Obtaining Consistent Answers from Databases

Leopoldo Bertossi

School of Computer Science

Carleton University

bertossi@scs.carleton.ca

www.scs.carleton.ca/~bertossi

The Context

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

  Each data source may be consistent and have an IC checking mechanism

  But the integrated (possibly virtual, mediated) global system not ...

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 Problem

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?

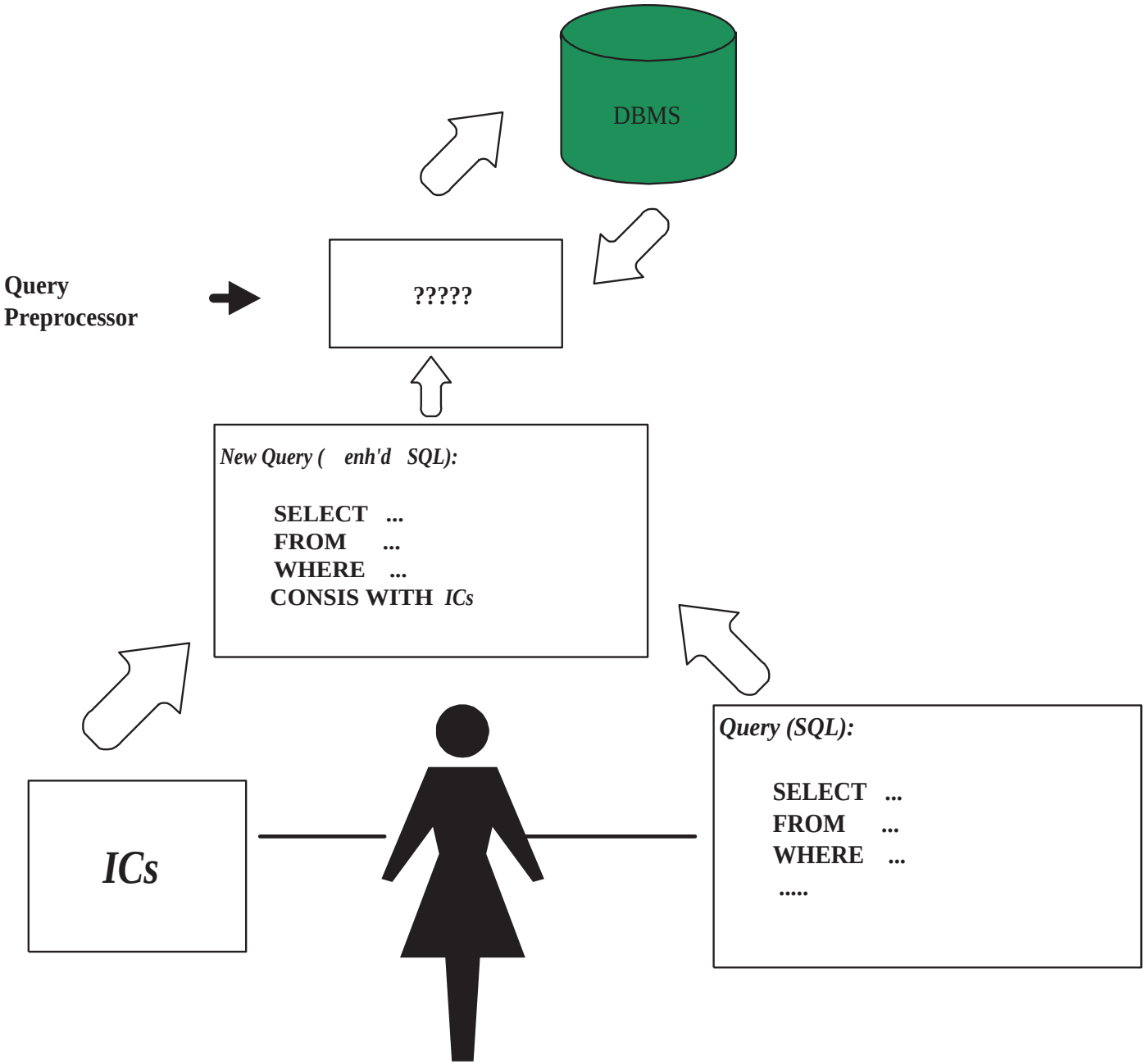
In particular, when we query the DB: what are the "correct answers"?

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 I

A computational mechanism to compute consistent answers

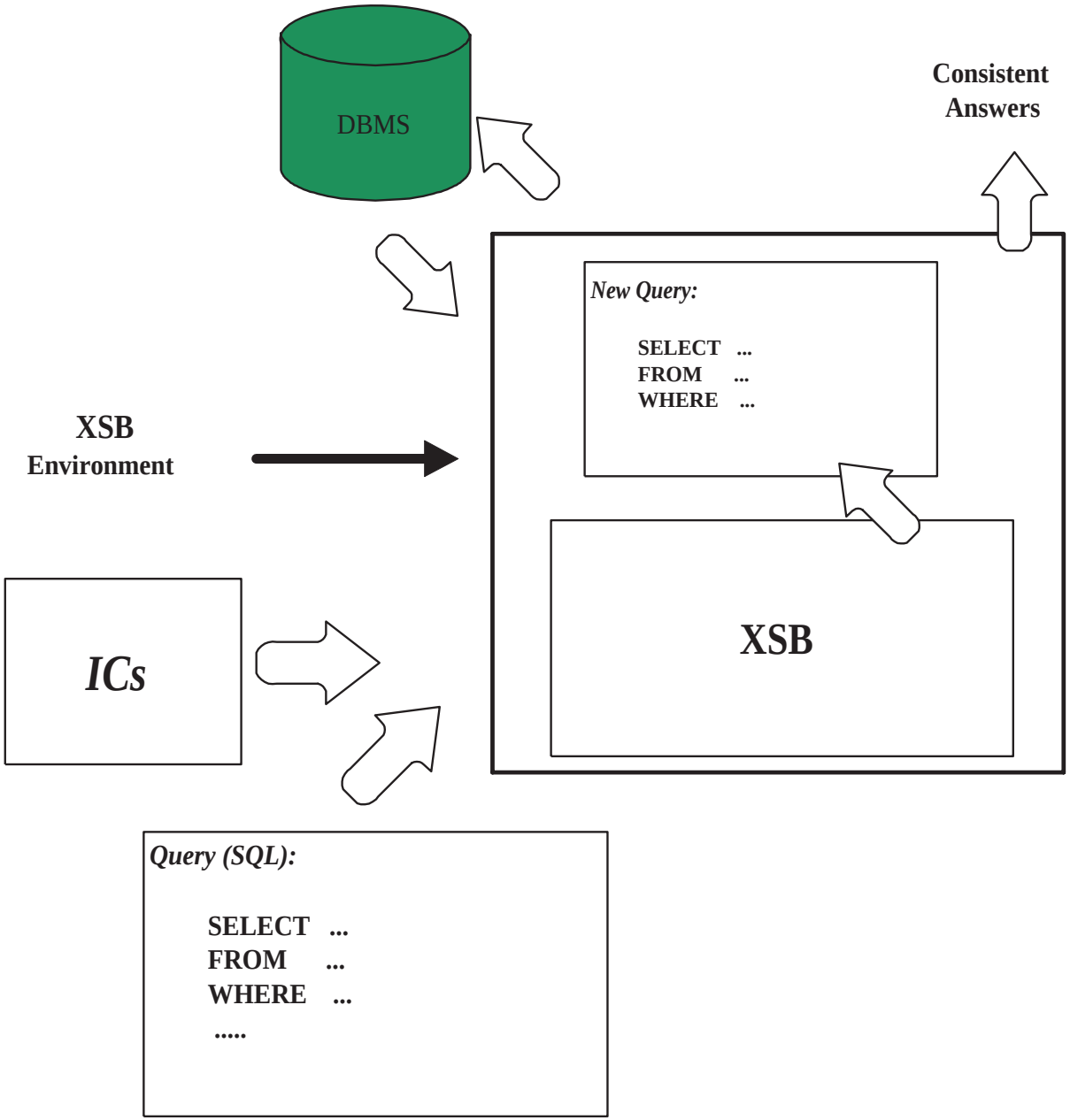
Does not produce the repairs

It queries the only explicitly available database instance

Query is **transformed** and posed as new query

Implementation on top of XSB, a deductive database system, connected to DB2 via ODBC

Input is an SQL query, the algorithm (implemented in XSB) produces a new SQL query that is posed to the DB2 DB


DBMS
Consistent
Answers
New Query:
SELECT ...
FROM ...
WHERE ...
XSB
Environment
XSB
ICs
Query (SQL):
SELECT ...
FROM ...
WHERE ...
.....

Example: The FD: $Name \rightarrow Salary$ can be written

$$\forall XYZ\ (\neg Employee(X,Y)\ \vee\ \neg Employee(X,Z)\ \vee\ Y = Z)$$

Query: $Employee(X,Y)$?

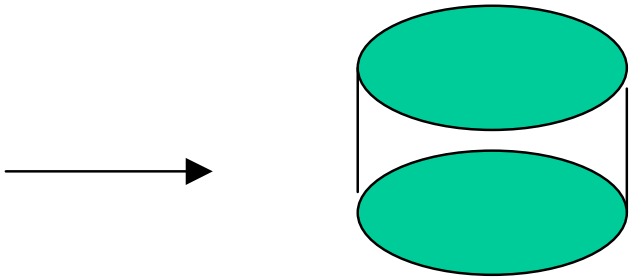
Consistent answers: $(V.Smith,\ 3{,}000),\ (M.Stowe,\ 7{,}000)$ (but not $(J.Page,\ 5{,}000),\ (J.Page,\ 8{,}000)$)

Can be obtained by means of the transformed query

$$T(Employee(X,Y)) := Employee(X,Y)\ \wedge$$
$$\forall Z\ (\neg Employee(X,Z)\ \vee\ Y = Z)$$

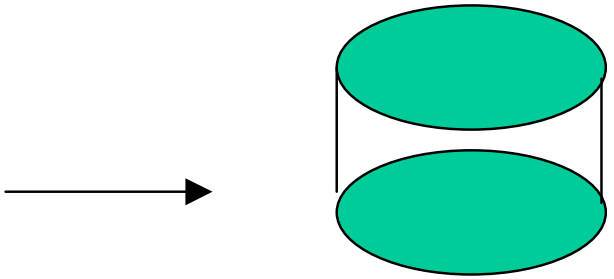
... those tuples (X,Y) in the relation for which X does not have and associated Z different from Y ...

```
SELECT  Name, Salary
FROM      Employee
CONSISTENT WITH
        FD(Name;Salary)
```



r

```
SELECT  Name, Salary
FROM      Employee  E
WHERE   Not exists (
        SELECT E.Salary
        FROM     E
        WHERE  E.Name = Name
           AND  E.Salary <> Salary)
```



r

Ordinary answers to new query are the consistent answers to the original query

[Arenas, Bertossi, Chomicki. PODS'99]

[Celle, Bertossi. DOOD'2000]

Methodology based on query transformation restricted to:

- Certain SQL queries, essentially conjunctions of DB tables

- Certain ICs, essentially universal ICs

  This covers most ICs found in DB praxis, except for referential ICs

- More expressive queries? Referential ICs?

Addressing the Problem II

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

To obtain consistent answers to a FO SQL query:

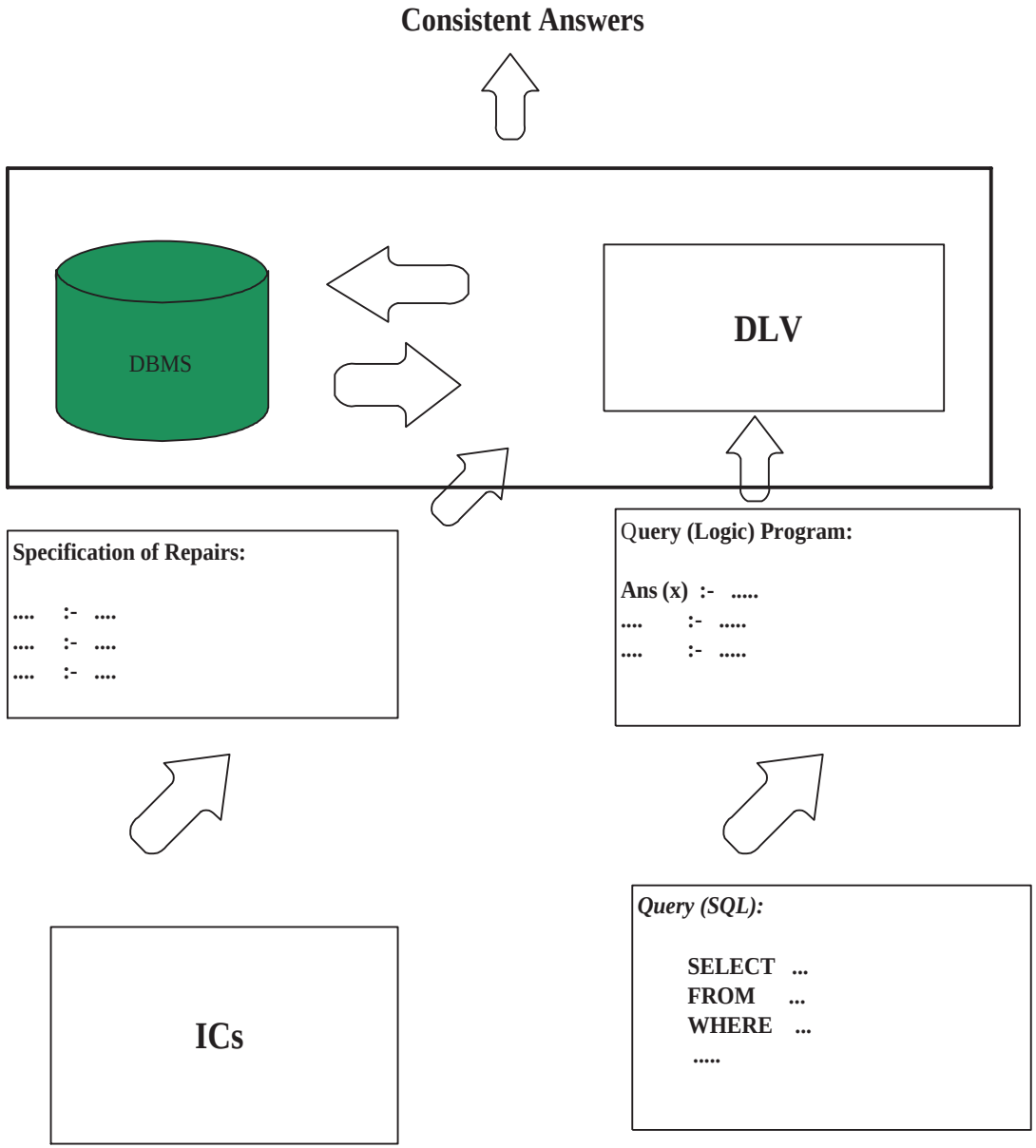
- Transform (internally) the query into a logic program (standard)

- Run that program together with the program that specifies the repairs

Can be implemented on top of DLV, a logic programming system with essentially a stable models semantics that computes the desired models

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

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

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

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

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

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

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

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

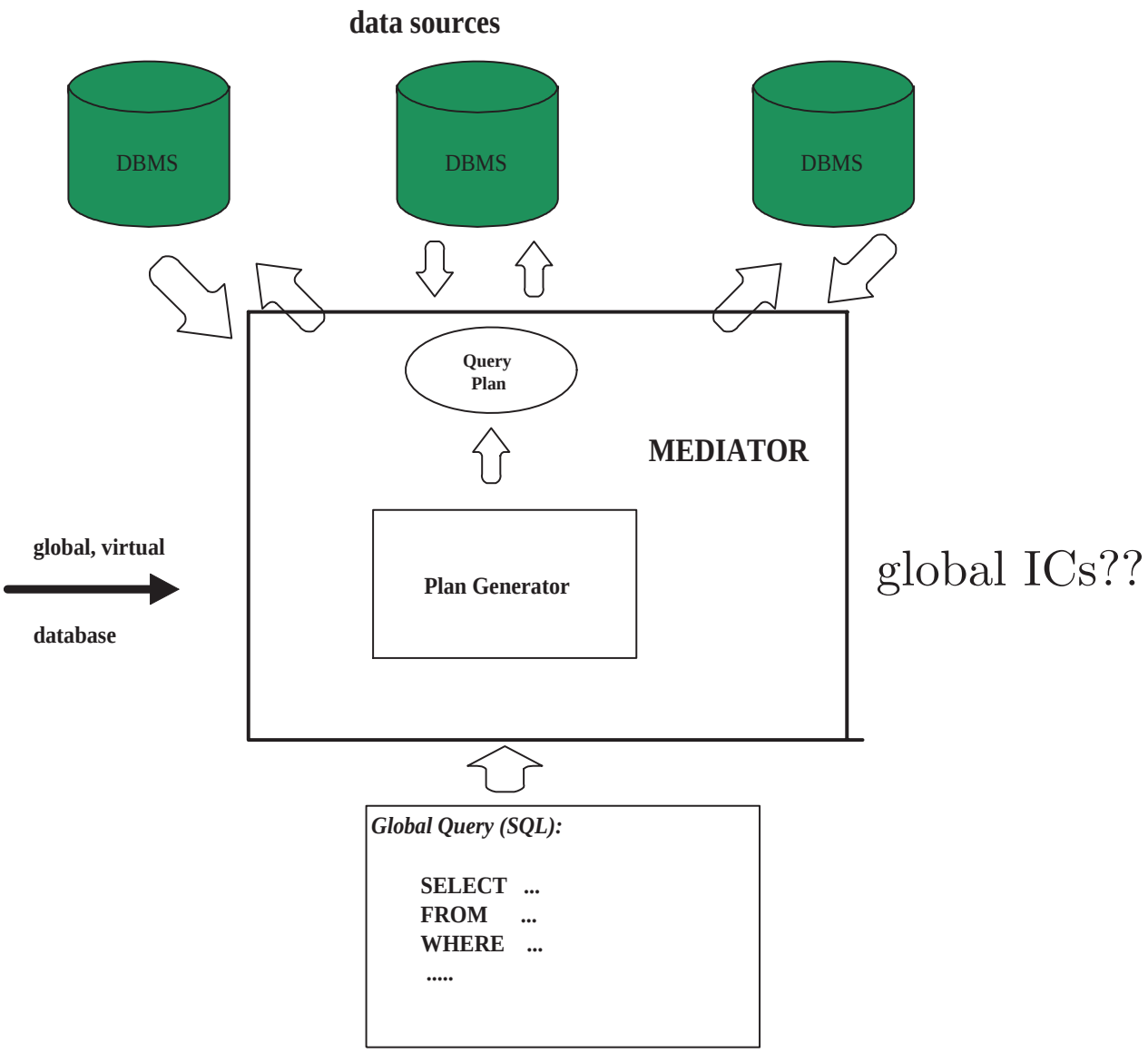
Answer: $Ans = \{d\}$

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

Given a (global) query Q to $\mathcal{G}$, one can generate a *query plan* that extracts and combines the information from the sources

Usually one **assumes** that certain ICs hold at the global level, and they are used in the generation of the query plan


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

global ICs??

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

They are not maintained at the global level

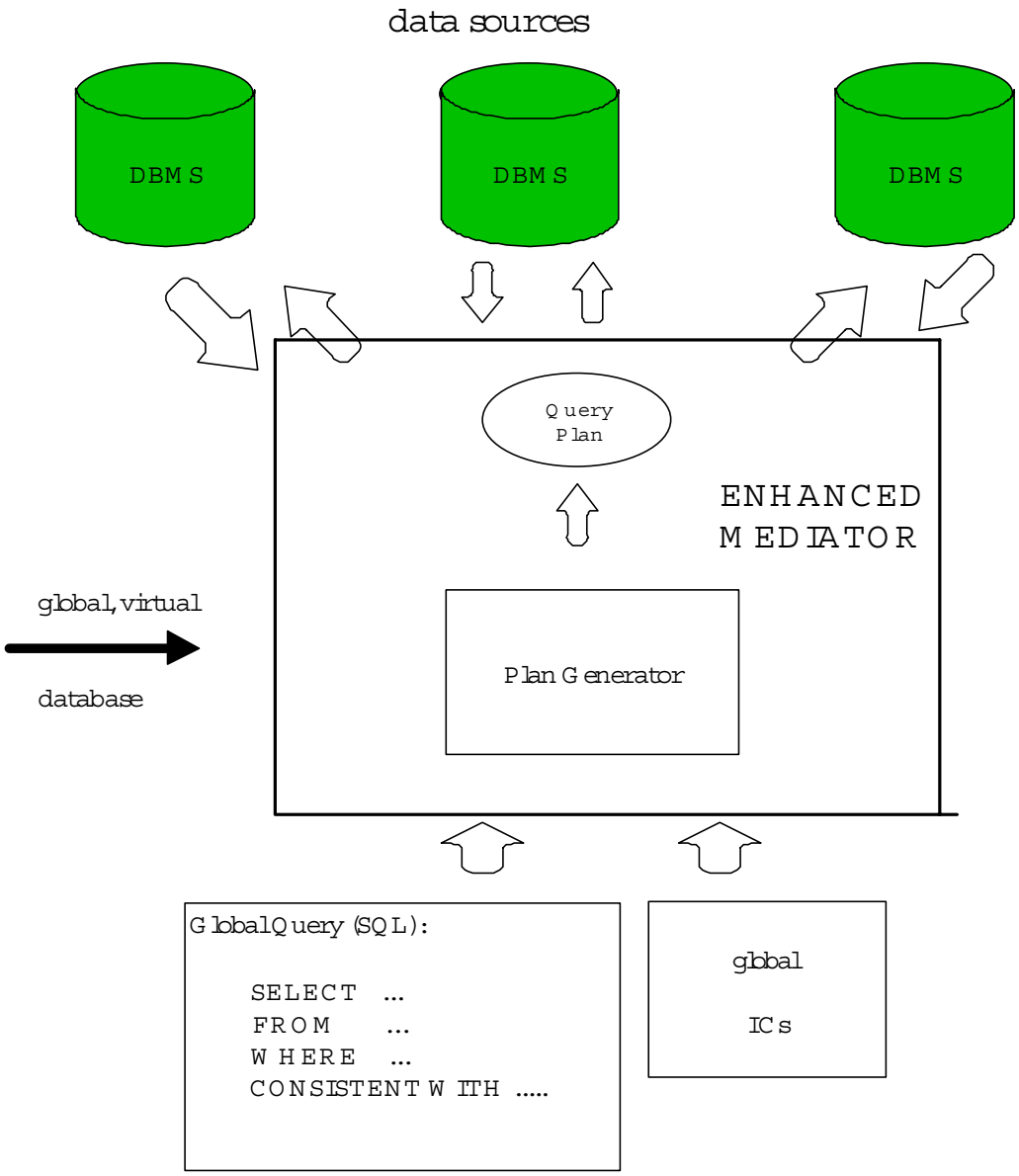
A natural scenario for applying our methodology: retrieve only information from the global database that is consistent with IC

New issues appear:

- What is a repair of the global, virtual database?

- How to retrieve consistent information from the global, virtual DB $\mathcal{G}$?

  At query time ...


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

A Solution for Data Integration

- Global schema $\mathcal{R}$
- Local Sources V
  - Local Schemas
  - Type: open, closed
  - Contents v
- Mapping:
  - Global-as-View (GAV)
  - Local-as-View (LAV)

- **Global-as-View (GAV)**: the global schema relations are described as views of the local relations

$$YearMovie(Title,\ Year) \leftarrow BD_1(Title,\ Director,\ Year)$$

$$YearMovie(Title,\ Year) \leftarrow BD_2(Title,\ Director,\ Year)$$

$$MovieReview(Title,\ Director,\ Review) \leftarrow BD_1(Title,\ Director,\ Year),\ BD_3(Title,\ Review)$$

- **Local-as-View (LAV*)**: each local source is described as a view of the relations of the global schema

$$V_1:\ BD_1(Title,\ Year,\ Director) \leftarrow Movie(Title,\ Year,\ Director,\ Genre),\ Canadian(Director),\ Year \geq 1960,\ Genre = Comedy$$

$$V_2:\ BD_2(Title,\ Review) \leftarrow Movie(Title,\ Year,\ Director,\ Genre),\ MovieReview(Title, Review),\ Year \geq 1990$$

A source can be:

- *open:* the source is incomplete
- *closed:* the source is complete (but may not be sound)
- *clopen:* the source is complete and sound

$$V_1 : \ BD_1(Title,\ Year,\ Director) \leftarrow Movie(Title,\ Year,\ Director,\ Genre),$$
$$Canadian(Director),$$
$$Year \geq 1960,\ Genre = Comedy$$

The *Legal Global Instances* are the ones that satisfy the mappings of the sources

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

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

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

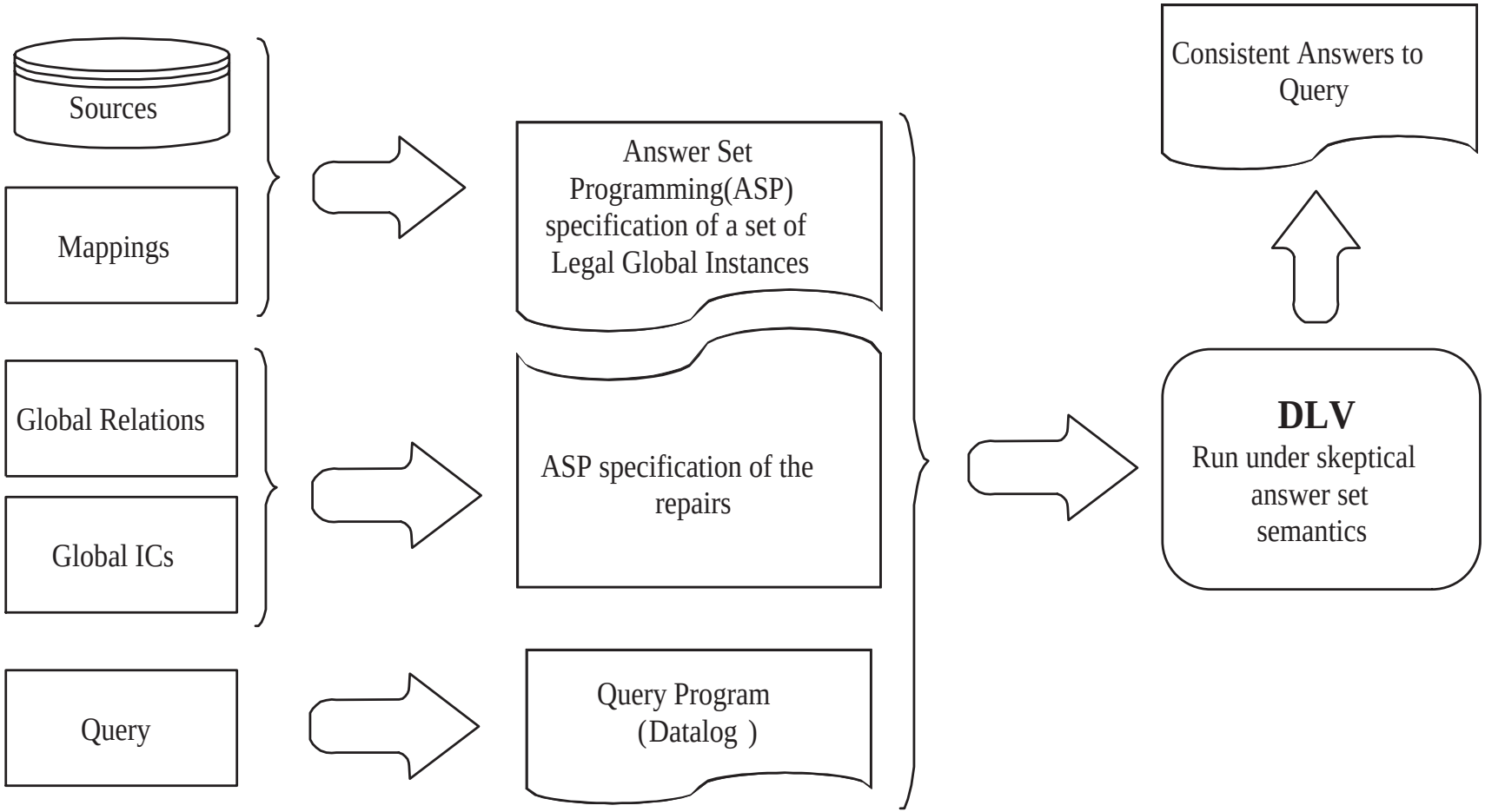
Supersets of D are all legal global instances; no subset of D is

Query Q: $\quad R(X,Y)? \Rightarrow Certain_{\mathcal{G}}(Q) = \{(a,b),(c,d),(a,c),(d,e)\}$

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

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

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


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

Repair Program for running Example:

```
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., $\{(c, d), (d, e)\}$

In [Bravo, Bertossi. IJCAI'03]:

It is assumed:

- LAV mapping
  - More challenging
  - Inconsistency issues are more interesting
- Open Sources

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

We are already extending it to consider clopen and closed sources

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 (XSB, DLV)

  * Some related research is being carried out in this direction by the developers of DLV (Vienna, Calabria, Roma)