Obtaining Certain and Consistent Answers from Data Integration Systems

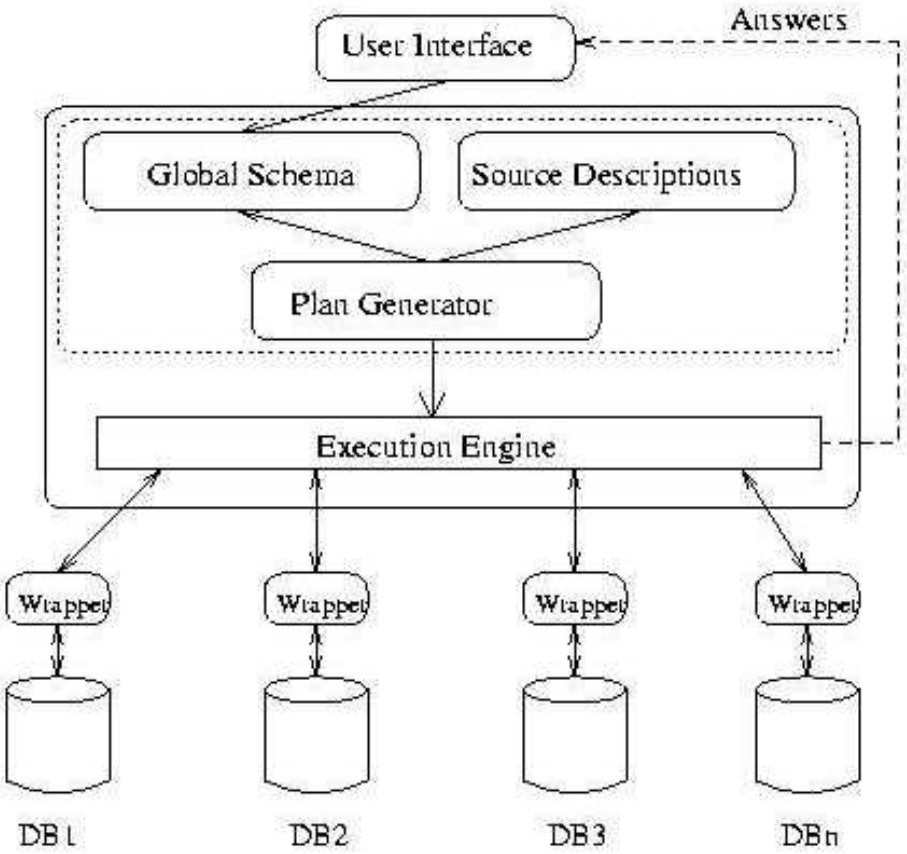
Loreto Bravo
Computer Science Department
Pontificia Universidad Católica de Chile
lbravo@ing.puc.cl

Joint work with:
Leopoldo Bertossi
School of Computer Science
Carleton University. Ottawa, Canada
bertossi@scs.carleton.ca

Presentation Outline

- Preliminaries
  - Virtual Data Integration Systems
- Minimal Instances
  - Consistency
  - Logic Programming Specification of Minimal Instances
    - Open Case
    - Mixed Case
- Computing Consistent Answers
  - Logic Programming Specification of Repairs
  - Retrieving Consistent Answers
- Conclusions

Virtual Data Integration Systems



Global schema $\mathcal{R}$

Local Sources

Schema

Type: open, closed

Contents v

Mapping:

Global-as-View (GAV)

Local-as-View (LAV)

$$V(\bar{X}) \leftarrow \varphi_V(\bar{X})$$

LAV: tables in local sources are described as (conjunctive) views of the global schema

$$V_1(Brand,\ Model,\ Year,\ Color) \leftarrow Car(Brand,\ Model,\ Year,\ Color),\\ Brand = Susuki,\ Year \geq 1990$$

$$V_2(Brand,\ Model,\ Country) \leftarrow Car(Brand,\ Model,\ Year,\ Color),\\ BrandOrigin(Brand, Country)$$

Open, Closed and Clopen

Virtual Global Instances?

$$
\begin{aligned}
V_i(\bar{X}) \leftarrow \varphi_{V_i}(\bar{X}) &\quad : \text{LAV mapping} \\
D &\quad : \text{global database instance} \\
\varphi_{V_i}(D) &\quad : \text{extension of } V_i \text{ obtained by applying } \varphi_{V_i} \text{ to } D \\
v_i &\quad : \text{source contents}
\end{aligned}
$$

The legal instances of $\mathcal{G}$ are:

$$
\begin{aligned}
Linst(\mathcal{G}) = \{D \text{ over } \mathcal{R} \mid\ & v_i \subseteq \varphi_{V_i}(D), \text{for all open sources} \\
& v_i \supseteq \varphi_{V_i}(D), \text{for all closed sources} \\
& v_i = \varphi_{V_i}(D), \text{for all clopen sources}\}
\end{aligned}
$$

The *certain answers* to a global query Q are those that can be obtained as answers from *every* legal instance: $Certain_{\mathcal{G}}(Q)$

Example 1: Global system $\mathcal{G}_1$

$V_1(X,Y) \leftarrow R(X,Y)$ with $v_1 = \{(a,b),(c,d)\}$ open

$V_2(X,Y) \leftarrow R(Y,X)$ with $v_2 = \{(c,a),(e,d)\}$ open

$V_3(X) \leftarrow P(X)$ with $v_3 = \{(a),(d)\}$ clopen

Global instance $D = \{R(a,b), R(c,d), R(a,c), R(d,e), P(a), P(d)\}$ is legal

- $v_1 \subseteq \varphi_1(D) = \{(a,b),(c,d),(a,c),(d,e)\}$
- $v_2 \subseteq \varphi_2(D) = \{(b,a),(d,c),(c,a),(e,d)\}$
- $v_3 = \varphi_3(D) = \{(a),(d)\}$

The legal instances are supersets of D that do not add tuples to the predicate P. No proper subsets of D are legal instances.

Query Q: $R(X,Y)$?

$Certain_{\mathcal{G}}(Q) = \{(a,b),(c,d),(a,c),(d,e)\}$

Presentation Outline

- ✓ Preliminaries
  - ✓ Virtual Data Integration Systems
- Minimal Instances
  - Consistency
  - Logic Programming Specification of Minimal Instances
    - Open Case
    - Mixed Case
- Computing Consistent Answers
  - Logic Programming Specification of Repairs
  - Retrieving Consistent Answers
- Conclusions

Consistency

Example 1 (continued)

$V_1(X,Y) \leftarrow R(X,Y)$ with $v_1 = \{(a,b),(c,d)\}$ open

$V_2(X,Y) \leftarrow R(Y,X)$ with $v_2 = \{(c,a),(e,d)\}$ open

$V_3(X) \leftarrow P(X)$ with $v_3 = \{(a),(d)\}$ clopen

Query Q: $R(X,Y)$?

$Certain_{\mathcal{G}}(Q) = \{(a,b),(c,d),(a,c),(d,e)\}$

Local FDs $V_1\colon\ X \rightarrow Y$, $V_2\colon\ X \rightarrow Y$ are satisfied in the sources

But the global FD $R\colon X \rightarrow Y$ is not satisfied by legal instance
$D = \{(a,b),(c,d),(a,c),(d,e)\}$

Only $(c,d),(d,e)$ should be consistent answers

Several questions arise:

- What does it mean for $\mathcal{G}$ to satisfy global ICs?

- What are the consistent answers to a global query?

  How to compute them?

These questions were addressed in

[Bertossi, Chomicki, Cortés and Gutiérrez; FQAS02]

First we briefly review some notions introduced there

A *minimal global instance* is a legal instance that does not properly contain any other legal instance

$Mininst(\mathcal{G})$:= set of minimal instances of $\mathcal{G}$

The *minimal answers* to a query are those that can be contained from *every* minimal instance:

$$Certain_{\mathcal{G}}(Q) \subseteq Minimal_{\mathcal{G}}(Q)$$

For monotone queries they coincide; with negation, possibly not

$\mathcal{G}$ is **consistent** wrt ICs if every *minimal* instance satisfies the ICs

Specification of Minimal Instances: Open Case

Example 2: $\mathcal{D} = \{a, b, c, \dots\}$ $\mathcal{G}_2$:

$$V_1(X,Z) \leftarrow P(X,Y), R(Y,Z) \qquad \{v_1(a,b)\} \quad \text{open}$$
$$V_2(X,Y) \leftarrow P(X,Y) \qquad \{v_2(a,c)\} \quad \text{open}$$

Inverse Rules [Duschka,Genesereth; SAC97]:

$$P(X, f(X,Z)) \leftarrow V_1(X,Z) \qquad R(f(X,Z),Z) \leftarrow V_1(X,Z)$$
$$P(X,Y) \leftarrow V_2(X,Y)$$

Try to use something like this to specify minimal instances ...

Answer set program $\Pi(\mathcal{G})$:

1. Fact $dom(a)$ for every constant $a \in \mathcal{D}$

2. Fact $V_i(\bar{a})$ whenever $V_i(\bar{a}) \in v_i$ for a source extension v_i in $\mathfrak{G}$

3. For every view (source) predicate V_i with definition $V_i(\bar{X}) \leftarrow P_1(\bar{X}_1), \ldots, P_n(\bar{X}_n)$, the rule

$$P_j(\bar{X}_j) \leftarrow V(\bar{X}), \bigwedge_{X_i \in (\bar{X}_j \setminus \bar{X})} F_i(\bar{X}, X_i)$$

4. For every predicate $F_i(\bar{X}, X_i)$ introduced in 3., the rule

$$F_i(\bar{X}, X_i) \leftarrow V_i(\bar{X}), dom(X_i), choice((\bar{X}), (X_i))$$

$choice((\bar{X}),(X_i))$: non-deterministically chooses a unique value for X_i for each value of $\bar{X}$

[Giannotti,Pedreschi,Sacca,Zaniolo; DOOD'91]

Models are the *choice models*, but the program can be transformed into one with answer sets (stable models)

$$Mininst(\mathcal{G}) \quad \subseteq \quad \text{stable models of } \Pi(\mathcal{G}) \quad \subseteq \quad Linst(\mathcal{G})$$

Queries expressed as logic programs can be answered from the query program together with $\Pi(\mathfrak{G})$ under cautious stable model semantics

For monotone queries Q, answers obtained using $\Pi(\mathcal{G})$ coincide with $Certain_{\mathcal{G}}(Q)$ and $Minimal_{\mathcal{G}}(Q)$

Example 2 (continued) $\mathcal{D} = \{a, b, c, \dots\}$ $\mathcal{G}_2$:

$$V_1(X, Z) \leftarrow P(X, Y), R(Y, Z) \qquad \{v_1(a, b)\} \quad \text{open}$$
$$V_2(X, Y) \leftarrow P(X, Y) \qquad \{v_2(a, c)\} \quad \text{open}$$

$\Pi(\mathcal{G}_2)$:

$$dom(a)., \quad dom(b)., \quad dom(c)., \quad \dots, V_1(a, b)., \quad V_2(a, c).$$
$$P(X, Z) \leftarrow V_1(X, Y), F_1(X, Y, Z)$$
$$R(Z, Y) \leftarrow V_1(X, Y), F_1(X, Y, Z)$$
$$P(X, Y) \leftarrow V_2(X, Y)$$
$$F_1(X, Y, Z) \leftarrow V_1(X, Y), dom(Z), choice((X, Y), (Z))$$

Example 2 (continued) $\mathcal{G}_2$:

$$V_1(X,Z) \leftarrow P(X,Y), R(Y,Z) \qquad \{v_1(a,b)\} \quad \text{open}$$
$$V_2(X,Y) \leftarrow P(X,Y) \qquad \{v_2(a,c)\} \quad \text{open}$$

$Mininst(\mathcal{G}) = \{\{P(a,c), P(a,z), R(z,b)\} \mid z \in \{a,b,c,...\}\}$

The stable models of $SV(\Pi(\mathcal{G}_2))$ are:

$\mathcal{M}_b = \{dom(a), \ldots, V_1(a,b), V_2(a,c), \underline{P(a,c)}, diffChoice_1(a,b,a),$
$chosen_1(a,b,b), diffChoice_1(a,b,c), F_1(a,b,b), \underline{R(b,b)}, \underline{P(a,b)}\}$

$\mathcal{M}_a = \{dom(a), \ldots, V_1(a,b), V_2(a,c), \underline{P(a,c)}, chosen_1(a,b,a),$
$diffChoice_1(a,b,b), diffChoice_1(a,b,c), F_1(a,b,a), \underline{R(a,b)}, \underline{P(a,a)}\}$

$\mathcal{M}_c = \{dom(a), \ldots, V_1(a,b), V_2(a,c), \underline{P(a,c)}, diffChoice_1(a,b,a),$
$diffChoice_1(a,b,b), chosen_1(a,b,c), F_1(a,b,c), \underline{R(c,b)}\}$
$\ldots$

Here: 1-1 correspondence with $Mininst(\mathcal{G})$

Example 3: $\mathcal{G}_3$:

$$\begin{array}{rcll} V_1(X) & \leftarrow & P(X,Y) & \{v_1(a)\} \quad \text{open} \\ V_2(X,Y) & \leftarrow & P(X,Y) & \{v_2(a,c)\} \quad \text{open} \end{array}$$

$Mininst(\mathcal{G}_3) = \{\{P(a,c)\}\}$

However, the legal global instances corresponding to models of $\Pi(\mathcal{G}_3)$ are of the form $\{\{P(a,c), P(a,z)\} \mid z \in \mathcal{D}\}$

As V_2 is open, it forces $P(a,c)$ to be in all legal instances, and with this, same condition on V_1 is automatically satisfied, and no other values for Y are needed

But the choice operator still has freedom to chose other values (the $z \in \mathcal{D}$)

We want $\Pi(\mathcal{G})$ to capture **only** the minimal instances

A revised version of $\Pi(\mathcal{G})$ is able to detect in which cases it is necessary to use the function predicates.

$$F_i(\bar{X}, X_i) \leftarrow add\_V_i(\bar{X}), dom(X_i), choice((\bar{X}), (X_i))$$

where $add\_V_i(\bar{X})$ is true only when the openness of source V_i is not satisfied through other views.

$$\text{stable models of } \Pi(\mathfrak{G}) \quad \equiv \quad Mininst(\mathcal{G}).$$

This program not only specifies the minimal instances, but can be also used to compute certain answers to monotone queries

Specification of Minimal Instances: Mixed Case

Example 4: $\mathcal{D} = \{a, b, c, \dots\}$ $\mathcal{G}_4$:

$V_1(X, Z) \leftarrow P(X, Y), R(Y, Z)$ $\{v_1(a, b)\}$ open

$V_2(X, Y) \leftarrow P(X, Y)$ $\{v_2(a, c)\}$ clopen

In Example 2 we had the same sources but they were all open. There we had:

$$Mininst(\mathcal{G}_2) = \{\{P(a, c), P(a, z), R(z, b)\} \mid z \in \{a, b, c, ...\}\}$$

Here the second sources restricts predicate P to (a, c) so we have that in this case we only get:

$$Mininst(\mathcal{G}_4) = \{\{P(a, c), R(c, b)\}\}$$

This closure condition imposes restrictions on the legal instances, but does not force to add new tuples to them.

$\Pi(\mathcal{G})^{mix}$:

- The same clauses as $\Pi(\mathcal{G})$, but now taking atoms from open and clopen sources.
- For every view predicate V of a clopen or closed source with description $V(\bar{X}) \leftarrow P_1(\bar{X}_1), \ldots, P_n(\bar{X}_n)$, the denial constraint:

$$\leftarrow P_1(\bar{X}_1), \ldots, P_n(\bar{X}_n),\ not\ V(\bar{X}).$$

(notice that in the specification program, V is purely extensional)

If the simple version of $\Pi(\mathcal{G})$ is considered we have that:

$$Mininst(\mathcal{G}) \quad \subseteq \quad \text{stable models of } \Pi(\mathcal{G})^{mix} \quad \subseteq \quad Linst(\mathcal{G})$$

If the revised version of $\Pi(\mathcal{G})$ is considered we have that:

$$\text{stable models of } \Pi(\mathfrak{G})^{mix} \quad \equiv \quad Mininst(\mathcal{G})$$

Example 4: $\mathcal{D} = \{a, b, c, \dots\}$ $\mathcal{G}_4$:

$$V_1(X,Z) \leftarrow P(X,Y), R(Y,Z) \qquad \{v_1(a,b)\} \quad \text{open}$$
$$V_2(X,Y) \leftarrow P(X,Y) \qquad \{v_2(a,c)\} \quad \text{clopen}$$

$\Pi(\mathcal{G}_4)^{mix}$:

$$dom(a)., \quad dom(b)., \quad dom(c)., \quad \dots, V_1(a,b)., \quad V_2(a,c).$$
$$P(X,Z) \leftarrow V_1(X,Y), F_1(X,Y,Z)$$
$$R(Z,Y) \leftarrow V_1(X,Y), F_1(X,Y,Z)$$
$$P(X,Y) \leftarrow V_2(X,Y)$$
$$F_1(X,Y,Z) \leftarrow V_1(X,Y), dom(Z), choice((X,Y),(Z))$$

$$\leftarrow P(X,Y),\ not\ V_2(X,Y)$$

Example 4: $\mathcal{D} = \{a, b, c, \dots\}$ $\mathcal{G}_4$:

$V_1(X, Z) \leftarrow P(X, Y), R(Y, Z)$ $\{v_1(a, b)\}$ open

$V_2(X, Y) \leftarrow P(X, Y)$ $\{v_2(a, c)\}$ clopen

$Mininst(\mathcal{G}) = \{\{P(a, c), R(c, b)\}\}$

The stable model of $\Pi(\mathcal{G}_4)^{mix}$ is:

$\{domd(a), \dots, v1(a, b), v2(a, c), \underline{P(a, c)}, diffchoice_1(a, b, a),$
$diffchoice_1(a, b, b), chosen_1(a, b, c), f_1(a, b, c), \underline{R(c, b)}\}$

Relation between answers obtained from different types of queries and programs:

$\Pi(\mathcal{G})$	Query	$Certain_{\mathcal{G}}(Q)$	$Minimal_{\mathcal{G}}(Q)$
Revised	Monotone	$=$	$=$
	General	$\neq$	$=$
Simple	Monotone	$=$	$=$
	General	$\neq$	$\neq$

Presentation Outline

- ✓ Preliminaries
  - ✓ Virtual Data Integration Systems
- ✓ Minimal Instances
  - ✓ Consistency
  - ✓ Logic Programming Specification of Minimal Instances
    - ✓ Open Case
    - ✓ Mixed Case
- Computing Consistent Answers
  - Logic Programming Specification of Repairs
  - Retrieving Consistent Answers
- Conclusions

Computing Consistent Answers

In [Bertossi, Chomicki, Cortés and Gutiérrez; FQAS02] (inspired by [Arenas, Bertossi, Chomicki; PODS'99]):

A **repair** of a global system $\mathcal{G}$ wrt to global ICs IC is:

- a global instance that satisfies IC, that
- minimally differs from a minimal instance (wrt to inclusion of sets of tuples)

$$Repairs^{IC}(\mathcal{G}) := \text{set of repairs of } \mathcal{G} \text{ wrt } IC$$

A tuple $\bar{t}$ is a **consistent answer** to query Q wrt IC if for every $D \in Repairs^{IC}(\mathcal{G})$: $D \models Q[\bar{t}]$

Intuitively, consistent answers are invariant under minimal restorations of consistency

Example 1 (continued)

Since $Mininst(\mathcal{G}_1) = \{\{R(a,b), R(c,d), R(a,c), R(d,e), P(a), P(d)\}\}$

$\mathcal{G}_1$ is inconsistent wrt FD $R: X \rightarrow Y$

$Repairs^{FD}(\mathcal{G}_1) = \{D^1, D^2\}$

- $D^1 = \{R(a,b), R(c,d), R(d,e), P(a), P(d)\}$
- $D^2 = \{R(c,d), R(a,c), R(d,e), P(a), P(d)\}$

Queries:

$Q(X,Y)$: $R(X,Y)$?

$(c,d), (d,e)$ are the consistent answers

$Q_1(X)$: $\exists Y\ R(X,Y)$?

a is a consistent answer

Specification of Repairs

So far: specification of minimal instances of an integration system

Minimal instances can be inconsistent

In consequence, we want to specify their repairs

[Barcelo, Bertossi; NMR'02], [Barcelo, Bertossi; PADL'02]:

Specification of the repairs of a single, inconsistent relational database using disjunctive logic programs with stable model semantics

We can apply those ideas here ...

We can combine the programs that specify the minimal instances and the (single DB) repair programs into a new program $\Pi(\mathcal{G}, IC)$ that specifies the repairs of an integration system $\mathcal{G}$ wrt IC

Example 3 (continued) $\mathcal{G}_3$:

$$\begin{array}{rcll} V_1(X) & \leftarrow & P(X,Y) & \{v_1(a)\} \quad \text{open} \\ V_2(X,Y) & \leftarrow & P(X,Y) & \{v_2(a,c)\} \quad \text{open} \end{array}$$

IC: $\forall x, y(P(x,y) \rightarrow P(y,x))$

$Mininst(\mathcal{G}_3) = \{\{P(a,c)\}\}$ ... inconsistent system

Repair Program using DLV syntax:

```
dom(a).  dom(c).  v1(a).  v2(a,c).  %begin subprogram for minimal instances

P(X,Y,td) :- P(X,Y,v1).
P(X,Y,td) :- P(X,Y,to).

P(X,Y,nv1) :- P(X,Y,to).
addv1(X) :- v1(X), not auxv1(X).
auxv1(X) :- P(X,Z,nv1).
fz(X,Z) :- addv1(X), dom(Z), chosenv1z(X,Z).
chosenv1z(X,Z) :- addv1(X), dom(Z), not diffchoicev1z(X,Z).
diffchoicev1z(X,Z) :-  chosenv1z(X,ZZ), dom(Z), ZZ!=Z.
P(X,Z,v1) :- addv1(X), fz(X,Z).
P(X,Y,v2) :- v2(X,Y).

P(X,Y,ts) :- P(X,Y,ta), dom(X), dom(Y).      %begin repair subprogram
P(X,Y,ts) :- P(X,Y,td), dom(X), dom(Y).
P(X,Y,fs) :- dom(X), dom(Y), not P(X,Y,td).
P(X,Y,fs) :- P(X,Y,fa), dom(X), dom(Y).
P(X,Y,fa) v P(Y,X,ta) :-  P(X,Y,ts), P(Y,X,fs), dom(X), dom(Y).
P(X,Y,tss) :- P(X,Y,ta), dom(X), dom(Y).
P(X,Y,tss) :- P(X,Y,td), dom(X), dom(Y), not P(X,Y,fa).
P(X,Y,fss) :- P(X,Y,fa), dom(X), dom(Y).
P(X,Y,fss) :- dom(X), dom(Y), not P(X,Y,td), not P(X,Y,ta).
:- p(X,Y,ta), p(X,Y,fa).
```

Repair subprogram annotations

Annotation	Atom	The tuple $P(\bar{a})$ is...
td	$P(\bar{a}, td)$	a fact of the database
fd	$P(\bar{a}, fd)$	not a fact in the database
ta	$P(\bar{a}, ta)$	advised to be made true
fa	$P(\bar{a}, fa)$	advised to be made false
ts	$P(\bar{a}, ts)$	true or becomes true
fs	$P(\bar{a}, fs)$	false or becomes false
tss	$P(\bar{a}, tss)$	it is true in the repair
fss	$P(\bar{a}, fss)$	it is false in the repair

Repair subprogram

```
P(X,Y,ts) :- P(X,Y,ta), dom(X), dom(Y).
P(X,Y,ts) :- P(X,Y,td), dom(X), dom(Y).
P(X,Y,fs) :- dom(X), dom(Y), not P(X,Y,td).
P(X,Y,fs) :- P(X,Y,fa), dom(X), dom(Y).

P(X,Y,fa) v P(Y,X,ta) :-  P(X,Y,ts), P(Y,X,fs), dom(X), dom(Y

P(X,Y,tss) :- P(X,Y,ta), dom(X), dom(Y).
P(X,Y,tss) :- P(X,Y,td), dom(X), dom(Y), not P(X,Y,fa).
P(X,Y,fss) :- P(X,Y,fa), dom(X), dom(Y).
P(X,Y,fss) :- dom(X), dom(Y), not P(X,Y,td), not P(X,Y,ta).
:- p(X,Y,ta), p(X,Y,fa).
```

Stable models obtained with DLV:

$\mathcal{M}_1^r =$ {dom(a), dom(c), v1(a), v2(a,c), P(a,c,nv1), P(a,c,v2), P(a,c,td), P(a,c,ts), auxv1(a), P(a,a,fs), P(c,a,fs), P(c,c,fs), P(a,a,fss), P(c,a,ta), P(c,c,fss), P(a,c,tss), P(c,a,ts), P(c,a,tss)}
$\equiv \{P(a,c), P(c,a)\}$

$\mathcal{M}_2^r =$ {dom(a),dom(c), v1(a), v2(a,c), P(a,c,nv1), P(a,c,v2), P(a,c,td), P(a,c,ts), auxv1(a), P(a,a,fs), P(c,a,fs), P(a,c,fs), P(c,c,fs), P(a,a,fss), P(c,a,fss), P(a,c,fss), P(c,c,fss), P(a,c,fa)} $\equiv \emptyset$

Repair programs specify exactly the repairs of an integration system for universal and simple referential ICs.

Computing Consistent Answers

Consistent answers $\bar{t}$ to a query posed to a global integration system $Q(\bar{x})$?

Methodology:

1. $Q(\cdots P(\bar{u}) \cdots \neg R(\bar{v}) \cdots) \mapsto$
$$Q' := Q(\cdots P(\bar{u}, \mathbf{tss}) \cdots R(\bar{v}, \mathbf{fss}) \cdots)$$

2. $Q'(\bar{x}) \mapsto (\Pi(Q'), Ans(\bar{X}))$
   - $\Pi(Q')$ is a query program
   - $Ans(\bar{X})$ is a query atom defined in $\Pi(Q')$

3. "Run" $\Pi := \Pi(Q') \cup \Pi(\mathcal{G}, IC)$

4. Collect ground atoms
$Ans(\bar{t}) \in \bigcap\{S \mid S \text{ is a stable model of } \Pi\}$

Example 3 (continued) Query Q: $P(x, y)$

1. Q': $P(x, y, \mathbf{tss})$

2. $\Pi(Q')$: $Ans(X, Y) \leftarrow P(X, Y, \mathbf{tss})$

3. $\Pi(\mathcal{G}_3, IC)$ as before; form $\Pi = \Pi(\mathcal{G}_3, IC) \cup \Pi(Q')$

4. The repairs corresponding to the stable models of the program Π extended with query atoms are

$$\overline{\mathcal{M}}_1^r = \mathcal{M}_1^r \cup \{Ans(a, c),\ Ans(c, a)\};$$
$$\overline{\mathcal{M}}_2^r = \mathcal{M}_2^r$$

5. No Ans atoms in common, then query has no consistent answers

Conclusions

- A general specification, under the LAV paradigm, of minimal global instances of an integration system of open, closed and clopen sources
- Certain Answers can be retrieved for monotone queries
- General approach to specifying the database repairs of a mediated integration system with open, closed and clopen sources under the LAV approach
- The methodology works for conjunctive view definitions, arbitrary universal ICs, simple referential ICs (in particular, no cycles), and queries expressed as Datalog^{not} programs

- Can be extended to the case of views defined using disjunctions of conjunctive queries

- Conditions over the views are being identified in order to be able to calculate *Certain* answers for any query.

Our computations are based on the current implementations of stable models.

However we are not interested in the repairs *per se*, we are interested in answers to queries.

Optimizations are being developed:

- Select the data from the sources that is relevant to answer the query
- Magic sets to use the query to guide the computation