Carleton
UNIVERSITY

Consistent Query Answering in Virtual Data Integration

Leopoldo Bertossi

Carleton University
School of Computer Science
Ottawa, Canada

bertossi@scs.carleton.ca
www.scs.carleton.ca/~bertossi

Simon Fraser University, March 2005.

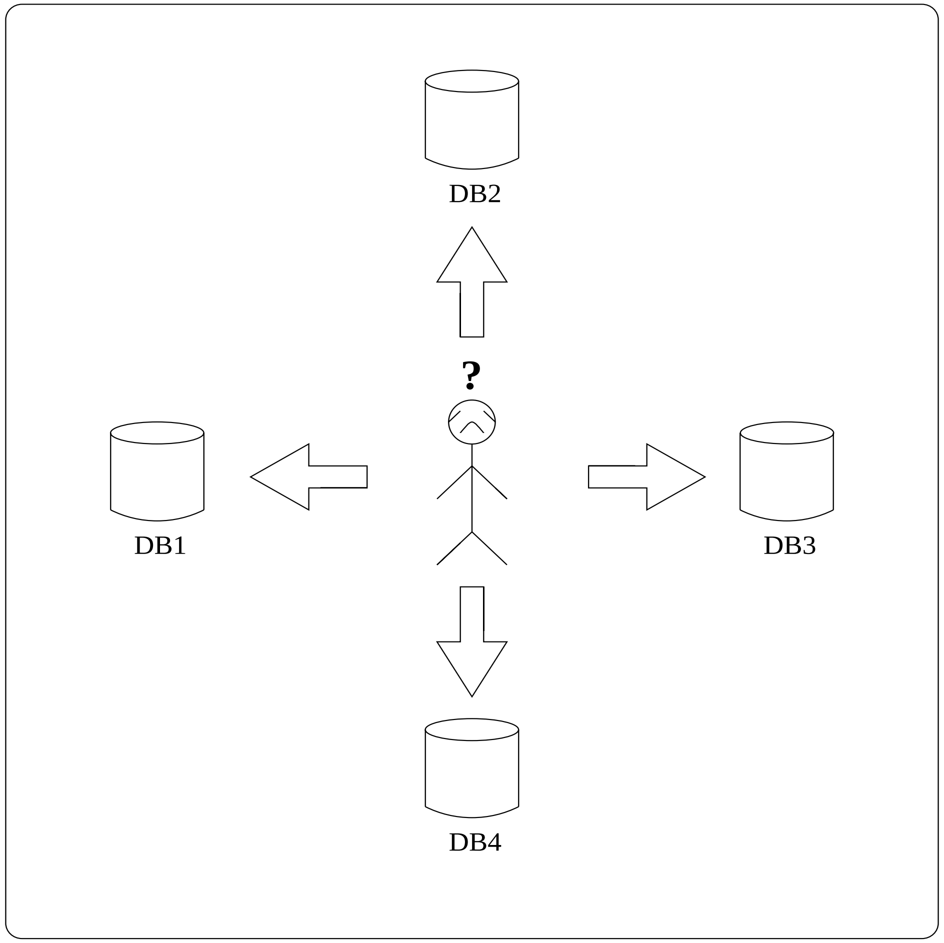
Virtual Data Integration

Number of available on-line information sources has increased dramatically:

How can users confront such a large and increasing number of information sources?

Interacting with each of the sources by means of queries?

- Considering all available sources?
- Selecting only those to be queried?
- Querying the relevant sources on an individual basis?
- Handcraft the combination of results from different sources?


DB2
?
DB1
DB3
DB4

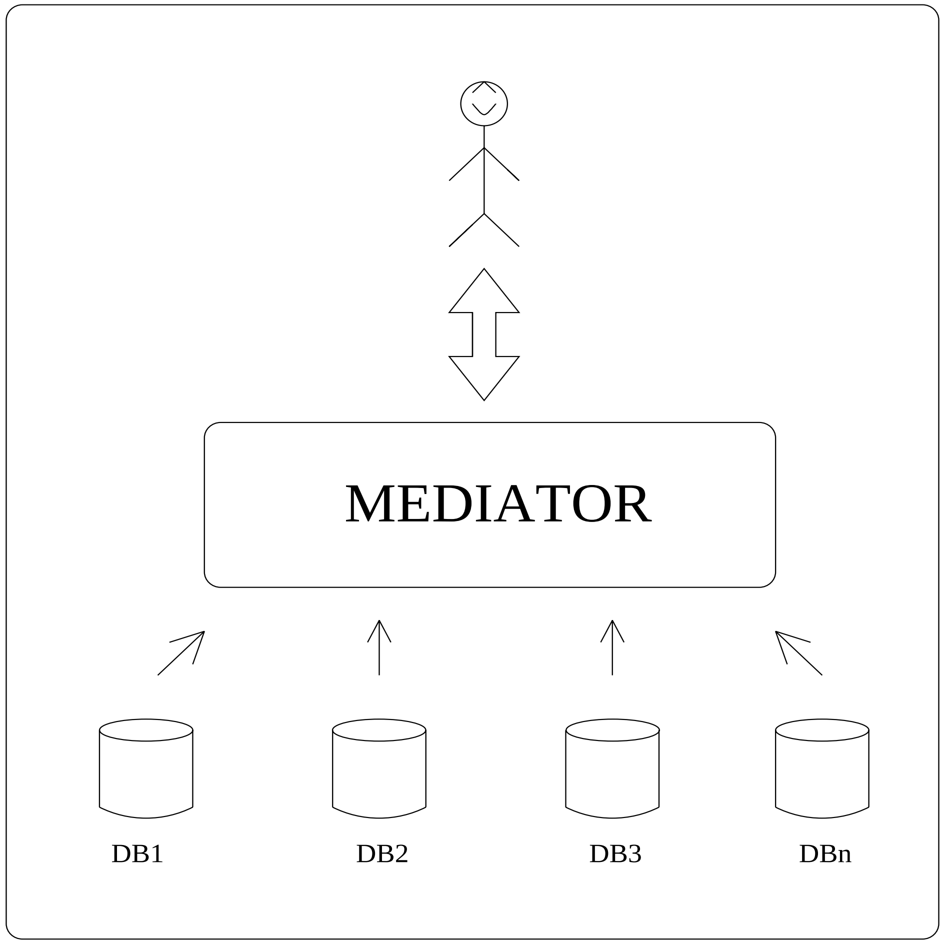
This can be a long, tedious, complex and error prone process

Independently of other "intrinsic" issues like:

- Incomplete data
- Redundant data
- Inconsistent data
- Restricted query languages/patterns to access sources

One way to attack the problem: Virtual integration of sources via a mediator

That is, a software system for an architecture that offers a common query interface to a set of heterogeneous information sources


MEDIATOR
DB1
DB2
DB3
DBn

Main features of a mediator based integration system (MIS)

- Sources are mutually independent
- Sources do not necessarily cooperate with each other
- Set and number of participating sources may be flexible and open
- System allows sources to get in and out
- Data sources may participate in different mediated systems
- Interaction with the system via queries
- Mediator combines and arranges for the user the query results

- Update operations of any kind are not allowed

  Only at the individual database source, but not through the MIS

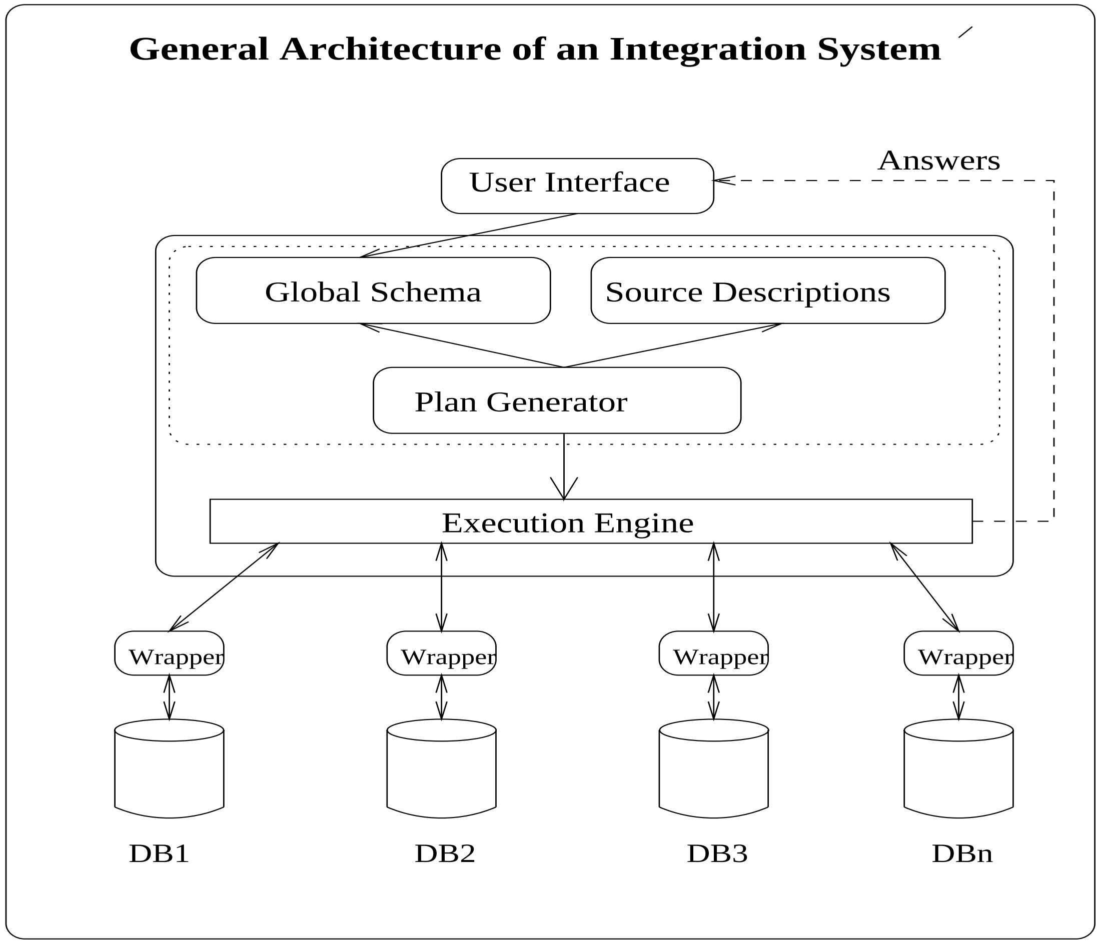
- Data is kept in the local, individual sources, and extracted at the mediator's request

- System is responsible of solving problems of data ...

  - redundancy: to avoid unnecessary computations
  - complementarity: data of the same kind may be spread through different sources and has to be detected and combined
  - inconsistency: two sources, independently, may be consistent, but taken together, possibly not

    E.g. Same ID card number may be assigned to different people in different sources

General Architecture



Main components of a MIS

There is a global schema used to present and export the data from the MIS, e.g. a relational schema

Like in a usual relational DB, from the user point of view, but data is not stored in “tables” of the global schema, but in the sources

The DB “instance” corresponding to the global schema is virtual

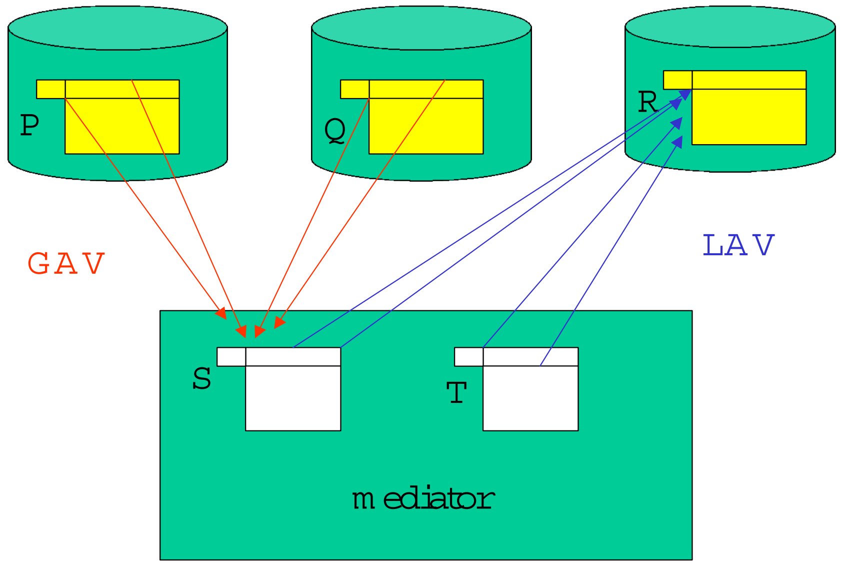
User poses queries in terms of the relations in the global schema

Relationship between the global schema and the data sources (and their local schemas) is specified at the mediator level

Description of the Sources:

- Mediator needs to know what is available in the sources and how that data relates to the global schema
- The sources are described by means of a set of logical formulas; like those used to express queries and define views
- Those formulas define the mappings between the global schema and the local schemas
- Two main approaches:
  - Global-as-View (GAV): Relations in the global schema are described as views over the tables in the local schemas

- Local-as-View (LAV): Relations in the local schemas (at the source level) are described as views over the global schema



Plan Generator:

- Gets a user query in terms of global relations
- Uses the source descriptions and rewrites the query as a query plan

  Which involves a set of queries expressed in terms of local relations
- Rewriting process depends on LAV or GAV approach
- Query plan includes a specification of how to combine the results from the local sources

LAV

Each local relation is described as a view (as a query expression) in terms of the global relations

LAV is more flexible wrt participation of sources; but more complex for query answering

Example: Sources:

$$\begin{aligned} S_1: \quad V_1(Title, \; Year, Director) \leftarrow \; & Movie(Title, Year, Director, Genre), \\ & American(Director), Genre = comedy, \\ & Year \geq 1960. \end{aligned}$$

S_1: comedies, after 1960, with American directors and their years

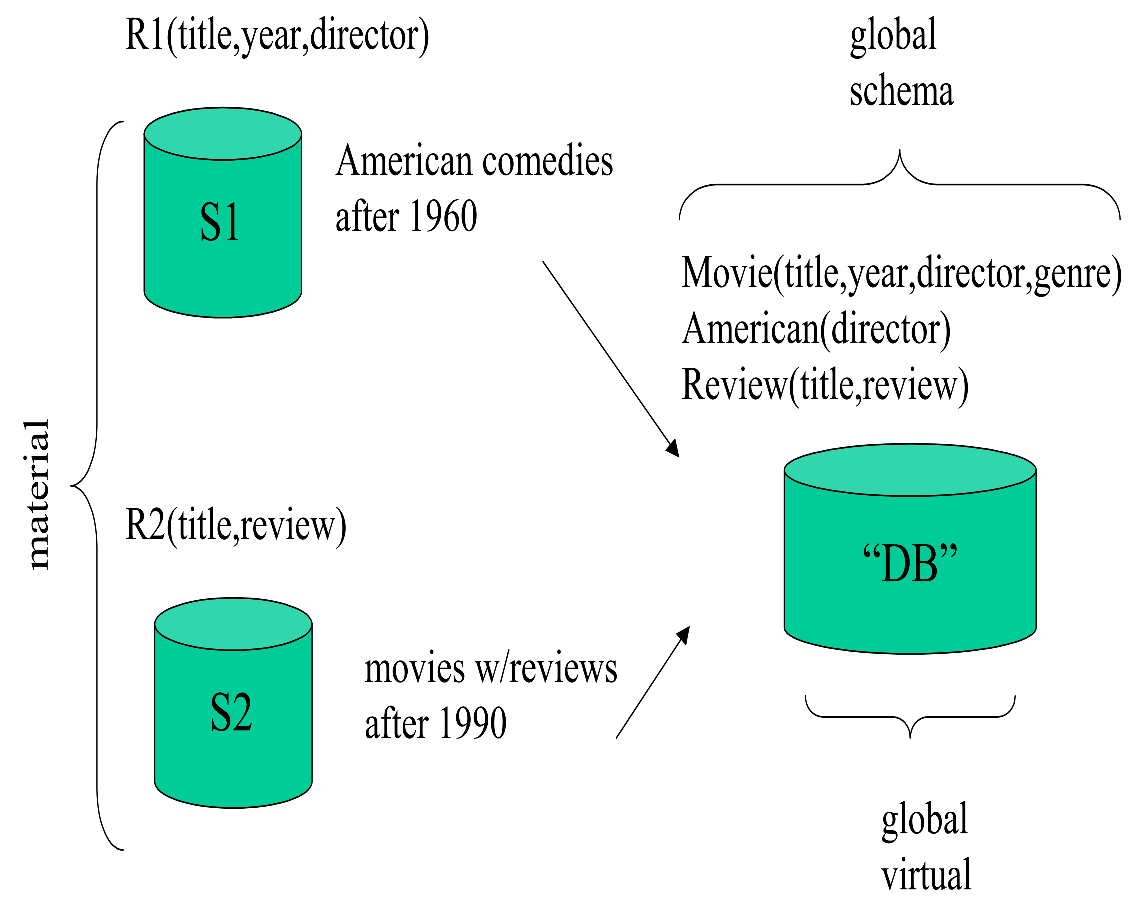
$$\begin{aligned} S_2: \quad V_2(\mathit{Title},\ \mathit{Review}) \leftarrow & \\ & \mathit{Movie}(\mathit{Title}, \mathit{Year}, \mathit{Director}, \mathit{Genre}), \\ & \mathit{Review}(\mathit{Title}, \mathit{Review}), \mathit{Year} \geq 1990. \end{aligned}$$

S_2: movies after 1990 with their reviews, but no directors

Sources defined as conjunctive queries (views) with built-ins

The global schema $\mathcal{G}$:

$$\begin{aligned} & \mathit{Movie}(\mathit{Title}, \mathit{Year}, \mathit{Director}, \mathit{Genre}), \\ & \mathit{Review}(\mathit{Title}, \mathit{Review}),\ \ \mathit{American}(\mathit{Director}) \end{aligned}$$


R1(title,year,director)
S1
American comedies after 1960
material
R2(title,review)
S2
movies w/reviews after 1990
global schema
Movie(title,year,director,genre)
American(director)
Review(title,review)
"DB"
global virtual

Definition of each source does not depend on other sources

From the perspective of S_2, there could be other sources containing information about comedies after 1990 with their reviews

In this sense, information in S_2 could be "incomplete" wrt what $\mathcal{G}$ "contains" (or might contain), containing only a part of the information of the same kind in the global system

Query posed to $\mathcal{G}$: "Comedies with their reviews produced since 1950?"

$$\begin{aligned} &Ans(Title,\ Review) \leftarrow \\ &\qquad Movie(Title, Year, Director, comedy), \\ &\qquad Review(Title, Review), Year \geq 1950. \end{aligned}$$

Query expressed as usual, in terms of the relations in the global schema only

Not possible to obtain answers by a simple and direct computation of the RHS of the query: Information is in the sources, now, views ...

A plan is a rewriting of the query as a set of queries to the sources and a prescription on how to combine their answers

A possible query plan for our query (more on this later):

$$Ans'(Title, Review) \leftarrow V_1(Title, Year, Director), V_2(Title, Review).$$

Query was rewritten in terms of the views ...

Can be computed:

1. Extract tuples from V_1
2. Extract the tuples from V_2
3. At the mediator level, compute the join via $Title$ and project

Due to the limited contents of the sources, we obtain comedies by American directors with their reviews filmed after 1990

The Semantics of a Data Integration System

What kind of answers to global queries are we obtaining with a query plan? How good is the plan generation mechanism?

This can be answered once we define the semantics of a virtual data integration system

We assume:

- The LAV approach is adopted
- The source relations are declared as open:

  Given extensions for the local sources, the global relations can be materialized in different ways, still satisfying the source descriptions; so different global instances are possible

Let v_1, v_2 be the given extensions for the local relations V_1, V_2

A global (material) instance D is legal if the view definitions applied to it compute extensions $V_1(D), V_2(D)$ such that $v_1 \subseteq V_1(D)$ and $v_2 \subseteq V_2(D)$

That is, each source relation contains a possibly proper subset of the data of its kind in the global system

Given a global query $Q(\bar{x})$, the intended answers to Q from the system $\mathcal{G}$ are the answers that can be obtained from all the legal instances for $\mathcal{G}$

$Certain_{\mathcal{G}}(Q) := \{\bar{t} \mid$ for every legal instance D it holds $D \models Q[\bar{t}]\}$

Example: Global system $\mathcal{G}_1$ with sources

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

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

$D = \{R(a,b), R(c,d), R(a,c), R(d,e)\}$ and its supersets are the legal instances

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

$$\mathit{Certain}_{\mathcal{G}_1}(Q) = \{(a,b),(c,d),(a,c),(d,e)\}$$

Certain answers to a query are true in all the legal instances

Consistency in Virtual Data Integration

Usually one assumes that certain ICs hold at the global level; and they are used in the generation of query plans

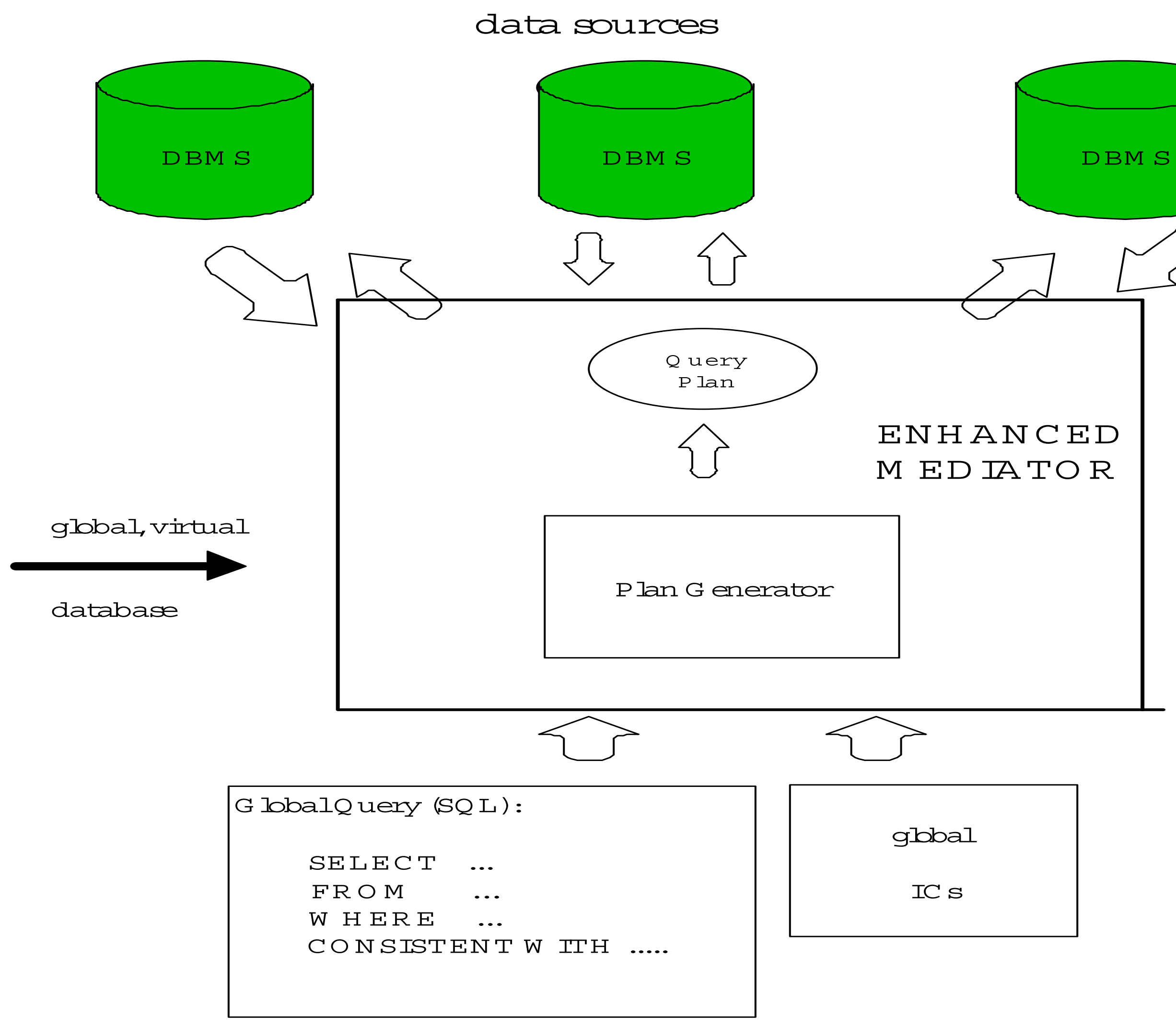
How can we be sure that those global ICs hold?

They are not maintained at the global level

Most likely they are not fully satisfied

The goal is to retrieve answers to global queries from the virtual integration system that are "consistent with the ICs"

We need a characterization of consistent answers and a mechanism to obtain them ...


data sources
DBMS
DBMS
DBMS
Query Plan
ENHANCED MEDIATOR
global, virtual
database
Plan Generator
Global Query (SQL):
SELECT ...
FROM ...
WHERE ...
CONSISTENT WITH .....
global
ICs

Example: (continued) Global system $\mathcal{G}_1$

What if we had a global functional dependency $R\colon X \rightarrow Y$?

(local FDs $V_1\colon X \rightarrow Y$, $V_2\colon X \rightarrow Y$ satisfied in the sources)

Global FD not satisfied by $D = \{(a, b), (c, d), (a, c), (d, e)\}$
(nor by its supersets)

From the certain answers to the query Q: $R(X, Y)?$, i.e. from

$$Certain_{\mathcal{G}_1}(Q) = \{(a, b), (c, d), (a, c), (d, e)\}$$

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

Minimal Legal Instances and Consistent Answers

There are algorithms for generating plans to obtain the certain answers (with some limitations)

Not much for obtaining consistent answers

Here we do both, in stages ...

First concentrating on the minimal legal instances of a virtual systems, i.e. those that do not properly contain any other legal instance

- Minimal legal instances do not contain unnecessary information; that could, unnecessarily, violate global ICs

In the example, $D = \{R(a,b), R(c,d), R(a,c), R(d,e)\}$ is the only minimal instance

The minimal answers to a query are those that can be obtained from every minimal legal instance:

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

For monotone queries they coincide

By definition, consistent answers to a global query wrt IC are those obtained from all the repairs of all the minimal legal instances wrt IC

(Bertossi, Chomicki, Cortes, Gutierrez; FQAS 02)

In the example:

- The only minimal legal instance

$$D = \{R(a,b), R(c,d), R(a,c), R(d,e)\}$$

  violates the FD $R\colon X \rightarrow Y$

- Its repairs wrt FD are

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

  A repair of an instance D wrt a set of ICs is an instance D' that satisfies the ICs and minimally differs from D (under set inclusion, considering a DB as a set of facts)

- Consistent answers to query Q: $R(X,Y)$?

  Only $\{(c,d), (d,e)\}$

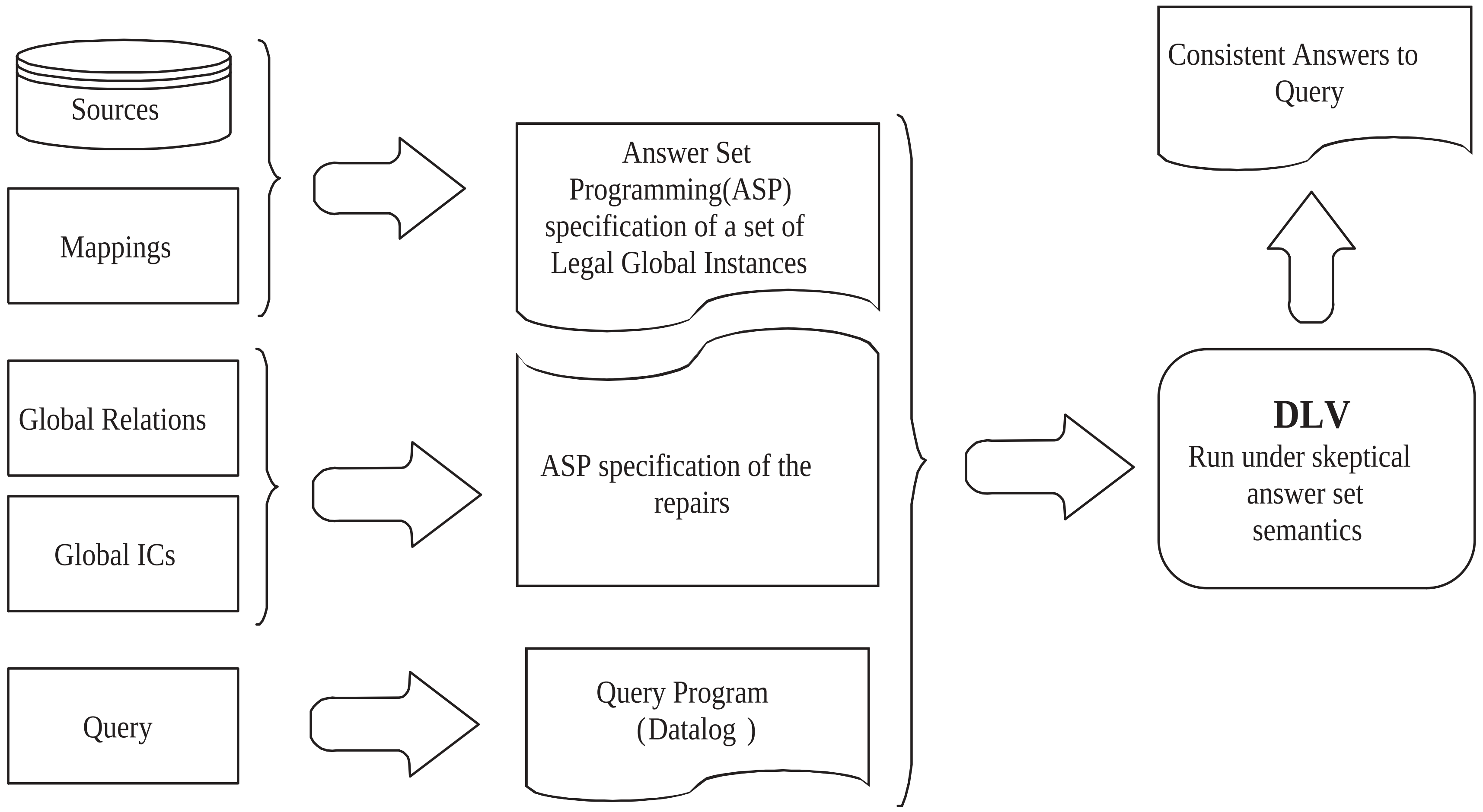
Computing consistent answers? (Idea)

(Bravo, Bertossi; IJCAI 03)

- Answer set programming (ASP) based specification of minimal instances of a virtual data integration system
- ASP based specification of repairs of minimal instances (we saw how to do this, e.g. programs with annotation constants)
- Global query in Datalog (or its extensions) to be answered consistently
- Run combined programs above under skeptical answer set semantics (stable model semantics)

- Methodology works for first-order queries (and Datalog extensions), and universal ICs combined with (acyclic) referential ICs

- Important subproduct: A methodology to compute certain answers to monotone queries


Sources
Mappings
Global Relations
Global ICs
Query
Answer Set Programming(ASP) specification of a set of Legal Global Instances
ASP specification of the repairs
Query Program (Datalog)
DLV
Run under skeptical answer set semantics
Consistent Answers to Query

Specifying Minimal Instances

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

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

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

Specification of minimal instances: $\Pi(\mathcal{G}_2)$

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

Inspired by inverse rules algorithm for computing certain answers (Duschka, Genesereth, Levy; JLP 00)

Now the global relations are being defined in terms of the local relations

F_1 is a functional predicate, whose functionality on the second argument is imposed by the choice operator

$choice((X,Y)),(Z))$: non-deterministically chooses a unique value for Z for each combination of values for X,Y
(Giannotti, Pedreschi, Sacca, Zaniolo; DOOD 91)

Models of $\Pi(\mathcal{G}_2)$ are the choice models, but the program can be transformed into one with stable models semantics

$$M_b = \{dom(a), \ldots, V_1(a,b), V_2(a,c), \underline{P(a,c)}, \mathit{diffChoice}_1(a,b,a), chosen_1(a,b,b), \mathit{diffChoice}_1(a,b,c), F_1(a,b,b), \underline{R(b,b)}, \underline{P(a,b)}\}$$

$$M_a = \{dom(a), \ldots, V_1(a,b), V_2(a,c), \underline{P(a,c)}, chosen_1(a,b,a), \mathit{diffChoice}_1(a,b,b), \mathit{diffChoice}_1(a,b,c), F_1(a,b,a), \underline{R(a,b)}, \underline{P(a,a)}\}$$

$$M_c = \{dom(a), \ldots, V_1(a,b), V_2(a,c), \underline{P(a,c)}, \mathit{diffChoice}_1(a,b,a), \mathit{diffChoice}_1(a,b,b), chosen_1(a,b,c), F_1(a,b,c), \underline{R(c,b)}\}$$

$\ldots$

Here: 1-1 correspondence between stable models and minimal instances of $\mathcal{G}_2$

In general:

- The minimal instances are all among the models of the program
- All the models of the program are (determine) legal instances
- In consequence, the program can be used to compute all the certain answers to monotone queries
- The program can be refined to compute all and only the minimal legal instances

Example: $\mathcal{G}_3$

$V_1(X)$	$\leftarrow$	$P(X,Y)$	$\{v_1(a)\}$	open
$V_2(X,Y)$	$\leftarrow$	$P(X,Y)$	$\{v_2(a,c)\}$	open

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

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

More legal instances (or stable models) than minimal instances

As V_2 is open, it forces $P(a,c)$ to be in all legal instances

What makes the same condition on V_1 automatically satisfied (no other values for Y needed)

Choice operator, as used above, may still choose other values $z \in \mathcal{D}$

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

A refined version of $\Pi(\mathcal{G})$ detects in which cases it is necessary to use the function predicates

$$F_1(X,Y) \leftarrow add\_V_1(X), dom(X), choice((X),Y)$$

where $add\_V_1(X)$ is true only when the openness of V_1 is not satisfied through other views (which has to be specified)

$$\text{stable models of } \Pi(\mathcal{G}) \equiv MinInst(\mathcal{G})$$

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

More general than any other algorithm for LAV ...

Specification programs can be produced for case where sources may be closed or clopen

Minimal Instances: Mixed Source Labels

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

$$
\begin{array}{lll}
V_1(X,Z) \leftarrow P(X,Y), R(Y,Z) & \{v_1(a,b)\} & \text{open} \\
V_2(X,Y) \leftarrow P(X,Y) & \{v_2(a,c)\} & \text{clopen}
\end{array}
$$

Like $\mathcal{G}_2$, that had the same sources but open; before:

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

Now second source restricts P to (a,c), so in this case:

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

Closure condition restricts the tuples in the legal instances, but does not add new tuples

In general: $\Pi(\mathcal{G})^{mix}$ built as follows:

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

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

It says it is not possible for a model to satisfy the conjunction at the RHS of the arrow; then it filters out models of the program

It captures closure of the source that for legal instances D:

$V(D) = P_1^D(\bar{X}_1), \ldots, P_n^D(\bar{X}_n) \subseteq v =$ set of facts for V in $\Pi(\mathcal{G})$

If the simple version of $\Pi(\mathcal{G})$ is considered, it holds:

$MinInst(\mathcal{G}) \subseteq$ stable models of $\Pi(\mathcal{G})^{mix} \subseteq LegInst(\mathcal{G})$

If the refined version of $\Pi(\mathcal{G})$ is used we have:

stable models of $\Pi(\mathcal{G})^{mix} \equiv MinInst(\mathcal{G})$

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

$$\begin{array}{lll} V_1(X,Z) \leftarrow P(X,Y), R(Y,Z) & \{v_1(a,b)\} & \text{open} \\ V_2(X,Y) \ \leftarrow \ P(X,Y) & \{v_2(a,c)\} & \text{clopen} \end{array}$$

$\Pi(\mathcal{G}_4)^{mix}$: (simple version, refined not needed here)

$$\begin{array}{l} dom(a).,\ \ dom(b).,\ \ dom(c).,\ \ \dots, V_1(a,b).,\ \ V_2(a,c). \\ P(X,Z) \leftarrow V_1(X,Y), F_1((X,Y),Z) \\ R(Z,Y) \leftarrow V_1(X,Y), F_1((X,Y),Z) \\ P(X,Y) \leftarrow V_2(X,Y) \\ F_1((X,Y),Z) \leftarrow V_1(X,Y), dom(Z), choice((X,Y),(Z)) \\ \leftarrow P(X,Y),\ not\ V_2(X,Y) \end{array}$$

(V_2 stores the source contents, and P is used to compute the view extension, so we are requiring that $P \subseteq V_2$)

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

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

Corresponding, as expected, to the fact that

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

Relation between answers obtained from different types of queries and programs (the same applies to the mixed case):

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

Repairs and Consistent Answers

Intuitively, consistent answers are invariant under minimal restorations of consistency

Definition based on notion of repair

A repair is a global instance that minimally differs from a minimal legal instance

Example: Global system $\mathcal{G}_1$ (extended)

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

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

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

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

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

$Repairs^{FD}(\mathcal{G}_1)$:

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

(we relax legality for repairs)

Queries:

- $Q(X,Y): \quad R(X,Y)?$

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

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

  a is a consistent answer, together with c, d

Specification of Repairs

So far: specification of minimal instances of an integration system; they can be inconsistent

Now, specify their repairs

Idea: Combine the program that specifies the minimal instances with the "repair program" that specifies the repairs of each minimal instance

We use the techniques developed for specifying repairs of single inconsistent databases (Barcelo, Bertossi; PADL 03)

$\Pi(\mathcal{G}, IC)$ is the program with annotations constants that specifies the repairs of an integration system $\mathcal{G}$ wrt IC

Example: $\mathcal{G}_3$

$V_1(X)$	$\leftarrow$	$P(X,Y)$	$\{v_1(a)\}$	open
$V_2(X,Y)$	$\leftarrow$	$P(X,Y)$	$\{v_2(a,c)\}$	open

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

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

In the program, the $P(\cdot,\cdot,\mathbf{t_d})$ are essentially the output of the first layer, that specifies the minimal instances

They are taken by the second layer specifying the repairs, whose output are the $P(\cdot,\cdot,\mathbf{t}^{\star\star})$

A third layer can be the query program, that uses the $P(\cdot,\cdot,\mathbf{t}^{\star\star})$

Repair Program: (as used by DLV) First Layer is refined program for minimal
instances (w/standard version of Choice)

$$\begin{array}{rcl}
dom(a).\ \ dom(c). & & V_1(a).\ \ V_2(a,c). \\
P(X,Y,\mathbf{t_d}) & \leftarrow & P(X,Y,v_1) \\
P(X,Y,\mathbf{t_d}) & \leftarrow & P(X,Y,t_o) \\
P(X,Y,nv_1) & \leftarrow & P(X,Y,t_o) \\
addV_1(X) & \leftarrow & V_1(X),\ \ not\ \ auxV_1(X) \\
auxV_1(X) & \leftarrow & P(X,Z,nv_1) \\
fz(X,Z) & \leftarrow & addV_1(X), dom(Z), chosenv1z(X,Z) \\
chosenv1z(X,Z) & \leftarrow & addV_1(X), dom(Z),\ \ not\ \ diffchoicev1z(X,Z) \\
diffchoicev1z(X,Z) & \leftarrow & chosenv1z(X,U), dom(Z), U \neq Z \\
P(X,Z,v_1) & \leftarrow & addV_1(X), fz(X,Z) \\
P(X,Y,t_o) & \leftarrow & V_2(X,Y)
\end{array}$$

Second Layer computes the repairs

$$
\begin{array}{rcl}
P(X,Y,t^\star) & \leftarrow & P(X,Y,t_a) \\
P(X,Y,t^\star) & \leftarrow & P(X,Y,\mathbf{t_d}) \\
P(X,Y,f_a) & \vee & P(Y,X,t_a) \leftarrow P(X,Y,t^\star), P(Y,X,f_a) \\
P(X,Y,f_a) & \vee & P(Y,X,t_a) \leftarrow P(X,Y,t^\star), \\
 & & \qquad\qquad\qquad\quad \mathit{not}\ P(Y,X,\mathbf{t_d}) \\
P(X,Y,\mathbf{t^{\star\star}}) & \leftarrow & P(X,Y,t_a) \\
P(X,Y,\mathbf{t^{\star\star}}) & \leftarrow & P(X,Y,\mathbf{t_d}),\ \mathit{not}\ P(X,Y,f_a) \\
 & \leftarrow & P(X,Y,t_a), P(X,Y,f_a).
\end{array}
$$

Disjunctive rules are crucial; they repair: If a violation of IC occurs (c.f. body), then either delete or insert tuples (c.f. head)

Stable models obtained with DLV: (parts of them)

$\mathcal{M}^r_1 =$ {dom(a), dom(c), v1(a), v2(a,c), P(a,c,nv1), P(a,c,v2), P(a,c,td), P(a,c,t*), auxv1(a), P(c,a,ta), P(a,c,t**), P(c,a,t*), P(c,a,t**)}
$\equiv \{P(a,c), P(c,a)\}$

$\mathcal{M}^r_2 =$ {dom(a),dom(c), v1(a), v2(a,c), P(a,c,nv1), P(a,c,v2), P(a,c,td), P(a,c,t*), auxv1(a), P(a,c,fa)} $\equiv \emptyset$

Repair programs specify exactly the repairs of an integration system for universal and simple (non cyclic) referential ICs

Computing Consistent Answers

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

Methodology:

1. $Q(\cdots P(\bar{u}) \cdots) \longmapsto Q' := Q(\cdots P(\bar{u}, \mathbf{t}^{\star\star}) \cdots)$
2. $Q'(\bar{x}) \longmapsto (\Pi(Q'), Ans(\bar{X}))$
   (Lloyd-Topor transformation)
   - $\Pi(Q')$ is a query program (the Third Layer)
   - $Ans(\bar{X})$ is a query atom defined in $\Pi(Q')$
3. "Run" $\Pi := \Pi(Q') \cup \Pi(\mathcal{G}, IC)$
4. Collect ground atoms
   $Ans(\bar{t}) \in \bigcap\{S \mid S$ is a stable model of $\Pi\}$

Example: $\mathcal{G}_3$ Query $Q:\ P(x,y)$

1. $Q':\ P(x,y,\mathbf{t}^{\star\star})$

2. $\Pi(Q'):\ Ans(X,Y) \leftarrow P(X,Y,\mathbf{t}^{\star\star})$

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

4. Repairs corresponding to the stable models of the program Π become extended with query atoms

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

5. No Ans atoms in common, then query has no consistent answers (as expected)

Example: Repair program for $\mathcal{G}_1$ with the FD

```
domd(a).    domd(b).    domd(c).                        %begin subprogram for minimal instances
domd(d).    domd(e).    v1(a,b).
v1(c,d).    v2(c,a).    v2(e,d).

R(X,Y,td) :- v1(X,Y).
R(Y,X,td) :- v2(X,Y).

R(X,Y,ts) :-  R(X,Y,ta), domd(X), domd(Y).              %begin repair subprogram
R(X,Y,ts) :-  R(X,Y,td), domd(X), domd(Y).
R(X,Y,fs) :- domd(X), domd(Y), not R(X,Y,td).
R(X,Y,fs) :-  R(X,Y,fa), domd(X), domd(Y).

R(X,Y,fa) v R(X,Z,fa) :-  R(X,Y,ts), R(X,Z,ts), Y!=Z, domd(X),domd(Y),domd(Z).

R(X,Y,tss) :- R(X,Y,ta), domd(X), domd(Y).
R(X,Y,tss) :- R(X,Y,td), domd(X), domd(Y), not R(X,Y,fa).
:- R(X,Y,fa), R(X,Y,ta).

Ans(X,Y) \la  R(X,Y,tss).                                %query subprogram
```

The consistent answers obtained for the query Q: $R(X,Y)$, correspond to the expected ones, i.e., $\{(c,d),(d,e)\}$

Conclusions

See tomorrow ...