Logic Programs for Consistently Querying Data Integration Systems

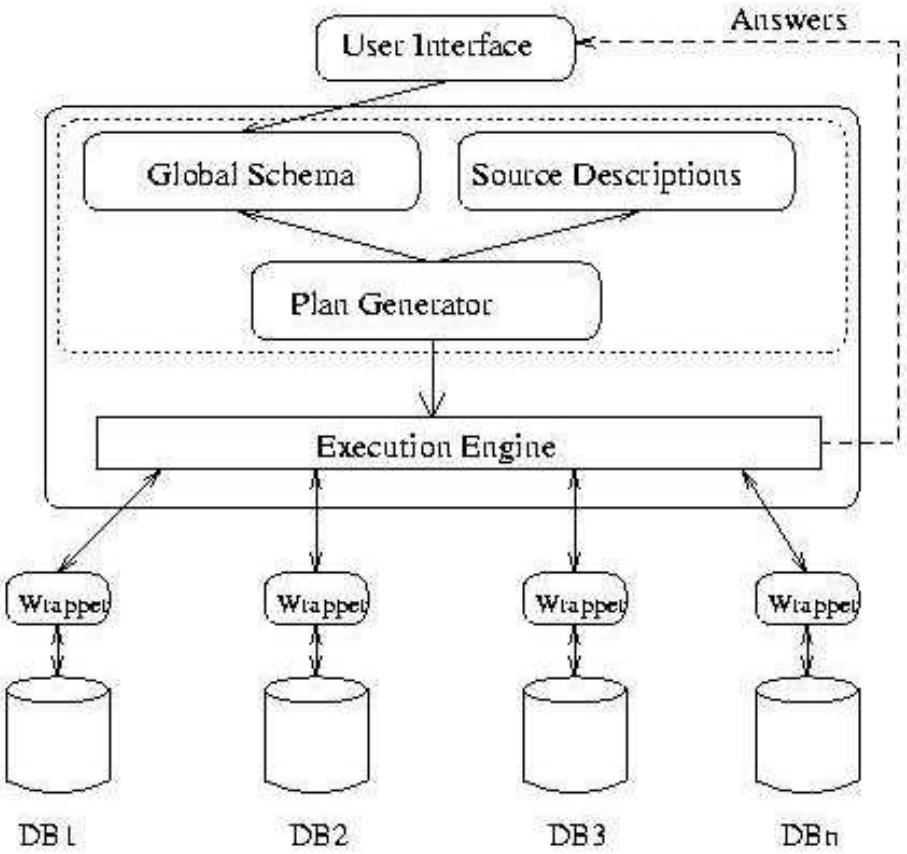
Loreto Bravo
Computer Science Department
Pontificia Universidad Católica de Chile
lbravo@ing.puc.cl

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
  - Query Answering over Minimal Instances
- 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

Virtual Global Instances?

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

Since the data sources are open, the legal instances of $\mathcal{G}$ are:

$$
Linst(\mathcal{G}) = \{ \quad D \text{ instance over } \mathcal{R} \mid v_i \subseteq \varphi_{V_i}(D), \quad i = 1, \ldots, n\}
$$

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 $v_1 = \{(a,b),(c,d)\}$

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

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

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

All supersets of D are legal global; but not the proper subsets of D

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
- Computing Consistent Answers
  - Logic Programming Specification of Repairs
  - Retrieving Consistent Answers
- Conclusions

Consistency

Example 1 (continued)

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

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

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

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

$$\begin{array}{ll} V_1(X,Z) \leftarrow P(X,Y), R(Y,Z) & \{V_1(a,b)\}, \\ V_2(X,Y) \leftarrow P(X,Y) & \{V_2(a,c)\} \end{array}$$

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

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

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

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

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

$$V_1(X,Z) \leftarrow P(X,Y), R(Y,Z) \qquad \{V_1(a,b)\},$$
$$V_2(X,Y) \leftarrow P(X,Y) \qquad \{V_2(a,c)\}$$

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

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: $\mathcal{G}_3$:

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

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

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

Presentation Outline

- ✓ Preliminaries
  - ✓ Virtual Data Integration Systems
- ✓ Minimal Instances
  - ✓ Consistency
  - ✓ Logic Programming Specification of Minimal Instances
- 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 $Mininst(\mathcal{G}_1) = \{\{(a,b),(c,d),(a,c),(d,e)\}\}$,
  $\mathcal{G}_1$ is inconsistent wrt $FD: X \rightarrow Y$
- $Repairs^{FD}(\mathcal{G}_1) = \{D^1, D^2\}$
  - $D^1 = \{(a,b),(c,d),(d,e)\}$
  - $D^2 = \{(c,d),(a,c),(d,e)\}$

- 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) $\mathcal{G}_3$:

$$\begin{aligned} V_1(X) &\leftarrow P(X,Y) && \{V_1(a)\}, \\ V_2(X,Y) &\leftarrow P(X,Y) && \{V_2(a,c)\} \end{aligned}$$

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 open integration system
- Certain Answers can be retrieved for monotone queries
- General approach to specifying the database repairs of a mediated integration system with open sources under the LAV approach
- The methodology works for conjunctive view definitions, arbitrary universal ICs, simple referential ICs, and queries expressed as Datalog^{not} programs
- Can be extended to the case of views defined using disjunctions of conjunctive queries (open sources)

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 the answers to queries.

Optimizations are being developed:

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

Logic Programs for Consistently Querying 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

Choice's Stable Version

The *choice models* can be obtained as the stable models of the associated program $SV(\Pi(\mathcal{G}))$, the stable version of $\Pi(\mathcal{G})$

- Replace each choice rule $r: \quad H \leftarrow B, choice((\bar{X}), (Y))$ by

$$H \leftarrow B, chosen_r(\bar{X}, Y)$$

- Introduce the new rules

$$chosen_r(\bar{X}, Y) \leftarrow \quad B, not \ diffChoice_r(\bar{X}, Y)$$
$$diffChoice_r(\bar{X}, Y) \leftarrow \quad chosen_r(\bar{X}, Y'), Y \neq Y'$$

stable models of $SV(\Pi(\mathcal{G}))$ $\equiv$ choice models of $\Pi(\mathcal{G})$

Revised Version

$\Pi(\mathcal{G})$ revisited, to capture **only** the minimal instances:

Annotation	Atom	The tuple $P(\bar{a})$ is...
t_d	$P(\bar{a}, t_d)$	an atom of the minimal legal instances
t_o	$P(\bar{a}, t_o)$	is an obligatory atom in all the minimal legal instances
v_i	$P(\bar{a}, v_i)$	an optional atom introduced to satisfy the openness of view v_i
nv_i	$P(\bar{a}, nv_i)$	an optional atom introduced to satisfy the openness of view that is not v_i

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

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

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

   a) For every P_k with no existential variables, the rules

   $$P_k(\bar{X}_k, t_o) \leftarrow V_i(\bar{X}) \quad k = 1, \ldots n$$

   b) For every set S_{ij} of predicates of the description's body that are related by common existential variables $(Z_1, \ldots, Z_m)$, the rules,

   $$add_{v_{ij}}(\bar{X}') \leftarrow V_i(\bar{X}),\ not\ aux_{v_{ij}}(\bar{X}')$$
   $$\text{where } \bar{X}' = \bar{X} \cap \{\bigcup_{P_k \in S_{ij}} X_k\}$$

$$aux_{v_{ij}}(\bar{X}') \leftarrow \bigwedge_{i=1}^{m} var_{v_1 Z_l}(\bar{X}_{Z_l})$$

$$var_{v_1 Z_l}(\bar{X}_{Z_l}) \leftarrow \bigwedge_{P_k \in S_{ij} \wedge Z_l \in \bar{X}_k} P_k(\bar{X}_k)$$
$$\text{where } \bar{X}_{Z_l} = \{\bigcup_{P_k \in S_{ij} \wedge Z_l \in \bar{X}_k} X_k\}$$
$$\text{for } l = 1, \ldots m$$

$$P_k(\bar{X}_k, v_{ij}) \leftarrow add_{v_{ij}}(\bar{X}'),\ \bigwedge_{Z_l \in (\bar{X}_k \setminus \bar{X}')} F_l(\bar{X}', Z_l),$$
$$\text{for } P_k \in S_{ij}.$$

4. For every predicate $F_l(\bar{X}', Z_l)$ introduced in 3.b., the rules,

$$F_l(\bar{X}', Z_l) \leftarrow add_{v_{ij} Z_l}(\bar{X}'), dom(Z_l), choice((\bar{X}'), (Z_l)).$$

$$add_{v_{ij} Z_l}(\bar{X}') \leftarrow add_{v_{ij}}(\bar{X}'),\ not\ aux_{v_{ij} Z_l}(\bar{X}')$$
$$\text{for } l = 1, \ldots m$$

$$aux_{v_{ij}Z_l}(\bar{X}') \leftarrow var_{v_1 Z_l}(\bar{X}_{Z_l}), \bigwedge_{Z_k \neq Z_l} F_k(\bar{X}', Z_k)$$
$$\text{for } l = 1, \ldots m$$

5. For every global relation $P(\bar{X})$ the rules

$$P(\bar{X}, nv_{ij}) \leftarrow P(\bar{X}, v_{hk}) \quad \text{for } \{(ij, hk) | P(\bar{X}) \in S_{ij} \text{ and } S_{hk}\}$$

$$P(\bar{X}, nv_{ij}) \leftarrow P(\bar{X}, t_o) \quad \text{for } \{(ij) | P(\bar{X}) \in S_{ij}\}$$

$$P(\bar{X}, t_d) \leftarrow P(\bar{X}, v_{ij}) \quad \text{for } \{(ij) | P(\bar{X}) \in S_{ij}\}$$

$$P(\bar{X}, t_d) \leftarrow P(\bar{X}, t_o)$$

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

Logic Specification of Repairs

Annotation	Atom	The tuple $P(\bar{a})$ is...
t_d	$P(\bar{a}, t_d)$	a fact of the database
f_d	$P(\bar{a}, f_d)$	not a fact in the database
t_a	$P(\bar{a}, t_a)$	adviced to be made true
f_a	$P(\bar{a}, f_a)$	adviced to be made false
$t^\star$	$P(\bar{a}, t^\star)$	true or becomes true
$f^\star$	$P(\bar{a}, f^\star)$	false or becomes false
$t^{\star\star}$	$P(\bar{a}, t^{\star\star})$	it is true in the repair
$f^{\star\star}$	$P(\bar{a}, f^{\star\star})$	it is false in the repair

Example 6: A full inclusion dependency

$$IC: \ \forall x(\neg P(x) \vee Q(x))$$

An inconsistent database instance $DB = \{P(a)\}$

Database atoms become facts of the *repair* program:

1. $P(a, \mathbf{t_d}) \leftarrow$

Whatever was true (false) or becomes true (false), gets annotated with $\mathbf{t}^\star$ ($\mathbf{f}^\star$):

2. $P(x, \mathbf{t}^\star) \leftarrow P(x, \mathbf{t_a})$.
   $P(x, \mathbf{t}^\star) \leftarrow P(x, \mathbf{t_d})$.
   $P(x, \mathbf{f}^\star) \leftarrow P(x, \mathbf{f_a})$.
   $P(x, \mathbf{f}^\star) \leftarrow \ not \ P(x, \mathbf{t_d})$.
   ... the same for Q ...

3. $P(x, \mathbf{f_a}) \vee Q(x, \mathbf{t_a}) \leftarrow P(x, \mathbf{t}^\star), Q(x, \mathbf{f}^\star).$

   One rule per IC; it says how to repair the IC

   Having passed to annotations $\mathbf{t}^\star$ and $\mathbf{f}^\star$ keeps repairing the IC if it becomes violated due to the repair of a different IC

The repair process must be coherent, this is captured by the denial program constraints

4. $\leftarrow P(\bar{x}, \mathbf{t_a}), P(\bar{x}, \mathbf{f_a}).$
   $\leftarrow Q(\bar{x}, \mathbf{t_a}), Q(\bar{x}, \mathbf{f_a}).$

Annotations constants $\mathbf{t}^{\star\star}$ and $\mathbf{f}^{\star\star}$ are used to read off the literals that are inside (outside) a repair

5. $P(x, \mathbf{t}^{\star\star}) \leftarrow P(x, \mathbf{t_a})$.

   $P(x, \mathbf{t}^{\star\star}) \leftarrow P(x, \mathbf{t_d}),\ not\ P(x, \mathbf{f_a})$.

   $P(x, \mathbf{f}^{\star\star}) \leftarrow P(x, \mathbf{f_a})$.

   $P(x, \mathbf{f}^{\star\star}) \leftarrow\ not\ P(x, \mathbf{t_d}),\ not\ P(x, \mathbf{t_a})$.

   ... the same for Q ...

   These rules are used to interpret the models as database repairs

The program has two stable models:

- $\{P(a, \mathbf{t_d}), P(a, \mathbf{t}^\star), Q(a, \mathbf{f}^\star), Q(a, \mathbf{t_a}), \underline{P(a, \mathbf{t}^{\star\star})}, Q(a, \mathbf{t}^\star), \underline{Q(a, \mathbf{t}^{\star\star})}\}$

  That corresponds to $DB_1 = \{P(a), Q(a)\}$

- $\{P(a, \mathbf{t_d}), P(a, \mathbf{t}^\star), P(a, \mathbf{f}^\star), Q(a, \mathbf{f}^\star), \underline{P(a, \mathbf{f}^{\star\star})}, \underline{Q(a, \mathbf{f}^{\star\star})}, P(a, \mathbf{f_a})\}$

  That corresponds to $DB_2 = \emptyset$

Theorem: For universal IC, there is a one to one correspondence between the stable models of the *repair* program and the repairs of DB wrt IC

Computing Minimal Answers

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

Methodology:

1. $Q(\cdots P(\bar{u}) \cdots \neg R(\bar{v}) \cdots) \;\; \mapsto$

   $$Q' := \; Q(\cdots P(\bar{u}, \mathbf{t_d}) \cdots \; \neg R(\bar{v}, \mathbf{t_d}) \cdots)$$

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" $\Pi := \; \Pi(Q') \; \cup \; \Pi(\mathcal{G})$

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': $P(x, y, \mathbf{t_d})$

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

3. $\Pi = \Pi(\mathcal{G}_3) \cup \Pi(Q')$

4. The stable model of the program Π extended with query atoms is:

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

5. $Ans(a, c)$ is the minimal answer to the query.