(Z, Y) \end{aligned}$$

Data source:

$$\mathrm{s}(\mathrm{X}, \mathrm{Y}) :- \mathrm{edge}(X, Z), \mathrm{edge}(Z, Y)$$

Query plan $(\mathcal{Q}, \mathcal{V}^{-1})$:

$$\begin{aligned} \mathrm{q}(X, Y) &:- \mathrm{edge}(X, Y) \\ \mathrm{q}(X, Y) &:- \mathrm{edge}(X, Z), \mathrm{edge}(Z, Y) \\ \mathrm{edge}(X, f(X, Y)) &:- \mathrm{s}(X, Y) \\ \mathrm{edge}(f(X, Y), Y) &:- \mathrm{s}(X, Y) \end{aligned}$$

Inverse rules algorithm (1)

Example

Query $\mathcal{Q}$:

Data sou

Query pla

No longer datalog, but
we can evaluate it in two stages:

- start with the inverse rules (they introduce function symbols but are not recursive),
- apply the rules of $\mathcal{Q}$, (they are recursive but do not introduce function symbols).

In fact, with a little bit of bureaucracy
we can get rid of function symbols at all.

$$\begin{aligned} \text{edge}(X, f(X, Y)) &:- \text{s}(X, Y) \\ \text{edge}(f(X, Y), Y) &:- \text{s}(X, Y) \end{aligned}$$

Inverse rules algorithm (2)

Inverse rules algorithm

- Compute plan $(\mathcal{Q}, \mathcal{V}^{-1})\downarrow$ that returns the same set of tuples as $(\mathcal{Q}, \mathcal{V}^{-1})$ but filters out the tuples that contain function symbol(s).
- Evaluate $(\mathcal{Q}, \mathcal{V}^{-1})\downarrow$ on a set of data sources.

Something similar: chase

Applying inverse rules is like chasing $\mathcal{I}$ with the view definitions.

Example

$$\begin{array}{lcl} \mathrm{s(X, Y)} & :- & \mathrm{edge}(X, Z), \mathrm{edge}(Z, Y) \\ \forall X, Y\ \mathrm{s(X, Y)} & \rightarrow & \exists Z\ \mathrm{edge}(X, Z), \mathrm{edge}(Z, Y) \end{array}$$

$\mathcal{I}$:

s:		
0	0	
0	1	
3	2	

$\mathcal{V}^{-1}(\mathcal{I})$:

edge:	
0	Z_1
Z_1	0
0	Z_2
Z_2	1
3	Z_3
Z_3	2

Outline

Probles with the inverse rules algorithm

The inverse rules algorithm produces expensive query plans

- Does not use views directly, forces a lot of recomputation.
  e.g. $Q(\bar{x}) = V(\bar{x})$
- May compute useless tuples i.e. may invert the extensions of the views that are not needed in the rewriting.

Another approach: bucket algorithm

1. Create buckets, one for each subgoal g in $\mathcal{Q}$.
   The bucket for g contains the views with subgoals to which g can be mapped.
2. For each element of the Cartesian product of the buckets
   1. construct a conjunctive rewriting r,
   2. check the containment of r in $\mathcal{Q}$
      (equate some pairs of variables in r, if necessary).

Example

$$
\begin{array}{lll}
q(X) & :- & e(X,Y), e(Y,X), p(X,Y) \\
v_1(U) & :- & e(U,V), e(V,U) \\
v_2(U,V) & :- & p(U,V) \\
v_3(U,W) & :- & e(U,V), e(V,W), p(U,V)
\end{array}
$$

$e(X,Y)$	$e(Y,X)$	$p(X,Y)$
$v_1(X), v_1(Y)$	$v_1(X)$	$v_2(X,Y)$
$v_3(X,Y)$	$v_3(X,Y)$	$v_3(X,Y)$

Rewritings

$$
\begin{array}{lll}
q_0(X) & :- & v_1(X), v_2(X,Y) \\
\ldots & & \\
q_i(X) & :- & v_3(X,X)
\end{array}
$$

Another approach: bucket algorithm

1. Create buckets, one for each subgoal g in $\mathcal{Q}$.
   The bucket for g contains the views with subgoals to which g can be mapped.
2. For each element of the Cartesian product of the buckets
   1. construct a conjunctive rewriting r,
   2. check the containment of r in $\mathcal{Q}$
      (equate some pairs of variables in r, if necessary).

Example

$$
\begin{array}{lcl}
\mathrm{q}(\mathrm{X}) & :- & \mathrm{e}(X, Y), \mathrm{e}(Y, X), \mathrm{p}(X, Y) \\
\mathrm{v}_1(\mathrm{U}) & :- & \mathrm{e}(U, V), \mathrm{e}(V, U) \\
\mathrm{v}_2(\mathrm{U}, \mathrm{V}) & :- & \mathrm{p}(U, V) \\
\mathrm{v}_3(\mathrm{U}, \mathrm{W}) & :- & \mathrm{e}(U, V), \mathrm{e}(V, W), \mathrm{p}(U, V)
\end{array}
$$

$\mathrm{e}(X, Y)$	$\mathrm{e}(Y, X)$	$\mathrm{p}(X, Y)$
$\mathrm{v}_1(X), \mathrm{v}_1(Y)$	$\mathrm{v}_1(X)$	$\mathrm{v}_2(X, Y)$
$\mathrm{v}_3(X, Y)$	$\mathrm{v}_3(X, Y)$	$\mathrm{v}_3(X, Y)$

Problems

- Expensive - tries many useless combinations,
- e.g. v_1 useless in the rewriting: since Y is not distinguished it is not possible to join $\mathrm{e}(X, Y)$ and $\mathrm{p}(X, Y)$.

MiniCon algorithm

Better idea

- As before, for each subgoal g in $\mathcal{Q}$ find the views with subgoals to which g can be mapped.
- But then, given such a partial mapping, finds minimal additional set of subgoals in the query that have to be mapped together.

MiniCon Descriptions (MCDs)

MCD C for a query $\mathcal{Q}$ over a view v consists of

- head homomorphism h (may equate head variables e.g. $v_3(X, X)$),
- partial mapping φ from Vars($\mathcal{Q}$) to Vars(V).
- some subset G of subgoals in $\mathcal{Q}$ that are covered by some subgoal in $h(V)$ using φ.

MCDs

Key Property

MCD C for $\mathcal{Q}$ over V can only be used in a non-redundant rewriting of $\mathcal{Q}$ if:

- C1 For each head variable x of $\mathcal{Q}$ in domain of φ, $\varphi(x)$ is head variable in MCD view (i.e. in $h(V)$).
- C2 If a variable participates in a join predicate (in Q) which is not enforced by V, then it must be in the head of the view. (new!)

Example

$$
\begin{array}{lll}
\mathrm{q(X)} & :- & \mathrm{e}(X, Y), \mathrm{e}(Y, X), \mathrm{p}(X, Y) \\
\mathrm{v_1(U)} & :- & \mathrm{e}(U, V), \mathrm{e}(V, U) \\
\mathrm{v_2(U, V)} & :- & \mathrm{p}(U, V) \\
\mathrm{v_3(U, W)} & :- & \mathrm{e}(U, V), \mathrm{e}(V, W), \mathrm{p}(U, V)
\end{array}
$$

$v(\bar{Y})$	h	φ	goals covered
~~$v_1(U)$~~	~~$U \to U$~~	~~$X \to U, Y \to V$~~	~~1, 2?~~ C2
$v_2(U, V)$	$U \to U, V \to V$	$X \to U, Y \to V$	3
$v_3(U, U)$	$U \to U, W \to U$	$X \to U, Y \to V$	1, 2, 3

MiniCon Algorithm: Phase 2

Minimality of MCDs

Only the minimal set of subgoals required to satisfy the Key Property is included in the set of goals G that are covered by MCD

Phase 2: combining MCDs

The only combinations of MCDs that can result in a non-redundant rewriting s of $\mathcal{Q}$ are such that:

- the sets of subgoals covered by the MCDs form a partition of the set of subgoals of $\mathcal{Q}$.

MiniCon Algorithm: Phase 2

Running time (worst-case)

The running time of the MiniCon algorithm is $\mathcal{O}(nmM)^n$, where

- n is the number of subgoals in the query,
- m is the maximal number of subgoals in a view,
- M is the number of views.

Outline

Back to inverse rules - full dependencies

Extending the inverse rules algorithm

Inverse rule algorithm can be extended to deal with full dependencies and access patterns.
The idea: add new datalog rules to the rewriting.

Rectification

New relation e is added with an intention to capture equality.
Queries should be modified to be able to use e.
For example

$$\mathrm{q}(X) \quad :- \quad \mathrm{pred}(c, X, Y, Y)$$

should be rewritten to its **rectified** version

$$\bar{\mathrm{q}}(X) \quad :- \quad \mathrm{pred}(Z, X', Y, Y'), e(X, X'), e(c, Z), e(Y, Y')$$

Back to inverse rules - full dependencies

The rules: chase(Δ)

For each rectified full dependency in Δ

$$\forall \bar{X}\ p_1(\bar{X}_1) \wedge \ldots \wedge p_{n-1}(\bar{X}_{n-1}) \quad \rightarrow \quad p_n(\bar{X}_n)$$

(p_i are global relations or the relation e, $\bar{X}_n \subseteq \bar{X}$, and $\bar{X} = \bar{X}_1, \ldots, \bar{X}_{n-1}$)

introduce a new datalog rule in chase(Δ)

$$p_n(\bar{X}_n) \quad :- \quad p_1(\bar{X}_1) \wedge \ldots \wedge p_{n-1}(\bar{X}_{n-1})$$

The rewriting

Let Δ be a set of full dependencies, $\mathcal{V}$ a set of conjunctive source descriptions, and let $\mathcal{Q}$ be a (rectified) query.
Let $\mathcal{R}$ be the set of rules $\mathcal{V}^{-1} \cup \text{chase}(\Delta) \cup \text{Equiv}(e)$.
Then $(\mathcal{Q}, \mathcal{R})\downarrow$ is maximally-contained in $\mathcal{Q}$ relative to Δ.

Access patterns - domain enumeration

Access patterns

$$
\begin{array}{lll}
s_1^o(X) & :- & \text{podsPaper}(X) \\
s_2^{io}(X, Y) & :- & \text{cites}(X, Y) \\
s_3^i(X) & :- & \text{awarded}(X)
\end{array}
$$

Query

$$
q(X) \quad :- \quad \text{awarded}(X)
$$

The executable rewriting

$$
\begin{array}{lll}
\text{domain}(X) & :- & s_1^o(X) \\
\text{domain}(X) & :- & \text{domain}(Y), s_2^{io}(Y, X) \\
\text{q}(X) & :- & \text{domain}(X), s_3^i(X)
\end{array}
$$

Outline

Once again: our setting

We are given:

- a query Q over global schema,
- a set of views with access patterns,
- a set of constraints Σ_c,

We have to find a query E such that

- E mentions the views literals only,
- E is executable w.r.t. access patterns,
- E is equivalent (or at least minimally-containing) to Q relative to Σ

where Σ contains $\Sigma_c \cup \Sigma_f^{\mathcal{V}} \cup \Sigma_b^{\mathcal{V}}$

Forward constraints $\Sigma_f^{\mathcal{V}}$ and backward constraints $\Sigma_b^{\mathcal{V}}$

For each V_i in $\mathcal{V}$ we have

- forward constraint: $\forall \bar{X}_i, \bar{Y}_i \, (\mathrm{body}(V_i) \rightarrow \mathrm{head}(V_i))$
- backward constraint: $\forall \bar{X}_i \, (\mathrm{head}(V_i) \rightarrow \exists \bar{Y}_i \mathrm{body}(V_i))$

Chase: handling negation

Constraints IC(UCQ$^\neg$)

$$\sigma : \forall \bar{X}\ \psi(\bar{x}) \rightarrow \bigvee_{i=1}^{l} \exists \bar{Y}_i\ \xi_i(\bar{X}, \bar{Y}_i)$$

where ψ and ξ_i are quantifier-free CQ$^\neg$

Step for Q in CQ$^\neg$

- Chase step of Q with σ applies iff
  there is homomorphism h from ψ to Q such that
  for each i, h has no extension to a homomorphism from $\psi \wedge \xi_i$ to Q.
- The result is $\bigvee_{i=1}^{l} Q \wedge h'(\xi_i)$ (h' extends h to be the identity on $\bar{Y}_i$)

Negation Constraints $\Sigma^\tau_\neg$

For each relation r in the schema τ, the set $\Sigma^\tau_\neg$ includes
the constraint $\forall \bar{X}\, \text{true} \rightarrow (r(\bar{X}) \vee \neg r(\bar{X}))$

Rewriting and the chase

ViewRewrite($Q, \Sigma_c, \mathcal{V}$)

1. $Q^1 = \text{chase}(Q, \Sigma_c \cup \Sigma^{\tau}_{\neg})$
2. $Q^2 = \text{chase}(Q^1, \Sigma^{\mathcal{V}}_f \cup \Sigma^{\tau_{\mathcal{V}}}_{\neg})$
3. $Q^3 = Q^2|_{\tau_{\mathcal{V}}}$ (leave the view literals only)
4. $Q^4 = \text{ans}(Q^3)$

If there exists an executable rewriting using views that contains Q (and the chase terminates) then ViewRewrite($Q, \Sigma_c, \mathcal{V}$) returns the minimal executable overestimate of Q.

Is the rewriting equivalent (relative to Σ)?

ViewFeasible($Q, \Sigma_c, \mathcal{V}$)

1. Q^4 = ViewRewrite($Q, \Sigma_c, \mathcal{V}$)
2. if Q^4 is undefined then return false
3. $Q^5 = \text{chase}(Q^4, \Sigma_b^{\mathcal{V}} \cup \Sigma_{\neg}^{\tau\mathcal{V}})$
4. $Q^6 = \text{chase}(Q^5, \Sigma_c \cup \Sigma_{\neg}^{\tau})$
5. $Q^7 = Q^6|_{\tau\mathcal{V}}$ (drop the view literals)
6. if Q^7 is contained in Q return true otherwise return false

If ViewFeasible($Q, \Sigma_c, \mathcal{V}$) terminates then it returns true iff there is an executable rewriting of Q using $\mathcal{V}$ that is equivalent to Q relative to Σ.

Conclusions

- Connections of query answering in data integration to incomplete databases as well as to the problem of query containment,
- Inverse rules and MiniCon algorithm that compute maximally-contained rewritings w.r.t conjunctive views,
- Inverse rules algorithm allows for processing recursive queries, full dependencies and access patterns,
- Recursive plans may be necessary when rewriting with access patterns or under functional dependencies,
- Access patterns, constraints and negation can be treated in a uniform way (chase).