Consistent Answers from Integrated Data Sources

Leopoldo Bertossi

Carleton University
School of Computer Science
Ottawa, Canada

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

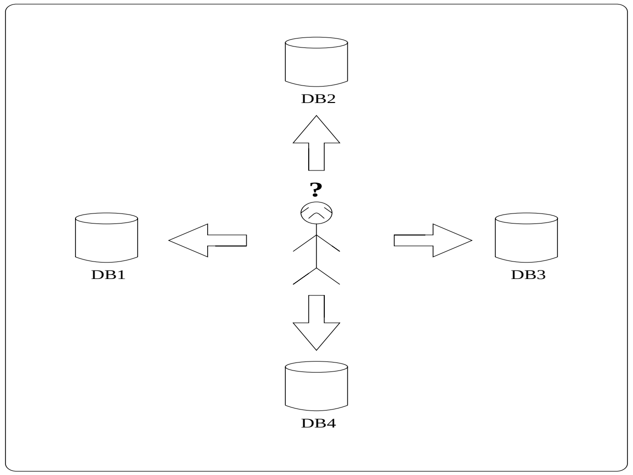
University at Buffalo, April 2005.

Virtual Data Integration

Number of available on-line information sources has increased dramatically

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

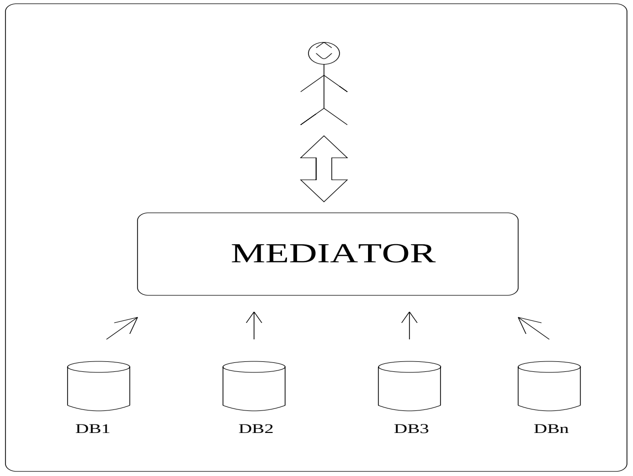
In particular, when information has to be integrated?



- Interacting with each of the sources independently?
- 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

An approach: Virtual integration of sources via a mediator

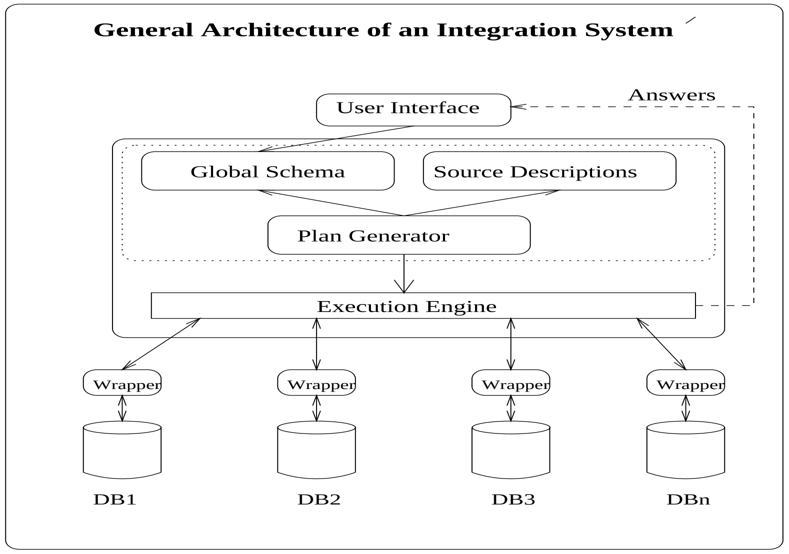


A software system that offers a common user interface to query a set of heterogeneous and independent data sources

System offers a single integrated, global schema

User feels like interacting with a single database

- Sources are mutually independent and non cooperative
- Data kept in sources, and extracted at mediator's request
- Interaction from the system to the sources via queries
- Mediator composes query results for the user
- Update operations are not supported via the mediator
- System should allow sources to get in and out



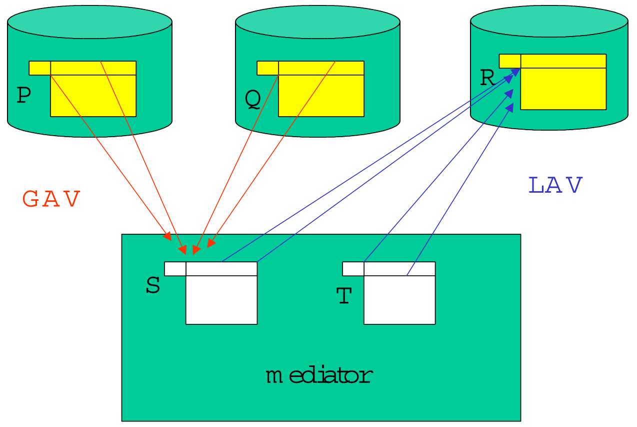
- User poses queries in terms of global schema
- Relationship between global schema and local, source schemas specified in the mediator, as source descriptions
- Mediator is responsible of solving problems of data:
  - redundancy: to avoid unnecessary computations
  - complementarity: data of the same kind may be spread through different sources
  - inconsistency: sources, independently, may be consistent, but together, possibly not

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

Description of the Sources:

- Mediator needs to know what data is in the sources and how it relates to the global schema
- Sources are described by means of logical formulas; like those used to express queries and define views
- Those formulas define the mappings between the global schema and the local schemas
- There are two main approaches (and a combination of them):

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



Plan Generator:

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

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

LAV

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

A global data schema is designed, and contributors of data (sources) describe how their data fits into the integration system

Example: Definition of tables at the sources:

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

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

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

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

Here, sources defined as conjunctive queries (views) with built-ins

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, i.e. data in the sources could be "incomplete"

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

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

Query expressed in terms of relations in global schema only

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

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

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

$$\begin{aligned} \mathit{Ans}'(\mathit{Title}, \mathit{Review}) \quad \leftarrow \quad & S_1(\mathit{Title}, \mathit{Year}, \mathit{Director}), \\ & S_2(\mathit{Title}, \mathit{Review}). \end{aligned}$$

Query was rewritten in terms of the views; and now can be easily computed

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

We get correct answers; and also the most we can get ...

Meaning?

Semantics of a LAV Data Integration System

We assume source relations are open or incomplete

Example: Global system $\mathcal{G}_1$ with source definitions and sources extensions

$$S_1(X,Y) \leftarrow R(X,Y) \quad \text{with} \quad s_1 = \{(a,b),(c,d)\}$$

$$S_2(X,Y) \leftarrow R(Y,X) \quad \text{with} \quad s_2 = \{(c,a),(e,d)\}$$

The global relations can be materialized in different ways, still satisfying the source descriptions, so different global instances are possible

A global (material) instance D is legal if the view definitions applied to it compute extensions $S_1(D), S_2(D)$ such that $s_1 \subseteq S_1(D)$ and $s_2 \subseteq S_2(D)$

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

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

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

What is a correct answer to the query, considering that there are many possible legal global instances?

The intended answers to a global query are the certain answers, those that can be obtained from all the legal instances

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

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

Consistency in Virtual Data Integration

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

How can we be sure that those global ICs hold?

They are not maintained at the global level

Most likely they are not fully satisfied

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

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

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

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

(local FDs $S_1 : X \rightarrow Y$, $S_2 : X \rightarrow Y$ satisfied in the sources)

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

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

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

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

Minimal Legal Instances and Consistent Answers

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

Not much for obtaining consistent answers

Here we do both, in stages ...

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

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

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

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

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

For monotone queries they coincide

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

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

In the example:

- The only minimal legal instance

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

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

- Its repairs wrt FD are

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

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

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

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

Computing consistent answers? (Idea)

(Bravo, Bertossi; IJCAI 03)

- Answer set programming (ASP) gives a semantics to datalog (logic) programs with negation (and possibly disjunction in the heads)
- ASP based specification of minimal instances of a virtual data integration system
- ASP based specification of repairs of minimal instances
- Global query in Datalog (or its extensions) to be answered consistently
- Run the three 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

Specifying Minimal Instances

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

$S_1(X, Z) \leftarrow P(X, Y), R(Y, Z)$ $s_1 = \{(a, b)\}$ open
$S_2(X, Y) \leftarrow P(X, Y)$ $s_2 = \{(a, c)\}$ open

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

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

$P(X, Z) \leftarrow S_1(X, Y), F((X, Y), Z)$
$P(X, Y) \leftarrow S_2(X, Y)$
$R(Z, Y) \leftarrow S_1(X, Y), F((X, Y), Z)$
$F((X, Y), Z) \leftarrow S_1(X, Y), dom(Z), choice((X, Y), (Z))$
$dom(a).,\quad dom(b).,\quad dom(c).,\quad \dots, S_1(a, b).,\quad S_2(a, c).$

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

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

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

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

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

$$M_b = \{dom(a), \ldots, S_1(a,b), S_2(a,c), \underline{P(a,c)}, choice((a,b), b),$$
$$F(a,b,b), \underline{R(b,b)}, \underline{P(a,b)}\}$$

$$M_a = \{dom(a), \ldots, S_1(a,b), S_2(a,c), \underline{P(a,c)}, choice(a,b,a),$$
$$F((a,b), a), \underline{R(a,b)}, \underline{P(a,a)}\}$$

$$M_c = \{dom(a), \ldots, S_1(a,b), S_2(a,c), \underline{P(a,c)}, choice((a,b), c),$$
$$F(a,b,c), \underline{R(c,b)}\}$$

$\ldots$

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

In general:

- The minimal instances are all among the stable models of the program
- All the models of the program are (determine) legal instances
- In consequence, the program can be used to compute all the certain answers to monotone queries
- The program can be refined to compute all and only the minimal legal instances (appendix 1)
- The program can also be used to compute certain answers to monotone queries; more general than any other algorithm for LAV
- Specification programs can be produced when there are also closed or clopen sources (appendix 2)

Repairs and Consistent Answers

Intuitively, consistent answers are invariant under minimal restorations of consistency

Definition based on notion of repair
(Arenas, Bertossi, Chomicki; PODS 99)

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

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

$S_1(X,Y) \leftarrow R(X,Y)$ with $s_1 = \{(a,b),(c,d)\}$ open

$S_2(X,Y) \leftarrow R(Y,X)$ with $s_2 = \{(c,a),(e,d)\}$ open

$S_3(X) \leftarrow P(X)$ with $s_3 = \{(a),(d)\}$ open

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

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

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

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

(we relax legality for repairs)

Queries:

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

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

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

  a is a consistent answer, together with c, d

Specification of Repairs

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

Now: Specify their repairs

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

Repairs of single databases are specified using disjunctive logic programs with stable model semantics and are used to compute consistent answers to queries (Barcelo, Bertossi; PADL03)

The high expressive power of such logic programs in needed in the general case of CQA under first-order queries and ICs due to the intrinsic complexity of the problem

(Chomicki, Marcinkowski; Inf. & Comp. 2005)

(Cali, Lembo, Rosati; PODS03) (Fuxman, Miller; ICDT05)

However, for restricted but useful classes of queries and ICs, specialized and more efficient methods have been developed

(Arenas,Bertossi,Chomicki; PODS99) (Celle,Bertossi; DOOD00)

(Chomicki,Marcinkowski,Staworko; CIKM04) (Fuxman, Miller; ICDT05)

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

If a repair can be obtained by inserting a tuple $P(\bar{t})$ into the DB, then the program makes the atom $P(\bar{t}, \mathbf{t_a})$ true

If $P(\bar{t})$ has to be deleted from the DB, then the program makes the atom $P(\bar{t}, \mathbf{f_a})$ true ...

Example: $\mathcal{G}_3$

$$\begin{array}{rclll} S_1(X) & \leftarrow & P(X,Y) & \{s_1(a)\} & \text{open} \\ S_2(X,Y) & \leftarrow & P(X,Y) & \{s_2(a,c)\} & \text{open} \end{array}$$

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

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

In the program, the $P(\cdot, \cdot, \mathbf{t_d})$ are 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:

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

$$
\begin{array}{rcl}
dom(a).\ \ dom(c). & & S_1(a).\ \ S_2(a,c).\\
P(X,Y,\mathbf{t_d}) & \leftarrow & P(X,Y,s_1)\\
P(X,Y,\mathbf{t_d}) & \leftarrow & P(X,Y,t_o)\\
P(X,Y,ns_1) & \leftarrow & P(X,Y,t_o)\\
addS_1(X) & \leftarrow & S_1(X),\ \ not\ \ auxS_1(X)\\
auxS_1(X) & \leftarrow & P(X,Z,ns_1)\\
fz(X,Z) & \leftarrow & addS_1(X), dom(Z), chosens1z(X,Z)\\
chosens1z(X,Z) & \leftarrow & addS_1(X), dom(Z),\ \ not\ \ diffchoices1z(X,Z)\\
diffchoices1z(X,Z) & \leftarrow & chosens1z(X,U), dom(Z), U \neq Z\\
P(X,Z,s_1) & \leftarrow & addS_1(X), fz(X,Z)\\
P(X,Y,t_o) & \leftarrow & S_2(X,Y)
\end{array}
$$

Second Layer computes the repairs

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

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

Stable models obtained with DLV: (parts of them)

$\mathcal{M}_1^r =$ {dom(a), dom(c), S1(a), S2(a,c), P(a,c,ns1),
P(a,c,s2), P(a,c,td), P(a,c,t*), auxS1(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), S1(a), S2(a,c), P(a,c,ns1),
P(a,c,s2), P(a,c,td), P(a,c,t*), auxS1(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. $Q(\cdots P(\bar{u})\cdots) \quad \longmapsto \quad Q' := Q(\cdots P(\bar{u}, \mathbf{t}^{\star\star})\cdots)$
2. $Q'(\bar{x}) \quad \longmapsto \quad (\Pi(Q'), Ans(\bar{X}))$
   (Lloyd-Topor transformation)

   - $\Pi(Q')$ is a query program (the Third Layer)
   - $Ans(\bar{X})$ is a query atom defined in $\Pi(Q')$
3. "Run" $\Pi := \Pi(Q') \cup \Pi(\mathcal{G}, IC)$
4. Collect ground atoms

   $Ans(\bar{t}) \in \bigcap\{S \mid S$ is a stable model of $\Pi\}$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Conclusions

Alternative semantics for query answering in virtual data integration systems under ICs have been introduced and studied
(Cali, Calvanese, De Giacomo, Lenzerini; CAISE02)
(Lembo, Lenzerini, Rosati; KRDB02) (Cali, Lembo, Rosati; PODS 03)

There are many open problems

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

- Existing implementations of stable models semantics are based on grounding the rules
- In database applications, this may lead to huge ground programs; so "intelligent grounding" is applied
- Implementations are geared to computing (some) stable model(s) and answering ground queries
- For database applications, posing and answering open queries is more natural
- Computing all the the stable models completely is undesirable
- Generation of "partial" repairs, relative to the ICs that are "relevant" to the query at hand (Eiter, Fink, G.Greco, Lembo; ICLP 03)
- Magic sets, query-directed methodologies, for evaluating logic programs for CQA (S.Greco et al.)

- Efficient integration of database management systems and logic programs (e.g. DLV)
- Recent experiments with consistent query answering in virtual data integration systems are encouraging (INFOMIX Project)

Appendix 1: The Refined Program

Example: $\mathcal{G}_3$

$S_1(X)$	$\leftarrow$	$P(X,Y)$	$\{s_1(a)\}$	open
$S_2(X,Y)$	$\leftarrow$	$P(X,Y)$	$\{s_2(a,c)\}$	open

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

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

More legal instances (or stable models) than minimal instances

As S_2 is open, it forces $P(a,c)$ to be in all legal instances

What makes the same condition on S_1 automatically satisfied (no other values for Y needed)

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

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

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

$$F(X,Y) \leftarrow add\_S_1(X), dom(X), choice((X),Y)$$

where $add\_S_1(X)$ is true only when the openness of S_1 is not satisfied through other views (which has to be specified)

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

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

More general than any other algorithm for LAV ...

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

Appendix 2: Minimal Instances for Mixed Sources

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

$$\begin{array}{lll} S_1(X,Z) \leftarrow P(X,Y), R(Y,Z) & \{s_1(a,b)\} & \text{open} \\ S_2(X,Y) \leftarrow P(X,Y) & \{s_2(a,c)\} & \text{clopen} \end{array}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\begin{array}{lll} S_1(X,Z) \leftarrow P(X,Y), R(Y,Z) & \{s_1(a,b)\} & \text{open} \\ S_2(X,Y) \ \leftarrow \ P(X,Y) & \{s_2(a,c)\} & \text{clopen} \end{array}$$

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

$$\begin{array}{l} dom(a).,\ \ dom(b).,\ \ dom(c).,\ \ \dots, S_1(a,b).,\ \ S_2(a,c). \\ P(X,Z) \leftarrow S_1(X,Y), F((X,Y),Z) \\ R(Z,Y) \leftarrow S_1(X,Y), F((X,Y),Z) \\ P(X,Y) \leftarrow S_2(X,Y) \\ F((X,Y),Z) \leftarrow S_1(X,Y), dom(Z), choice((X,Y),(Z)) \\ \leftarrow P(X,Y),\ not\ S_2(X,Y) \end{array}$$

(S_2 stores the source contents, and P is used to compute the view extension, so we are requiring that $P \subseteq S_2$)

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

$\{domd(a), \ldots, s1(a,b), s2(a,c), \underline{P(a,c)}, \mathit{diffchoice}(a,b,a),$
$\mathit{diffchoice}(a,b,b), chosen(a,b,c), f(a,b,c), \underline{R(c,b)}\}$

Corresponding, as expected, to the fact that

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

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

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