Static, Incremental and Parameterized Complexity of Consistent Query Answering in Databases Under Cardinality-Based Semantics

Leopoldo Bertossi

Carleton University

Ottawa, Canada

Based in part on join work with:

Andrei Lopatenko (U. Bolzano-Bozen)

Introduction

Databases may become inconsistent wrt a given set of integrity constraints (ICs)

- DBMS has no mechanism to maintain certain classes of ICs
- Data of different sources are being integrated

  Even if the independent data sources are consistent, integrated data may not
- New constraints are imposed on pre-existing, legacy data
- Soft, user, or informational constraints, to be considered at query answering, but without being enforced

Most likely most of the data in the DB is still "consistent"

Considerable amount of research on consistent query answering (CQA) has been carried out in the last 6 years

In [Arenas, Bertossi, Chomicki. PODS 99]:

- Characterization of consistent answers to queries as those that are invariant under minimal repairs of the original database; i.e. true in all minimally repaired versions of the DB
- Mechanism for computing them (for certain classes of queries and ICs)

Here, repairs of databases are consistent instances that minimize under set inclusion the set of insertions/deletions of whole database tuples

D (set of ground atoms); inconsistent wrt IC: A repair D' of D satisfies IC and makes $\Delta(D, D')$ minimal under set inclusion

Example 1: Inconsistent DB instance D wrt FD: $Name \rightarrow Salary$

$Employee$	$Name$	$Salary$
	$Page$	5000
	$Page$	8000
	$Smith$	3000
	$Stowe$	7000

Repairs D_1, resp. D_2

$Employee$	$Name$	$Salary$
	$Page$	5000
	$Smith$	3000
	$Stowe$	7000

$Employee$	$Name$	$Salary$
	$Page$	8000
	$Smith$	3000
	$Stowe$	7000

Consistent answers to the queries:

- $Employee(x,y)?$: $(Smith, 3000), (Stowe, 7000)$
- $\exists y Employee(x,y)?$: $Page, Smith, Stowe$

First algorithm for CQA was based on *FO query rewriting*

Example 2: (continued)

FD: $\forall XYZ\ (\neg \mathit{Employee}(X,Y)\ \vee\ \neg \mathit{Employee}(X,Z)\ \vee Y = Z)$

Query: $\mathit{Employee}(x,y)$?

Consistent answers can be obtained by means of the transformed query

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

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

FO query rewriting is defined for (actually applies to) limited classed of queries and ICs

A more general mechanism is based on specifying database repairs using disjunctive logic programs with stable model semantics

It is a general methodology, that can be applied to any FO queries (and beyond)

CQA using these programs gets a Π_2^P complexity upper bound in data

[Greco, Greco, Zumpano. IEEE TKDE 2003]
[Arenas, Bertossi, Chomicki. TPLP 2003]
[Barcelo, Bertossi. PADL 2003]

Example 3: (continued) Repair program $\Pi(D, FD)$:

1. Original data as facts: $Employee(Page, 5000, \mathbf{t_d})$, etc.

2. $Employee(x, y, \mathbf{f_a}) \ \vee \ Employee(x, z, \mathbf{f_a}) \ \leftarrow$
$$Employee(x, y, \mathbf{t_d}), Employee(x, z, \mathbf{t_d}), y \neq z$$

3. Annotation constant $\mathbf{t}^{\star\star}$ is used to read off the literals that are inside a repair
$$Employee(\bar{x}, \mathbf{t}^{\star\star}) \leftarrow Employee(\bar{x}, \mathbf{t_d}), \ not \ Employee(\bar{x}, \mathbf{f_a})$$

Query $\exists y Employee(x, y)$?

Add to the repair program the rule

$$Ans(x) \leftarrow Employee(x, y, \mathbf{t}^{\star\star})$$

Answer under the skeptical stable model semantics

Today we have a quite clear picture of tractable/intractable cases for CQA

- Complete analysis of complexity of CQA for simple aggregate queries under FDs

  Most cases are NP-complete

  [Arenas, Bertossi, Chomicki. ICDT 01]

- Complexity results for conjunctive queries under denial ICs and referential ICs

  Π_2^P-complete cases identified (and below)

  Even undecidability (cyclic referential ICs)

  [Chomicki, Marcinkowski. I&C 05], [Cali, Lembo, Rosati. PODS 2003]

- Conjunctive queries and FDs: broad and tight tractable classes identified

  [Chomicki, Marcinkowski. I&C 05], [Fuxman, Miller. ICDT 2005]

Other Forms of Repairs

Except for a few exceptions, most of the research has concentrated on this repair semantics

Repairs minimize under set inclusion the set of insertions or deletions of whole database tuples

In different forms, and exploiting different aspects, two alternative repair semantics have been considered in the literature

- Cardinality-based repair semantics:

Repairs minimize the cardinality of the set of whole tuples by which it differs from the original DB

[Arenas, Bertossi, Chomicki. TPLP 2003]

Example 4: DB schema $P(X, Y, Z)$, $FD\colon X \rightarrow Y$

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

D has two repairs wrt set inclusion

They have a set-minimal difference with D

- $D_1 = \{P(a, b, c)\}$: $\Delta(D, D_1) = \{P(a, c, d), P(a, c, e)\}$
- $D_2 = \{P(a, c, d), P(a, c, e)\}$: $\Delta(D, D_2) = \{P(a, b, c)\}$

Only D_2 is a cardinality-based repair: $|\Delta(D, D_2)|$ is minimum

- Attribute-based repair semantics:

Repair minimize a numerical aggregation function over differences between attribute values in the original tuples and their repaired versions

[Franconi, Laureti Palma, Leone, Perri, Scarcello. LPAR 01]
[Wijsen. ICDT 03], [Flesca, Furfaro, Parisi. DBPL 05]

[Bertossi, Bravo, Franconi, Lopatenko. DBPL 05]:

- Schemas with key constraints that are always satisfied
- Some attributes take possibly erroneous numerical values
- ICs are expressed by denial constraints that prohibit certain combinations of data values
- Changes in values of numerical and fixable attributes are allowed to restore consistency
- Quadratic distance is kept to a minimum

Example 5:

$IC_1: \forall ID, P, A, M \neg(Buy(ID, I, P), Client(ID, A, M), A < 18, P > 25)$

$IC_2: \forall ID, A, M \neg(Client(ID, A, M), A < 18, M > 50)$

D:

Client	**ID**	**A**	**M**
	1	15	52
	2	16	51
	3	60	900
Buy	**ID**	**I**	**P**
	1	CD	27
	1	DVD	26
	3	DVD	40

A (minimal) fix D' with $cost = 1^2 + 2^2 + 1^2 + 2^2 = 10$

Client'	**ID**	**A**	**M**
	1	15	~~52~~ 50
	2	16	~~51~~ 50
	3	60	900
Buy'	**ID**	**I**	**P**
	1	CD	~~27~~ 25
	1	DVD	~~26~~ 25
	3	DVD	40

A fix D'' with $cost = 1^2 + 3^2 = 10$

Client''	**ID**	**A**	**M**
	1	~~15~~ 18	52
	2	16	~~51~~ 50
	3	60	900
Buy''	**ID**	**I**	**P**
	1	CD	27
	1	DVD	26
	3	DVD	40

We concentrated on the complexity of deciding the existence of a fix close enough to the original instance: NP-complete

Relevant for the kind of applications that motivated this research: editing of census data

Provably no $PTAS$ exists ...

Provide $PTIME$ approximation within a constant factor

CQA for ground atomic queries: P^{NP}-hard

Terminology: (repair semantics)

- S-repairs: Repairs based on minimal set difference
- C-repairs: Repairs based on minimum cardinality set difference
- A-repairs: Repairs based on minimization of aggregation over attribute changes

$Rep(D, IC, \mathcal{S})$: The repairs of a database instance D (a finite set of ground atoms) wrt integrity constraints IC, and a repair semantics $\mathcal{S}$

$CQA(Q, IC, \mathcal{S}) := \{\ (D, \bar{t})\ \mid\ D' \models Q(\bar{t})$ for all $D' \in Rep(D, IC, \mathcal{S})\ \}$

(the decision problem of CQA)

We are interested in data complexity

Recent Research: Motivation

• Provide a better understanding of complexity of CQA under the C-repair semantics

Formulation in graph-theoretic terms allows us to find interesting relationships between CQA (a *certain semantics*) and the *possible semantics* (an answer is consistently true when true in some repair)

A formulation that is closer to the essence of our problem as found in DBs ...

There has been relevant research in belief revision/update wrt complexity of different semantics; but it is not clear how and if they can be applied here
[Eiter, Gottlob. AIJ 92]

• Provide the first steps towards a study of CQA in a dynamic setting, when the DB undergoes some updates

The complexity analysis considers all the three (families of) semantics above

An assumption is that the DB is (was) consistent wrt the given ICs before the updates

Dynamic aspects of CQA have been largely ignored; but more research on incremental properties and algorithms is necessary to make ideas and techniques applicable

In general, what is the complexity of CQA from the instance $U(D)$ obtained by applying a finite sequence U of update operations to the consistent instance D?

Example 6: (example 4 continued)

Instance $D_1 = \{P(a,c,d), P(a,c,e)\}$ is consistent

After the update operation $insert(P(a,f,d))$ it becomes inconsistent

The only C-repair of $D_1 \cup \{P(a,f,d)\}$ is D_1 itself

Thus, CQA from $D_1 \cup \{P(a,f,d)\}$ amounts to classic query answering from D_1

If we start from the consistent instance $D_2 = \{P(a,c,d)\}$, the same update action leads to two C-repairs: D_2, but also $\{P(a,f,d)\}$

Now CQA from D_2 is different from classic query answering from D_2 (two repairs to consider)

Graph-Theoretic Representation of C-Repairs

Given: IC, a set of denial ICs; and D, a DB instance, we consider the conflict hyper-graph:

- Vertices are the DB tuples
- Hyper-edges are formed by DB tuples that simultaneously violate an IC

[Arenas, Bertossi, Chomicki. ICDT 01], [Chomicki, Marcinkowski. I&C 05]

Repairs obtained by tuple deletions only

There is a one-to-one correspondence between C-repairs of D wrt IC and the maximum independent sets, i.e. independent sets of maximum cardinality (MIS)

A ground atomic query is consistently true if it is a vertex in every MIS

Every tuple in D belongs to an S-repair, but not necessarily to a C-repair; so testing membership of vertices to some (or all, of course) MIS becomes relevant

We can prove these useful polynomial time self-reducibility properties of MIS independent sets:

- Given a graph G and a vertex v in it, there is a graph G', such that v belongs to some MIS of G iff v belongs to all MIS of G' iff the sizes of MIS in G and G' differ by one
- Given a graph G and a vertex v there is a graph G', such that v belongs to all MISs of G iff v belongs to some MIS of G'

Computing the size of a maximum clique in a graph belongs to $FP^{NP(log(n))}$ [Krentel. JCSS 88]; then we obtain:

Proposition: The problems of deciding for a vertex in a graph if it belongs to some MIS and if it belongs to all MISs are both in $P^{NP(log(n))}$

As a consequence: For denial constraints and ground atomic queries, CQA under C-repair semantics belongs to $P^{NP(log(n))}$

The results above show that the problems of CQA under the *certain* and *possible* C-repair semantics are polynomially reducible to each other

What about completeness?

Lemma: There is a fixed database schema $\mathcal{D}$ and a denial constraint, such that for every graph G, there is an instance D over $\mathcal{D}$, whose C-repairs are in one-to-one correspondence with the MISs of G

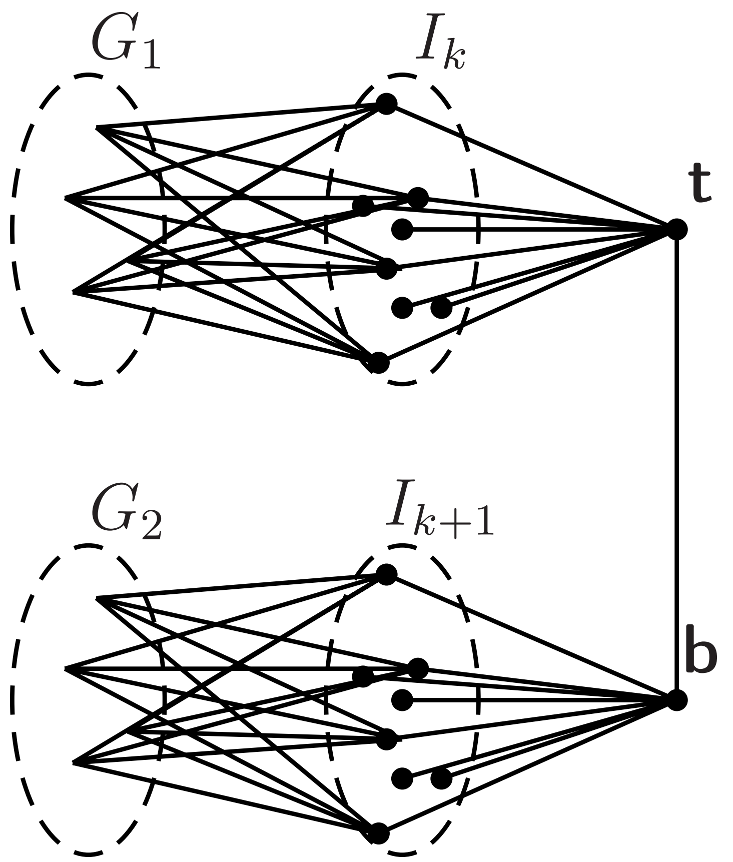
D can be built in polynomial time in the size of G

From the $P^{NP(log(n))}$-completeness of determining the size of a maximum clique [Krentel. Op. cit.], we obtain:

Determining the size of a C-repair for denial constraints is $P^{NP(log(n))}$-complete

Coming back to CQA ...

For hardness results we use a graph-theoretic polynomial-time construction, the block $B_k(G, \mathbf{t})$



- G_1, G_2 are copies of G
- k a natural number
- $\mathbf{t}$ is a distinguished vertex
- Two internally disconnected subgraphs I_k, I_{k+1}, with k and $k+1$ nodes, resp.
- Every vertex in G_1 (G_2) connected to every vertex in I_k (I_{k+1})

It holds: $\mathbf{t}$ belongs to all MISs of $B_k(G, \mathbf{t})$ iff the cardinality of a MIS of G is equal to k

Proposition: Deciding if a vertex belongs to all MISs of a graph is $P^{NP(log(n))}$-hard

Can be proved by reduction from this $P^{NP(log(n))}$-complete [Krentel. Op. cit.]: Given a graph G and an integer k, is the size of a maximum clique in G equivalent to $0 \ mod \ k$?

G is reduced to a graph G' that is built by combining a number of versions of the block construction

G' can be represented as a database consistency problem (previous lemma):

Theorem: For denial constraints, CQA for ground atomic queries under the C-repair semantics is $P^{NP(log(n))}$-complete

Notice: For S-repair semantics this problem can be solved in polynomial time [Chomicki, Marcinkowski. I&C 05]

Incremental CQA and Parameterized Complexity

Given:

- D, IC, with $D \models IC$
- $U: \; U_1, \ldots, U_m$, with $m < c \cdot |D|$, and each U_i is an insertion, deletion or attribute change

What about incremental CQA, i.e. CQA from $U(D)$ wrt IC?

In contrast to what we had for the "static" case:

Proposition: For the C-repair semantics, first-order boolean queries, and denial constraints: incremental CQA is in *PTIME* in the size of D

How does the algorithm do in terms of m?

The proof of this proposition provides an upper bound of $O(n \times n^m)$, which is exponential in m

Can we do better? Say in time $O(f(m) \times n^c)$?

A natural question in the context of *fixed parameter tractability*, i.e. we are considering:

$CQA^p(Q, IC, \mathcal{S}, \bar{t}) := \{(D, U) \mid U$ is an update sequence, and Q is consistently true in $U(D)$ under repair semantics $\mathcal{S}\}$

The parameter is the update sequence U (or its size m)

Proposition: Incremental CQA for ground atomic queries and functional dependencies under the C-repair semantics is in FPT, actually in time $O(log(m) \times (1{,}2852^m + m \cdot n))$

Proof uses the FPT of Vertex Cover

Again conflict graph G (totally disconnected by consistency)

Update: m tuples are inserted, conflict graph G'

$VC(G', k)$ decides if there is a vertex cover of size not bigger than k

Use binary search starting from m with $VC(G', \_)$ to determine the size of a minimum vertex cover

This is the minimum number of tuples that have to be removed to restore consistency

A ground atomic query (a vertex v of G') is consistently true if it does not belong to any minimum vertex cover

Answer is *yes* iff the sizes of minimum vertex covers for G' and $G' \smallsetminus \{v\}$ are the same

For denial ICs we have hyper-graphs ...

The FPT of incremental CQA still holds for denial ICs if the number d of atoms in them is fixed

We can use the FPT of the d-Hitting Set: Finding the size of a minimum HS for a hyper-graph with hyper-edges bounded in size by d

Incremental CQA: S-Repair Semantics

For conjunctive boolean queries and denial ICs, static CQA is:

- $P^{NP(log(n))}$-complete under the C-repair semantics (this work)
- $coNP$-complete under the S-repair semantics [Chomicki, Marcinkowski. I&C 05]

For boolean queries and denial ICs, incremental CQA is:

- In $PTIME$ under the C-repair semantics (this work)

However, we can prove:

For boolean conjunctive queries and denial ICs, incremental CQA is in $coNP$-complete under the S-repair semantics

(can be obtained from the static case for the same semantics)

Why the difference?

The cost of a C-repair cannot exceed the size of an update, whereas the cost of an S-repair may be unbounded wrt the size of an update

Example 7: Schema $R(\cdot), S(\cdot)$; denial $\forall x \forall y \neg(R(x) \wedge S(y))$

Consistent $D = \{R(1), \ldots, R(n)\}$ (empty table for S)

After $U = insert(S(0))$, the database becomes inconsistent, and the S-repairs are $\{R(1), \ldots, R(n)\}$ and $\{S(0)\}$

Only the former is a C-repair, at a distance 1 from D

The second S-repair is at a distance n

Incremental CQA: A-Semantics

For comparison with the static case, we consider first a quite general, weighted version of the A-repair semantics

Numerical weight function w on tuples of the form $(R(\bar{t}), A, \mathit{newValue})$

- $R(\bar{t}) \in D$
- A is an attribute of R, and
- $\mathit{newValue}$ is a new value for A in $R(\bar{t})$

The wA-repairs are consistent instances that minimize an aggregate function g on the values $w(R(\bar{t}), A, \mathit{newValue})$

Typically: $w(R(\bar{t}), A, \mathit{newValue}) = \delta(R(\bar{t}).A, \mathit{newValue})$ and $g := \mathit{sum}$, i.e. only number of changes is counted

In Example 5, we can take:

- $w((R(\bar{t}), A, newValue) := (R(\bar{t}).A - newValue)^2$
- $g = sum$

Proposition: Static CQA for ground atomic queries and denial constraints under the wA-repair semantics is P^{NP}-hard

Obtained by reduction from the P^{NP}-complete problem:

Given a Boolean formula $\psi(X_1, \cdots, X_n)$ in 3CNF, decide if the last variable X_n is equal to 1 in the lexicographically maximum satisfying assignment (answer is No if ψ is not satisfiable) [Krentel. Op.cit.]

For the incremental part:

We use: 3-Colorability for regular graphs of degree 4 is NP-complete

Next we encode as a consistent database problem a 4-regular graph G with a fixed set of first-order constraints:

- Schema $E(X, Y)$, $Coloring(X, V)$, $Colors(X)$
- Colors must be legal (belong to $Colors$)
- Usual colorability condition
- etc.

G is 4-colorable and uses all four values available in a table of colors as specified through a constraint

Update part: Delete one color, say C, from $Colors$

To restore consistency, we have to color the graph with 3 colors; so colors have to be reassigned

There are wA-repairs iff the graph is colorable with 3 colors

Then, the query $Colors(C)$ is (trivially) consistently true iff there are no wA-repairs iff the graph is not colorable with 3 colors

We obtain:

Theorem: For wA-repairs, ground atomic queries, first-order ICs, and update sequences consisting of tuple deletions, incremental CQA is *coNP*-hard

What about denial ICs?

In this case, deletions are trivial; they do not introduce any violations

Theorem: Incremental CQA wrt denial constraints and atomic queries under the wA-repair semantics is P^{NP}-hard

By reduction from static CQA for A-repairs as in
[Bertossi, Bravo, Franconi, Lopatenko. DBPL 05]

Here one tuple insertion is good enough for the update part

Attribute values changes are used to restore consistency

Ongoing and Future Work

- Analyze the complexity of CQA from the point of view of
  - *Dynamic complexity* [Immerman; 99]
  - *Incremental complexity*
    [Miltersen, Subramanian, Vitter, Tamassia. TCS 94]

  Here the DB is not necessarily consistent before the update

  Auxiliary data structures can be used for incremental computation

- Parameterized complexity

  In many forms in CQA, both static and incremental

  Several parameters naturally offer themselves:

- number of inconsistencies in the database
- degree of inconsistency, i.e. the maximum number of violations per database tuple
- complexity of inconsistency, i.e. the length of the longest path in the conflict graph or hypergraph
- ...

These parameters are practically relevant in many applications, where inconsistencies are “local”

[Bertossi, Bravo, Franconi, Lopatenko. DBPL 05]