VIRTUAL DATA INTEGRATION

Dr. Leopoldo Bertossi

Carleton University
School of Computer Science
bertossi@scs.carleton.ca

www.scs.carleton.ca/~bertossi

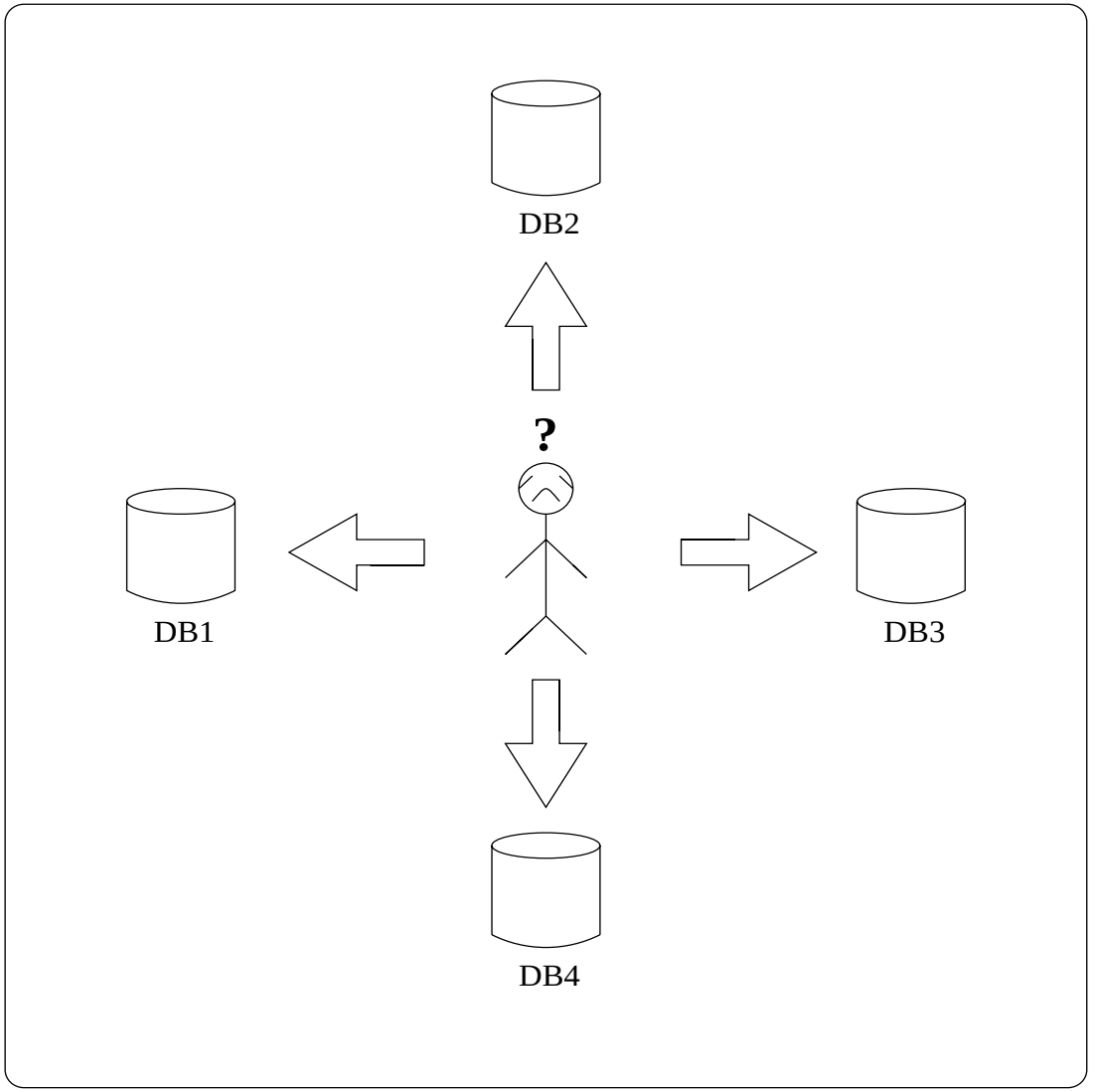
PART I: Introduction, Basic Notions, Problems ...

Mediators for Data Integration Systems

There is a dramatically increasing number of available information sources, many of them on-line :

- Organizational databases
- Libraries, catalogues, etc.
- Non structured and semi structured documents, e.g. in the WWW
- Data repositories, e.g. scientific (genome databases, etc.), also distributed, ...

How can a user interact simultaneously with so many data sources?



An alternative: bring all the data that might be needed into a single, huge, new database

Actually a *data warehouse*, that physically integrates all the information

Sometimes this can be done, however

- Difficult to design
  - What views should be materialized?
  - Semantic and syntactic (format) reconciliation of data?
- Data cleaning
- Difficult to populate
- Difficult to refresh or update
- Sometimes not possible to bring all the data from a given source
- ...

Alternative: keep data in their sources

- Interacting with each of the sources via queries
- Considering all available sources
- Selecting only those to be queried
- Querying the relevant sources on an individual basis
- Combining results from different sources

A long, tedious, complex and error prone process ...

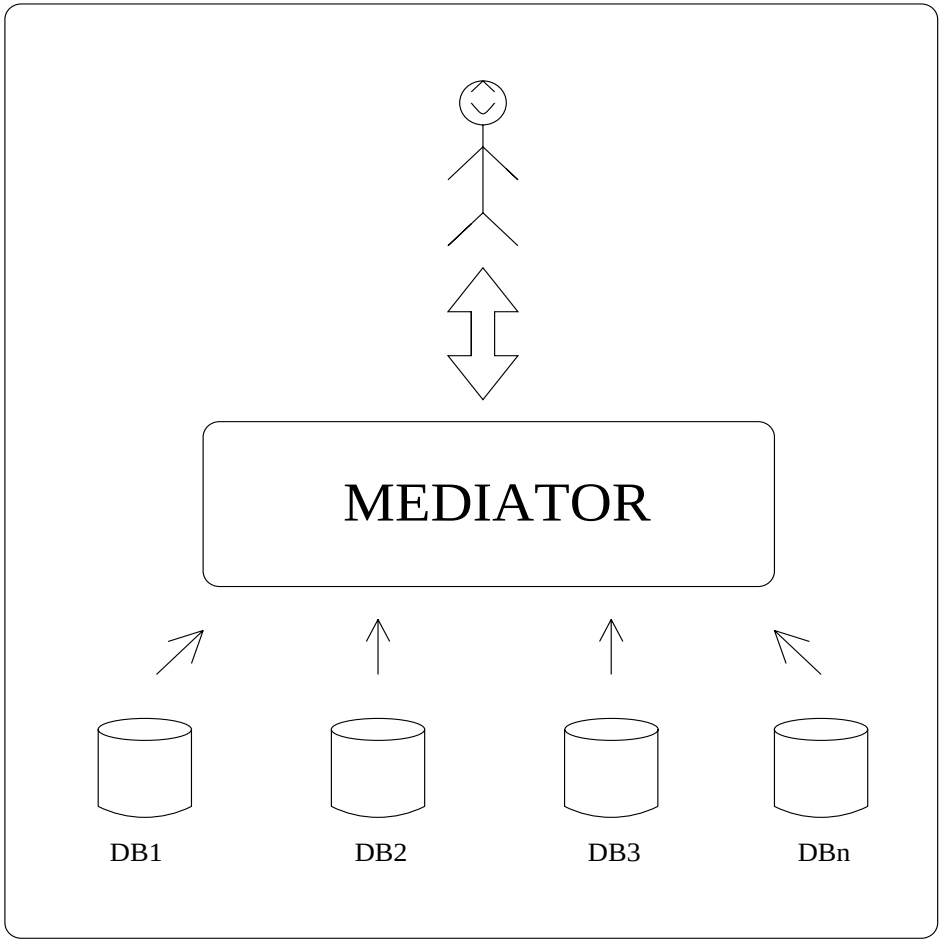
To implement this, an automated, robust and uniform approach is needed

A solution consists in *virtual integration of data sources via mediators*

Mediator: Software system that offers a common, data base like interface to a set of autonomous, independent and possibly heterogeneous data sources

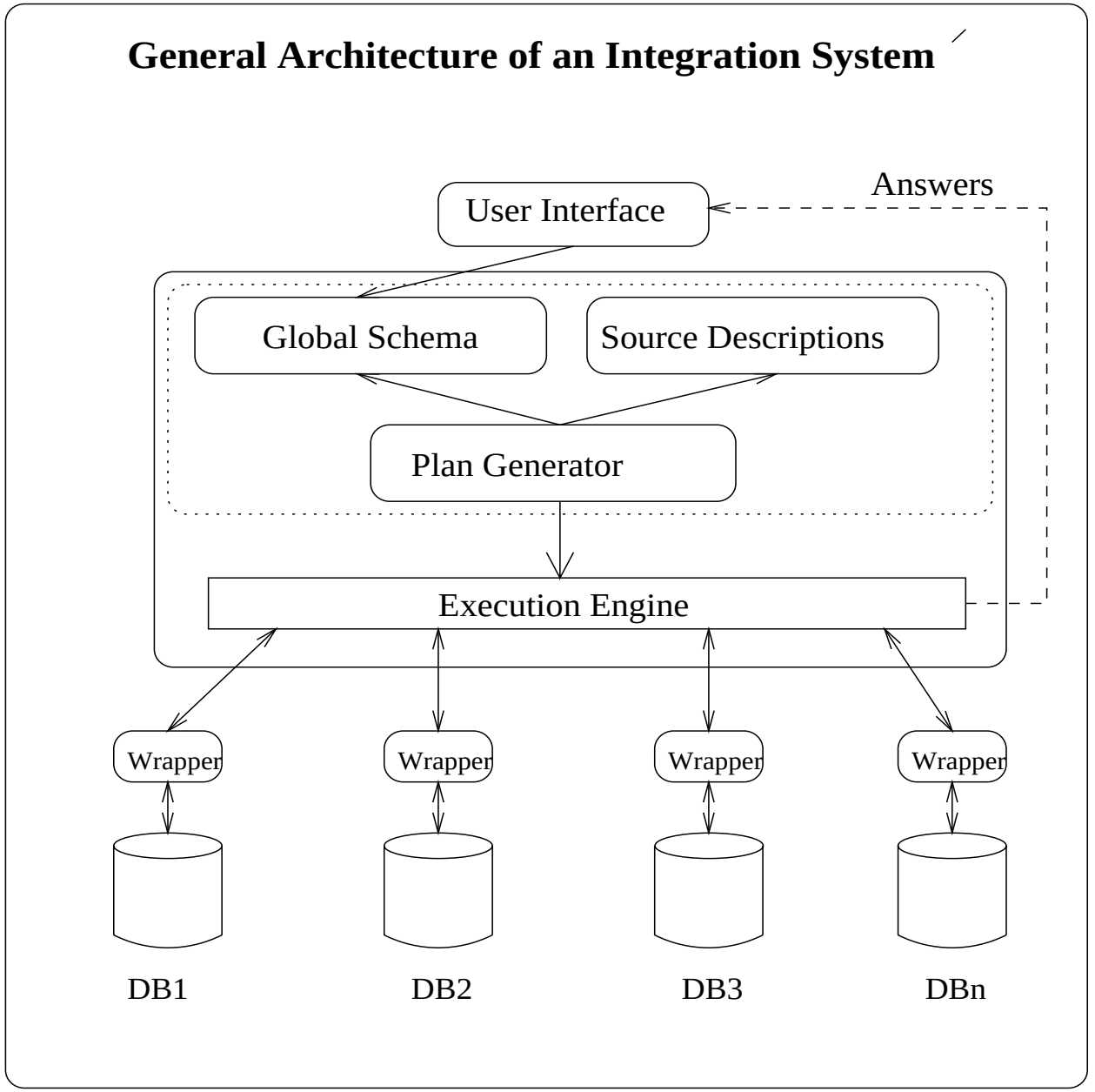
- Virtual integration because data stays in the sources
- User interacting with mediator feels like interacting with a single database
- The sources may not cooperate with each other
- No control on individual sources
- No central control or maintenance mechanism
- Set of participating sources is flexible and open

Interaction with a mediated integrated information system



Main features of a mediator based system:

- Interaction with the system via queries posed to the mediator
- Updates via the mediator are not allowed
- Data sources are mutually independent and may participate in different mediated systems at the same time
- Integration system allows sources to get in and out
- Data is kept in the local, individual sources, and extracted at mediator's request


General Architecture of an Integration System
Answers
User Interface
Global Schema
Source Descriptions
Plan Generator
Execution Engine
Wrapper
Wrapper
Wrapper
Wrapper
DB1
DB2
DB3
DBn

Global or Mediated Schema:

- Set of names for relations (virtual tables) and their attributes
- Determines a query language for the global system
- Application dependent
- Like in a normal, usual relational DB, from the user point of view
- The "DB" corresponding to the global schema is virtual
- User poses queries in terms of the relations in the global schema

- Mediator knows the correspondence between the global schema and the local schemas (source descriptions)

- Mediator receives a query and develops a *query plan* that determines
  - the relevant portions of data in the relevant data sources
  - how to extract that data from the sources via queries, and
  - how to combine the answers received into a final answer for the user

- Mediator is responsible for solving problems of *redundancy*, *complementarity*, *incompleteness*, and *consistency* of data in the integration system

  What to do if we ask about a person's ID card number and we get two different numbers, each coming from a different source?

  The two sources, independently, may be consistent, but taken together, possibly not *Consistency is a problem!*

Example: Consider global schema for a DB "containing" information about scientific publications:

$$Conference(Paper, Conference)$$
$$Year(Paper, Year)$$
$$Place(Conference, Year, Place)$$

User wants to know where conference ACM PODS'89 was held

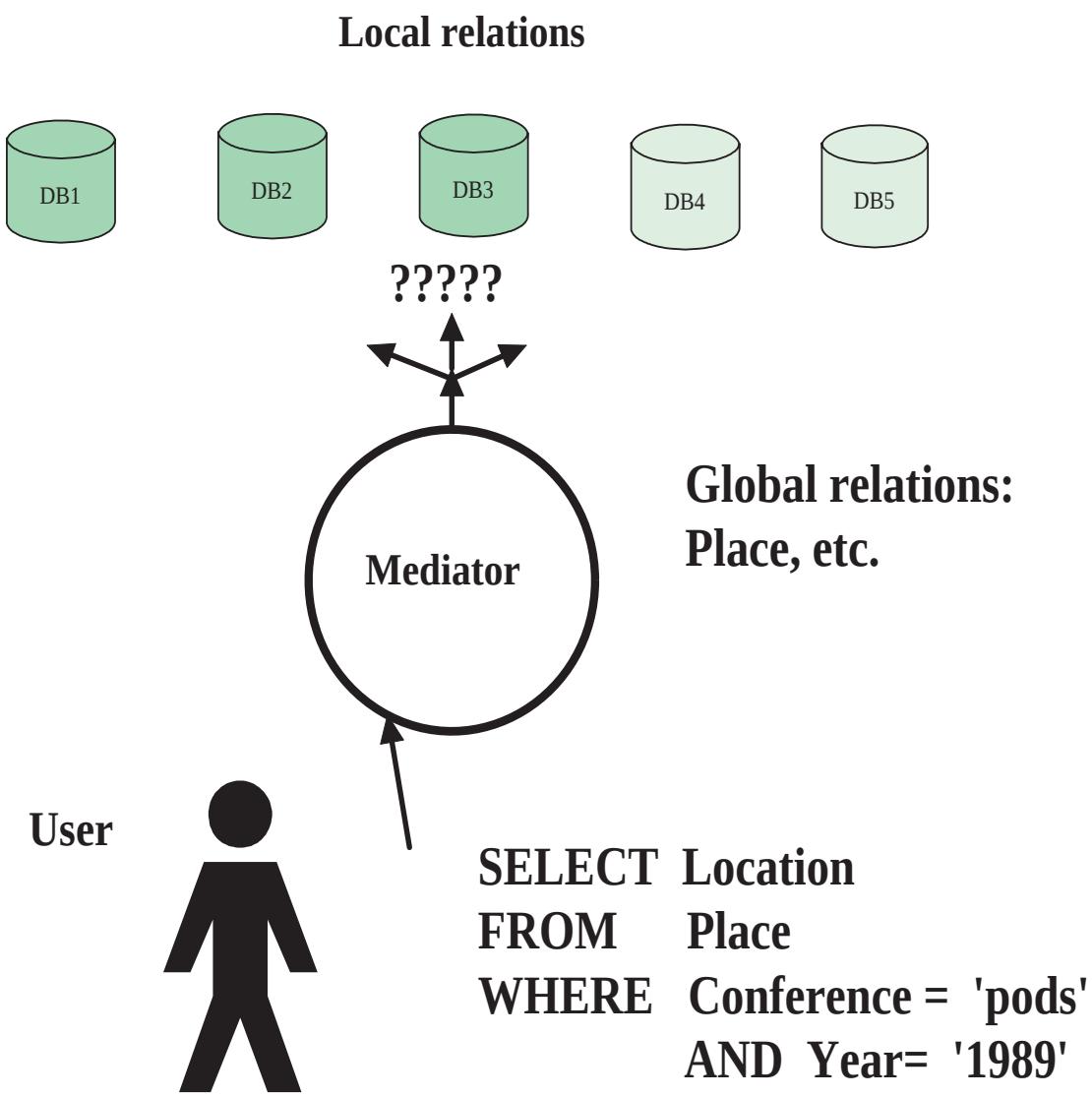
Query to global system $\quad Q: \quad Ans(L) \leftarrow Place(pods, 1989, L)$

Predicate *Ans* contains the answers, that are computed by computing the RHS of the rule

In this case it is the selection $\quad \texttt{SELECT}_{X=pods, Y=1989}\ \texttt{Place(X, Y, L)}$

However, the data is not in table *Place*

Query plan needed to extract and combine the relevant data from sources


Local relations
DB1
DB2
DB3
DB4
DB5
?????
Mediator
Global relations:
Place, etc.
User
SELECT Location
FROM Place
WHERE Conference = 'pods'
AND Year= '1989'

Wrapper:

- Module that is responsible of wrapping a data source, so that it can interact with the rest of integration system
- There is a wrapper for each data source
- It presents the data source as a convenient database, with the right schema and data
- This presentation schema may be different from the real, internal one
- Data provided by the wrapper my be different from the real one in the local source

- Preliminary transformations, cleaning, etc. may be necessary before exporting the data to the integration system

- Provides data as requested by the execution engine

We will assume that each data sources have already a wrapper that presents it as a, say relational database

Source Description:

- The mediator needs to know the available sources and their contents (as presented by the wrapper)

- This is done by describing the relationships (mappings) between the global schema and the local schemata

- The description is given by means of a set of logical formulas

  Think of the way views are defined in terms of base tables in a relational DB, using queries ...

  Logical formulas are used, e.g. expressed in SQL

- How are the mappings defined?

There are two main approaches (and combinations of them)

- **Global-as-View (GAV)**:

  The relations in the global schema are described as views of the collection of local relations

- **Local-as-View (LAV)**:

  Each relation in a local source is described as a view of the global schema

- **GLAV**: a combination of GAV and LAV

  (Friedman, Levy, Millstein. AAAI 99)

(More on all this soon ...)

Plan Generator:

- Gets a user query in terms of global relations
- Uses the source descriptions to design a *query plan*
- *Rewrites* the query as a set of subqueries queries expressed in terms of the local relations
- Rewriting depends on whether the LAV or the GAV approach is followed
- Query plan includes the way in which the different results from local sources are combined

Still much theoretical and technical research to do on query plan generation

Execution Engine:

- The query plan is just that, a plan; another issue is to execute it

- The EE could solve, e.g. inconsistencies, etc.

  Unless we are able to anticipate potential inconsistencies and "solve them in advance", when the plan is generated

  That is, the plan generator could take care of potential inconsistencies ... we'll see ...

Description of Data Sources

Achieved through logical formulas that relate the global relations with local relations

Global-as-View (GAV):

Relations in the global schema are described as *views* of the union of the local schemata

Conceptually very natural: usually views are virtual relations defined in terms of material relations (tables)

Since global relations are virtual and local sources, material, it makes sense ...

Example: Integrating information about movies, from local sources:

- $DB_1(Title, Dir, Year)$
- $DB_2(Title, Dir, Year)$
- $DB_3(Title, Review)$
- A global relation containing movies and their years:

$$MovieYear(Title, Year) \leftarrow DB_1(Title, Dir, Year)$$
$$MovieYear(Title, Year) \leftarrow DB_2(Title, Dir, Year)$$

That is, $MovieYear$ is defined as the union of two projections, of DB_1 and DB_2 on attributes $Title$, $Year$, i.e.

$$MovieYear = \quad \Pi_{Movie, Year}(DB_1) \cup \Pi_{Movie, Year}(DB_2)$$

- The movies, their directors and reviews:

  $$MovieRev(Title, Dir, Review) \leftarrow DB_1(Title, Dir, Year), DB_3(Title, Review)$$

This is a view defined as, first, the join of DB_1 and DB_3 via attribute $Title$, and then a projection on $Title, Dir, Review$

The view is defined by means of a *rule*

The rule says that in order to compute the tuples in the relation in the LHS (the *head* of the rule), we have to go to the RHS (the *body* of the rule) and compute whatever is specified there

The attributes appearing in the head indicate that we are interested in them only, so the others (in the body) can be projected out at the end

If there are more that one rule to compute a relation, we use all of them and we take the union of the results, like in $MovieYear$

Instead of using a rule as above, we can use relational algebra (or relational calculus, or SQL2):

$$MovieRev = \Pi_{Title,Dir,Review}(DB_1 \bowtie_{Title} DB_2)$$

Why rules then?

The language of rules is more expressive than relational algebra, e.g. recursive views can be defined with rules, but not with relational algebra

Queries:

Posed via the mediator and in terms of global relations

How to answer them? The global relations have no material data ...

Under GAV it is simple: just rule unfolding

Example: Query: Movies filmed in year 2001, with their reviews?

$$Ans(Title, Review) \leftarrow \underline{MovieYear(Title, 2001)}, \underline{\underline{MovieRev(Title, Dir, Review)}}$$

Expressed in terms of the global schema

The data has to be obtained from the sources

The query has to be *rewritten* in terms of the source relations

"Unfold"each global relation by replacing it by its definition in terms of the local relations

$$Ans'(Title, Review) \leftarrow \underline{DB_1(Title, Dir, 2001)}, \underline{\underline{DB_1(Title, Dir, 2001)}, \underline{\underline{DB_3(Title, Review)}}}$$

$$Ans'(Title, Review) \leftarrow \underline{DB_2(Title, Dir, 2001)}, \underline{\underline{DB_1(Title, Dir, 2001)}, \underline{\underline{DB_3(Title, Review)}}}$$

These new queries do get answers directly from the sources

Final answer is the union of two answer sets, one for each rule

But notice: there is a redundant condition (subgoal) on the RHS of the first rule; and the second rule is completely redundant

The mediator should realize this before performing redundant computations

Local-as-View (LAV):

Each table in each local data source is described as a view (as a query expression) in terms of the global relations

Somehow unnatural from the conceptual point of view, and from perspective of usual databases practice

... here views contain data, but "base tables"don't ...

But this approach has some advantages

Scenario:

- Collection of material data sources $S_1, \ldots, S_n$
- Global, virtual database G
- G integrates data from $S_1, \ldots, S_n$
- Tables in $S_1, \ldots, S_n$ are seen as *views* of G

Example: Sources:

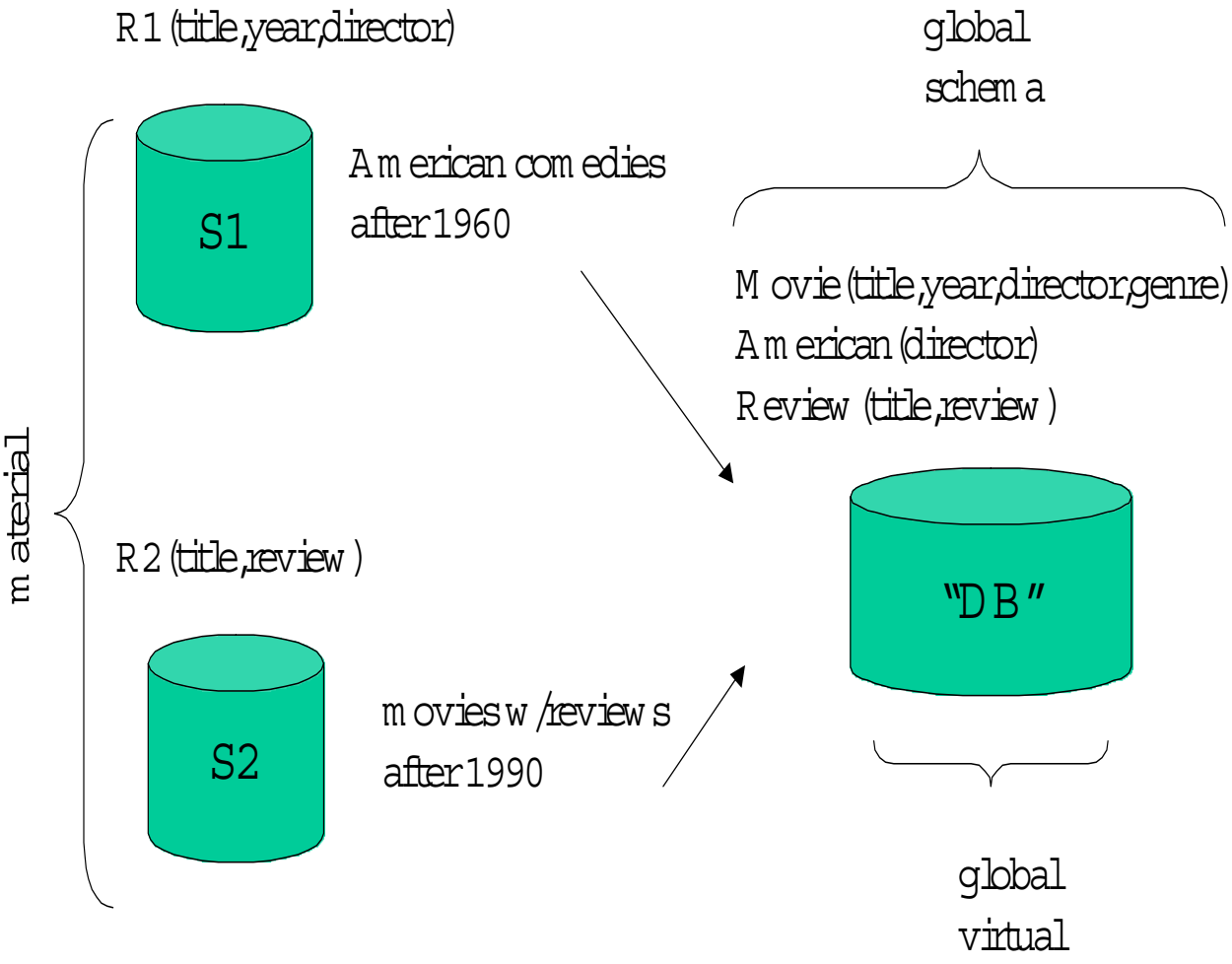
$$S_1\text{:}\quad V_1(\mathit{Title}, \mathit{Year}, \mathit{Director}) \leftarrow \quad \mathit{Movie}(\mathit{Title}, \mathit{Year}, \mathit{Director}, \mathit{Genre}),$$
$$\mathit{American}(\mathit{Director}),\ \mathit{Year} \geq 1960,$$
$$\mathit{Genre} = \mathit{comedy}.$$

S_1 contains comedies, not older than 1960, with American directors and their years

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

S_2 contains movies not older than 1990 with their reviews, but not their directors

Then, global schema G: $\mathit{Movie}(\mathit{Title}, \mathit{Year}, \mathit{Director}, \mathit{Genre}),$
$\mathit{Review}(\mathit{Title}, \mathit{Review}),$
$\mathit{American}(\mathit{Director})$


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

Notice: 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, the information in S_2 could be "incomplete" wrt what G contains (or might contain)

... containing only a part of the data of the same kind in the global system

More on this later ...

Queries:

Query posed to G: "Comedies with their reviews filmed since 1950?"

$$Ans(Title, Review) \leftarrow Movie(Title, Year, Director, comedy), \\ Review(Title, Review), Year \geq 1950.$$

Query expressed as usual, in terms of global relations only

Not possible to obtain answers by a simple and direct computation of the RHS of the query

No direct rule unfolding for the relations in the body: no explicit definitions for them now

Data is in the sources, now views ...

Plan generation to extract information from the sources is more complex than with GAV

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 (needed in this example)

A query plan for our query is (believe this for the moment ...)

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

Query has been rewritten in terms of the views ...

Can be computed:

1. Extract values for *Title* from V_1
2. Extract the tuples from V_2
3. At the mediator level, compute the join via *Title*

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

Remark: In LAV, we have to answer a query posed in terms of certain relations (the global ones), but we have to answer using the contents of certain views only (the local relations)

Then, query plan generation becomes an instance of a more general and traditional problem in DBs: *query rewriting using views*

- Given is a collection views $V_1, \ldots, V_n$, whose contents are already computed
- A new query Q arrives

  Instead of computing its answers directly,
- Try to use the answers (contents) to (of) $V_1, \ldots, V_n$

Query can be seen as views, but usually are not defined as such, and answers may not be kept

Instead, views will be used during a session or across sessions

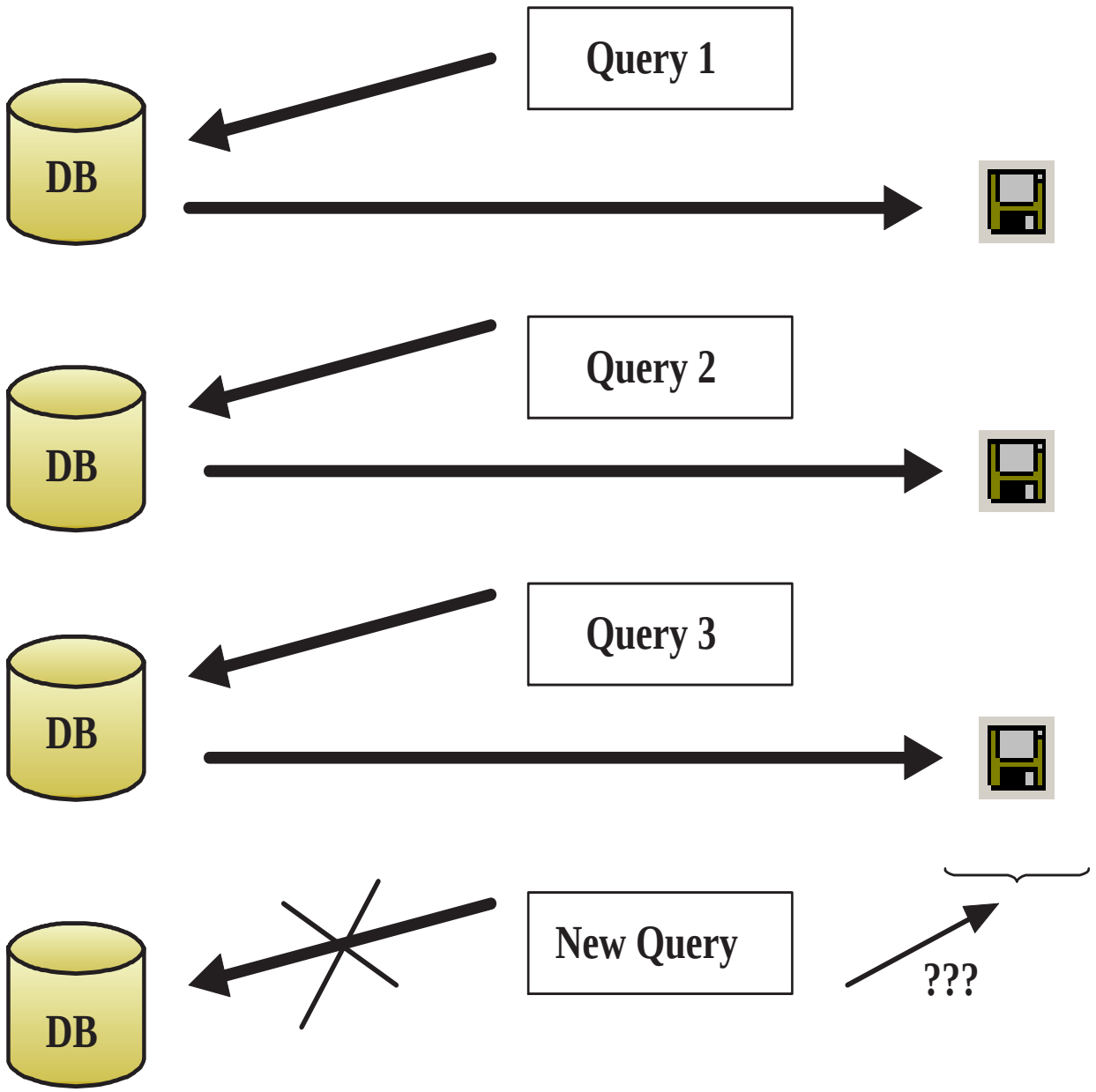
Views usually kept virtual (during a session), and are recomputed when needed (if there are relevant updates and full re-computation is easy)

May be useful to materialize contents for certain views and answers to queries

When computing those answers has been expensive and the information obtained is potentially useful and related to other queries

When a new query arrives, try to take advantage of those pre-computed, cached results ... How?

Problem: How much from the real answer do we get by using the pre-computed views only? What is the maximum we can get?


Query 1
DB
Query 2
DB
Query 3
DB
New Query
DB
???

Comparison of Paradigms:

- GAV:
  - Rule unfolding makes plan generation simple and direct
  - Not flexible to accept new sources or eliminate sources into/from the system
  - Adding new sources implies modifying definitions of global relations
- LAV:
  - More flexibility to add new sources to the integration system
  - A new source is just a new view definition
  - Other sources do not need to be considered at this level, no interaction with other sources
  - Only the plan generator has to be aware of the new source
  - Plan generation is more difficult, actually provably more complex

Data Integration and Consistency

In virtual data integration, one usually **assumes** that certain integrity constraints (ICs) hold at the global level

Actually, they are **used** in the generation of the query plan

There are situations where without ICs no query plan can be generated

Example: Global schema $Conferences(Paper, Conference)$
$Years(Paper, Year)$
$Locations(Conference, Year, Location)$

Global functional dependencies (FDs):

$$Conferences\text{: } Paper \rightarrow Conference$$
$$Years\text{: } Paper \rightarrow Year$$
$$Locations\text{: } Conference, Year \rightarrow Location$$

Data sources as views of the global DB (i.e. LAV):

$$S_1(P, C, Y) \leftarrow Conferences(P, C), Years(P, Y)$$
$$S_2(P, L) \leftarrow Conferences(P, C), Years(P, Y), Locations(C, Y, L)$$

S_1 contains papers with their conferences and years

S_2 contains the papers and their presentation locations

Want location of PODS99

$$Ans(L) \leftarrow Locations(pods, 1999, L)$$

Query plan:

$$Ans'(L) \leftarrow S_1(P, pods, 1999), S_2(P, L).$$

That is:

- First find some paper presented at PODS99 using source S_1
- Next, find the location of the conference at which this paper was presented using source S_2

Plan is correct: every paper is presented at one conference and in one year only

But only if the global FDs are satisfied

BUT, how can we be sure that such global ICs hold?

They are not maintained or checked at the global level and could be violated

... think of two consistent data sources (DBs) S_1, S_2 with the same kind of information that merged produce violations of a FD

Actually, the flexibility to add/remove sources, e.g. in LAV, is likely to introduce extra sources of inconsistency

Example:

S_1	Paper	Conference	Year
	querying inconsistent databases	pods	1999
	workflow specifications	sigmod	1998
	...	...	...

S_2	Paper	Location
	querying inconsistent databases	philadelphia
	querying inconsistent databases	schloss dagstuhl
	...	...

Each local source seems to be consistent, but not the global system

Conference, Year associated to two *Location*s via *Paper*

... intuitively speaking at least, because we have not defined what a *legal and consistent* global instance *would be* ...

Desired:

- When queries are posed to the integrated system, retrieve only those answers that are "consistent with" the ICs (which are they?)

- That is, solve inconsistencies at query time

- By the way, this an interesting new and more general view of ICs:

  They represent semantic constraints on query answers, rather than on database states

  A much more flexible approach to database integrity than the classical approach

  ... deserves investigation on its own ...

PART II: Query Plans and Semantics of Virtual Data Integration Systems

Query Plans

We will concentrate on the LAV approach

Given a global query Q posed in terms of the global schema, we need to go to the sources for the data

How?? We need a plan for evaluating Q???

Q has to be "rewritten" in terms of the views, i.e. the relations in the sources

Some rewriting algorithms:

- Bucket [Halevy et al.]
- Inverse Rules [Duschka, Genesereth]
- MiniCon [Halevy et al.]
- Deductive [Grant, Minker]

Inverse Rules Algorithm

Given: Set of rules describing the source relations as conjunctive (SPJ) views of the global schema

Descriptions contain rules of the form:

$$S(\bar{X}) \leftarrow P_1(\bar{X}_1), \ldots, P_n(\bar{X}_n)$$

S is a source relation; P_is are global relations

Incomplete (open) sources (more on this coming ...)

Input: global query expressed in Datalog (may be recursive, but no negation)

Output: A new Datalog program expressed in terms of the source relations

Example: V_1, V_2 are local; R_1, R_2, R_3 are global

Source descriptions $\mathcal{V}$:

S_1: $V_1(X, Z) \leftarrow R_1(X, Y), R_2(Y, Z)$
S_2: $V_2(X, Y) \leftarrow R_3(X, Y)$

IR Algorithm: obtain from these descriptions, "inverses rules" describing the global relations

Idea: V_2 is incomplete, i.e. its contents is contained in the contents of the global relation R_3 (which one? more on this coming ...)

That is, the only way to get tuples for V_2 is by going to pick up tuples on the RHS

That is, V_2 "$\subseteq$" R_3, i.e. V_2 "$\Rightarrow$" R_3

More precisely, we invert the rule in the description of V_2

We get: $R_3(X, Y) \leftarrow V_2(X, Y)$ a rule describing R_3!!

There may be other rules describing R_3, from other source description rules containing R_3 on the RHS (take the union ...)

What about inverting the rule for V_1?

$R_1(X, Y), R_2(Y, Z) \leftarrow V_1(X, Z)$???

What kind of rule (head) is this? Maybe the conjunction ...

$R_1(X, Y) \leftarrow V_1(X, Z)$ and $R_2(Y, Z) \leftarrow V_1(X, Z)$

The two occurrences of Y are independent now, and it was a shared variable

What does Y mean in the heads? No condition on Y in the bodies

Any value for Y? Not the idea ...

Better: $V_1(X, Z) \leftarrow R_1(X, Y), R_2(Y, Z)$ is equivalent to

$$V_1(X, Z) \leftarrow \exists Y \ (R_1(X, Y) \wedge R_2(Y, Z)) \quad \text{(join and projection)}$$

Inverting, we obtain

$$\exists Y (R_1(X, Y) \wedge R_2(Y, Z)) \leftarrow V_1(X, Z)$$

There is an implicit universal quantification on X, Z

Each value for Y possibly depends on the values for X, Z, i.e. Y is a function of X, Z

To capture this dependence, replace Y by a "function symbol" $f_1(X, Y)$

(we may need new functions, for dependencies between variables in other rules)

$$R_1(X, f_1(X, Z)) \wedge R_2(f_1(X, Z), Z) \leftarrow V_1(X, Z)$$

Then

$$R_1(X, f_1(X, Z) \leftarrow V_1(X, Z)$$

$$R_2(f_1(X, Z), Z) \leftarrow V_1(X, Z)$$

Finally, we have the following inverse rules $\mathcal{V}^{-1}$:

$$\begin{aligned} R_1(X, f_1(X, Z)) &\leftarrow V_1(X, Z) \\ R_2(f_1(X, Z), Z) &\leftarrow V_1(X, Z) \\ R_3(X, Y) &\leftarrow V_2(X, Y) \end{aligned}$$

We can use them to compute global queries

(introduce one function symbol for each variable in the body of a view definition that is not in the head; the function evaluated in the variables in the head)

Query Q

$$\begin{aligned} Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z), R_4(X) \\ R_4(X) &\leftarrow R_3(X,Y) \\ R_4(X) &\leftarrow R_7(X,Y) \\ R_7(X) &\leftarrow R_1(X,Y), R_6(X,Y) \end{aligned}$$

Goal R_6 cannot be computed, there is no description in $\mathcal{V}_1$ for it (it did not appear in any source description)

Then, R_7 cannot be evaluated either; delete that rule

$$\begin{aligned} Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z), R_4(X) \\ R_4(X) &\leftarrow R_3(X,Y) \\ R_4(X) &\leftarrow R_7(X,Y) \end{aligned}$$

For the same reason, second rule cannot be evaluated; delete it

$$\begin{aligned} Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z), R_4(X) \\ R_4(X) &\leftarrow R_3(X,Y) \end{aligned}$$

We obtain a pruned query Q^-

The plan returned by the IRA is $Q^- \cup \mathcal{V}^{-1}$; a Datalog program with functions

$Plan(Q)$:

$$\begin{aligned} Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z), R_4(X) \\ R_4(X) &\leftarrow R_3(X,Y) \\ R_1(X, f_1(X,Z)) &\leftarrow V_1(X,Z) \\ R_2(f_1(X,Z), Z) &\leftarrow V_1(X,Z) \\ R_3(X,Y) &\leftarrow V_2(X,Y) \end{aligned}$$

"The best we have" to answer the original query (this may be given a precise meaning)

We will compute *some* answers to Q, but "the most" we can

The Datalog query can be evaluated, e.g. bottom-up, from concrete source contents, e.g.

$$V_1 = \{(a, b), (a, a), (c, a), (b, a)\}$$
$$V_2 = \{(a, c), (a, a), (c, d), (b, b)\}$$

Delete from the final answer set those tuples with the auxiliary function symbol f_1, if any

Let's see this with a different query

$$Ans(X) \leftarrow R_1(X, Y), R_2(Y, Z), R_4(X)$$
$$Ans(X) \leftarrow R_2(X, Y)$$
$$R_4(X) \leftarrow R_3(X, Y)$$
$$R_4(Y) \leftarrow R_1(X, Y), Y = a$$

(This query can be expressed as the SQL query

```
SELECT R1.1 FROM R1, R2, R4 WHERE R1.2 = R2.1 AND R1.1 = R4.1
UNION SELECT R2.1 FROM R2
```

that involves the previously defined view

```
CREATE VIEW R4 AS SELECT R3.1 FROM R3 UNION SELECT R1.2 FROM R1
WHERE R1.2 = 'a'
```

)

We use the inverse rules (the same for all the queries)

$$R_1(X, f(X, Z)) \leftarrow V_1(X, Z)$$
$$R_2(f(X, Z), Z) \leftarrow V_1(X, Z)$$
$$R_3(X, Y) \leftarrow V_2(X, Y)$$

Prune query rules that cannot be evaluated due to the inverse rules

$$Ans(X) \leftarrow R_1(X,Y), R_2(Y,Z), R_4(X)$$
$$Ans(X) \leftarrow R_2(X,Y)$$
$$R_4(X) \leftarrow R_3(X,Y)$$

(last rule in the query does not contribute to R_4; a cannot be a f-value)

Rest is just bottom-up evaluation of

$$Ans(X) \leftarrow R_1(X,Y), R_2(Y,Z), R_4(X)$$
$$Ans(X) \leftarrow R_2(X,Y)$$
$$R_4(X) \leftarrow R_3(X,Y)$$
$$R_1(X, f(X,Z)) \leftarrow V_1(X,Z)$$
$$R_2(f(X,Z), Z) \leftarrow V_1(X,Z)$$
$$R_3(X,Y) \leftarrow V_2(X,Y)$$

and a final deletion of the tuples from Ans that are f-values:

$R_1 = \{(a, f(a,b)), (a, f(a,a)), (c, f(c,a)), (b, f(b,a))\}$
$R_2 = \{(f(a,b), b), (f(a,a), a), (f(c,a), a), (f(b,a), a)\}$
$R_3 = \{(a,c), (a,a), (c,d), (b,b)\}$
$R_4 = \{a, c, b\}$
$\Pi_X(R_1 \bowtie R_2) = \{a, c, b\}$

From the 1st rule for Ans: $Ans = \{a, c, b\} \cap \{a, c, b\} = \{a, c, b\}$

From the second rule for Ans: $Ans = \{f(a,b), f(a,a), f(c,a), f(b,a)\}$

These tuple do not contribute to the final answer

Then, finally $Ans = \{a, c, b\}$.

Properties of the Inverse Rules Algorithm

- The query plan obtained is not exactly a Datalog program, because it may contain function symbols

- The query plan obtained can be evaluated in a bottom up manner and always has a unique fix point

- The query plan can be constructed in polynomial time in the size of the original query and the source desciptions

- The plan obtained is the best we can get under the circumstances, i.e. given the query, the sources and their descriptions

  More precisely, query plan is *maximally contained* in the original query Q

- There is no (other) query plan that retrieves a proper superset of *answers* to Q from the integration system

  This can be made precise ...

  Theorem: For every Datalog program Q and every set of conjunctive source descriptions $\mathcal{V}$, the query plan obtained with the IRA is maximally contained in Q

What *answers* are we talking about?

A subtle point, since there is no global instance and then no answer to a global query in the classical sense

To answer this question, we need to give a *semantics* to an open global system under the LAV approach

Semantics:

An open mediated system $\mathcal{G}$ under the LAV approach determines *a set of legal global instances* as follows

- $\mathcal{G}$: $V_1(\bar{X}_1) \leftarrow \varphi_1(\bar{X}_1)$ v_1

  .....

  $V_n(\bar{X}_n) \leftarrow \varphi_n(\bar{X}_n)$ v_n

  Each v_i is an extension (material data source) for view V_i

- Let D be a **global** instance, i.e. its domain contains at least the constants appearing in the source extensions and the view definitions; and has relations (and contents) for the global schema

- $\varphi_i(D)$ is the definition of view V_i applied to D; what gives an extension for V_i in (wrt) global instance D

We define $\quad Legal(\mathcal{G}) := \{ \text{ global } D \mid v_i \subseteqq \varphi_i(D);\ i = 1, \ldots, n\}$

Legal instances respect the fact that sources are open, aka *incomplete* or *sound* sources

The global schema may have many instances, but only the legal ones will determine which are the intended answers to a global query

Example: A global system that integrates data sources about teams participating in the soccer world cup 2002

Global relation $Team(Country, Group)$

One source of information contains the countries whose matches in the first round were shown on TV

Can be represented as the view

$$ShownOnTV(X) \leftarrow Team(X, Y)$$

It contains only a subset of the expected entries for relation $Team$:

$$ShownOnTV \subseteq \Pi_{Country}(Team)$$

The source is open or incomplete

A second source, *Qual*, contains all the countries participating in the qualifying matches, e.g. Germany, Canada, ...

$$Qual(X) \leftarrow Team(X, Y)$$

Canada did not participate in the World Cup, but did participate in the qualifying matches, then $Qual \not\supseteq \Pi_{Country}(Team)$

This source is not open, rather

$$Qual \supseteq \Pi_{Country}(Team)$$

We say the source is *closed*, aka *complete*

We could admit this source into the system, but the legal instances would have to respect its closedness ...

A third source, *First*, contains the countries participating in the first round

$$First(X) \leftarrow Team(X, Y)$$

Now $First = \Pi_{Country}(Team)$

The source is *open and closed* (clopen) or *sound and complete*

In what follows, open sources only ...

Example: Global system $\mathcal{G}_1$ with relation $R(X, Y)$ and open sources

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

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

Legal instance: $D = \{(a, b), (c, d), (a, c), (d, e)\}$

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

All supersets of D are also legal global instances; e.g.

$$\{(a, b), (c, d), (a, c), (d, e), (c, e)\} \in \mathit{Legal}(\mathcal{G})$$

But no subset of D is legal, e.g.

$$\{(a, b), (c, d), (a, c)\} \notin \mathit{Legal}(\mathcal{G})$$

The legal instances give the semantics to the virtual data integration system
We are not interested in the legal instances *per se*

But now we can define the intended answers to a global query

The *certain answers* to a global query are those that can be obtained from every legal global instance

$$Certain_{\mathcal{G}}(Q) := \{\bar{t} \mid \bar{t} \text{ is an answer to } Q \text{ in } D \text{ for all } D \in Legal(\mathcal{G})\}$$

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

Global query Q : $Ans(X, Y) \leftarrow R(X, Y)$

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

The algorithms for constructing query plans should be able to produce plans that will get all the certain answers from a data integration system

Of course, without explicitly computing all the legal instances

It is possible to prove that, for open sources, the IRA returns all the certain answers to conjunctive queries

There are other complete algorithms, but IRA illustrates the idea and we will use it later on ...

The Need for Recursion

Assume the original global query is conjunctive, i.e. of the form

$$Q: \quad Ans(\bar{X}) \leftarrow A_1(\bar{X}_1), \ldots, A_n(\bar{X}_n)$$

where the A_i are database atoms, and the body may contain some variables that are not in the head (i.e. existential quantifiers or projections); in particular, no negations

Conjunctive queries are part of RA, and also of Datalog, but less than Datalog (that also has union and recursion)

Given Q, IRA will produce a query plan that is a Datalog program with functions (but without negations)

So, do we need recursion?

If the original global query is recursive, yes ... not surprising ...

What if the query is not recursive, e.g. when it is conjunctive?

If we want the query plan to be maximally contained in the original query, yes ...

At least when there are restrictions on the patterns of queries, i.e. when data sources may be accessed only with particular patterns

E.g. one restriction specifies in an atom the variables that must have concrete values (bindings) in order to access the data, i.e. there are *binding patterns*

For example, $Grades^{bf}(Student, Grade)$ (b for *bound*, and f for *free*)

Meaning that

- we cannot ask "give me all the students with their grades", i.e. $Grades(X, Y)$?
- but we can ask for the grades of specific students, e.g. "give me John's grades", i.e. $Grades(john, Y)$?

Example: Three global relations

- $AAAIpapers(X)$ "contains" papers presented at the AAAI conference
- $Cites(X, Y)$ contains papers citing other papers; papers presented anywhere
- $AwardPaper(X)$ contains award winning papers (presented anywhere)

Three open data sources

- one containing papers presented at the AAAI conference, no access restrictions

$$AAAIdb^{f}(X) \leftarrow AAAIpapers(X)$$

- another containing papers and their citations, but can be accessed providing the title of the citing paper

$$CitationDB^{bf}(X, Y) \leftarrow Cites(X, Y)$$

- finally, one containing award winning papers, that can be accessed to check specific papers

$$AwardDB^b(X) \leftarrow AwardPaper(X)$$

Global query: "Give me all award winning papers", i.e.

$$Q: \quad Ans(X) \leftarrow AwardPaper(X)$$

AwardPaper cannot be used directly

Need to go through *AAAIpapers*, but also through *Cites* if we want to find as many papers as possible

Q':

$$Ans'(X) \leftarrow AAAIdb(X), AwardDB(X)$$

$$Ans'(X) \leftarrow AAAIdb(V), CitationDB(V, X_1), \ldots, CitationDB(X_m, X), \\ AwardDB(X)$$

No plan that fixes the length m of the chain of citations will be maximally contained in the original query

Instead, the following query Q'' will do ...

$$Ans(X) \leftarrow papers(X), AwardDB(X)$$

$$papers(X) \leftarrow AAAIdb(X)$$

$$papers(X) \leftarrow papers(Y), CitationDB(Y, X)$$

PART III: Consistent Query Answers from Virtual Data Integration Systems

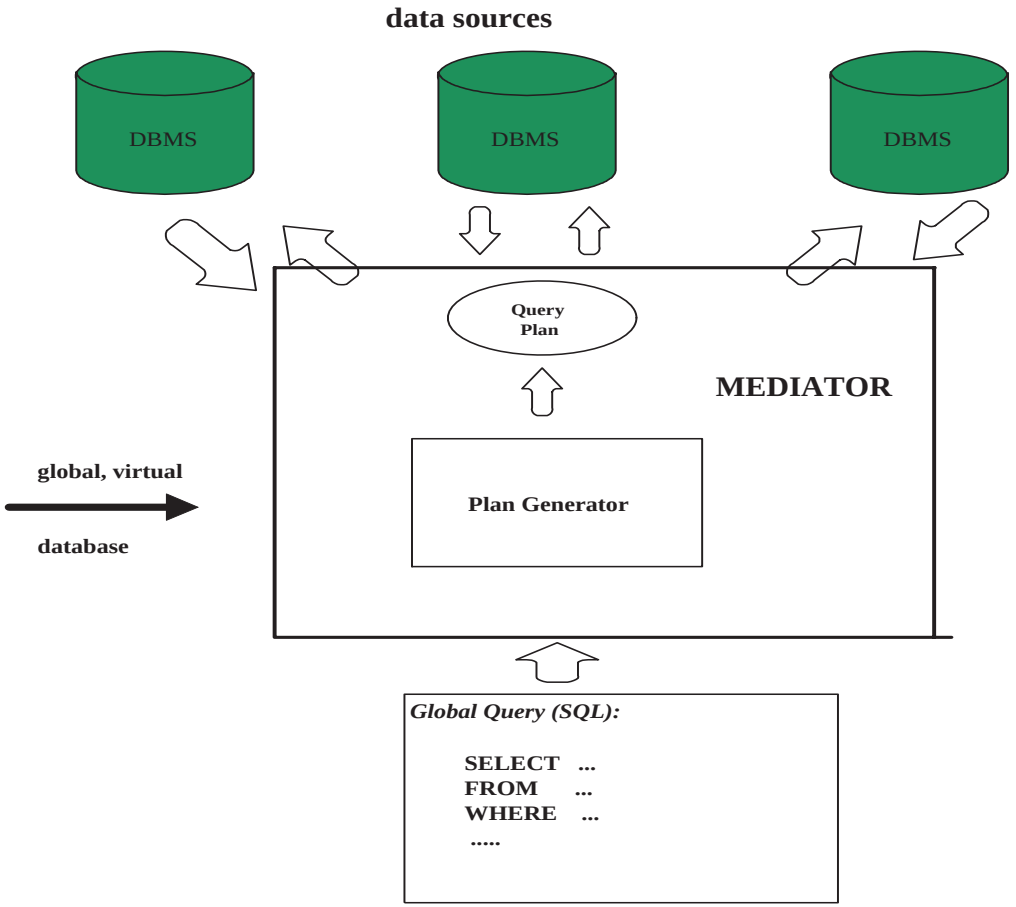
Data Integration

Given a collection of (materialized) data sources $S_1, \ldots, S_n$, and a global, virtual database $\mathcal{G}$, that integrates the data sources

There might be a set IC of global ICs that are not maintained

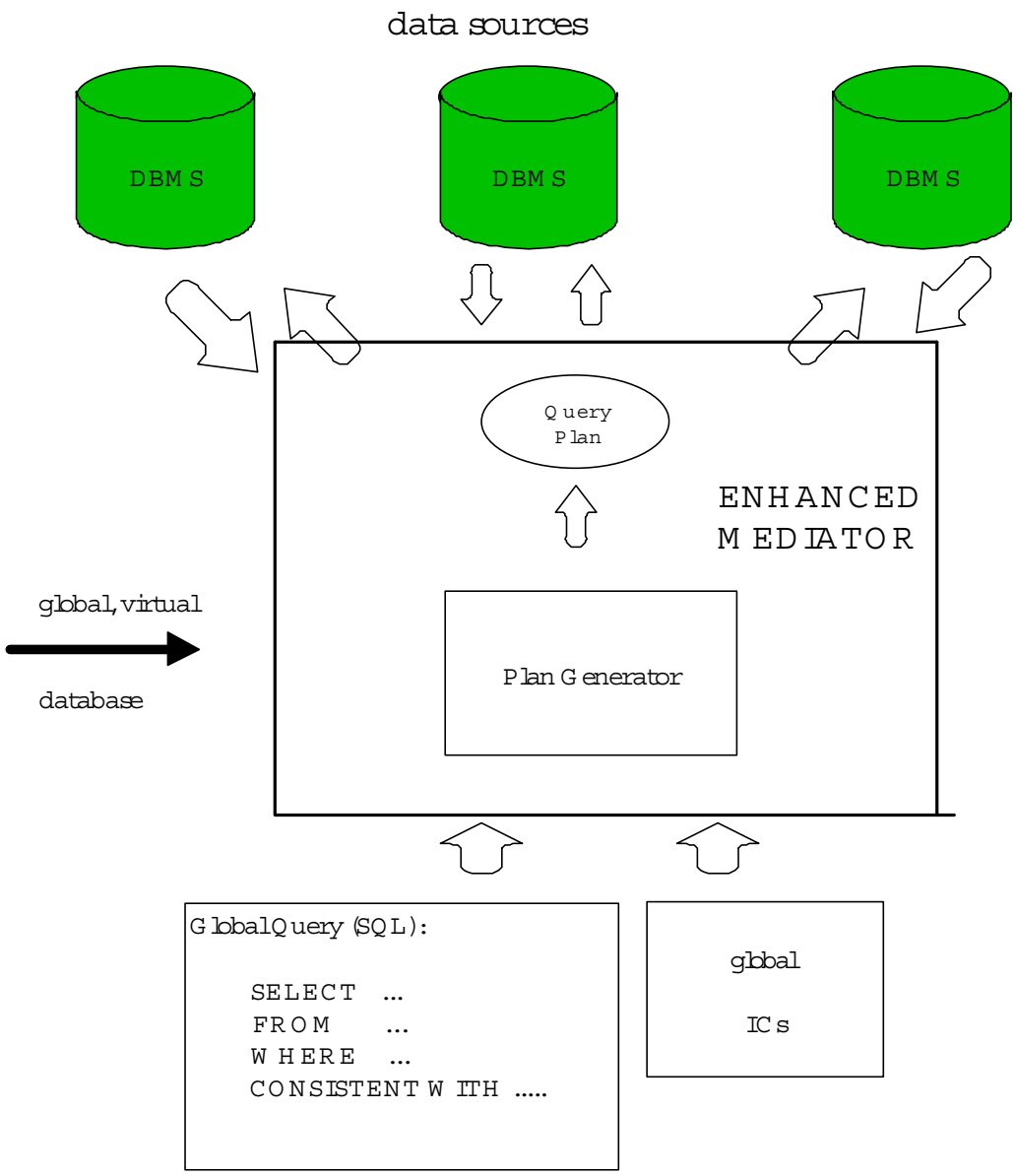
We would like to obtain those *answers to global queries that are consistent wrt to* IC

The query plan generated by the mediator should incorporate new elements which *enforce the satisfaction of the ICs by the query answers*



global ICs??

VS.


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

Apart from and before developing new algorithms for generating query plans, we need to characterize what is an answer to a query that is consistent wrt a set of ICs

ICs that may be violated by the database or the integration system

Otherwise, we will not be in position to evaluate the quality of our algorithmic solution

I we will not know for we are searching for ...

To solve the problem in virtual data integration systems, we can reuse some notions and techniques developed for retrieving consistent answers from single, stand alone, relational databases

INTERLUDE: Consistent Query Answering from Single Relational Databases

The Problem

We need to live with databases that are inconsistent

With information that contradicts given integrity constraints

There are many reasons, among them

- Inconsistency wrt integrity constraints that current commercial DBMS cannot check or maintain
- User constraints than cannot be checked

  A user wants or needs to impose his/her view of the world (semantics) on data that is out of his/her control
- Legacy data on which we want to impose (new) semantic constraints
- Integration of independent data sources (we saw ...)

It may be impossible/undesirable to repair the database (to restore consistency)

- No permission
- Inconsistent information can be useful
- Restoring consistency can be a complex process

The inconsistent database can still give us "correct" answers to certain queries!

Not all data participates in the violation of the ICs

What is "correct" ("consistent") information in an inconsistent database?

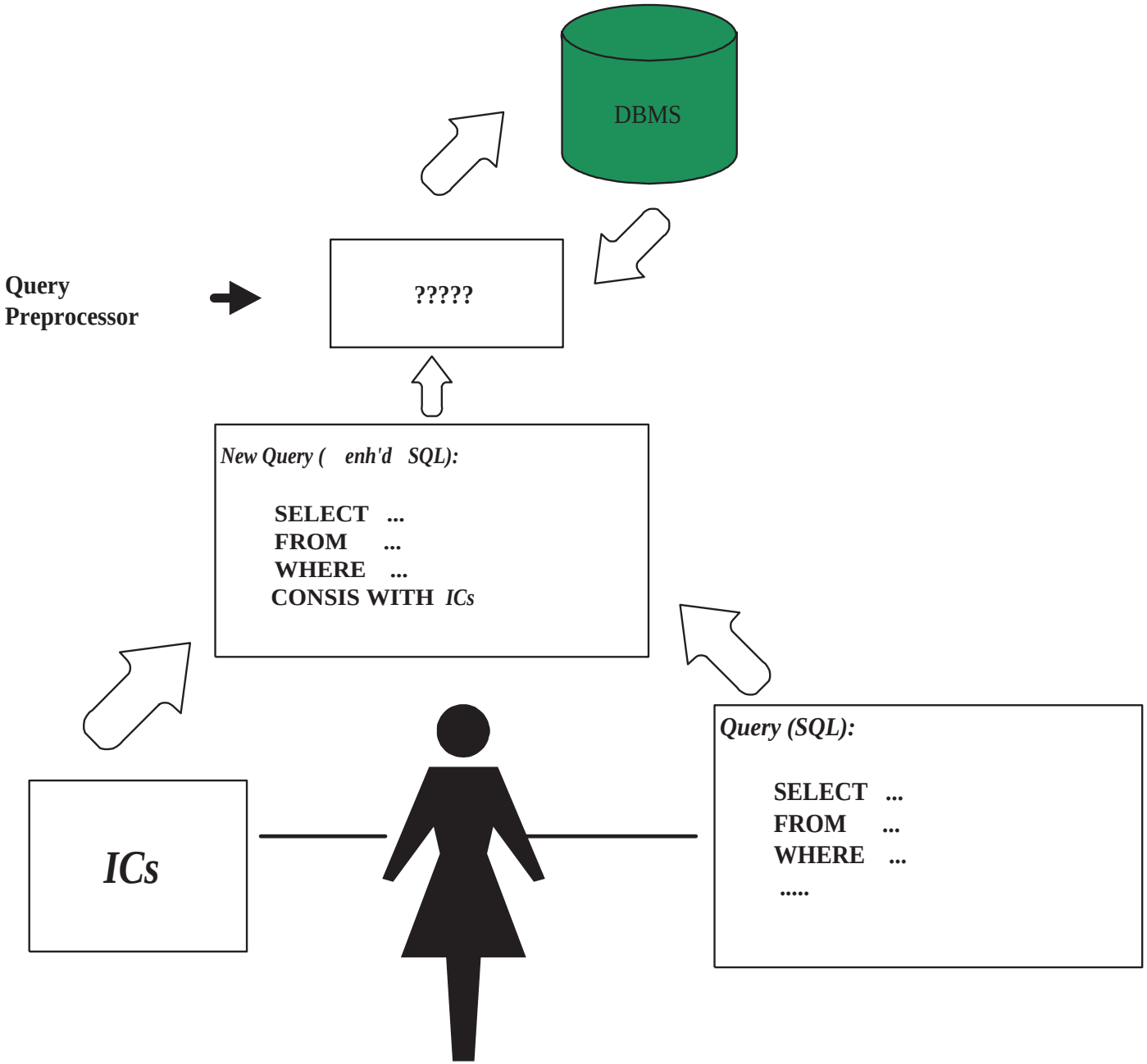
In particular, when we query the DB: what are the "correct answers"?

The research problem requires

- A precise characterization of consistent answers to a query in an inconsistent database

- Mechanisms for retrieving such consistent information from the the database

  Without changing the database ...


DBMS
Query
Preprocessor
?????
New Query (enh'd SQL):
SELECT ...
FROM ...
WHERE ...
CONSIS WITH ICs
ICs
Query (SQL):
SELECT ...
FROM ...
WHERE ...
.....

Consistent Answers

Given a database instance r, a query Q, and a set of ICs IC

Tuple $\bar{t}$ is a **consistent answer** to query Q in r wrt IC whenever $\bar{t}$ is an answer to Q in every *repair* of r

Where: a **repair** of a database instance r is a database instance r'

- over the same schema and domain
- satisfies IC
- differs from r by a minimal set of changes (insertions/deletions of whole tuples)

Intuitively, consistent answers are invariant under minimal ways of restoring consistency

We use repairs as an auxiliary concept, but **we are not interested in repairs per se**

We want to **compute** consistent answers, ideally without computing all repairs, but by querying the original instance r

(Arenas, Bertossi, Chomicki. ACM PODS'99)

Example: r inconsistent wrt $Name \rightarrow Salary$

Employee	*Name*	*Salary*
	V.Smith	3,000
	P.Jones	5,000
	P.Jones	8,000
	M.Stowe	7,000

Repair 1	*Name*	*Salary*
	V.Smith	3,000
	P.Jones	5,000
	M.Stowe	7,000

Repair 2	*Name*	*Salary*
	V.Smith	3,000
	P.Jones	8,000
	M.Stowe	7,000

In r it is consistently true that

- $Employee(M.Stowe, 7{,}000)$
- $Employee(P.Jones, 5{,}000) \ \vee \ Employee(P.Jones, 8{,}000)$
- $\exists X \, Employee(P.Jones, X)$

Addressing the Problem

Represent in a compact form the collection of all database repairs

Use disjunctive logic (answer set) programs

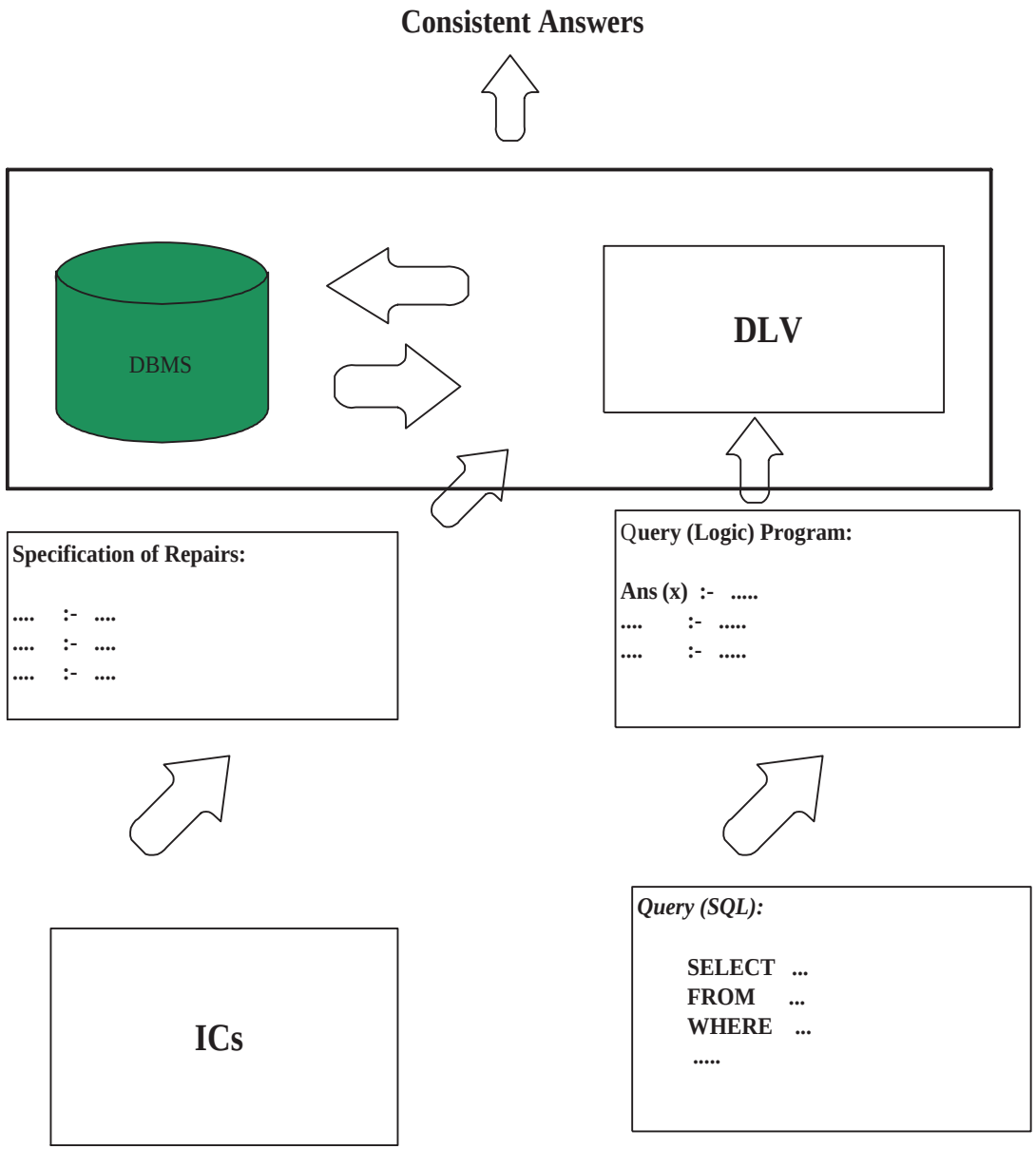
Repairs correspond to certain distinguished models of the program, the *stable models* basically

To obtain consistent answers to a FO SQL query:

- Transform (internally) the query into a logic program (standard)
- Run that program together with the program that specifies the repairs

Can be implemented on top of DLV, a logic programming system that computes according to the stable models semantics

(Arenas, Bertossi, Chomicki. TPLP 2003)
(Barcelo, Bertossi. NMR'02, PADL'03)


Consistent Answers
DBMS
DLV
Specification of Repairs:
.... :- ....
.... :- ....
.... :- ....
Query (Logic) Program:
Ans (x) :- .....
.... :- .....
.... :- .....
ICs
Query (SQL):
SELECT ...
FROM ...
WHERE ...
.....

Example: Full inclusion dependency

$$IC\colon\ \forall \bar{x}(P(\bar{x}) \rightarrow Q(\bar{x}))$$

Inconsistent instance $r = \{P(\bar{c}), P(\bar{d}), Q(\bar{d}), Q(\bar{e})\}$

The programs use annotation constants

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

Repair program $\Pi(r, IC)$:

1. $P(\bar{c}, \mathbf{t_d}) \leftarrow$
   $P(\bar{d}, \mathbf{t_d}) \leftarrow$
   $Q(\bar{d}, \mathbf{t_d}) \leftarrow$
   $Q(\bar{e}, \mathbf{t_d}) \leftarrow$

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

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

   ... the same for Q ...

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

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

   Passing to annotations $\mathbf{t}^\star$ and $\mathbf{f}^\star$ allows to keep repairing the DB wrt to all the ICs until the process stabilizes

Repairs must be *coherent*: use denial constraints at the program level, to prune some models

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

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

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

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

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

   Used to interpret the models as database repairs

The program has two stable models (and two repairs):

$\{P(\bar{c}, \mathbf{t_d}), ..., P(\bar{c}, \mathbf{t}^{\star}), Q(\bar{c}, \mathbf{f}^{\star}), Q(\bar{c}, \mathbf{t_a}), P(\bar{c}, \mathbf{t}^{\star\star}), Q(\bar{c}, \mathbf{t}^{\star}), Q(\bar{c}, \mathbf{t}^{\star\star}), ...\} \equiv$
$\{P(\bar{c}), Q(\bar{c}), P(\bar{d}), Q(\bar{d}), Q(\bar{e})\}$

$\{P(\bar{c}, \mathbf{t_d}), ..., P(\bar{c}, \mathbf{t}^{\star}), P(\bar{c}, \mathbf{f}^{\star}), Q(\bar{c}, \mathbf{f}^{\star}), P(\bar{c}, \mathbf{f}^{\star\star}), Q(\bar{c}, \mathbf{f}^{\star\star}), P(\bar{c}, \mathbf{f_a}), ...\} \equiv$
$\{P(\bar{d}), Q(\bar{d}), Q(\bar{e})\}$

Consistent answers to query $P(\bar{x}) \wedge \neg Q(\bar{x})$?

Run repair program $\Pi(r, IC)$ together with query program

$Ans(\bar{x}) \leftarrow P(\bar{x}, \mathbf{t}^{\star\star}), Q(\bar{x}, \mathbf{f}^{\star\star})$

Answer: $Ans = \emptyset$

Query: $Ans(\bar{x}) \leftarrow P(\bar{x}, \mathbf{t}^{\star\star})$

Answer: $Ans = \{d\}$

PART III (cont'd): Certain and Consistent Answers from Virtual Data Integration Systems

A Solution for Data Integration

(Bravo and Bertossi. IJCAI 2003)
(Bertossi,Cortes, Chomicki, Gutierrez. FQAS 2002)

Assumptions:

- Local-as-View (LAV)
  (more challenging than GAV and inconsistency issues more relevant)
- Conjunctive view definitions
- Open sources

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

Can be extended to clopen sources, and views defined as disjuntions of conjuntive queries

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

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

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

Legal instance: $D = \{(a,b),(c,d),(a,c),(d,e)\}$ (and supersets)

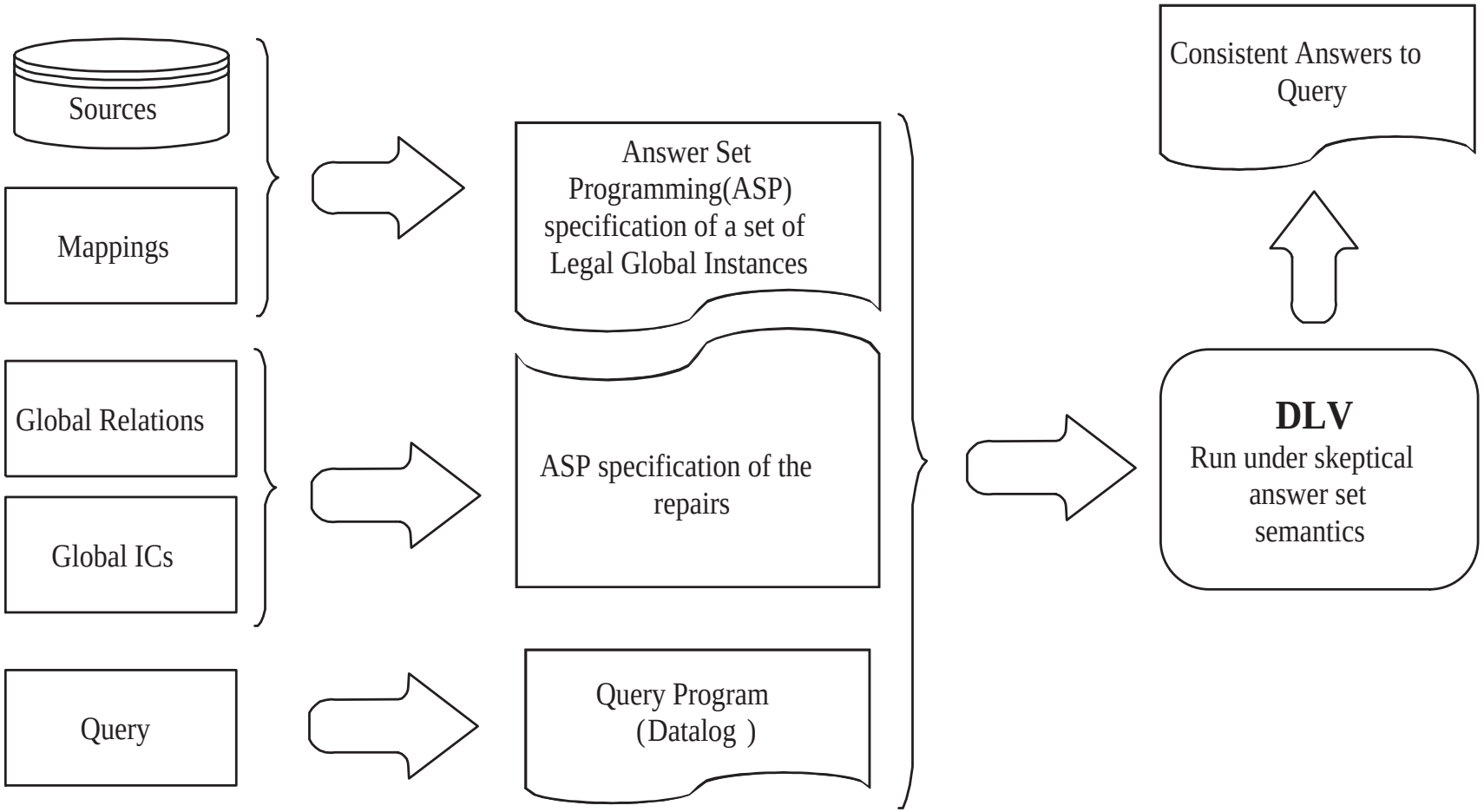
We found for query Q: $R(X,Y)$?

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

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

But global FD R: $X \rightarrow Y$ not satisfied by $D = \{(a,b),(c,d),(a,c),(d,e)\}$ (or its supersets)

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


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

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

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

Inverse rules:

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

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 $\mathcal{G}$
3. For every view (source) predicate V_i with definition

$$V_i(\bar{X}) \leftarrow P_1(\bar{X}_1), \ldots, P_n(\bar{X}_n), \quad \text{the rule}$$

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

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

$$F_i^l(\bar{X}, X_l) \leftarrow V_i(\bar{X}), dom(X_l), choice((\bar{X}), (X_l))$$

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

(Giannotti,Pedreschi,Sacca,Zaniolo. DOOD 1991)

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

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

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

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

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

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: $\mathcal{G}_3$:

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

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

However, the legal global instances corresponding to 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 chose other values $z \in \mathcal{D}$

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

A refined version of $\Pi(\mathcal{G})$ detects 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 V_i is not satisfied through other views

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

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

More general than any other algorithm for LAV ...

Consistent Answers

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 $D \in Repairs^{IC}(\mathcal{G})$: $D \models Q[\bar{t}]$ (i.e. $\bar{t}$ is an answer to Q in D)

Intuitively, consistent answers are invariant under minimal restorations of consistency

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

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

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

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

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

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

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

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

(repairs may not be legal instances)

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

Apply the ideas and techniques developed for single databases ...

Combine the programs that specify the minimal instances and the repair programs for single instances

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

Example: $\mathcal{G}_3$:

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

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

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

Repair Program using DLV syntax:

```
dom(a).  dom(c).  v1(a).  v2(a,c).  %begin refined 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,to) :- 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).
```

Stable models obtained with DLV: (parts of them)

$\mathcal{M}_1^r =$ {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 (non cyclic) 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: $\mathcal{G}_3$ 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. Repairs corresponding to the stable models of the program Π become extended with query atoms

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

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

Example: Repair program for first example ($\mathcal{G}_1$ with the FD) in this section

```
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) :-  R(X,Y,tss).                              %query subprogram
```

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

Ongoing and Future Work

- Several implementation issues, in particular in the case of most common SQL queries and constraints

  Specially those that are not maintained by commercial DBMSs

- Research on many issues related to the evaluation of logic programs for consistent query answering (CQA) in the context of databases
  - Optimization of the logic programs for CQA
  - Optimization of the access to the DB, to the relevant portions of it
  - Generation of "partial" repairs, relative to the ICs that are "relevant" to the query at hand

- Magic sets (or similar query-directed methodologies) for evaluating logic programs for CQA

- Efficient integration of databases (DB2) and logic programs (DLV)

Literature

General Papers and Surveys:

- Wiederhold, G. and Genesereth, M. The Conceptual Basis for Mediation Services. IEEE Expert, 1997, 12(5): 38-47.
- Lenzerini, M. Data Integration: A Theoretical Perspective. Proc. ACM Symposium on Principles of Database Systems (PODS), 2002, pp. 233-246.
- Levy, A. Combining Artificial Intelligence and Databases for Data Integration. Artificial Intelligence Today, Springer LNAI 1600, 1999, pp. 249-268.
- Levy, A. Logic-Based Techniques in Data Integration. Chapter in 'Logic Based Artificial Intelligence', J. Minker (ed.), Kluwer Publishers, 2000.
- Ullman, J.D. Information Integration Using Logical Views. Proc. 6th Int. Conf on Database Theory (ICDT), Springer LNCS 1186, 1997, pp. 19-40.
- Ullman, J.D. Information Integration Using Logical Views. Theoretical Computer Science 239(2): 189-210 (2000).

Query Answering in VDISs:

- Duschka, O. Query Planning and Optimization in Information Integration. PhD thesis, Stanford University, December 1997.
- Duschka, O., Genesereth, M. and Levy, A. Recursive Query Plans for Data Integration. Journal of Logic Programming, 2000, 43(1):49-73.

- Grahne, G. and Mendelzon, A. Tableau Techniques for Querying Information Sources through Global Schemas. Proc. of the Int. Conf. on Database Theory (ICDT), Springer LNCS 1540, 1999, pp. 332–347.
- Friedman, M., Levy, A., Millstein, T. Navigational Plans for Data Integration. Proc. Sixteenth National Conference on Artificial Intelligence (AAAI), AAAI Press, 1999, pp. 67-73.
- Grant, J. and Minker, M. A Logic-based Approach to Data Integration. Theory and Practice of Logic Programming, 2002, 2(3):323-368.
- Gryz, J. Query Rewriting Using Views in the Presence of Functional and Inclusion Dependencies. Information Systems, 1999, 24(7):597–612.
- Levy, A., Rajaraman, A. and Ordille, J. Querying Heterogeneous Information Sources using Source Descriptions. Proc. 22nd International Conference on Very Large Databases (VLDB), Morgan Kaufmann Publishing Co., 1996, pp. 251–262.
- Pottinger, R. and Levy, A. A Scalable Algorithm for Answering Queries Using Views. Proc. 26th International Conference on Very Large Databases (VLDB), Morgan Kaufmann Publishing Co., 2000, pp. 484–495.
- Pottinger, R. and Halevy, A. MiniCon: A Scalable Algorithm for Answering Queries using Views. VLDB Journal, 10(2-3): 182-198, 2001.

Complexity Issues in VDIS:

- Abiteboul, A. and Duschka, O. Complexity of Answering Queries Using Materialized

Views. Proc. ACM Symp. on Principles of Database Systems, 1998, pp. 254-263.
- Cali, A., Calvanese, D., De Giacomo, G., Lenzerini, M. On the Expressive Power of Data Integration Systems. Proc. ER 2002, pp. 338-350.

CQA in Single Relational Databases:

- Arenas, M., Bertossi, L., and Chomicki, J. Consistent Query Answers in Inconsistent Databases. Proc. 18th ACM Symposium on Principles of Database Systems (PODS), 1999, pp. 68–79.
- Arenas, M., Bertossi, L., and Chomicki, L. Answer Sets for Consistent Query Answering in Inconsistent Databases. Theory and Practice of Logic Programming, 3(4-5): 393-424, 2003.
- Barcelo, P., and Bertossi, L. Logic Programs for Querying Inconsistent Databases. Proc. International Symposium on Practical Aspects of Declarative Languages, pp. 208–222. Springer-Verlag, LNCS 2562, 2003.
- Barcelo, P., Bertossi, L. and Bravo, L. Characterizing and Computing Semantically Correct Answers from Databases with Annotated Logic and Answer Sets. Chapter in book 'Semantics of Databases', Springer LNCS 2582, 2003, pp. 1–27.
- Bertossi, L. and Chomicki, J. Query Answering in Inconsistent Databases. Chapter in book 'Logics for Emerging Applications of Databases', J. Chomicki, G. Saake and R. van der Meyden (eds.), Springer, 2003.
- Cali, A., Lembo, D., Rosati, R. On the Decidability and Complexity of Query An-

swering over Inconsistent and Incomplete Databases. Proc. ACM PODS 2003, pp. 260-271.
- Greco, G., Greco, S., and Zumpano, E. A Logic Programming Approach to the Integration, Repairing and Querying of Inconsistent Databases. Proc. 17th International Conference on Logic Programming, pp. 348–364. Springer-Verlag, LNCS 2237, 2001.
- Wijsen, J. Condensed Representation of Database Repairs for Consistent Query Answering. Proc. 9th International Conference on Database Theory (ICDT), pp. 378–393, Springer-Verlag, LNCS 2572, 2003.

CQA in VDIS:

- Bertossi, L., Chomicki, J., Cortes, A. and Gutierrez, C. Consistent Answers from Integrated Data Sources. In Flexible Query Answering Systems, Springer LNAI 2522, 2002, pp. 71–85.
- Bravo, L., and Bertossi, L. Logic Programs for Consistently Querying Data Integration Systems. Proc. International Joint Conference on Artificial Intelligence (IJCAI), 2003, pp. 10–15.
- Cali, A., Lembo, D., and Rosati, R. Query Rewriting and Answering under Constraints in Data Integration Systems. Proc. IJCAI, 2003.
- Lembo, D., Lenzerini, M., and Rosati, R. Source Inconsistency and Incompleteness in Data Integration. Proc. International Workshop Knowledge Representation meets Databases (KRDB), 2002.

- Eiter, T., Fink, M., Greco, G., and Lembo, D. Efficient Evaluation of Logic Programs for Querying Data Integration Systems. Proc. 19th International Conference on Logic Programming (ICLP), 2003, Springer LNCS, 2003.

Role of ICs in VDIS:

- Cali, A.; Calvanese, D.; De Giacomo, G. and Lenzerini, M. Data Integration Under Integrity Constraints. Proc. Conf. on Advanced Information Systems Engineering (CAISE), Springer LNCS 2348, 2002, pp. 262–279.
- Cali, A., Calvanese, D., De Giacomo, G., Lenzerini, M. On the Role of Integrity Constraints in Data Integration. IEEE Data Engineering Bulletin 25(3): 39-45 (2002).

Schema Mapping, Data Exchange, Peer-to-Peer, Incomplete Information:

- Doan, A., Domingos, P. and Halevy, A. Learning to Match the Schemas of Data Sources: A Multistrategy Approach. Machine Learning 50(3): 279-301, 2003.
- Fagin, R., Kolaitis, P., Miller, R. and Popa, L. Data Exchange: Semantics and Query Answering. Proc. Int. Conf on Database Theory (ICDT), Springer LNCS 2572, pp. 207-224.
- Fagin, R., Kolaitis, P., and Popa, L. Data Exchange: Getting to the Core. Proc. ACM PODS 2003, pp. 90-101.
- Grahne, G. Information Integration and Incomplete Information. IEEE Computer Society Bulletin on Data Engineering, September 2002, pp. 46-52.

- Halevy, A., Ives, Z., Suciu, D., and Tatarinov, I. Schema Mediation in Peer Data Management Systems. Proceedings of the International Conference on Data Engineering, ICDE, 2003.
- Halevy, A. Corpus-Based Knowledge Representation. Proc. IJCAI, 2003, pp. 1567-1572.
- Kementsietsidis, A., Arenas, M., Miller, R.J. Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. SIGMOD Conference 2003, pp. 325-336.
- Pottinger, R., and Bernstein, Ph. Creating a Mediated Schema Based on Initial Correspondences. IEEE Data Engineering Bulletin 25(3): 26-31 (2002).

Answering Queries Using Views:

- Flesca, S., Greco, S. Rewriting Queries Using Views. Transactions on Knowledge and Data Engineering 13(6): 980-995 (2001).
- Halevy, A. Theory of Answering Queries Using Views. SIGMOD Record 29(4): 40-47 (2000).
- Halevy, A.Y. Answering Queries Using Views: A Survey. VLDB Journal 10(4): 270-294, 2001.
- Millstein, T., Halevy, A., Friedman, M. Query Containment for Data Integration Systems. Journal of Computer and Systems Sciences, 66(1): 20-39 (2003).