Query Answering in Peer-to-Peer Data Exchange Systems

Leopoldo Bertossi

Carleton University
School of Computer Science
Ottawa, Canada

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

Join work with: **Loreto Bravo (Carleton University)**

The Context

Consider a system consisting of peers who exchange data when they answer local queries

Each peer has a local and autonomous database

Different peer's databases may have a different (relational) schemas

Data at two different peers' sites may be related by data exchange constraints (DECs)

Each peer has a set of DECs expressed as first order formulas that relate its schema with those of some other peers

Each peer does not update its instance according to its DECs and other peers' instances

However, if a peer `P` is answering a (local) query $Q_{\mathtt{P}}$, it may, at query time

- Import data from other peers to complement its data
- Ignore part of its own data

All this depending upon its own DECs and the peers' instances

But also upon the trust relationships that `P` has with other peers

An additional element to consider is `P`'s local semantic constraints

Example 1: Peers, their schemas, instances, DECs, trust relationships:

- $\texttt{P1}, \quad \mathcal{R}_1 = \{R^1\}, \quad r(\texttt{P1}) = \{R^1(a,b), R^1(s,\ t)\}$

  $\Sigma(\texttt{P1},\texttt{P2})\colon\ \forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))$

  $\Sigma(\texttt{P1},\texttt{P3})\colon\ \forall x \forall y \forall z (R^1(x,y) \wedge R^3(x,z) \quad \rightarrow \quad y = z)$

  `P1` trusts `P2` more than itself

  `P1` trusts `P3` the same as itself

  (also some local ICs $IC(\texttt{P1})$)

Those answers returned by $\mathtt{P}$ to $Q_{\mathtt{P}}$ that give an account of all these extra elements are $\mathtt{P}$'s peer consistent answers (PCAs) to $Q_{\mathtt{P}}$

In this presentation:

- We make these intuitions precise
- We give a semantics to PCAs
- Specification can be used as the basis for computing them
- We establish connections to virtual data integration

The Solutions for a Peer

Assume the peers's schemas are disjoint, with the possible exception of a shared domain

The union of all peers' instances $r(\texttt{P})$ can be seen as a single *global* instance $\bar{r}$

A solution for a peer `P` is a global instance that respects `P`'s DECs and trust relationships with its *immediate neighbors* and stays "as close as possible" to $\bar{r}$

`P`'s peer consistent answers (PCA) are those answers that can be retrieved from `P`'s portion of data in *every* solution for `P`

The notion of solution can be captured by means of the notion of repair used to characterize the notion of *consistent answer* to a query in a database r that fails to satisfy given ICs
(Arenas, Bertossi, Chomicki; PODS'99)

A repair satisfies the originally violated ICs and minimizes the sets of tuples by which it departs from r

A solution may virtually change `P`'s data

Solutions -as repairs in consistent query answering (CQA)- are virtual and used as an auxiliary tool to semantically characterize the notion of PCA

Then correct, intended answer to queries posed to a peer are captured by appealing to alternative models

Emphasis is not on computing repairs, but on defining and computing PCAs

Ideally, `P` should be able to obtain its PCAs just by querying its and its neighbors' instances

We are first dealing with the *direct case*, that considers the immediate neighbors; the *transitive case* is examined later

Example 1: (cont.)

- $\texttt{P1}$, $\quad \mathcal{R}_1 = \{R^1\}$, $\quad r(\texttt{P1}) = \{R^1(a,b), R^1(s,\ t)\}$

  $\Sigma(\texttt{P1},\texttt{P2})\colon\ \forall x \forall y (R^2(x,y) \rightarrow R^1(x,y))$

  $\Sigma(\texttt{P1},\texttt{P3})\colon\ \forall x \forall y \forall z (R^1(x,y) \wedge R^3(x,z) \quad \rightarrow \quad y = z)$

  $\texttt{P1}$ trusts $\texttt{P2}$ more than itself

  $\texttt{P1}$ trusts $\texttt{P3}$ the same as itself

- $\texttt{P2}$, $\ \mathcal{R}_2 = \{R^2\}$, $\ r(\texttt{P2}) = \{R^2(c,d), R^2(a,e)\}$
- $\texttt{P3}$, $\ \mathcal{R}_3 = \{R^3\}$, $\ r(\texttt{P3}) = \{R^3(a,f), R^3(s,u)\}$

Global instance does not satisfy the DECs

$$\bar{r} = \{R^1(a,b), R^1(s,t), R^2(c,d), R^2(a,e), R^3(a,f), R^3(s,u)\}$$

P does not change its or other peers' data

Rather P solves its conflicts at query time, when it queries its own and other peers' databases

Obtained answers should be sanctioned as correct wrt to the "solution based semantics"

The solutions for P1 are obtained by:

1. **First repairing $\bar{r}$ wrt $\Sigma(\mathtt{P1}, \mathtt{P2})$**

   Changing P1's data only (less trustable that P2)

   Only one repair is obtained:

$$\bar{r}_1 = \{R^1(a,b), R^1(s,t), R^1(c,d), R^1(a,e), R^2(c,d), R^2(a,e), R^3(a,f), R^3(s,u)\}$$

2. **This repair has to be repaired on its own wrt $\Sigma(\texttt{P1}, \texttt{P3})$**

   Keeping $\Sigma(\texttt{P1}, \texttt{P2})$ satisfied

   Now, data in `P1` or `P3` can change (equally trustable)

   Two repairs are obtained; and then solutions:

   $\bar{r}' = \{R^1(a,b), R^1(s,t), R^1(c,d), R^1(a,e), R^2(c,d), R^2(a,e)\}$

   $\bar{r}'' = \{ R^1(a,b), R^1(c,d), R^1(a,e), R^2(c,d), R^2(a,e), R^3(s,u)\}$

There is a precise model theoretic definition of solution that corresponds to this process

It involves a minimization with fixed predicates as found in non-monotonic reasoning

Actually these two layered process can be merged into a single one

Definition: **Given a FO query $Q(\bar{x}) \in \mathcal{L}(\mathtt{P})$, posed to peer P, a ground tuple $\bar{t}$ is a peer consistent answer for P iff $\bar{r}'|\mathtt{P} \models Q(\bar{t})$ for every solution $\bar{r}'$ for P**

Example 1: **(cont.) The query $Q\colon R^1(x,y)$ posed to P1 (in the language of P1) has the PCAs: $(a,b),(c,d),(a,e)$**

This answer has values that did not exists in P1's instance

Data originally in P1 is now missing in the set of PCAs

Computation of PCAs

In example 1, the PCAs can be obtained by a FO rewriting of Q using first $\Sigma(\texttt{P1},\texttt{P2})$ and then $\Sigma(\texttt{P1},\texttt{P3})$

$$Q''\colon \quad [R^1(x,y) \wedge \forall z_1((R^3(x,z_1) \wedge \neg\exists z_2 R^2(x,z_2)) \ \rightarrow \\ z_1 = y)] \quad \vee \quad R^2(x,y)$$

I.e. `P1` first issues a query to `P2` to retrieve the tuples in R^2

Next, a query is sent to `P3` to discard tuples from R^1 with the same first but not the same second argument in R^3
(as long as there does not exist a tuple in R^2 that "protects" the tuple in R^1)

Rewritten query gives exactly the PCAs to Q

This FO query rewriting approach cannot be extended much

It inherits the limitations of FO query rewriting for CQA

Better look for alternative methodologies

A general approach: answer set programming based specification of a peer's solutions ...

Mixed Referential DECs

In most applications we may expect the DECs $\Sigma(\mathtt{P},\mathtt{Q})$ for peer $\mathtt{P}$ to consist of formulas of the form

$$\forall\bar{x}\exists\bar{y}(R^{\mathtt{Q}}(\bar{x}) \wedge \varphi \quad \longrightarrow R^{\mathtt{P}}(\bar{z},\bar{y}) \wedge \psi)$$

with $R^{\mathtt{Q}}, R^{\mathtt{P}}$ relations for peers $\mathtt{Q}$ and $\mathtt{P}$, resp., φ,ψ formulas in terms of built-ins, $\bar{z} \subseteq \bar{x}$

Peer $\mathtt{P}$ wants to import data from the more trusted peer $\mathtt{Q}$

The same kind of formula could belong to $\mathtt{Q}$, if $\mathtt{Q}$ wants to validate its own data against $\mathtt{P}$'s data

We may have even more involved cases of referential DECs

Mixing tables from the two peers on each side of the implication

Example 2: Peers: `P` **with schema** $\{R_1(\cdot,\cdot), R_2(\cdot,\cdot)\}$
`Q` **with schema** $\{S_1(\cdot,\cdot),\ S_2(\cdot,\cdot)\}$

`P`**'s DEC:**

$$\forall x \forall y \forall z \exists w (R_1(x,y) \wedge S_1(z,y) \ \rightarrow\ R_2(x,w) \wedge S_2(z,w))$$

Assume `P` **considers** `Q`**'s data more reliable than its own**

(the case where `P` **and** `Q` **are equally trustable according to** `P` **can be handled similarly)**

If `P`'s DEC is not satisfied by the combination of the data in `P` and `Q`, alternative solutions for `P` have to be found, keeping `Q`'s data fixed in the process

This is the case, when it holds: $R_1(d,m), S_1(a,m)$**, but for no** t **both** $R_2(d,t)$ **and** $S_2(a,t)$

Obtaining PCAs for `P` amounts to virtually restoring the satisfaction of `P`'s DEC by virtually modifying `P`'s data

In order to specify `P`'s (virtually) modified relations, introduce virtual versions R_1', R_2' **of** R_1, R_2

`P`'s queries will be expressed in terms of relations R_1', R_2' **only (plus built-ins)**

Contents for R_1', R_2' are obtained from the material sources R_1, R_2, S_1, S_2

Since S_1, S_2 are fixed, the satisfaction of `P`'s DEC requires R_1' to be a subset of R_1, and R_2', a superset of R_2

Specification of R_1', R_2' is done by means of a disjunctive extended logic program Π with answer set semantics

First rules:

$$\begin{aligned} R_1'(x,y) &\leftarrow R_1(x,y),\ \mathit{not}\ \neg R_1'(x,y) & (1)\\ R_2'(x,y) &\leftarrow R_2(x,y),\ \mathit{not}\ \neg R_2'(x,y) & (2)\end{aligned}$$

i.e. by default, the tuples in the sources are copied into the virtual versions, with some exceptions ...

Some of the exceptions for R_1':

$$\begin{aligned}
\neg R_1'(x,y) &\leftarrow R_1(x,y), S_1(z,y),\ \ not\ \ aux_1(x,z),\ \ not\ \ aux_2(z) && (3)\\
aux_1(x,z) &\leftarrow R_2(x,w), S_2(z,w) && (4)\\
aux_2(z) &\leftarrow S_2(z,w) && (5)
\end{aligned}$$

I.e. $R_1(x,y)$ is deleted if simultaneously:

- **It participates in a violation of DEC**

  (captured by the first three literals in (3) plus rule (4))

- **There is no way to restore consistency by inserting a tuple into R_2, because there is no possible matching tuple in S_2 for the possibly new tuple in R_2**

  (captured by last literal in (3) plus rule (5))

In case there is such a tuple in S_2, we can either delete a tuple from R_1 or insert a tuple into R_2:

$$\neg R_1'(x,y) \vee R_2'(x,w) \;\; \leftarrow \;\; R_1(x,y), S_1(z,y), \;\; not \;\; aux_1(x,z), \\ S_2(z,w), choice((x,z),w) \qquad (6)$$

I.e. in case of a violation of DEC, when there is tuple of the form (a,t) in S_2 for the combination of values (d,a), then the *choice operator* non-deterministically chooses a unique value for t, so that the tuple (d,t) is inserted into R_2

(to minimize differences between material and virtual versions)

(as an alternative to deleting (d,m) from R_1)

choice **predicate can be replaced by a standard predicate plus extra rules**

Modified program has a usual answer set semantics

No exceptions are specified for R_2', which makes sense since R_2' is a superset of R_2

Then, the negative literal in the body of (2) can be eliminated

However, new tuples can be inserted into R_2' (captured by rule (6)

Finally, the program contains as facts the tuples in the material relations R_1, R_2, S_1, S_2

If `P` equally trusts itself and `Q`, both `P` and `Q`s' relations are flexible when searching for a solution

Since S_1, S_2 may also change, virtual versions for them must be introduced and specified, and the program becomes more involved

Program Π represents in a compact form all the solutions for a peer

PCAs for a peer can be obtained by running a query program expressed in terms of the virtually repaired tables, in combination with program Π

The combined program is run under the *skeptical answer set semantics*

E.g. the query $Q(x,z):\ \exists y(R_1(x,y) \land R_2(z,y))$ **to** `P` **is peer consistently answered by running program** Π **together with**

$$Ans_Q(x,z) \leftarrow R_1'(x,y), R_2'(x,y)$$

Only the virtual versions of `P`**'s relations appear in the query, but the program will make** `P` **import** `Q`**'s data**

Other Considerations

(A) With referential DECs, the *choice operator* may have to choose values from the infinite domain

Several alternatives considered in the literature

The notion of solution in this regard and the class of referential DECs to deal with will have an impact on decidability, complexity, ...

1. **Open infinite domain, repairing picking up elements from it**

   PCA becomes undecidable with cyclic referential DECs (Cali, Lembo, Rosati; PODS'03)

2. **Repair assigning null values which do not propagate through DECs (Barcelo, Bertossi, Bravo; 2003)**

   It becomes decidable even with cyclic referential DECs

3. **Consider an appropriate finite and closed proper superset of the active domains (Bravo, Bertossi; IJCAI'03)**

4. **Introduce fresh constants whenever needed from a separate domain (Calvanese, Damaggio,DeGiacomo,Lenzerini,Rosati; DBISP2P'03)**

(B) A peer may have local ICs, e.g. a FD

$$\forall x \forall y \forall z (R_1(x,y) \wedge R_1(x,z) \rightarrow y = z)$$

The peer's program that specifies its solutions should take care of them, *at query time*

They can be integrated in our framework by treating them as DECs $\Sigma(\mathtt{P},\mathtt{P})$ **with** $(\mathtt{P},\mathtt{P},\mathit{same})$ **in the trust relationship**

Interaction of Peers' DECs

Peers may be indirectly related by "composition" of DECs, by transitivity ...

Peer `A` gets a query, then gets data from peer `B`, who requests data from peer `C`, ...

`A` may not even know about `C`'s existence ...

There won't be any explicit DECs from `A` to `C`; and we do not want to derive them

We propose that the semantics for such a global exchange system should be given by the "stabilized interaction" of the pair-based solutions

More precisely, by those *global instances that correspond to the stable models of the program that combines the specification programs we had for one peer and its direct neighbors*

In particular, the absence of solutions is reflected in the absence of stable models for the program

Example 2: (cont.)

- `P`: **schema** $\{R_1(\cdot,\cdot), R_2(\cdot,\cdot)\}$
  instance $r(\mathtt{P})$ **with** $r_1 = \{(a,b)\},\ r_2 = \{\}$

  $\Sigma(\mathtt{P},\mathtt{Q})$:

  $$\forall x \forall y \forall z \exists w (R_1(x,y) \wedge S_1(z,y) \ \rightarrow\ R_2(x,w) \wedge S_2(z,w))$$

  `P` trusts `Q` more than itself

- `Q`: **schema** $\{S_1(\cdot,\cdot), S_2(\cdot,\cdot)\}$
  instance $r(\mathtt{Q})$ **with** $s_1 = \{\},\ s_2 = \{(c,e),(c,f)\}$

  $\Sigma(\mathtt{Q},\mathtt{C})$: $\forall x \forall y (U(x,y) \rightarrow S_1(x,y))$

  `Q` trusts `C` more than itself

- `C`: **schema** $\{U(\cdot,\cdot)\}$
  instance $r(\mathtt{C})$ **with** $u = \{(c,b)\}$

Now $\mathtt{Q}$'s relations may also change, actually in this case only S_1, so we also need a virtual version S_1'

Rules (3), (6) are replaced by (7) and (8), resp.

$$\begin{aligned} \neg R_1'(x,y) \;&\leftarrow\; R_1(x,y), S_1'(z,y),\ not\ \ aux_1(x,z), \\ &\qquad not\ \ aux_2(z) \qquad\qquad\qquad\qquad (7) \\ \neg R_1'(x,y) \vee R_2'(x,w) \;&\leftarrow\; R_1(x,y), S_1'(z,y),\ not\ \ aux_1(x,z), \\ &\qquad S_2(z,w), choice((x,z),w) \qquad\quad (8) \end{aligned}$$

Combination program consists of (1), (2),(4), (5), (7), (8) plus

$$\begin{aligned} S_1'(x,y) \;&\leftarrow\; S_1(x,y),\ not\ \neg S_1'(x,y) &\quad (9) \\ S_1'(x,y) \;&\leftarrow\; U(x,y),\ not\ S_1(x,y). &\quad (10) \end{aligned}$$

(9) is a persistence rule for S_1, and (10) enforces $\Sigma(\mathtt{Q},\mathtt{C})$

The solutions obtained from the stable models of the combined program (plus the material sources) are the expected ones:

$\bar{r}' = \{S_2(c,e), S_2(c,f), U(c,b), S_1'(c,b), R_2'(a,f), R_1'(a,b)\}$,

$\bar{r}'' = \{S_2(c,e), S_2(c,f), U(c,b), S_1'(c,b)\}$

$\bar{r}''' = \{S_2(c,e), S_2(c,f), U(c,b), S_1'(c,b),\ R_2'(a,e), R_1'(a,b)\}$

P2P Data Exchange and Data Integration

There are clear connections between PCAs and querying virtual data integration systems

The logic programming-based apprach can be seen as *global-as-view* (GAV) approach: relations in the solutions are specified as views over the peers' original schemas

We explore the connection to the *local-as-view* (LAV) approach, where relations in the (local) data sources are expressed as views of virtual global relations

In example 2, we introduce virtual, global versions S_1', S_2' of S_1, S_2

We propose the following specification:

View definitions	**label**	**source**
$R_1(x,y) \leftarrow R_1'(x,y)$	*closed*	r_1
$R_2(x,y) \leftarrow R_2'(x,y)$	*open*	r_2
$S_1(x,y) \leftarrow S_1'(x,y)$	*clopen*	s_1
$S_2(x,y) \leftarrow S_2'(x,y)$	*clopen*	s_2

Labels (for the sources) in the second column depend on the kind of DECs & the trust relationships

They indicate that S_1, S_2 **do not change;** R_1, R_2 **do change, by deletion or insertion of tuples, resp.**

A query posed to `P` **has to be first reformulated in terms of** R_1', R_2'

Its PCAs can be obtained by querying the integration system subject to the global IC:

$$\forall xyz \exists w (R_1'(x,y) \wedge S_1'(z,y) \rightarrow R_2'(x,w) \wedge S_2'(z,w))$$

There are methodologies for obtaining consistent answers to queries posed to virtual data integration systems with open sources

Also for the LAV approach with mixed sources (Bertossi & Bravo, to appear)

Again this is a specification based on answer set programming of repairs of the virtual system

Some small adjustments required in the P2P scenario

Final Remarks

(A) What we have presented provides *semantics* and *specifications* for peer consistent aswers

In principle it is possible to compute answers from those specifications (and the data available)

As ongoing work, most urgent future work on peer consistent query answering (PCQA):

“Translate” the specifications into concrete algorithms to query the peers’ databases and integrate their answers

(B) At the answer set programming level:

- **It becomes necessary to derive *specialized specifications*, that are easier to handle and compute for particular classes of DECs and queries**

  And from them also *specialized algorithms* for PCQA (c.f. (A))

- **The *specifications* themselves have to be optimized (as logic programs)**

- *Computations* of/under the answer set semantics have to be optimized

  Avoid extra complexity in cases where complexity of PCQA is lower that general data complexity of disjunctive answer set programming

  (The latter is not higher that the *general* data complexity of peer consistent query answering)

- The *interaction* between the logic programming system and the data sources has to be optimized (Eiter, Fink, G.Greco, Lembo; ICLP'03)

Appendix I: Definition of solution

Definition: (direct case) Given a peer $\mathtt{P}$ in a P2P data exchange system and an instance $\bar{r}$ on $\mathcal{R}$, an instance $\bar{r}'$ on $\mathcal{R}$ is a solution for $\mathtt{P}$ if $\bar{r}'$ is a repair of $\bar{r}$ wrt to $\Sigma(\mathtt{P}) \cup IC(\mathtt{P})$ that does not change the more trusted relations

More precisely:
(a) $\bar{r}' \models \bigcup\{\Sigma(\mathtt{P},\mathtt{Q}) \mid (\mathtt{P}, \mathit{less}, \mathtt{Q})$ or $(\mathtt{P}, \mathit{same}, \mathtt{Q}) \in \mathit{trust}\} \cup IC(\mathtt{P})$
(b) $\bar{r}'|P = \bar{r}|P$ for every predicate $P \in \mathcal{R}(\mathtt{Q})$, where $\mathtt{Q}$ is a peer with $(\mathtt{P}, \mathit{less}, \mathtt{Q}) \in \mathit{trust}$
(c) $\bar{r}'$ minimally differs from $\bar{r}$ in the sense that $(\bar{r}' \smallsetminus \bar{r}) \cup (\bar{r} \smallsetminus \bar{r}')$ is minimal under set inclusion among those instances that satisfy (a) and (b)

Intuitively, a solution for `P` repairs the global instance wrt the DECs with peers that `P` trusts more than or the same as itself, but leaving unchanged the tables that belong to more trusted peers

As a consequence of the definition, tables belonging to peers that are not related to `P` or are less trustable are not changed

That is, `P` tries to change its own tables according to what the dependencies to more or equally trusted peers prescribe

Appendix II: Program with three peers, direct case

Example 1: (cont.) R_1, R_3 **have to be flexible in the repair process, and we get interacting rules for** R_1

The logic program should have the effect of repairing the database

The repair process may need to execute several steps until it stabilizes

We use program rules with annotations as introduced for CQA in the presence of interacting ICs

Annotations are constants that are used in an extra argument introduced in each database relation

$\mathbf{t_d}$: used to annotate the atoms that are in the original database instance

Single repair steps are obtained by deriving the annotations $\mathbf{t_a}$ or $\mathbf{f_a}$ (atoms getting them are advised to be made true, resp. false)

This when each IC is considered in isolation, but there may be interacting ICs, which requires an iterative process; for this we use annotations $\mathbf{t^\star}$, $\mathbf{f^\star}$

E.g. $\mathbf{t_d}$ groups together the annotations $\mathbf{t_d}$ and $\mathbf{t_a}$ for the same atom

Derived annotations are used to propagate changes through several ICs

Annotations $\mathbf{t}^{\star\star}$ and $\mathbf{f}^{\star\star}$ are just used to read off the literals that are inside (resp. outside) a repair

Generic rules; to be found in any repair program with annotations

$$\begin{array}{rcl}
R_1(X, Y, \mathbf{t}^{\star}) & \leftarrow & R_1(X, Y, \mathbf{t_d}).\\
R_1(X, Y, \mathbf{t}^{\star}) & \leftarrow & R_1(X, Y, \mathbf{t_a}).\\
R_1(X, Y, \mathbf{f}^{\star}) & \leftarrow & R_1(X, Y, \mathbf{f_a}).\\
R_1(X, Y, \mathbf{f}^{\star}) & \leftarrow & \mathit{dom}(X), \mathit{dom}(Y), \mathit{not}\ R_1(X, Y, \mathbf{t_d}).\\
R_1(X, Y, \mathbf{t}^{\star\star}) & \leftarrow & R_1(X, Y, \mathbf{t_d}), \mathit{not}\ R_1(X, Y, \mathbf{f_a}).
\end{array}$$

$$\begin{array}{rcl}
R_1(X,Y,\mathbf{t}^{\star\star}) & \leftarrow & R_1(X,Y,\mathbf{t_a}). \\
 & \leftarrow & R_1(X,Y,\mathbf{t_a}), R_1(X,Y,\mathbf{f_a}). \\
R_2(X,Y,\mathbf{t}^{\star}) & \leftarrow & R_2(X,Y,\mathbf{t_d}). \\
R_2(X,Y,\mathbf{t}^{\star}) & \leftarrow & R_2(X,Y,\mathbf{t_a}). \\
R_2(X,Y,\mathbf{f}^{\star}) & \leftarrow & R_2(X,Y,\mathbf{f_a}). \\
R_2(X,Y,\mathbf{f}^{\star}) & \leftarrow & dom(X), dom(Y), not\ \ R_2(X,Y,\mathbf{t_d}). \\
R_2(X,Y,\mathbf{t}^{\star\star}) & \leftarrow & R_2(X,Y,\mathbf{t_d}), not\ \ R_2(X,Y,\mathbf{f_a}). \\
R_2(X,Y,\mathbf{t}^{\star\star}) & \leftarrow & R_2(X,Y,\mathbf{t_a}). \\
 & \leftarrow & R_2(X,Y,\mathbf{t_a}), R_2(X,Y,\mathbf{f_a}). \\
R_3(X,Y,\mathbf{t}^{\star}) & \leftarrow & R_3(X,Y,\mathbf{t_d}). \\
R_3(X,Y,\mathbf{t}^{\star}) & \leftarrow & R_3(X,Y,\mathbf{t_a}). \\
R_3(X,Y,\mathbf{f}^{\star}) & \leftarrow & R_3(X,Y,\mathbf{f_a}).
\end{array}$$

$$\begin{array}{rcl}
R_3(X,Y,\mathbf{f}^\star) & \leftarrow & dom(X), dom(Y), not\ R_3(X,Y,\mathbf{t_d}).\\
R_3(X,Y,\mathbf{t}^{\star\star}) & \leftarrow & R_3(X,Y,\mathbf{t_d}), not\ R_3(X,Y,\mathbf{f_a}).\\
R_3(X,Y,\mathbf{t}^{\star\star}) & \leftarrow & R_3(X,Y,\mathbf{t_a}).\\
 & \leftarrow & R_3(X,Y,\mathbf{t_a}), R_3(X,Y,\mathbf{f_a}).
\end{array}$$

Now we have only two specific rules, they express how to repair the databases when a violation of the DECs occurs:

$$\begin{array}{rcl}
R_1(X,Y,\mathbf{t_a}) & \leftarrow & R_2(X,Y,\mathbf{t}^\star), R_1(X,Y,\mathbf{f}^\star).\\
R_1(X,Y,\mathbf{f_a}) \vee R_3(X,Z,\mathbf{f_a}) & \leftarrow & R_1(X,Y,\mathbf{t}^\star), R_3(X,Z,\mathbf{t}^\star), Y \neq Z.
\end{array}$$

The first one corresponds to a violation of $\Sigma(\mathtt{P1},\mathtt{P2})$; the second one, to a violation of $\Sigma(\mathtt{P1},\mathtt{P3})$

The facts of the program:

$R_1(a, b, \mathbf{t_d})$. $R_1(s, t, \mathbf{t_d})$. $R_2(c, d, \mathbf{t_d})$. $R_2(a, e, \mathbf{t_d})$.$R_2(t, h, \mathbf{t_d})$. $R_3(a, f, \mathbf{t_d})$. $R_3(s, u, \mathbf{t_d})$. $R_3(t, u, \mathbf{t_d})$. $dom(a)$. $dom(b)$. $dom(s)$. $dom(t)$. $dom(c)$. $dom(d)$. $dom(e)$. $dom(f)$. $dom(u)$. $dom(h)$.

The non domain atoms say that originally
$R_1 = \{(a, b), (s, t)\}$
$R_2 = \{(c, d), (a, e), (t, h)\}$
$R_3 = \{(a, f), (s, u), (t, u)\}$

Here we do not need virtual versions R_1', R_3' for R_1, R_2, because their final contents will be read off from atoms annotated with $\mathbf{t}^{\star\star}$

Appendix III: Comparison with data integration

Methodology for CQA under LAV and mixed sources is based on a three-layered specification of the repairs:

- A first layer specifies the contents of the global relations in the minimal legal instances (to this layer only open and clopen sources contribute)
- A second layer consisting of program denial constraints that prunes the models that violate the closure condition for the closed sources
- A third layer specifying the minimal repairs of the legal instances left by the other layers wrt the global ICs (repairs may violate the source labels)

In the P2P scenario we consider only legal instances that:

- **Satisfy the mapping in the table**
- **In the case of closed sources, include the maximum amount of tuples from them (virtual relations must be kept as close as possible to their original, material versions)**

This can be achieved using the same specifications as for the mixed case, *but* considering the closed sources as clopen

They contribute with rules that import their contents into the system (maximizing tuples in the global relation) and denial program constraints

Trust relation also makes a difference: virtual relations must satisfy the original labels (which capture the trust relationships)

Then repairs of legal instances are based only on tuple deletions (insertions) for global relations corresponding to closed (resp. open) sources

For clopen sources the rules can neither add nor delete tuples

This preference criterion on repairs is similar to the *loosely-sound semantic* for integration of open sources under GAV
(Lembo, Lenzerini, Rosati; KRDB'02)

Methodology handles universal and acyclic referential DECs

This is when arbitrary elements from the infinite underlying domain can be picked up to satisfy the DECs

When repairs are done using null values that do not propagate through ICs, then cycles are allowed

The DEC in example 2 is not a typical referential IC, but the repair layer can be adjusted in order to generate the solutions for `P`

Assume the peers have the following instances:

$r_1 = \{(a, b)\}, s_1 = \{(c, b)\},\ r_2 = \{\}$ **and** $s_2 = \{(c, e), (c, f)\}$

The layer that specifies the preferred legal instances:

$$\begin{array}{rcl}
R_1'(X, Y, \mathbf{t_d}) & \leftarrow & R_1(X, Y). \\
S_1'(X, Y, \mathbf{t_d}) & \leftarrow & S_1(X, Y). \\
R_2'(X, Y, \mathbf{t_d}) & \leftarrow & R_2(X, Y). \\
S_2'(X, Y, \mathbf{t_d}) & \leftarrow & S_2(X, Y). \\
 & \leftarrow & R_1'(X, Y, \mathbf{t_d}), R_1(X, Y). \\
 & \leftarrow & S_1'(X, Y, \mathbf{t_d}), S_1(X, Y). \\
 & \leftarrow & S_2'(X, Y, \mathbf{t_d}), S_2(X, Y).
\end{array}$$

The layer that specifies the repairs of the legal instances:

(The annotation constants are used as auxiliary elements in the repairs process)

$$\begin{array}{rcl}
R'_1(X,Y,\mathbf{t^{\star\star}}) & \leftarrow & R'_1(X,Y,\mathbf{t_d}),\ \ not\ \ R'_1(X,Y,\mathbf{f_a}).\\
R'_1(X,Y,\mathbf{t^{\star\star}}) & \leftarrow & R'_1(X,Y,\mathbf{t_a}).\\
 & \leftarrow & R'_1(X,Y,\mathbf{t_a}), R'_1(X,Y,\mathbf{f_a}).\\
S'_1(X,Y,\mathbf{t^{\star\star}}) & \leftarrow & S'_1(X,Y,\mathbf{t_d}),\ \ not\ \ S'_1(X,Y,\mathbf{f_a}).\\
S'_1(X,Y,\mathbf{t^{\star\star}}) & \leftarrow & S'_1(X,Y,\mathbf{t_a}).\\
 & \leftarrow & S'_1(X,Y,\mathbf{t_a}), S'_1(X,Y,\mathbf{f_a}).\\
R'_2(X,Y,\mathbf{t^{\star\star}}) & \leftarrow & R'_2(X,Y,\mathbf{t_d}),\ \ not\ \ R'_2(X,Y,\mathbf{f_a}).
\end{array}$$

$$
\begin{array}{rcl}
R'_2(X,Y,\mathbf{t}^{\star\star}) & \leftarrow & R'_2(X,Y,\mathbf{t_a}).\\
 & \leftarrow & R'_2(X,Y,\mathbf{t_a}), R'_2(X,Y,\mathbf{f_a}).\\
S'_2(X,Y,\mathbf{t}^{\star\star}) & \leftarrow & S'_2(X,Y,\mathbf{t_d}),\ \ not\ \ S'_2(X,Y,\mathbf{f_a}).\\
S'_2(X,Y,\mathbf{t}^{\star\star}) & \leftarrow & S'_2(X,Y,\mathbf{t_a}).\\
 & \leftarrow & S'_2(X,Y,\mathbf{t_a}), S'_2(X,Y,\mathbf{f_a}).\\
R'_1(X,X,\mathbf{f_a}) & \leftarrow & R'_1(X,Y,\mathbf{t_d}), S'_1(Z,Y,\mathbf{t_d}),\\
 & & \ \ not\ \ aux_1(X,Z),\ \ not\ \ aux_2(Z).\\
aux_1(X,Z) & \leftarrow & R'_2(X,U,\mathbf{t_d}), S'_2(Z,U,\mathbf{t_d}).\\
aux_2(Z) & \leftarrow & S'_2(Z,W,\mathbf{t_d}).\\
R'_1(X,Y,\mathbf{f_a}) \vee R'_2(X,W,\mathbf{t_a}) & \leftarrow & R'_1(X,Y,\mathbf{t_d}), S'_1(Z,Y,\mathbf{t_d}),\\
 & & \ \ not\ \ aux_1(X,Z), S'_2(Z,W,\mathbf{t_d}),\\
 & & choice((X,Z),W).
\end{array}
$$

Running this program with DLV, we obtain the following solutions (they can be obtained by selecting only the tuples with annotation t** from a stable model):

$\bar{r}^1 = \{S'_1(c,b),\ S'_2(c,e),\ S'_2(c,f),\ R'_1(a,b),\ R'_2(a,f)\}$

$\bar{r}^2 = \{S'_1(c,b),\ S'_2(c,e),\ S'_2(c,f)\}$

$\bar{r}^3 = \{S'_1(c,b),\ S'_2(c,e),\ S'_2(c,f),\ R'_1(a,b),\ R'_2(a,e)\}$

$\bar{r}^4 = \{S'_1(c,b),\ S'_2(c,e),\ S'_2(c,f)\}$