



Carleton  
UNIVERSITY

From Consistent Query  
Answering to Query  
Rewriting:

A Detour around Answer Set Programs

Leopoldo Bertossi

Carleton University  
School of Computer Science  
Ottawa, Canada

## Why Consistent Query Answering?

A (relational) database instance  $D$  may be inconsistent

It does not satisfy a given set of integrity constraints  $IC$ :  $D \not\models IC$

$D$  is a FO structure, identified with a finite set of ground atoms of the FO language associated to the relational schema

However, we do not throw  $D$  away

Most of the data in it is still consistent, i.e. intuitively and informally, it does not participate in the violation of  $IC$

We can still obtain meaningful and correct answers from  $D$

Initial motivation for research in “Consistent Query Answering” (CQA):

- Characterize in precise terms the data in  $D$  that is consistent with  $IC$
- Develop mechanisms for computing/extracting the consistent information from  $D$

Obtain answers to queries from  $D$  that are consistent with  $IC$

(Arenas, Bertossi, Chomicki; Pods 1999)

## Consistent Answers and Repairs

**Example:** Database instance  $D$  and  $FD: Name \rightarrow Salary$

<i>Employee</i>	<i>Name</i>	<i>Salary</i>
	<i>Page</i>	5K
	<i>Page</i>	8K
	<i>Smith</i>	3K
	<i>Stowe</i>	7K

$D$  violates  $FD$  through the tuples with *Page* in *Name*

There are two possible ways to repair the database in a minimal way if only deletions/insertions of whole tuples are allowed

Repairs  $D_1$ , resp.  $D_2$

<i>Employee</i>	<i>Name</i>	<i>Salary</i>
	<i>Page</i>	<b>5K</b>
	<i>Smith</i>	3K
	<i>Stowe</i>	7K

<i>Employee</i>	<i>Name</i>	<i>Salary</i>
	<i>Page</i>	<b>8K</b>
	<i>Smith</i>	3K
	<i>Stowe</i>	7K

$(Stowe, 7K)$  persists **in all** repairs, and it does not participate in the violation of  $FD$ ; it is **invariant** under these minimal ways of restoring consistency

$(Page, 8K)$  does not persist in all repairs, and it does participate in the violation of  $FD$

Fixed: DB schema with (infinite) domain; and a set of first order integrity constraints  $IC$

**Definition:** (Arenas et al.; Pods 1999)

A **repair** of instance  $D$  is an instance  $D'$

- over the same schema and domain
- satisfies  $IC$ :  $D' \models IC$
- Makes  $\Delta(D, D')$  minimal wrt set inclusion

**Definition:** (Arenas et al.; Pods 1999)

Tuple of constants  $\bar{t}$  is a **consistent answer** to query  $Q(\bar{x})$  in  $D$  iff

$\bar{t}$  is an answer to query  $Q(\bar{x})$  in every repair  $D'$  of  $D$ :

$$D \models_{IC} Q(\bar{t}) \quad :\iff \quad D' \models Q(\bar{t}) \quad \text{for every repair } D' \text{ of } D$$

A model-theoretic definition ...

Example: (continued)

$$D \models_{FD} \text{Employee}(\text{Stowe}, 7K)$$

$$D \models_{FD} (\text{Employee}(\text{Page}, 5K) \vee \text{Employee}(\text{Page}, 8K))$$

$$D \models_{FD} \exists x \text{Employee}(\text{Page}, x)$$

Example:  $D = \{P(a, b), Q(c, b)\}$ ,  $IC: \forall x \forall y (P(x, y) \rightarrow Q(x, y))$

The repairs are:

- $D_1 = \{Q(c, b)\}$  with  $\Delta(D, D_1) = \{P(a, b)\}$
- $D_2 = \{P(a, b), Q(a, b), Q(c, b)\}$  with  $\Delta(D, D_2) = \{Q(a, b)\}$

But not  $D_3 = \{P(a, b), Q(a, b)\}$ , because

$$\Delta(D, D_3) = \{Q(a, b), Q(c, b)\} \not\subseteq \Delta(D, D_2)$$

## Computing Consistent Answers

We want to **compute** consistent answers, **but not** by computing all possible repairs and checking answers in common

Retrieving consistent answers via explicit and material computation of **all** database repairs may not be the right way to go

**Example:** An inconsistent instance wrt  $FD: X \rightarrow Y$

$D$	$X$	$Y$
	1	0
	1	1
	2	0
	2	1
	·	·
	$n$	0
	$n$	1

It has  $2^n$  possible repairs!

Try to avoid or minimize computation of repairs ...



## FO Query Rewriting (sometimes)

First-Order queries and constraints

ICs are universal (as in the rest of this presentation)

Approach: Transform the query and keep the database instance!

- Consistent answers to  $Q(\bar{x})$  in  $D$ ?
- Rewrite query:  $Q(\bar{x}) \mapsto Q'(\bar{x})$   
 $Q'(\bar{x})$  is a new FO query
- Retrieve from  $D$  the (ordinary) answers to  $Q'(\bar{x})$

Example:  $D = \{P(a), P(b), Q(b), Q(c)\}$

$IC: \forall x(P(x) \rightarrow Q(x))$

$Q(x): P(x)?$  (consistent answer should be  $(b)$ )

If  $P(x)$  holds, then  $Q(x)$  must hold

An answer  $t$  to  $P(x)$  is consistent if  $t$  is also answer to  $Q(x)$

Rewrite  $Q(x)$  into  $Q'(x) : P(x) \wedge Q(x)$  and pose it to  $D$

$Q(x)$  is a **residue** of  $P(x)$  wrt  $IC$

Residue obtained by resolution between query literal and  $IC$

Posing new query to  $D$  (as usual) we get only answer  $(b)$

Example: (continued) Same *FD*:

$$\forall xyz (\neg Employee(x, y) \vee \neg Employee(x, z) \vee y = z)$$

$$Q(x, y): Employee(x, y)$$

Consistent answers:  $(Smith, 3K), (Stowe, 7K)$   
 (but not  $(Page, 5K), (Page, 8K)$ )

Can be obtained via rewritten query:

$$T(Q(x, y)) := Employee(x, y) \wedge \forall z (\neg Employee(x, z) \vee y = z)$$

... those tuples  $(x, y)$  in the relation for which  $x$  does not have and associated  $z$  different from  $y$  ...

In general,  $T$  has to be applied iteratively

To the literals appearing in the residues appended at the previous step

In the two previous examples there were no (new) residues

We reached a fixed point

Does it always exist?

In general, an infinitary query:

$$T^\omega(Q(x)) := \bigcup_{n < \omega} \{T^n(Q(x))\}$$

Is  $T^\omega$  sound, complete, finitely terminating?

Several **sufficient and necessary syntactic conditions on ICs and queries** have been identified for these properties to hold

(Arenas et al.; Pods 1999)

Main **limitations** are:

- Universal ICs
- Queries are projection-free conjunctions of literals

**Example:**

$$IC = \{ \forall xy(P(x, y) \rightarrow R(x, y)), \forall xy(R(x, y) \rightarrow P(x, y)), \\ \forall xyz(P(x, y) \wedge P(x, z) \rightarrow y = z) \}$$

$$Q(x, y) : R(x, y) \wedge \neg P(x, y)$$

**Other approaches to FO query rewriting were in principle possible**

## What Kind of Logic for CQA?

From the **logical point of view**:

- We have not logically **specified** the database repairs
- We have a **model-theoretic definition** plus an incomplete computational mechanism
- From such a specification *Spec* we might:
  - Reason from *Spec*
  - Consistently answer queries:  $Spec \stackrel{?}{\models} Q(\bar{x})$
  - Derive algorithms for consistent query answering

Notice ...

Example: Database  $D$  and  $FD: Name \rightarrow Salary$

<i>Employee</i>	<i>Name</i>	<i>Salary</i>
	<i>Page</i>	5K
	<i>Smith</i>	3K
	<i>Stowe</i>	7K

It holds:  $D \models_{FD} Employee(Page, 5K)$

However

$$D \cup \{Employee(Page, 8K)\} \not\models_{FD} Employee(Page, 5K)$$

- Consistent query answering is non-monotonic
- A non-monotonic semantics for *Spec* and its logic is expected
- What other logical properties of CQA reasoning/entailment?

## Specifying Database Repairs

The class of all database repairs can be represented in a compact form

This class can be specified using logic programs

Use **disjunctive logic programs with stable model semantics**  
(a.k.a. Answer Set Programs) (Gelfond, Lifschitz; NGC 1991)

Here we use the ASPs essentially introduced in  
(Barcelo, Bertossi; PADL 2003)

Repairs correspond to distinguished models of the program,  
namely to its stable models

The programs use **annotation constants** in an extra attribute in  
the database relations



- To keep track of the atomic repair actions, i.e. insertions or deletions of tuples ( $\mathbf{t}, \mathbf{f}$ )
- To give feedback to the program in case additional changes become necessary due to interacting ICs ( $\mathbf{t}^*$ )
- To collect the tuples in the final, repaired instances ( $\mathbf{t}^{**}$ )

Annotation	Atom	The tuple $P(\bar{a})$ is ...
$\mathbf{d}$	$P(\bar{a}, \mathbf{d})$	fact in original database
$\mathbf{t}$	$P(\bar{a}, \mathbf{t})$	made true (inserted)
$\mathbf{f}$	$P(\bar{a}, \mathbf{f})$	made false (deleted)
$\mathbf{t}^*$	$P(\bar{a}, \mathbf{t}^*)$	true or made true
$\mathbf{t}^{**}$	$P(\bar{a}, \mathbf{t}^{**})$	true in the repair

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

$$D = \{P(c, l), P(d, m), Q(d, m), Q(e, k)\}$$

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

1. Original data facts:  $P(c, l, \mathbf{d})$ , etc.
2. Whatever was true or becomes true, is annotated with  $\mathbf{t}^*$ :

$$P(\bar{x}, \mathbf{t}^*) \leftarrow P(\bar{x}, \mathbf{d})$$

$$P(\bar{x}, \mathbf{t}^*) \leftarrow P(\bar{x}, \mathbf{t}) \quad (\text{the same for } Q)$$

3. There may be interacting ICs (not here), and the repair process may take several steps, changes could trigger other changes

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

$$P(\bar{x}, \mathbf{f}) \vee Q(\bar{x}, \mathbf{t}) \leftarrow P(\bar{x}, \mathbf{t}^*), \text{ not } Q(\bar{x}, \mathbf{d})$$

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

$$P(\bar{x}, \mathbf{f}) \vee Q(\bar{x}, \mathbf{t}) \leftarrow P(\bar{x}, \mathbf{t}^*), \text{ not } Q(\bar{x}, \mathbf{d})$$

Two rules per IC; that says how to repair the IC (c.f. the head) in case of a violation (c.f. the body)

Passing to annotation  $\mathbf{t}^*$  allows to keep repairing the DB wrt to all the ICs until the process stabilizes

4. Repairs must be **coherent**: Program denial constraints prune undesirable models

$$\leftarrow P(\bar{x}, \mathbf{t}), P(\bar{x}, \mathbf{f})$$

$$\leftarrow Q(\bar{x}, \mathbf{t}), Q(\bar{x}, \mathbf{f})$$

5. Annotations constants  $\mathbf{t}^{**}$  are used to read off the atoms in a repair

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

$$P(\bar{x}, \mathbf{t}^{**}) \leftarrow P(\bar{x}, \mathbf{d}), \text{ not } P(\bar{x}, \mathbf{f})$$

Etc.

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

$$\begin{aligned} \mathcal{M}_1 &= \{P(c, l, \mathbf{d}), \dots, P(c, l, \mathbf{t}^*), Q(c, l, \mathbf{t}), P(c, l, \mathbf{t}^{**}), \\ &\quad Q(c, l, \mathbf{t}^*), P(d, m, \mathbf{t}^{**}), Q(d, m, \mathbf{t}^{**}), \dots, Q(c, l, \mathbf{t}^{**})\} \\ &\equiv \{P(c, l), Q(c, l), P(d, m), Q(d, m), Q(e, k)\} \end{aligned}$$

... insert  $Q(c, l)$ !!

$$\begin{aligned} \mathcal{M}_2 &= \{P(c, l, \mathbf{d}), \dots, P(c, l, \mathbf{t}^*), P(d, m, \mathbf{t}^{**}), Q(d, m, \mathbf{t}^{**}), \\ &\quad \dots, P(c, l, \mathbf{f}), \dots\} \\ &\equiv \{P(d, m), Q(d, m), Q(e, k)\} \end{aligned}$$

... delete  $P(c, l)$ !!

One-to-one correspondence between repairs and stable models of the program

## Obtaining Consistent Answers

To obtain consistent answers to a (FO) query:

1. Transform or provide the query as a logic program (a standard process)

$$1. \quad Q(\dots P(\bar{u}) \dots) \longmapsto Q' := Q(\dots P(\bar{u}, \mathbf{t}^{**}) \dots)$$

$$2. \quad Q'(\bar{x}) \longmapsto (\Pi(Q'), Ans(\bar{X}))$$

$\Pi(Q')$  is a query program, a third layer on top of the DB and the repair program

$Ans(\bar{X})$  is a query atom defined in  $\Pi(Q')$

2. Run the query program together with the specification program under the **skeptical or cautious stable model semantics**

It sanctions as true of a program **what is true of all its stable models**

“Run”  $\Pi := \Pi(Q') \cup \Pi(D, IC)$

3. Collect ground atoms

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

Example: (continued)

Consistent answers to  $Q(x, y) : P(x, y)$

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

$$Ans(\bar{x}) \leftarrow P(\bar{x}, \mathbf{t}^{**})$$

The two previous stable models become extended with ground *Ans* atoms

$$\mathcal{M}'_1 = \mathcal{M}_1 \cup \{Ans(c, l), Ans(d, m)\}$$

$$\mathcal{M}'_2 = \mathcal{M}_2 \cup \{Ans(d, m)\}$$

Then the only answer is  $(d, m)$

- Use of DLP is a general methodology for CQA for universal ICs and referential ICs, general FO queries (and beyond)

- ASPs can be used to provide **declarative and executable specifications** of database repairs
- ASP based specification of repairs and CQAs as consequences from a program provide **some sort of logic for CQA**

A non-classical logic though ...

- ASPs extended with query programs provide **a form of query rewriting**:  $D, Q \mapsto \Pi(D, IC) \cup \Pi(Q)$

Leaving aside the instance (program facts), this is query rewriting; in a language that is more expressive than FOL ...

- The same repair program can be used with all queries, the same applies to the computed stable models

The query at hand adds a final layer on top



## Complexity: Immediate Results

- When a FO query rewriting approach works (e.g. correct and finitely terminating in case of  $T^\omega$ ), consistent answers to FO queries can be computed in *PTIME* in data
- The **problem of CQA** is a decision problem:

$$CQA(Q(\bar{x}), IC) := \{(D, \bar{t}) \mid D \models_{IC} Q(\bar{t})\}$$

Data complexity?

- Query answering from DLPs (under skeptical stable models semantics) is  $\Pi_2^P$ -complete in data  
(Dantsin, Eiter, Gottlob, Voronkov; ACM CSs 2001)

This provides an upper bound for CQA

- There are classes of disjunctive programs for which query answering has lower complexity

For **head-cycle free programs** (HCF), query answering becomes *coNP*-complete

HCF programs are defined in terms of a directed graph  $G(\Pi)$ :  $\Pi$  is HCF iff  $G(\Pi)$  has no cycles through positive atoms in bodies containing two atoms in the head of the same rule

- For some classes of ICs, repair programs become HCF

For sets  $IC$  of **denial constraints**,  $\Pi(D, IC)$  is HCF

$$\forall xyz \neg (R(x, y) \wedge S(y, z) \wedge T(x, z) \wedge z \neq y)$$

And we have a better upper bound for CQA

## Closing and Understanding the Gap

In a series of papers, *P*TIME algorithms for CQA were provided

Applying graph-theoretic methods to

- Repairs: they correspond to maximal independent sets in conflict (hyper)graphs or hypergraphs)

(Arenas, Bertossi, Chomicki; ICDT 2001),

(Chomicki, Marcinkowski; Inf&Comp 2005)

or

- Syntactic structure of queries

In all those cases, the query was also FO rewritable for CQA

1. FDs and projection-free conjunctive queries

Also some conjunctive queries with limited projection  
(Ch, M; I&C 2005)

2. Key Constraints (KCs) and some **syntactic classes of conjunctive queries** with restricted projection

(Fuxman, Miller; ICDT 2005)

Classes defined by the graph-theoretic syntactic structure of the query and its interaction with the KCs

For every query  $Q$  in the class,  $\mathcal{C}_{forest}$ , there is a FO rewriting  $Q'$  for CQA

$$Q : \exists x \exists y \exists z (R(\underline{x}, z) \wedge S(\underline{z}, y)) \mapsto$$

$$Q' : \exists x \exists z' (R(\underline{x}, z') \wedge \forall z (R(\underline{x}, z) \rightarrow \exists y S(\underline{z}, y)))$$

3. As in 2., but extending the class of queries (*rooted queries*)

Same property of FO rewritability (Wijsen; DBPL 2007)

Classes of queries above are rather sharp, i.e. not satisfying some of their syntactic conditions increases complexity

## Lower bounds?

- For arbitrary FDs and inclusion dependencies (deletions only) CQA becomes  $\Pi_2^P$ -complete (Ch,M; I&C 2005)
- For KCs and conjunctive queries (with some forms of projection) CQA becomes *coNP*-complete

$$Q : \exists z \exists y \exists z (R(\underline{x}, z) \wedge S(\underline{y}, z))$$

(Ch,M; I&C 2005), (F,M; ICDT 2005), (W; DBPL 2007)

- FO Rewriting vs. *PTIME*

There are sets of KCs  $\mathcal{K}$  and conjunctive queries  $Q$  for which CQA is in *PTIME*, but there is no FO rewriting of  $Q$  for CQA

- $Q : \exists x \exists y \exists z (R(\underline{x}, z) \wedge R(\underline{y}, z) \wedge x \neq y)$

Reduction techniques

(Fuxman, Miller; IJWeb 2003)

- $Q : \exists x \exists y (R(\underline{x}, y) \wedge R(\underline{y}, c))$

Using Ehrenfeucht-Fraïssé games

(Wijsen; DBPL 2007)

## Getting More from ASPs

- Complexity of query evaluation from disjunctive logic programs (DLPs) coincides with the complexity of CQA
- However, for some classes of queries and ICs, CQA has a lower complexity, e.g. in *PTIME*
- The landscape between FO rewritable cases and  $\Pi_2^P$ -completeness for CQA still not quite clear
- Results obtained in the middle ground are scattered, isolated, and rather ad hoc
- The “logics” of CQA is not fully understood yet

Some natural questions arise ...



- Can we identify classes of ICs and queries for which repair programs can be automatically “simplified” into queries of lower complexity?
  - Can we reobtain previous classes?
  - Can we identify new ones?
  - Can we obtain new complexity results?
- Can we better understand the logic of CQA through the analysis of repair programs?
- Can we take advantage of results about updates of LPs to deal with the problem of CQA under updates? (almost untouched problem)

Some progress in this research program ... (ongoing work)

## SO Rewriting of Repair Programs (and more)

**Example:**  $P(X, Y) : X \rightarrow Y$

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

Repair program:

$$P(x, y, \mathbf{f}) \vee P(x, z, \mathbf{f}) \leftarrow P(x, y, \mathbf{d}), P(x, z, \mathbf{d}), y \neq z$$

$$P(x, y, \mathbf{t}^{**}) \leftarrow P(x, y, \mathbf{d}), \text{ not } P(x, y, \mathbf{f})$$

$$P(a, b, \mathbf{d}). \quad P(a, c, \mathbf{d}). \quad P(d, e, \mathbf{d}).$$

This program can be seen as a FO specification (forget about the stable model semantics), i.e. a **FO conjunction**  $\Psi_\rho$  of

$$P(x, y) \wedge P(x, z) \wedge y \neq z \rightarrow P_f(x, y) \vee P_f(x, z)$$

$$P(x, y) \wedge \neg P_f(x, y) \rightarrow P_{**}(x, y)$$

$$P(a, b) \wedge P(a, c) \wedge P(d, e)$$

Quite recently a stable model semantics has been introduced for any FO sentence (Ferraris, Lee, Lifschitz; IJCAI 2007)

$\Psi \mapsto \Psi'$  and  $\Psi'$  is a **SO sentence** (same signature)

The stable models of  $\Psi$  are the Herbrand models of  $\Psi'$  (the stable sentence)

For DLPs, this “stable semantics” coincides with their original stable model semantics

In our case:

- The stable sentence for a repair program (as a FO sentence  $\Psi_\rho$ ) is always a circumscription

Parallel circumscription of all the predicates in the program

- Given the structure of the repair program (also including the query program), the circumscription becomes a prioritized circumscription

In the example, minimize predicates in this order: database predicates, predicates annotated with  $f$ , predicates annotated with  $t^{**}$ , the *Ans* predicate

- Most complex is minimization of predicates defined by disjunctive rules (those associated to the ICs)

For all the others we can apply predicate completion

In the example,  $\Psi'_\rho$  becomes

$$\forall xy(P(x, y) \equiv (x = a \wedge y = b) \vee (x = a \wedge y = c) \vee (x = d \wedge y = e)) \\ \wedge \forall xy((P(x, y) \wedge \neg P_f(x, y)) \equiv P_{\star\star}(x, y)) \wedge$$

$$\forall xyz(P(x, y) \wedge P(x, z) \wedge y \neq z \rightarrow (P_f(x, y) \vee P_f(x, z))) \wedge \\ \neg \exists U((U < P_f) \wedge \forall xyz(P(x, y) \wedge P(x, z) \wedge y \neq z \rightarrow \\ (U(x, y) \vee U(x, z)))) \quad (*)$$

- Predicate  $P_f$  minimized via the last conjunct (\*) of  $\Psi'_\rho$
- $U < P_f$  stands for
 
$$\forall xy(U(x, y) \rightarrow P_f(x, y)) \wedge \exists xy(P_f(x, y) \wedge \neg U(x, y))$$

Consistent query answering?

$$Q(x, y) : P(x, y)$$

$$\Psi'_\rho \wedge \forall x \forall y (Ans(x, y) \equiv P_{\star\star}(x, y)) \stackrel{?}{\models} Ans(x, y)$$

Classical logical entailment!

Anything else?

Eliminate SO quantifiers from  $\Psi'_\rho$  ...

(Doherty, Lukaszewicz, Szalas; JAR 1997)

Let  $\kappa(x, y, z)$  stand for  $P(x, y) \wedge P(x, z) \wedge y \neq z$

The negation of (\*) is logically equivalent to

$$\begin{aligned} \exists st \exists f \exists U \forall x \forall r (\forall x_1 y_1 z_1 (\neg \kappa(x_1, y_1, z_1) \vee f(x_1, y_1, z_1) = \vee(y_1, z_1)) \\ \wedge \forall yz (\neg \kappa(x, y, z) \vee r \neq f(x, y, z) \vee U(x, r))) \\ \wedge \forall uv (\neg U(u, v) \vee P_f(u, v)) \wedge (P_f(s, t) \wedge \neg U(s, t))) \end{aligned}$$

( $t = \vee(t_1, t_2)$  stands for  $t = t_1 \vee t = t_2$ )

Now we are ready to apply Ackermann's lemma

The formula is of the form

$$\exists st \exists f \exists U \forall x \forall r ((A(x, r) \vee U(x, r)) \wedge B \frac{U}{\neg U}) \quad (**)$$

$B \frac{U}{\neg U}$  is formula  $B$  with predicate  $U$  replaced by  $\neg U$

$$\begin{aligned}
A(x, r) : & \quad \forall yz(\forall yz(\neg\kappa(x, y, z) \vee r \neq f(x, y, z))) \\
B(U) : & \quad \forall x_1y_1z_1(\neg\kappa(x_1, y_1, z_1) \vee f(x_1, y_1, z_1) = \vee(y_1, z_1)) \wedge \\
& \quad \forall uv(U(u, v) \vee P_f(u, v)) \wedge (P_f(s, t) \wedge U(s, t))
\end{aligned}$$

$B$  is positive in  $U$

The subformula in (\*\*) starting with  $\exists U$  can be equivalently replaced by  $B \frac{U}{A(x,r)}$ , eliminating  $U$ :

$$\begin{aligned}
\exists st \exists f & (\forall x_1y_1z_1(\neg\kappa(x_1, y_1, z_1) \vee f(x_1, y_1, z_1) = \vee(y_1, z_1)) \wedge \\
& \quad \forall uv(\forall yz(\neg\kappa(u, y, z) \vee v \neq f(u, y, z) \vee P_f(u, v)) \wedge \\
& \quad (P_f(s, t) \wedge \forall y_1z_1(\neg\kappa(s, y_1, z_1) \vee t \neq f(s, y_1, z_1)))
\end{aligned}$$

Unskolemizing:

$$\begin{aligned}
\exists st \forall xyz \exists w & ((\neg\kappa(x, y, z) \vee w = \vee(y, z)) \wedge \\
& \quad (\neg\kappa(x, y, z) \vee P_f(u, w)) \wedge \\
& \quad (P_f(s, t) \wedge (x \neq s \vee \neg\kappa(x, y, z) \vee t \neq w)))
\end{aligned}$$

Its negation is equivalent (via other conjuncts in page 37) to

$$\forall st(P_f(s, t) \rightarrow \exists xyz\forall w(\kappa(x, y, z) \wedge (P_f(x, w) \rightarrow (x = s \wedge t = w))))),$$

which can be replaced for (\*) in page 37, obtaining an equivalent FO specification of predicate  $P_f$

A FO theory  $\Psi''_\rho$  we can do CQA with

$$Q(x, y) : P(x, y)$$

$$\Psi''_\rho \wedge \forall xy(Ans(x, y) \equiv P_{\star\star}(x, y)) \stackrel{?}{\models} Ans(x, y)$$

Classical FO entailment!

In this case, by simple logical transformation, equivalent to

$$D \models P(x, y) \wedge \neg\exists z(P(x, z) \wedge z \neq y)$$

Reobtaining the original FO rewriting!



## Final Remarks

- For FDs (and KCs), this methodology provably works

Has to be exploited now ...

- Apparently FO rewritings in (Arenas et al.; 1999) can be reobtained (for other universal ICs too)

- The query is posed on top of a FO specification of repairs

The database (its completion) lies at the bottom

Like doing query answering in DBs with complex, expressive FO views that are implicitly defined

- Interesting to investigate the kind of FO theories obtained

- The FO specification of repairs can be used to reobtain FO and other new rewritings for CQA
- Use FO theory to analyze complexity of CQA
- Ackermann's Lemma can be extended and SO quantifier elimination produces a Fixpoint formula (Nonnengart, Szalas; 1998)

Relevant cases in CQA?

Relevant for *P*TIME vs. FO rewriting?