Carleton
UNIVERSITY

Virtual Data Integration and Global Consistency

Leopoldo Bertossi

Carleton University
School of Computer Science
Ottawa, Canada

www.scs.carleton.ca/~bertossi

bertossi@scs.carleton.ca

Chapter 1: Introduction and Issues

Data in Different Forms

There is a large and increasing number of data sources

People and companies need to integrate data and the systems that handle that data

- Data in DBMSs: relational, OO, XML, ...
- Legacy data in different formats and systems
- Text files repositories
- Spread sheets
- Data on the Web

Database systems have to inter-operate, cooperate, and coordinate with each other

Data has to be shared, exchanged, integrated, ...

- In particular, it has to be reconciliated

  Syntactically: compatible formats and data types

  Semantically: compatible meanings

Data has to be queried

What if the data is spread out in different sources?

Some Forms of Data Integration

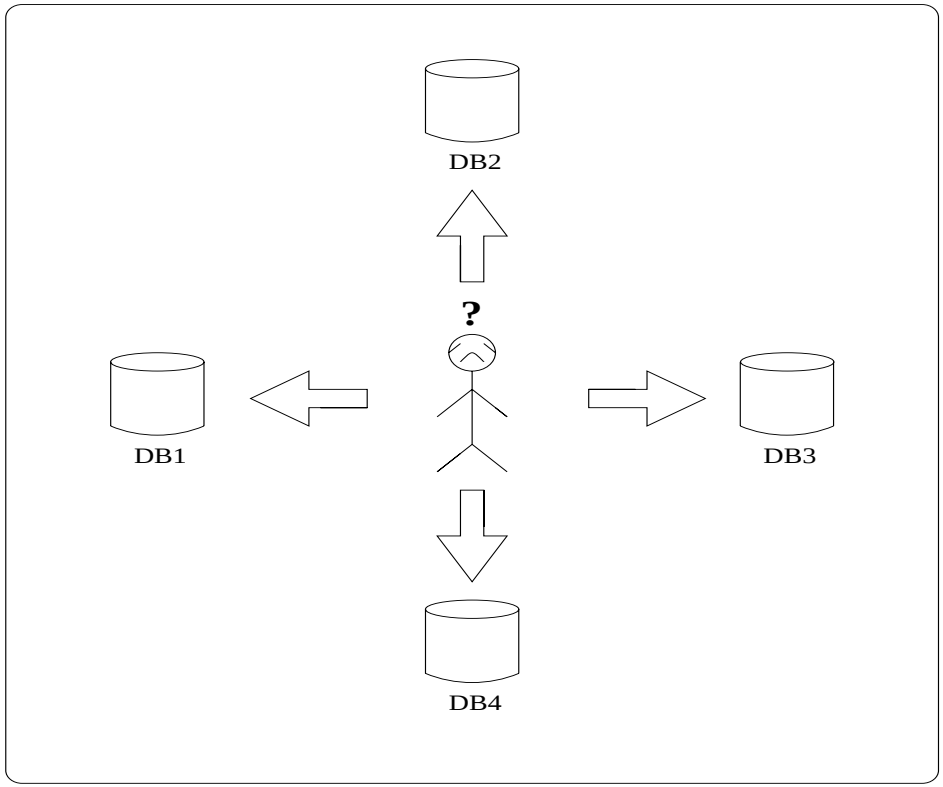
There are two main general approaches and paradigms for data integration

- Materialized: physical, integrated repository is created

  Data Warehouses: physical repositories of selected data extracted from a collection of DBs and other information sources

- Mediated: data stay at the sources, a virtual integration system is created ⟸

Mediator-Based Data Integration



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?

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

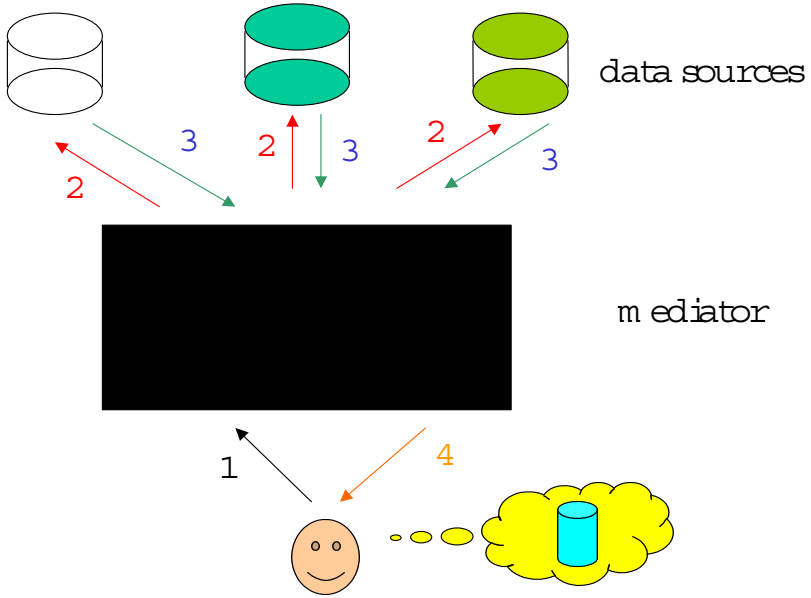
Independently of other "intrinsic" issues like:

- Incomplete data
- Redundant data
- Inconsistent data
- Different data formats, types, ...
- Different semantics
- Different forms of external data presentation
- Different and possibly limited query languages (if any)

  Common when interacting with the WWW: keyword search, fixed query patterns and templates, ...
- Restrictions on access to data

One way to go:

Use a mediator-based information integration system



A mediator is a software system that offers a common query interface to a set of heterogeneous information sources

We have:

- Collection of data sources that are independent and autonomous
- A virtual "database" is created which is accessed via the *mediator*
- A mediator that creates the illusion on users of being interacting with a real database
- Queries are posed and answered via the mediator

More precisely, the mediator:

- Accepts the participation of different data sources
- Contains information about the contents of the data sources
- Collects data from sources upon request; at query time
- Logically integrates the different data sources by means of a unifying, global or mediated schema
- Receives queries from users that are expressed in the language of the global schema
- In order to answer global queries, it sends appropriate queries to the sources
- Combines the answers received from the sources to build up the final answer to the user

Main Issues and Features Around VDISs

- Autonomy:

  Update operations on data sources via the mediator are not allowed

  Individual data source are updated in an independent and autonomous manner

  Sources do not necessarily cooperate with each other

  Data sources are mutually independent and may participate in different mediated systems at the same time

- Data location, presentation, organization, flexibility:

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

  The interaction with and within the system is realized through queries (and answers to them)

  System allows sources to get in and out

  Set and number of participating sources should be flexible and open

  Data sources have their own schema, the virtual database has its own data presentation schema

  Database schemas are a special kind of metadata, i.e. data about data

  In this case, schemas say how the data is logically structured

Mediator has to know what kind of data is offered by the sources and how they relate to the unifying global schema

A problem of describing data, data schemas, and specifying mappings between data schemas

At this stage the problem of semantic heterogeneity can be addressed

Description formalism should be expressive, computationally easy to use, and easy to maintain

Establishing the correspondence between the schemas of the sources and the global schema is a form and instance of metadata management

Metadata management appears in different forms in all the subjects around data integration and related subjects

- Problems with data:

  Global system is responsible for solving/addressing problems with data, like:

  Redundancy: to avoid unnecessary computations

  Complementarity: data of the same kind may be spread through different sources and has to be detected and combined

  Consistency: 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

Traditional data cleaning or consistency restoration techniques are not applicable here: data at the sources cannot be changed

Existing VDISs offer almost no support for consistency handling

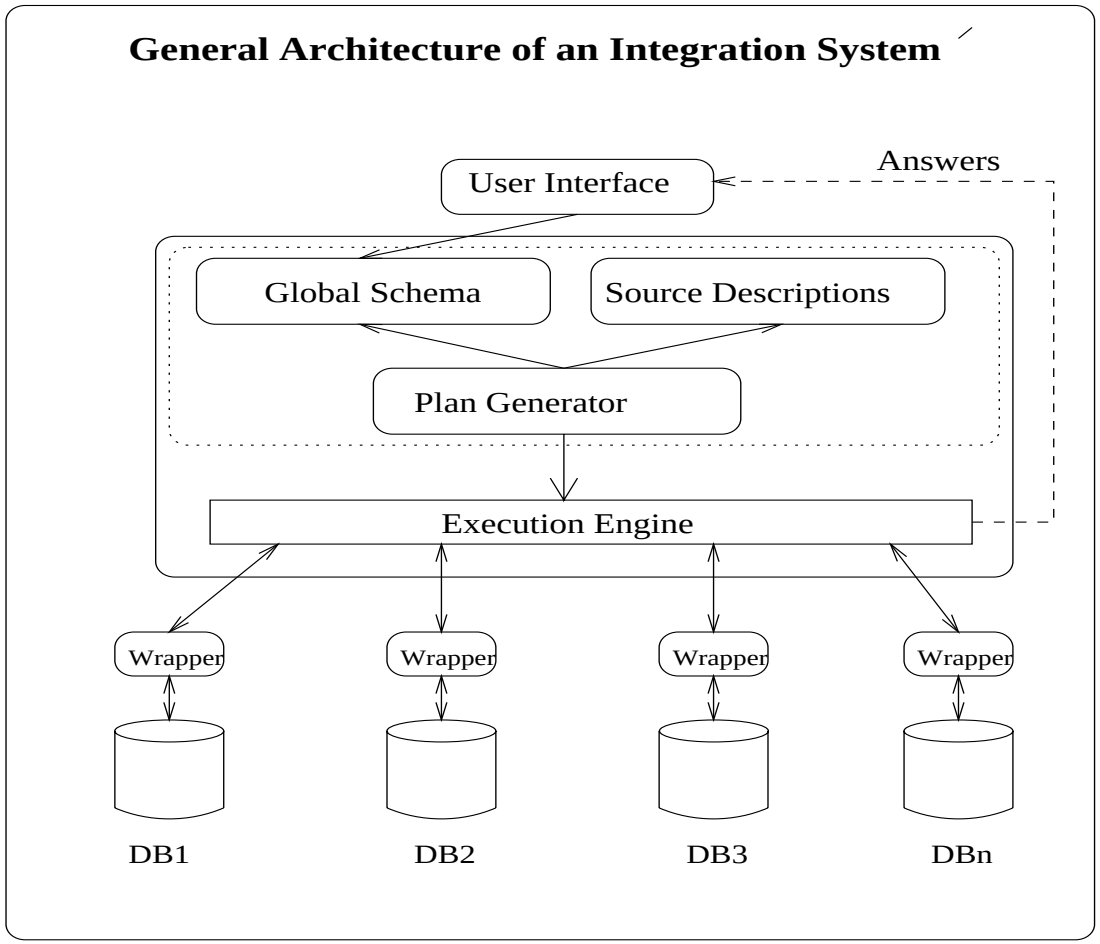
Many interesting research issues here ...

Even commercial DBMSs for stand alone databases (no integration) offer limited *general purpose support* in this direction

Consistency maintenance has to be achieved at the application level: rejection or compensation of updates via triggers or application programs

Chapter 2: Components of a Virtual Data Integration System

General Architecture

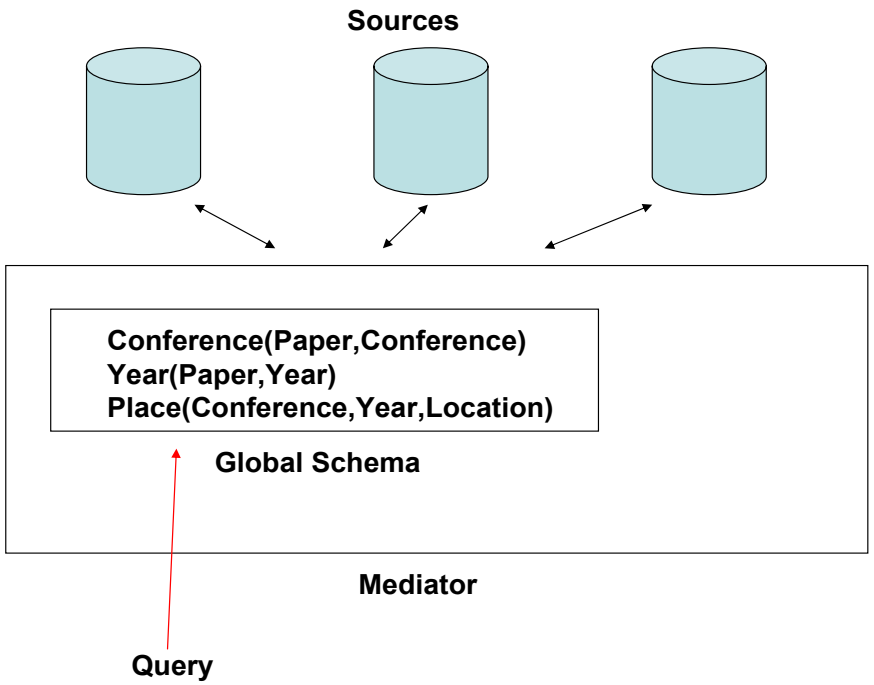


Main components of a mediator-based VDIS

Global or Mediated Schema:

- Schema used to present and export the data from the VDIS
- For example, if it is a relational schema, then it is a set of names for relations (tables), their attributes, etc.
- Application dependent
- Like in a normal, usual relational DB, from the user point of view
- The DB "instance" corresponding to the global schema is virtual
- User poses queries in terms of the relations in the global schema

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



$Conference(Paper, Conference), \quad Year(Paper, Year),$
$Place(Conference, Year, Location)$

User wants to know where conference PODS'89 was held

Query to global system: a simple selection

```
SELECT Location
FROM Place
WHERE conference = 'pods' AND year = '1989';
```

Or: $\Pi_L(\sigma_{C=pods,Y=1989}(\texttt{Place}(\texttt{C},\texttt{Y},\texttt{L})))$

Or using a rule based query language, like Datalog:

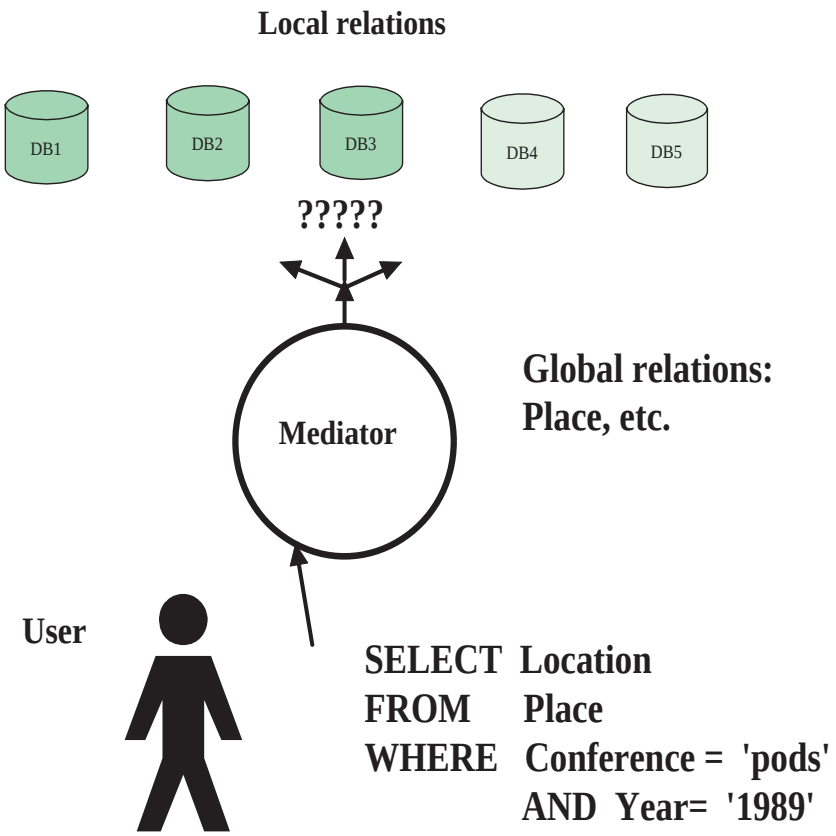
$$Q: \quad Ans(L) \leftarrow Place(pods, 1989, L)$$

Predicate Ans contains the answers, that are obtained by computing the RHS (the body) of the rule

But data is not in table $Place$

A query plan has to be generated to:

- identify relevant data sources
- identify relevant data in them
- determine (sub- or local) queries to be sent to the sources
- combine the answer sets obtained from the sources into one final answer set


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

Description of the Sources:

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

- 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 the formulas used to express queries and define views in terms of base tables in a relational DB

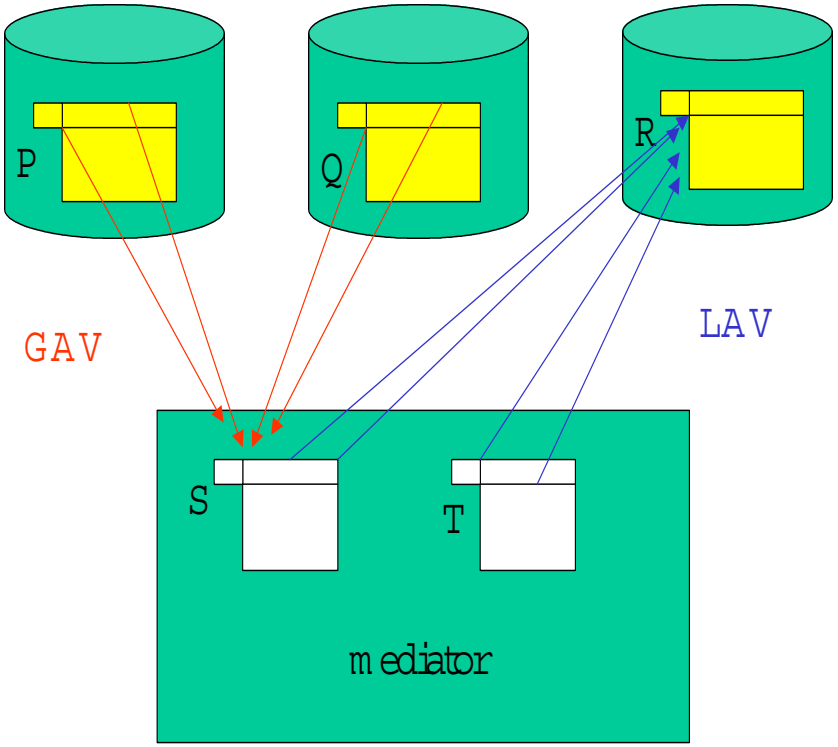
  We use logical formulas, usually incarnated as SQL queries or SQL view definitions, or, preferably, Datalog

- Those formulas define or are the mappings between the global schema (or "data") and the local schemas (or data)
- How data are described will heavily determine how query plans are computed

  And how complex it is to obtain query answers

  And also how flexible the system becomes

Main classical approaches to describe mappings:



- 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

A few obvious problems:

- GAV: Sources may contain more (and unnecessary) details wrt the mediated schema

  And still whole source relations have to be used in the definition (maybe with many projections)

- LAV: Mediated schema may contain details than are not in a source; same problem as in previous item

  Or too few details at the global level and still a whole relation at the source level has to be described

  How are those unrepresented attributes or fields "filled" with data?

To address these problems, a more recent approach:

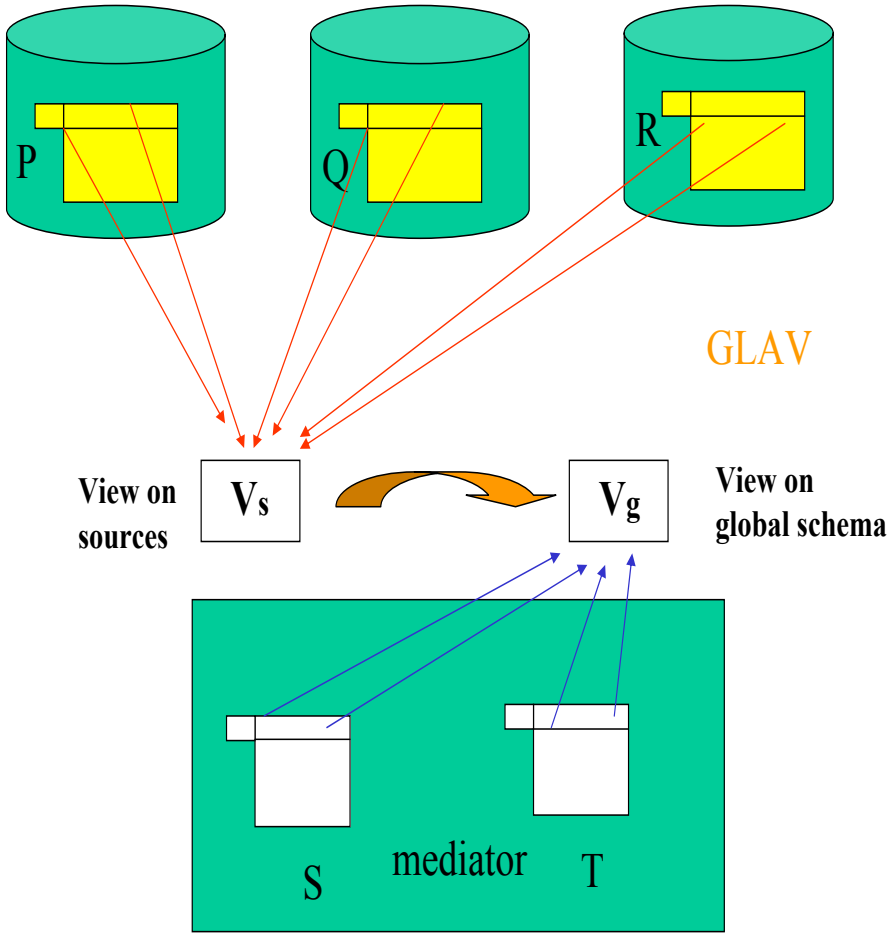
- Global-and-Local-as-View (GLAV): A correspondence between views on the global schema and views over the local schema are established and described

  Under certain assumptions on the sources, GAV and LAV can be obtained as special cases

  Gives more expressive power, more natural source descriptions, and more natural mediated schemas

M. Friedman, A. Levy, T. Millstein. Navigational Plans for Data Integration. Proc. AAAI'99.

The mappings (arrows in the pictures) require a language and a semantics ... Coming ...


P
Q
R
GLAV
View on sources
Vs
Vg
View on global schema
S
mediator
T

Plan Generator:

- It gets a user query in terms of global relations
- It uses the source descriptions and rewrites the query as a query plan

  Which involves a set of queries expressed in terms of local relations
- This is the most complex part
- Rewriting depends on mappings (LAV, GAV, or GLAV)
- Query plan includes a specification as to how to combine the results from the local sources
- At the stage of plan generation potential inconsistencies should be addressed; "anticipating solving them in advance", on-the-fly, when the plan is being generated ...

Description of Data Sources

Given as logical formulas in terms of relations in the global schema and the relations in the data sources (or their wrappers)

Since under any of the approaches mentioned above (GAV, LAV, GLAV) we are defining views, we will use a common language to define views in relational databases: Datalog formulas for view definitions

Example: A relational database with base relations

$$Employee(Id, Name, Position), \quad Salary(Id, Amount)$$

The definition of a view that gives the employee names and salaries

$$NameSal(x, y) \leftarrow Employee(u, x, v), Salary(u, y)$$

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

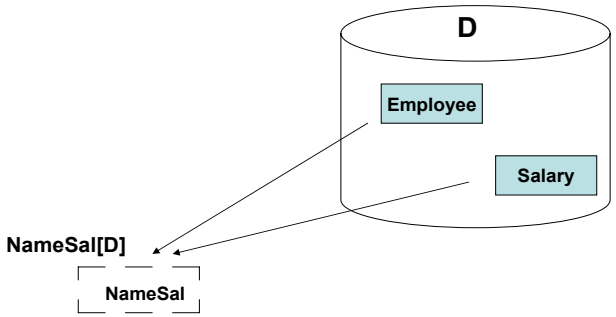
The values of variables in the body that do not appear in the head are projected out after computing a join via Id (variable u) of the two base relations

View is defined as a conjunctive query over the base relations

Beware: The semantics of Datalog rules does not follow classic predicate logic semantics

The tuples in the (usually virtual) extension of the view are only those that can be obtained by propagating body to head

Given an instance D for the base schema, we obtain a possibly virtual extension $NameSal[D]$ for the view



We will keep using the Datalog language for defining views, but being careful about their semantics in the context of virtual data integration

An Example of GAV

Relations in global schema are described as *views* of the relations in the (union of the) local schemas

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

Since global relations are virtual and local sources are material, it is conceptually as usual ...

Example: We have data sources with different kinds of data about movies

We integrate those data sources into a mediated system

New sources may be available later, some may disappear, ...

As modelers of the integration system, we decide what common data interface or schema we want to offer to the outside world

Local sources:

- $DB_1(Title, Dir, Year)$
- $DB_2(Title, Dir, Year)$
- $DB_3(Title, Review)$

The first two are complementary, the third of a different, but related kind

A global relation "containing" movies and their years:

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

A disjunctive view (defined as a disjunction of conjunctive queries)

Defined by two Datalog rules

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

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

Another global relation containing movies, their directors and reviews:

$$MovieRev(Title, Dir, Review) \leftarrow$$

$$DB_1(Title, Dir, Year), DB_3(Title, Review)$$

A view defined by, first, the join of DB_1 and DB_3 over attribute $Title$, and then a projection on $Title, Dir, Review$

A view defined by a conjunctive query, i.e. expressed in terms of

- conjunctions (or joins)
- projections

An important, useful, common, and well-studied class of queries

Using Datalog for view/query definitions makes use of recursion and syntactic processing of query/view definitions easier

And plan generation will rely on syntactic transformation of view definitions and queries ...

How to pose queries to the integration system?

In a language based on the mediated, global schema

Query: *Movies shown in year 2001, with their reviews?*

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

Query is expressed in terms of the global schema

There is no data in the global "DB"

The data has to be obtained from the sources

What are the intended, correct, expected answers to the query?

How can they be computed?

The first question should be answered by providing the semantics of a virtual data integration system under GAV

It is the semantics that determines the correct answers (coming)

But, how could we proceed to obtain answers following our first natural impulse?

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

The query can be rewritten in terms of the source relations

This is simple under GAV: rule unfolding

"Unfold" each global relation into its definition in terms of the local relations

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

These new queries do get answers directly from the sources
Final answer is the union of two answer sets, one for each rule

Under GAV it seems to be easy to obtain (correct?) answers to global queries

On the other side, if new sources join in the system or older leave it, definitions of global relations as views have to be rewritten

Not very flexible ...

An Example of LAV

Each table in each local data source is described as a view in terms of global relations

Somehow unnatural:

- From the conceptual point of view
- From perspective of usual databases practice

  Here, views contain data, but "base tables" don't

But this approach has some advantages

It also makes sense:

- A designer of a virtual, mediated system defines its own schema

  The way data will be offered to users

  Invites potential contributors of data to participate

  May not know who are or will be potential participants

  Or how their data is logically structured

- The latter have to describe their local relations in terms of the global relations (that are fixed)

  And pass the descriptions to the mediator

  Independently from other sources or contributors!

Example: Global schema offered by mediated system $\mathfrak{G}$:

$Movie(Title, Year, Director, Genre), AmerDir(Director),$
$Review(Title, Review)$

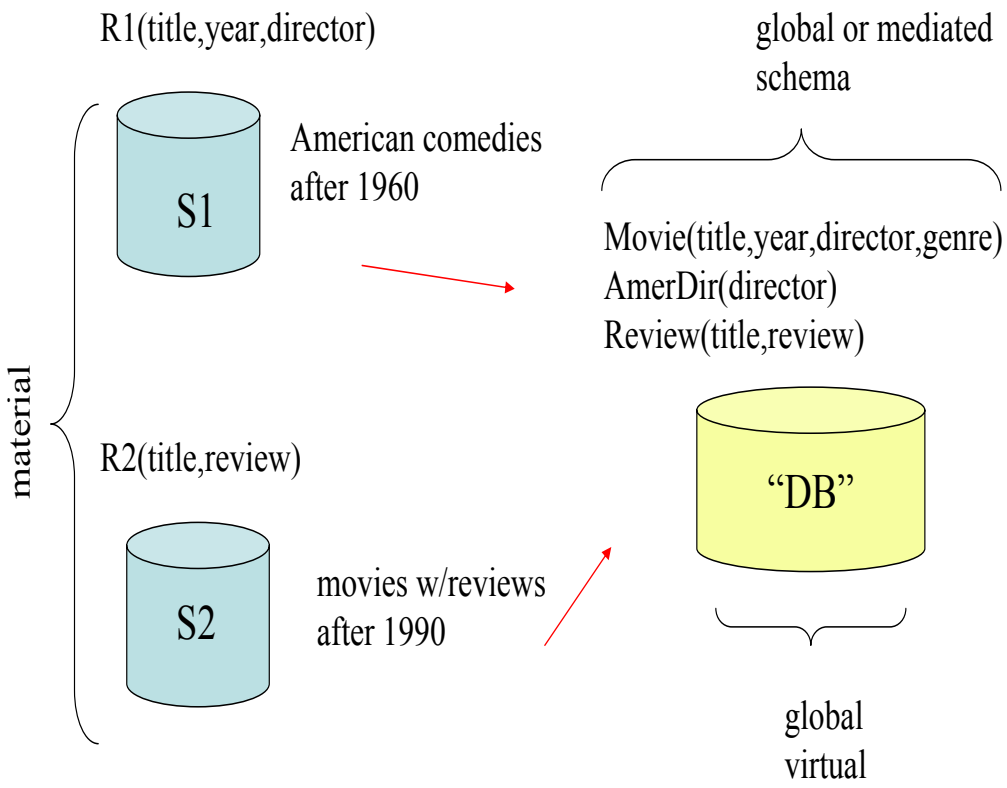
Sources S_1, S_2 are defined as views by means of conjunctive queries with built-ins:

$$S_1: \quad V_1(Title, Year, Director) \leftarrow$$
$$Movie(Title, Year, Director, Genre),$$
$$AmerDir(Director), Genre = comedy,$$
$$Year \geq 1960.$$

S_1: Has a relation V_1 containing comedies, filmed after 1960, with American directors and their years

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

S_2: Has a relation V_2 containing movies filmed after 1990 with their reviews, but no directors


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

- Definition of each source does not depend on other sources!

  Sources can easily leave the system or join in

  Other sources' definitions are not affected

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

  In this sense, information in S_2 could be "incomplete" wrt what $\mathfrak{G}$ "contains" (or might contain)

  So, S_2 could containing only a part of the information of the same kind in the global system

  This is the most common scenario: sources are incomplete

Query to $\mathfrak{G}$: *Comedies with their reviews produced since 1950?*

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

Query expressed in terms of mediated schema, as expected

Information is in the sources, now defined as views

Not possible to obtain answers by a simple, obvious or direct computation of the RHS of the query

No simple rule unfolding for the relations in the body: no definitions for them as in GAV

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 of how to combine their answers (which is needed here)

This is a query plan for our query: (we'll come back to this ...)

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

Query is rewritten in terms of the views; and 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

This is the best or most we can get from the sources

The plan is maximally contained in the original global query

Something that can be made precise and established after defining the semantics of the system

Chapter 3: Semantics of Virtual Data Integration Systems

Idea behind the Semantics of a VDIS

When we pose queries to a VDIS, we expect to receive answers

What answers are we talking about?

What are the correct answers?

It depends on the semantics of the system

So, what are the semantically correct answers?

Notice that there is no global instance and then no answer to a global query in the classical sense

We proceed by indicating what are the intended models of the system

More precisely, what are the intended global instances of the integration system

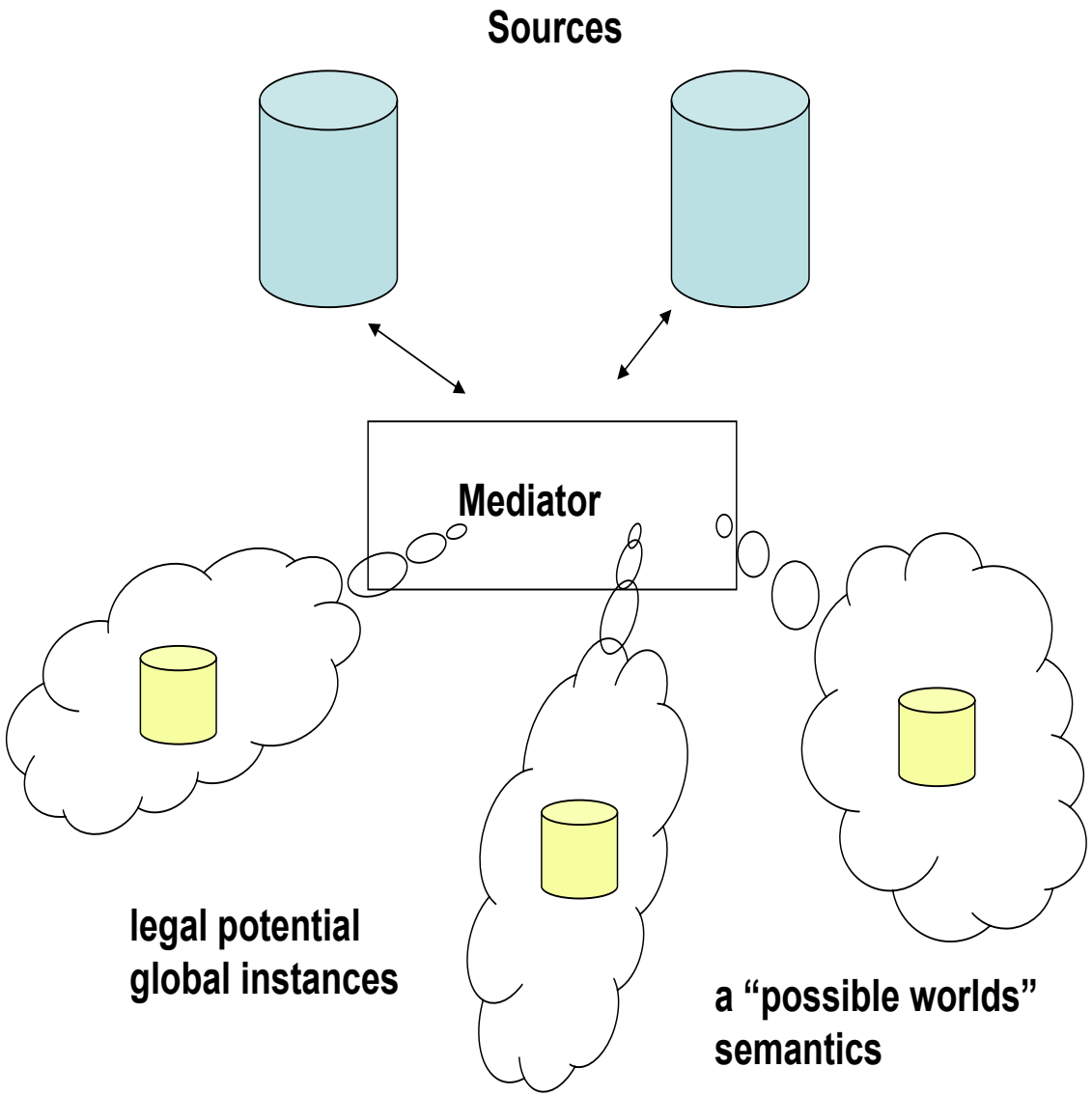
In general, we are not going to materialize those global instances

But the set of admissible, legal global instances will give a meaning to the system

Global instances (of the global schema) must satisfy the constraints and parameters of the problem, e.g.

- Source descriptions or mappings (e.g. LAV or GAV)
- Contents of the data sources
- Character of the sources, e.g. incomplete, complete, ... data
- Global ICs $\times$

We will assume there are no global integrity constraints


Sources
Mediator
legal potential
global instances
a “possible worlds”
semantics

Semantics under GAV

Example: Source relations:

$S_1(Title, Dir, Year)$, $S_2(Title, Dir, Year)$, $S_3(Title, Review)$

We can see the collection of local relations as forming a single source schema $\mathcal{S}$

For each local relation $S \in \mathcal{S}$, we have an extension s

And the extensions s as forming a single source instance I

Global relations:

$$
\begin{aligned}
MovieYear(Title, Year) &\leftarrow S_1(Title, Dir, Year) \\
MovieYear(Title, Year) &\leftarrow S_2(Title, Dir, Year) \\
MovieRev(Title, Dir, Review) &\leftarrow S_1(Title, Dir, Year), \\
&\qquad\qquad S_3(Title, Review)
\end{aligned}
$$

All this determines a global (mediated) integration system $\mathfrak{G}$, whith a global schema $\mathcal{G}$

The elements of $\mathcal{G}$ can be seen as views over $\mathcal{S}$

And each global relation $G \in \mathcal{G}$ gets an extension $G[I]$ by applying the view definition on instance I

This is as usual, propagating the data through Datalog rules in view definitions from body to head

Assume the sources are incomplete (aka. open, sound)

We also assume that the view definitions are given by positive, non-recursive Datalog rules, possibly with built-ins

A global instance D for schema $\mathcal{G}$ is legal if, for each global relation G, it holds: $G[D] \supseteq G[I]$

$G[D]$ is the extension of relation G in D

$Legal(\mathfrak{G})$ denotes the set of legal instances

Its elements are the "possible worlds"

What is true of the mediated system, at the global level, is what is true of all the possible worlds, i.e. in all the legal (global) instances

This applies in particular to query answers

If $Q(\bar{x})$ is a global query

$$Certain_{\mathfrak{G}}(Q) = \{\bar{t} \mid \bar{t} \text{ is a usual answer to } Q \text{ in } D, \text{ for every legal instance } D\}$$
$$= \bigcap_{D \in Legal(\mathfrak{G})} Q[D]$$

The certain answers to the global query Q

These are the semantically correct answers

We have a model theoretic definition of correct answer?

How to compute them?

What is the connection (if any) with the intuitive method we saw before (unfolding)?

It can be easily proved (using the semantics of Datalog programs) that the certain answers coincide with those obtained as follows:

1. Take source instance I
2. Propagate its data through the view definitions
3. Obtain extensions g for each of the global relations G
4. The g's form a global instance $\bar{D}$, the so-called retrieved database

   The retrieved database is the global instance obtained by propagating the data at the sources through the rules from right to left
5. Pose Q to $\bar{D}$ as usual
6. The obtained answers are exactly the certain answers

   From this perspective of query answering, this particular (and legal) global instance gives the semantics to the integration system A generic legal instance

   ...

It can also be proved that the unfolding method returns the same answers

So, it provides correct query plans!

We obtain immediately that obtaining certain answers from a mediated integration system can be done in polynomial time in the size of the combined data sources:

1. Computing the retrieved database can be done in polynomial time
2. Querying the retrieved database too

Example: (cont.) Given the material sources

$S_1 = \{(aaa, peter, 1989), (abc, john, 1960), (cdde, mary, 1978)\}$
$S_2 = \{(assd, steve, 1997), (shhhh, alice, 1920)\}$
$S_3 = \{(shhhh, good), (abc, awful), (kkkk, excellent), (cdde, mediocre)\}$

The retrieved database is

$$MovieYear = \{(aaa, 1989), (abc, 1960), (cdde, 1978), (assd, 1997), (shhhh, 1920)\}$$
$$MovieRev = \{(abc, john, awful), (cdde, mary, mediocre)\}$$

This an ordinary relational database

Given the global query

$$Q: \quad Ans(Dir, Year) \leftarrow MovieRev(Title, Dir, Review), MovieYear(Title, Year)$$

The correct answers are: $\{(john, 1960), (mary, 1978)\}$

Semantics under LAV

A virtual data integration system $\mathfrak{G}$ under LAV and open sources

$$\begin{array}{lcll} V_1(\bar{x}_1) & \leftarrow & \varphi_1(\bar{x}'_1) & v_1 \\ \ldots & \ldots & \ldots & \\ V_n(\bar{x}_n) & \leftarrow & \varphi_n(\bar{x}'_n) & v_n \end{array}$$

Here, the V_is are the source relations, and each v_i is an extension (material data source) for relation (view) V_i, the given contents

The $\varphi_i(\bar{x}_i)$ are conjunctions of global database atoms (and possibly built-ins), i.e. conjunctive view definitions; $\bar{x}_i \subseteq \bar{x}'_i$

$\mathfrak{G}$ determines a set of legal global instances Which ones?

Example: (views or sources on LHS, global relations on RHS)

$$\begin{aligned} DirYears(Dir, Year) &\leftarrow MovieRev(Title, Dir, Review), \\ &\quad\ MovieYear(Title, Year) \\ Movies(Title, Dir) &\leftarrow MovieRev(Title, Dir, Review) \end{aligned}$$

Given material extension for $DirYears$:

$$\{(peter, 1989), (john, 1960), (mary, 1978), \\ (steve, 1997), (alice, 1920)\}$$

Given material extension for $Movies$:

$$\{(aaa, peter), (abc, john), (cdde, mary), \\ (assd, steve), (shhhh, alice)\}$$

Let D be a concrete global instance

- Its underlying domain $\mathcal{U}$ contains all the constants in the sources and those appearing in the view definitions (and possibly others)
- $V_i[D]$ is the contents of the view V_i when its definition

$$V_i(\bar{x}_i) \leftarrow \varphi_i(\bar{x}_i)$$

is applied to D (the computed view on D)

Example: (cont.) Consider the global instance D_0 with

$$\begin{aligned} MovieYear &= \{(aaa, 1989), (abc, 1960), (cdde, 1978), \\ &\quad (assd, 1997), (shhhh, 1920)\} \\ MovieRev &= \{(shhhh, alice, good), (abc, john, awful), \\ &\quad (cdde, mary, mediocre)\} \end{aligned}$$

The view definitions evaluated on D_0 give:

- $DirYears[D_0] = \{(john, 1960), (mary, 1978), (alice, 1920)\}$
- $Movies[D_0] = \{(shhhh, alice), (abc, john), (cdde, mary)\}$

We can see that the computed views on D_0 differ from the given material contents of the views

Actually, the computed views are both strictly contained in the original source extensions

We would expect for open sources, their material extensions to be contained in the computed views, i.e. the other way around

The chosen global instance D_0 is not one of the intended instances of the integration system ...

$$Legal(\mathfrak{G}) := \{ \text{ global } D \mid v_i \subseteq V_i[D], \quad i = 1, \ldots, n\}$$

This is the set of intended global instances

Example: (cont.) Global instance D_0 is not legal, because

$$\{(peter, 1989), (john, 1960), (mary, 1978), (steve, 1997), (alice, 1920)\} \not\subseteq DirYears[D_0]$$

$$\{(aaa, peter), (abc, john), (cdde, mary), (assd, steve), (shhhh, alice)\} \not\subseteq Movies[D_0]$$

Legal instances give the semantics to the integration system

We may have several legal global instances

We can define, as before, the semantically correct answers from the integration system to a global query Q

But now, we may not have something like a single generic, representative global instance as in GAV with the retrieved DB

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

$Certain_{\mathfrak{G}}(Q) := \{\bar{t} \mid \bar{t}$ is an answer to Q in D

for all $D \in Legal(\mathfrak{G})\}$

Example: System $\mathfrak{G}_1$ under LAV with global relation $R(x, y)$ and open sources

$V_1(x, y) \leftarrow R(x, y)$ with $v_1 = \{(a, b), (c, d)\}$

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

Legal instance: $D_1 = \{R(a, b), R(c, d), R(a, c), R(d, e)\}$

- $v_1 \subseteq V_1[D_1] = \{(a, b), (c, d), (a, c), (d, e)\}$
- $v_2 \subseteq V_2[D_1] = \{(a, b), (c, d), (a, c), (d, e)\}$

All supersets of D_1 are also legal global instances; no subset of D_1 is legal, e.g.

$$\{R(a, b), R(c, d), R(a, c), R(d, e), R(c, e)\} \in Legal(\mathfrak{G}_1)$$

$$\{R(a, b), R(c, d), R(a, c)\} \notin Legal(\mathfrak{G}_1)$$

Consider now the global query

$$Q_1: \quad Ans(x,y) \leftarrow R(x,y)$$

By direct inspection of the legal instances we get

$$Certain_{\mathfrak{G}_1}(Q_1) = \{(a,b),(c,d),(a,c),(d,e)\}$$

Notice $(c,e) \notin Certain_{\mathfrak{G}_1}(Q_1)$

$$Q_2: \quad Ans(x) \leftarrow R(x,y)$$

$$Certain_{\mathfrak{G}_1}(Q_2) = \{(a),(c),(d)\}$$

We did not use any query plan to get them, only the semantics

Here, D_1 does act as a generic legal instance

Example: Global system $\mathfrak{G}_2$ with global relations P, Q, and sources

$$V_1(x,y) \leftarrow P(x,z), Q(z,y); \qquad v_1 = \{(a,b)\}$$
$$V_2(x,y) \leftarrow P(x,y); \qquad v_2 = \{(a,c)\}$$

$Legal(\mathfrak{G}_2)$?

- From definition of V_2 and the contents v_2, in any legal instance P is forced to contain (a,c)
- From definition of V_1 and the contents v_1, a legal global instance must contain at least a set of tuples of the form $\{P(a,e), Q(e,b)\}$

The legal instances of $\mathfrak{G}_2$ are all the supersets of instances of the form $\{P(a,c), P(a,z), Q(z,b) \mid z \in \mathcal{U}\}$ ($\mathcal{U}$: underlying domain)

- $\{P(a,c), Q(c,b)\} \in Legal(\mathfrak{G}_2)$
- $\{P(a,c), P(a,e), Q(e,b)\} \in Legal(\mathfrak{G}_2)$
- $\{P(a,c), Q(c,b), P(e,e), Q(e,a), Q(d,d), Q(a,c)\} \in Legal(\mathfrak{G}_2)$
- $\{P(a,c), Q(e,b)\} \notin Legal(\mathfrak{G}_2)$

Here we have legal instances that are incomparable under set inclusion!

Global query: $Q_1 : Ans(x,y) \leftarrow P(x,y)$

$$Certain_{\mathfrak{G}_2}(Q_1) = \{(a,c)\}$$

Global query: $Q_2 : Ans(x) \leftarrow Q(x,y)$

$$Certain_{\mathfrak{G}_2}(Q_2) = \{\}$$

Global query: $Q_3 : Ans(y) \leftarrow Q(x,y)$

$$Certain_{\mathfrak{G}_2}(Q_3) = \{(b)\}$$

Global query: Q_4: $Ans(x, y) \leftarrow P(x, y), not\ Q(x, y)$

Not a conjunctive query

In legal instances (under open sources) we can always add new tuples, so $not\ Q(x, y)$ can always be made false in some legal global instance: $Certain_{\mathfrak{G}_2}(Q_4) = \emptyset$

Notion of certain answer does not seem to be a sensible notion for non-monotone queries

Conjunctive queries (and others) are monotone: the set of answers may only grow if the database grows: for databases D_1, D_2

$D_1 \subseteq D_2 \implies$ Answers to Q in $D_1 \subseteq$ Answers to Q in D_2
(i.e. $Q[D_1] \subseteq Q[D_2]$)

General mechanisms for query planning available are for classes of monotone queries

Chapter 4: Query Answering under LAV

Query Plans

We have seen that generation of query plans is simple under GAV

Also that the naive algorithm for query answering is correct: it obtains all and only certain answers to global queries

We will concentrate on the LAV approach with open sources

Now query plans can be assessed against a precise semantics

The question now is whether a query plan to answer a global query is capable of retrieving all the certain answers

Given a global query Q posed in terms of the global schema, we need to go to the sources to find the local data to return global answers to the user

How?? We need a plan for evaluating Q

Query Q has to be "rewritten" in terms of the views, i.e. as a set of queries expressed in terms of the relations in the sources

Some rewriting algorithms:

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

They are designed to handle conjunctive global queries (and minor extensions)

We will concentrate on the "Inverse Rules Algorithm" (IRA) due to its conceptual interest, and to show some general issues

Inverse Rules Algorithm

Given: Set of rules describing the source relations as (non recursive Datalog) views of the global schema

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

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

Source relations described by conjunctive rules (queries):

$$S(\bar{x}) \leftarrow P_1(\bar{x_1}), \ldots, P_n(\bar{x_n})$$

S a source relation; P_is global relations, no built-ins

Example: V_1, V_2: local relations

$R_1, R_2, R_3, R4, R_5, R_6, R_7$: global relations

Source descriptions $\mathcal{V}$:

S_1: $V_1(x, z) \leftarrow R_1(x, y), R_2(y, z)$

S_2: $V_2(x, y) \leftarrow R_3(x, y)$

IRA obtains from these descriptions, "inverse rules" that describe the global relations

Idea: V_2 is incomplete, i.e. its contents is a subset of the "contents" of the (legal) global relation R_3

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:

$$R_3(x, y) \leftarrow V_2(x, y)$$

A rule describing R_3!!

We could obtain other rules describing R_3, e.g. if we had a third source description $V_3(x, y) \leftarrow R_3(x, y)$, we would get the inverse rule $R_3(x, y) \leftarrow V_3(x, y)$; and we take the union

Notice that both V_2, V_3 are incomplete, they have only part of the data in R_3

What about inverting the rule for V_1:

$$R_1(x, y), R_2(y, z) \leftarrow V_1(x, z) \quad ???$$

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

$R_1(x, y) \leftarrow V_1(x, z)$

$R_2(y, z) \leftarrow V_1(x, z)$

We lost the shared variable y (the join), the two occurrences of y are independent now

Furthermore: what does y in the head mean??

y does not appear in the bodies, so no condition on y ...

Any value for y? Not the idea ...

Those rules are not safe

Better:

$V_1(x, z) \leftarrow R_1(x, y), R_2(y, z)$ is logically equivalent to

$V_1(x, z) \leftarrow \exists y\ (R_1(x, y)\ \wedge\ R_2(y, z))$

Inverting the rule, 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

$$\forall x \forall z(\exists y(R_1(x, y)\ \wedge\ R_2(y, z)) \leftarrow V_1(x, z))$$

Each value for y possibly depends on the values for x, z, i.e. y is a function of x, z

To keep this dependence, replace y by a "function symbol" $f(x, z)$

f is a so-called "Skolem" function

We may need more of them, to capture other dependencies between variables

One for each rule and existential variable (projection) appearing in it

$$(R_1(x, f(x, z)) \ \wedge \ R_2(f(x, z), z)) \leftarrow V_1(x, z)$$

and then

$$R_1(x, f(x, z)) \leftarrow V_1(x, z)$$

$$R_2(f(x, z), z)) \leftarrow V_1(x, z)$$

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

$$\begin{aligned} R_1(x, f(x,z)) &\leftarrow V_1(x,z) \\ R_2(f(x,z), z) &\leftarrow V_1(x,z) \\ R_3(x,y) &\leftarrow V_2(x,y) \end{aligned}$$

The global relations are described as views of the local relations!

Notice: not exactly a Datalog program, it contains functions ...

In order to answer global queries we can append (always) the (same) inverse rules to any Datalog global query

We can use the inverse rules to compute global queries ...

For example, the query Q

$$\begin{aligned} Ans(x,z) &\leftarrow R_1(x,y), R_2(y,z), R_4(x) \\ R_4(x) &\leftarrow R_3(x,y) \\ R_4(x) &\leftarrow R_7(x) \\ R_7(x) &\leftarrow R_1(x,y), R_6(x,y) \end{aligned}$$

A Datalog query in terms of the "base" global relations R_1, R_2, R_3

We have descriptions for those three global relations; the inverse rules ...

The goal R_6 cannot be computed, there is no description for it, in particular, it does not appear in any source description, its contents is empty

Then, R_7 cannot be evaluated either (also empty contents); delete that rule; we are left with

$$\begin{aligned} Ans(x,z) &\leftarrow R_1(x,y), R_2(y,z), R_4(x) \\ R_4(x) &\leftarrow R_3(x,y) \\ R_4(x) &\leftarrow R_7(x) \end{aligned}$$

For the same reason, the second rule for R_4 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^-

Then the plan $Plan(Q)$ returned by the IRA is $Q^- \cup \mathcal{V}^{-1}$:

$$
\begin{aligned}
Ans(x,z) &\leftarrow R_1(x,y), R_2(y,z), R_4(x) \\
R_4(x) &\leftarrow R_3(x,y) \\
R_1(x, f(x,z)) &\leftarrow V_1(x,z) \\
R_2(f(x,z),z) &\leftarrow V_1(x,z) \\
R_3(x,y) &\leftarrow V_2(x,y)
\end{aligned}
$$

A "Datalog" program with functions ...

This is "the best we have for answering the original query"

We will get answers to Q, and "the most" that could be computed

These claims require a precise formulation and proof ...

This Datalog query can be evaluated, e.g. bottom-up, from concrete source contents, e.g. assume that the source relations are:

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

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

$$Ans(x, z) \leftarrow R_1(x, y), R_2(y, z), R_4(x) \quad (1)$$
$$R_4(x) \leftarrow R_3(x, y) \quad (2)$$
$$R_1(x, f_1(x, z)) \leftarrow V_1(x, z) \quad (3)$$
$$R_2(f_1(x, z), z) \leftarrow V_1(x, z) \quad (4)$$
$$R_3(x, y) \leftarrow V_2(x, y) \quad (5)$$

- Using rules (5), (3), (4) we get

  $R_3 = \{(a, c), (a, a), (c, d), (b, b)\}$

  $R_1 = \{(a, f(a, b)), (a, f(a, a)), (c, f(c, a)), (b, f(b, a))\}$

  $R_2 = \{(f(a, b), b), (f(a, a), a), (f(c, a), a), (f(b, a), a)\}$

- Now rule (2)

  $R_4 = \{(a), (c), (b)\}$

- Finally, rule (1)

  $\Pi_{x,z}(R_1 \bowtie R_2) = \{(a, b), (a, a), (c, a), (b, a)\}$

  We keep only those tuples in this relation such that the first argument appears in R_4; all of them in this case

  Finally, $Ans = \{(a, b), (a, a), (c, a), (b, a)\}$

Remarks:

- We process the functions symbolically

  They did not appear in the final answer set

  This may not be always the case

  If tuples with function symbols appear in the final answer set, we filter them out ...

- It may be necessary to introduce more than one function symbol

- We may get several rules describing the same global relation, in this case, that relation is described by a disjunctive query (view)

We illustrate all these issues with an example

Example: Source descriptions $\mathcal{V}$

$$\begin{aligned} V_1(x) &\leftarrow R(x,y), G(y,z) \\ V_2(x,y) &\leftarrow R(z,x), U(x,y) \end{aligned}$$

Inverse rules:

$$\begin{aligned} R(x, f_1(x)) &\leftarrow V_1(x) \\ G(f_1(x), f_2(x)) &\leftarrow V_1(x) \\ R(f_3(x,y), x) &\leftarrow V_2(x,y) \\ U(x,y) &\leftarrow V_2(x,y) \end{aligned}$$

Source contents:

$$\begin{aligned} V_1 &= \{(a),(d)\} \\ V_2 &= \{(a,c),(c,d),(b,a)\} \end{aligned}$$

Global query: $Ans(x) \leftarrow R(x, y)$

$$R = \{(a, f_1(a)), (d, f_1(d))\} \cup$$
$$\{(f_3(a, c), a), (f_3(c, d), c), (f_3(b, a), b)\}$$

$$\Pi_x(R) = \{(a), (d)\} \cup \{(f_3(a, c)), (f_3(c, d)), (f_3(b, a))\}$$

$$Ans = \{(a), (d)\}$$

The definition of R by the two rules

$$\begin{aligned} R(x, f_1(x)) &\leftarrow V_1(x) \\ R(f_3(x, y), x) &\leftarrow V_2(x, y) \end{aligned}$$

reflects the fact that each of V_1, V_2 contains only part of the data of its kind, i.e. that the source relations are incomplete wrt to the same kind of data in the system

Several questions:

- Are the answers obtained with IRA any good?

  In what sense?

- Is there a better plan?

  Or better mechanisms for generating plans?

  Better in what sense?

- What is the most information we can get from such a system?

- What is the "data contained in the system", i.e. in the global relations if they were to be materialized?

  There must be some sort of data, otherwise how can we be obtaining answers to global queries?

All these questions have to do with the semantics of a virtual data integration system we introduced before

Some of them have been answered already, and for answering the others we also have the basic semantic framework

Properties of the IRA

The query plan obtained may not be exactly a Datalog program, due to the auxiliary function symbols

The same inverse rules can be used with any global query; we compute them once

The query plan 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 descriptions

The resulting plan can be evaluated in polynomial time in the size of the underlying data sources (polynomial time in data complexity)

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

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

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

Some references:

Serge Abiteboul, Oliver M. Duschka: Complexity of Answering Queries Using Materialized Views. PODS 1998: 254-263

Oliver M. Duschka, Michael R. Genesereth, Alon Y. Levy: Recursive Query Plans for Data Integration. J. Log. Program. 43(1): 49-73 (2000)

Maurizio Lenzerini: Data Integration: A Theoretical Perspective. PODS 2002: 233-246

Chapter 5: Consistency

Data Integration and Consistency

In the virtual approach to data integration, one usually assumes that certain ICs hold at the global level

However, there is no global IC maintenance mechanism

So, no guarantee that global ICs on the global schema hold

Actually, under the LAV approach they are sometimes used to generate a query plan to answer a global query

There are situations where without assuming and using those global ICs no query plans can be generated

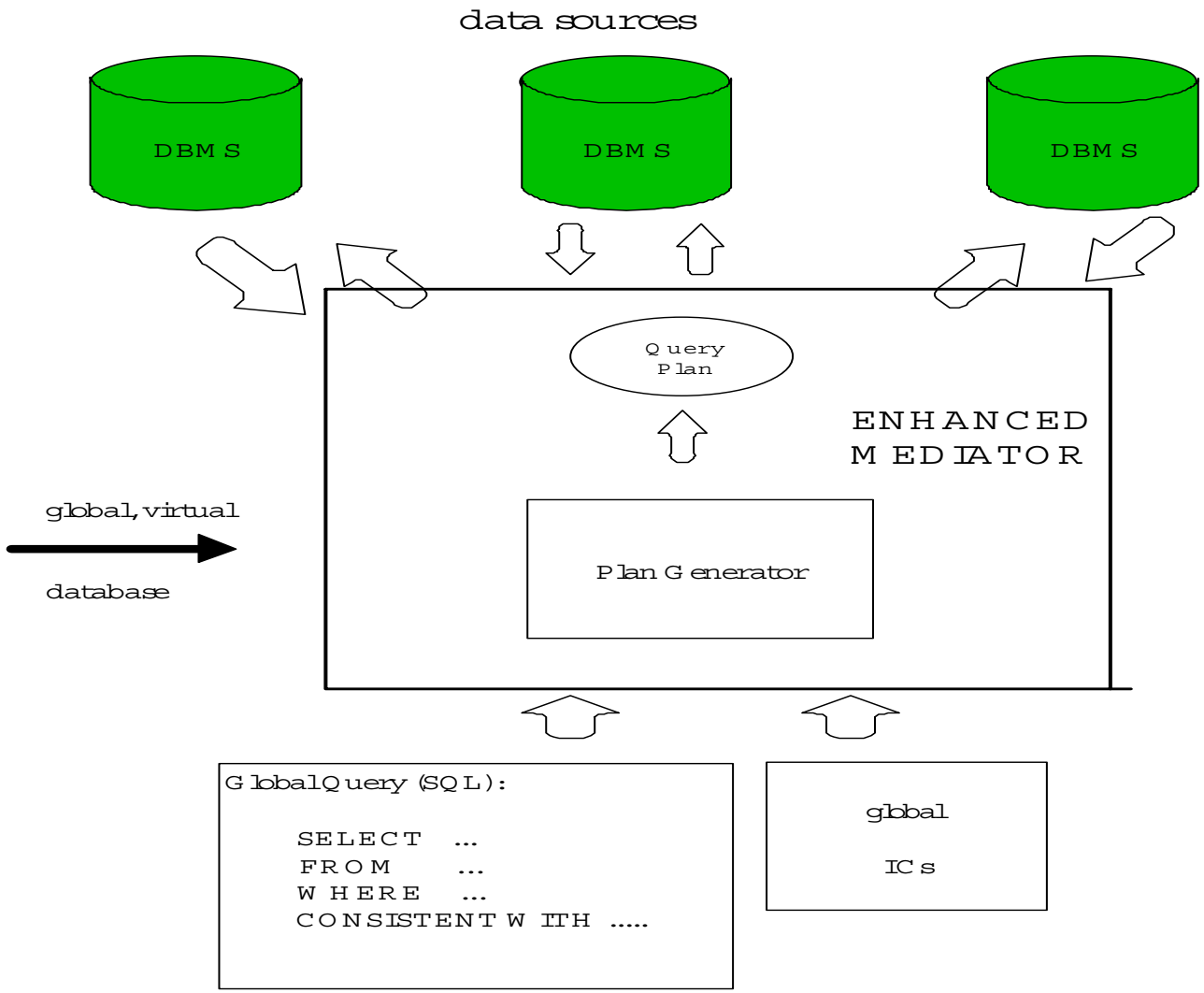
How can we be sure that those global ICs hold?

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
Plan Generator
global, virtual
database
GlobalQuery (SQL):
SELECT ...
FROM ...
WHERE ...
CONSISTENT WITH .....
global
ICs

An alternative approach: Consistent Query Answering

- Do not worry too much about the consistency of the data "contained" in the integration system (or stand alone DB)
- Better deal with the problem at query time
- When queries are posed to the integrated system, retrieve only those answers from the global database that are "consistent with" the global ICs

  What is a consistent answer to a query?
- The notion of consistency or ICs satisfaction is a holistic notion, the whole DB satisfies the ICs or not

  However, most likely most of the data in DB is still "consistent"

- When DB is queried, we want only the "consistent answers", a local notion ...
- We need a precise definition of what is a consistent answer to a query in an inconsistent DB
- We need to develop mechanisms for computing consistent answers, hopefully querying the only available, possibly inconsistent DB

Example: (informal and intuitive) Data sources

CUstudents	Number	Name
	101	john
	102	mary

OUstudents	Number	Name
	103	claire
	101	peter

Both sources satisfy the local FD: $Number \rightarrow Name$

Global relation: (defined using GAV)

$Students(x, y) \leftarrow CUstudents(x, y)$
$Students(x, y) \leftarrow OUstudents(x, y)$

The global data does not seem to satisfy the same FD, but now considered as a global IC: $Students : Number \rightarrow Name$

Consistency of global system cannot be restored from the mediator

Alternative: consider the global FD when queries are answered

Obtain only the consistent answers

Which is the consistent data in an inconsistent database, in particular, query answers?

The data that is invariant under all "minimal ways" of restoring consistency

How to compute them? Different techniques

In several cases (of queries and ICs): FO rewriting of the query

Compiling ICs into the query, to enforce consistency

This problem already appears in a stand alone database, e.g.

Students	Number	Name
	101	john
	103	claire
	102	mary
	101	peter

that is inconsistent wrt $FD: \ Number \rightarrow Name$

Without changing the database we can obtain consistent answers to queries

Repairs: (Arenas, Bertossi, Chomicki; PODS99)

Students1	Number	Name
	101	john
	102	mary
	103	claire

Students2	Number	Name
	103	claire
	101	peter
	102	mary

Obtained by deleting a minimal set of tuples (under set inclusion)

They determine the consistent information in the database

In particular, the consistent answers to queries

Some global queries:

- $Ans(x, y) \leftarrow Students(x, y)$
  Unrestricted answers (no FD considered):
  $\{(101, john), (101, peter), (102, mary), (103, claire)\}$
  Consistent answers (FD considered):
  $\{(102, mary), (103, claire)\}$

- $Ans(x) \leftarrow Students(x, y)$
  Unrestricted answers: $\{(101), (102), (103)\}$
  Consistent answers: $\{(101), (102), (103)\}$

Can they be computed without generating the repairs?

Using only the inconsistent database?

In some cases this can be done by rewriting the FO query into a new FO query (Arenas, Bertossi, Chomicki; 99)

E.g. the first one

$$Q(x, y): \ Students(x, y) \quad \mapsto \quad Q'(x, y)$$

$$Q'(x, y): \ Students(x, y) \wedge \neg\exists z(Students(x, z) \wedge z \neq y)$$

The rewritten query is posed to the original database

Its answers are all and only consistent answers

Rewritten query can still be posed using SQL, e.g.

The FD was "compiled" into the query

Adding conditions, to filter out answers ...

We make the answers respect the global ICs

Bertossi, L. Consistent Query Answering in Databases. ACM Sigmod Record, June 2006, 35(2):68-76.

Consistent Answers in VDISs: Idea

Example: Global LAV system $\mathfrak{G}_1$ 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: $R(x, y)$?

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

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

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

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

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

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

For $\mathfrak{G}_1$, $D = \{R(a,b), R(c,d), R(a,c), R(d,e)\}$ is the only minimal instance

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

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

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)

For $\mathfrak{G}_1$:

- The only minimal legal instance

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

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

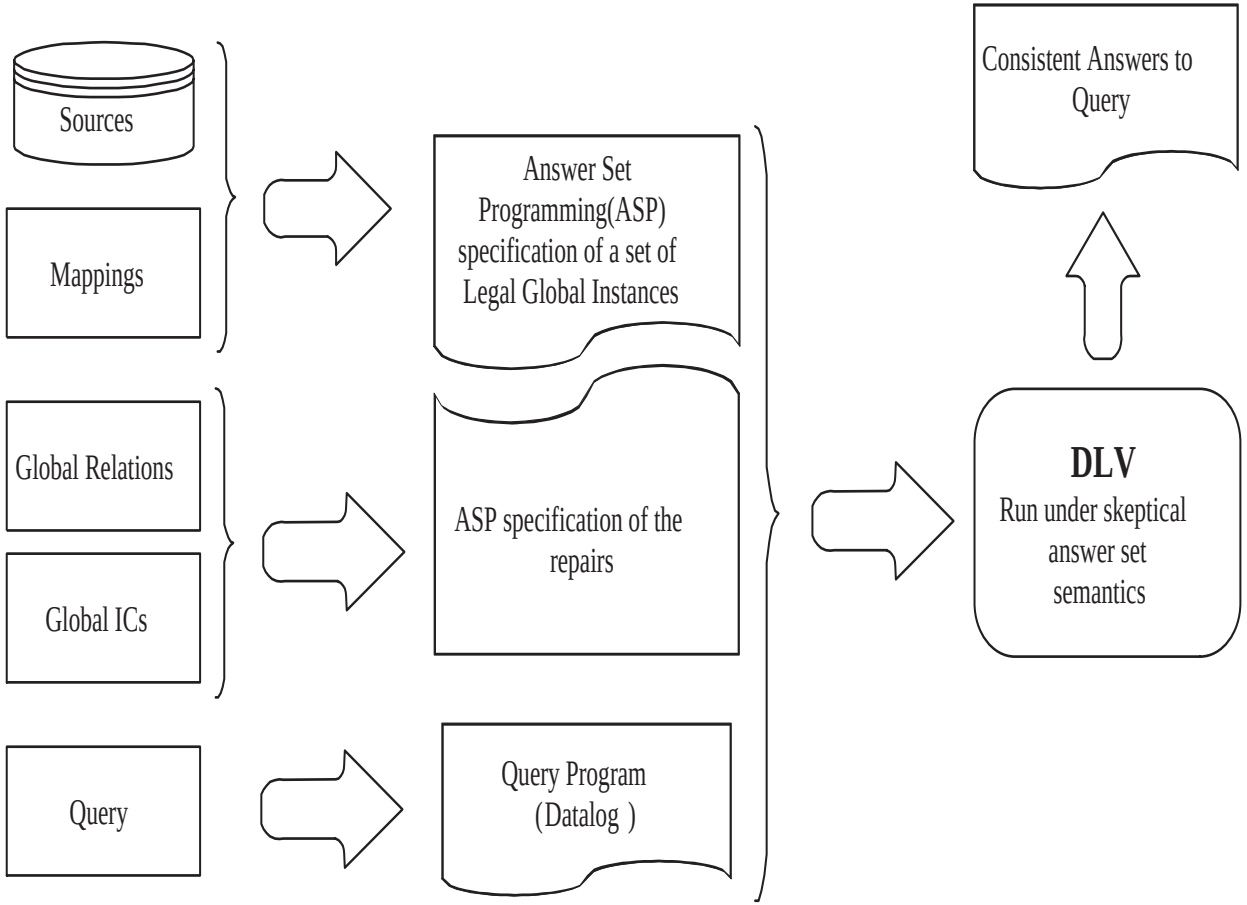
- Its repairs wrt FD are
  $D^1 = \{R(a, b), R(c, d), R(d, e)\}$ and
  $D^2 = \{R(c, d), R(a, c), R(d, e)\}$

  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 $Q\colon\ R(x, y)?$
  Only $\{(c, d), (d, e)\}$
  Those that are answers to original query in every repair!

Computing Consistent Answers

- 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



(Bravo, Bertossi; IJCAI 03)

Bertossi, L. and Bravo, L. Consistent Query Answers in Virtual Data Integration Systems. In 'Inconsistency Tolerance', Springer LNCS 3300, 2004, pp. 42-83.

Specifying Minimal Instances

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

$V_1(x, z) \leftarrow P(x, y), R(y, z)$ $\quad v_1 = \{(a, b)\}$ open

$V_2(x, y) \leftarrow P(x, y)$ $\quad v_2 = \{(a, c)\}$ open

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

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

$P(x, z) \leftarrow V_1(x, y), F_1((x, y), z)$

$P(x, y) \leftarrow V_2(x, y)$

$R(z, y) \leftarrow V_1(x, y), F_1((x, y), z)$

$F_1((x, y), z) \leftarrow V_1(x, y), dom(z), choice((x, y), (z))$

$dom(a).,\quad dom(b).,\quad dom(c).,\quad \dots, V_1(a, b).,\quad V_2(a, c).$

Inspired by inverse rules algorithm for computing certain answers

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

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

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

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

...

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

More generally:

- 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 minimal legal instances determine the certain answers)
- The program can be refined to compute all and only the minimal legal instances

Example: $\mathfrak{G}_3$

$$V_1(x) \leftarrow P(x,y) \qquad \{v_1(a)\} \qquad \text{open}$$

$$V_2(x,y) \leftarrow P(x,y) \qquad \{v_2(a,c)\} \qquad \text{open}$$

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

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

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

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

Which makes the same condition on V_1 automatically satisfied; so no other values for y are needed

We want $\Pi(\mathfrak{G}_3)$ to capture **only** the minimal instances

A refined version of $\Pi(\mathfrak{G}_3)$ detects in which cases it is necessary to use the functional 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)

With the general refined program, it holds

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

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 of Repairs

So far: specification of minimal instances of an integration system

Since they can be inconsistent, we need to 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)

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

Example: $\mathfrak{G}_3$

$$V_1(x) \leftarrow P(x,y) \qquad \{v_1(a)\} \qquad \text{open}$$

$$V_2(x,y) \leftarrow P(x,y) \qquad \{v_2(a,c)\} \qquad \text{open}$$

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

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

In the program below, 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: The refined program for minimal instances (w/standard version of Choice)

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

Only one stable model: $\mathcal{M} = \{P(a,c,\mathbf{t_d})\}$ (essentially)

Second Layer: The program that specifies the repairs

$$
\begin{array}{rcl}
P(x,y,t^\star) & \longleftarrow & P(x,y,t_a) \\
P(x,y,t^\star) & \longleftarrow & P(x,y,\mathbf{t_d}) \\
P(x,y,f_a) \quad \vee & & P(y,x,t_a) \longleftarrow P(x,y,t^\star), \\
 & & \qquad\qquad not\ P(y,x,\mathbf{t_d}) \\
P(x,y,f_a) \quad \vee & & P(y,x,t_a) \longleftarrow P(x,y,t^\star), P(y,x,f_a) \\
P(x,y,\mathbf{t}^{\star\star}) & \longleftarrow & P(x,y,t_a) \\
P(x,y,\mathbf{t}^{\star\star}) & \longleftarrow & P(x,y,\mathbf{t_d}),\ not\ P(x,y,f_a) \\
 & \longleftarrow & P(x,y,t_a), P(x,y,f_a).
\end{array}
$$

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

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,t*), auxv1(a), P(c,a,ta), P(a,c,t**), P(c,a,t*), P(c,a,t**)}
$\equiv \{P(a,c), P(c,a)\}$

$\mathcal{M}_2^r =$ {dom(a),dom(c), v1(a), v2(a,c), P(a,c,nv1), P(a,c,v2), P(a,c,td), P(a,c,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" $\Pi := \Pi(\mathfrak{G}, IC) \cup \Pi(Q')$
4. Collect ground atoms
   $Ans(\bar{t}) \in \bigcap\{S \mid S \text{ is a stable model of } \Pi\}$

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

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 in DLV for $\mathfrak{G}_1$ 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) :-  R(X,Y,tss).                            %query subprogram
```

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

Stay in touch ...

www.scs.carleton.ca/~bertossi

bertossi@scs.carleton.ca

For an extended version of this tutorial (but not much on consistency):

www.scs.carleton.ca/~bertossi/talks/datIntegr07.pdf

This one will be at:

www.scs.carleton.ca/~bertossi/talks/datIntPuc07.pdf