Carleton
UNIVERSITY

Static and Incremental Complexity of Consistent Query Answering under Attribute- and Cardinality-based Repairs

Leopoldo Bertossi
Carleton University
Ottawa, Canada

Join work with: Andrei Lopatenko, Loreto Bravo, Enrico Franconi

Introduction

For several reasons, databases may become inconsistent wrt a given set of integrity constraints (ICs)

In the last 6 years considerable amount of research carried out on consistent query answering (CQA)

In (Arenas, Bertossi, Chomicki; PODS 99):

- Characterization of consistent answers to queries as those that are invariant under minimal repairs of the original database

- Mechanism for computing them (for certain classes of queries and ICs)

Most of the research has concentrated on repairs of databases that minimize under set inclusion the set of insertions/deletions of whole database tuples (no matter what type of data they contain)

No much interest in the repairs *per se* or their computation (unless absolutely necessary), but in obtaining and checking consistent answers to queries

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 queries:

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

I. Fixing Numerical Attributes

In some scenarios or applications these assumptions or semantic commitments may not be the most natural ones

We were motivated in by census-like data, where other ontological assumptions, repairs, and computational needs appear (Franconi,Laureti Palma, Leone, Perri, Scarcello; LPAR 01)

- Some attributes take numerical values
- Most natural form of fix is to change some of the values of attributes in tuples (Wijsen; ICDT 03)

- The tuples are identified by a key (e.g. household identifier) that is fixed and not subject to changes

  Identifiers are kept in repaired versions of the database (census form)

- Only some attributes are subject to fixes
- Constraints are expressed by prohibitions on positive data, that can be captured by denial constraints
- Few relations in the database, actually not uncommon to have one single relation
- Computation of fixes (repairs) becomes relevant and a common task in census like applications
- Aggregate queries are most important, rather than queries about individual properties

The Problem

We propose a notion of fix (repair) of a database containing numerical, integer values that:

- Allows, as basic fixing actions, changes of values in (changeable or flexible) attributes
- For the first time takes into account the occurrence of numerical values and their nature when distances between databases are measured in numerical terms
- Considers the presence of non-contradictable key constraints

In this scenario, new issues and problems appear:

- Computing database fixes
- Determining existence of fixes, e.g. not beyond a certain distance from the original database
- Consistent query answering to aggregate queries wrt to denial constraints

Example 2: A denial constraint DC: *Pay at most* 6000 *to employees with less than 5 years of experience*

Database D, with Name as the key, is inconsistent wrt DC

Employee	Name	Experience	Salary
	Sarah	6	12000
	Robert	4	7000
	Daniel	5	8000

Under tuple and set oriented semantics for repairs, the only minimal repair corresponds to deleting the tuple
Employee(Robert, 4, 7000)

Other options may make more sense that deleting employee Robert (a value for the key):

- Change the violating tuple to Employee(Robert,5, 7000)
- Change it to Employee(Robert, 4, 6000)

They satisfy the implicit requirements that:

- Key values are kept
- Numerical values associated to them do not change too much

Minimum Numerical Fixes

We concentrate on linear denial constraints (LDCs) of the form

$$\forall \bar{x} \neg (A_1 \wedge \ldots \wedge A_m)$$

- A_i are database atoms, or built-in atoms of the form $X \theta c$, with $\theta \in \{=, \neq, <, >, \leq, \geq\}$, or $X = Y$

  (the latter can be replaced by different occurrences of the same variable)

- If we allow atoms of the form $X \neq Y$, we call them extended linear denial constraints (ELDs)

Example: *Pay at most* 6000 *to employees with less than 5 years of experience*

$$\forall Name, Experience, Salary \neg (Employee(Name, Experience, Salary), \\ Experience < 5, Salary > 6000)$$

Distance between Instances:

- When numerical values are updated to restore consistency, it is desirable to make the smallest overall variation of the original values
- A database and a repair (fix) share the same key values
- We can use key values to compute associated numerical variations

To compare instances, we need a common relational schema $\mathcal{R}$, a set of key constraints $\mathcal{K}$, assumed to be satisfied by all the instances, the set of attributes $\mathcal{A}$, a subset of changeable or flexible attributes $\mathcal{F}$ (not containing any attributes in a key)

For a tuple $\bar{k}$ of key values in relation R in instance D, $\bar{t}(\bar{k}, R, D)$ denotes the unique tuple $\bar{t}$ in relation R in instance D whose key value is $\bar{k}$

Example 3: D has tables $Client(\underline{ID}, A, M)$, A is age, M is money; and $Buy(\underline{ID, I}, P)$, I is items, P is price

Denials: *People younger than 18 cannot spend more than 25 on one item nor spend more than 50 in the store*

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

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

D:

Client	**ID**	**A**	**M**	
	1	15	52	t_1
	2	16	51	t_2
	3	60	900	t_3
Buy	**ID**	**I**	**P**	
	1	CD	27	t_4
	1	DVD	26	t_5
	3	DVD	40	t_6

IC_1 is violated by $\{t_1, t_4\}$ and $\{t_1, t_5\}$; IC_2 by $\{t_1\}$ and $\{t_2\}$

$\bar{t}((1, CD), Buy, D) = Buy(1, CD, 27)$, etc. $\mathcal{F} = \{A, M, P\}$

For instances D, D' over the same schema and same key values for each relation $R \in \mathcal{R}$, their square distance is

$$\Delta(D, D') = \sum \alpha_A [\pi_A(\bar{t}(\bar{k}, R, D)) - \pi_A(\bar{t}(\bar{k}, R, D'))]^2$$

- Sum is over all relations R, fixable numerical attributes A and tuples of key values $\bar{k}$
- π_A is the projection on attribute A
- α_A is the weight of attribute A

Other numerical distance functions can be considered, e.g. "city distance", obtaining results similar to those that follow

Given an instance D, with $D \models \mathcal{K}$, and a set of possibly violated ELDCs IC, a least squares fix (LS-fix) for D wrt IC is an instance D' with:

1. Same schema and domain as D
2. Same values as D in the non fixable numerical attributes $\mathcal{A} \setminus \mathcal{F}$ (in particular in key attributes)
3. $D' \models \mathcal{K} \cup IC$;
4. $\Delta(D, D')$ is minimum over all the instances that satisfy 1. - 3.

$Fix(D, IC) := \{D' \mid D' \text{ is an LS-fix of } D \text{ wrt } IC\}$

Example 3: (cont.)

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

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

D:

Client	**ID**	**A**	**M**	
	1	15	52	t_1
	2	16	51	t_2
	3	60	900	t_3
Buy	**ID**	**I**	**P**	
	1	CD	27	t_4
	1	DVD	26	t_5
	3	DVD	40	t_6

Assume $\alpha_A = \alpha_M = \alpha_P = 1$

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

Client'	**ID**	**A**	**M**	
	1	15	~~52~~ 50	t_1'
	2	16	~~51~~ 50	t_2'
	3	60	900	t_3
Buy'	**ID**	**I**	**P**	
	1	CD	~~27~~ 25	t_4'
	1	DVD	~~26~~ 25	t_5'
	3	DVD	40	t_6

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

Client"	**ID**	**A**	**M**	
	1	~~15~~ 18	52	t_1''
	2	16	~~51~~ 50	t_2''
	3	60	900	t_3
Buy"	**ID**	**I**	**P**	
	1	CD	27	t_4
	1	DVD	26	t_5
	3	DVD	40	t_6

In this example, it was possible to obtain LS-fixes by performing direct, local changes in the original conflictive tuples alone

No new, intermediate inconsistencies introduced in the repair process

This may not be always the case ...

Complexity of LS-fixes

It is relevant to ask about the existence of fixes

In contrast to the "classical" case of CQA, there may be no fixes

We concentrate on data complexity and denial constraints

The number of fixes can be exponential, actually

For (E)LDCs:

$NE(IC) := \{D \mid Fix(D, IC) \neq \emptyset\}$ is NP-complete

The Database Fix Problem

$DFP(IC) := \{(D,k) | \exists\ D' \in Fix(D,IC)$ with $\Delta(D,D') \leq k\}$

It is important for transformation of inconsistent database into the closest consistent state

- What is the distance to the closest consistent state?
- Make computations more efficient by cutting off incorrect (too expensive) branches during computation or materialization of a consistent state

Complexity of DFP provides lower bounds for CQA under some assumptions

Theorem: $DFP(IC)$, the problem of deciding whether there exists an LS-fix wrt IC at a distance $\leq k$, is NP-complete

Hardness: By encoding "Vertex Cover"

One LDC with three database atoms plus two built-ins is good enough

Membership:

- Possible values in repaired tuples are determined by the constants in the DCs
- $PTIME$ computation of the distance function

Approximate Solutions to DFP

Given that $DFP(IC)$ is NP-complete, can we get a good and efficient approximate solution?

$DFOP$ denotes the optimization problem of finding the minimum distance to a fix

Theorem: For a fixed set of LDCs, $DFOP$ is $MAXSNP$-hard

Proof by reduction from "Minimum Vertex Cover Problem for Graphs of Bounded Degree" which is $MAXSNP$-complete

$MAXSNP$-hardness implies (unless $P = NP$) that there exist constant δ such that $DFOP$ for LDCs cannot be approximated within an approximation factor less than δ

Can we provide an approximation within a constant factor, possibly with restriction to a still useful but hard class of LDCs?

Definition: A set of LDCs IC is local if:

- Equalities between attributes and joins involve only non fixable attributes
- There is a built-in with a fixable attribute in each IC
- No attribute A appears in IC both in comparisons of the form $A < c_1$ and $A > c_2$

Example 3: (cont.) The LDCs there are local

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

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

Why "local" and why interesting?

- Basically, inconsistencies can be fixed tuple by tuple, without introducing new violations
- LS-fixes always exist
- Local LDCs are the most common in census-like applications
  (Franconi,Laureti Palma, Leone, Perri, Scarcello; LPAR 01)
- DFP still NP-complete for local LDCs
- $DFOP$ still $MAXSNP$-hard for local LDCs
  (non-local LDCs can be eliminated from the proof for the general case)

We will transform $DFOP$ into a "Weighted Set Cover Problem"

A Weighted Set Cover Problem

The WSCP is defined as follows

Instance: $(U, \mathcal{S}, w)$

- $\mathcal{S}$ is a collection of subsets of set U
- $\bigcup \mathcal{S} = U$ (a cover)
- w assigns numerical weights to elements of $\mathcal{S}$

Question: Find a sub-collection of $\mathcal{S}$ with minimum weight that covers U

$WSCP$ is $MAXSNP$-hard

We create a "good" instance of $WSCP$ ...

1. A set I of database atoms (tuples) from D is a violation set for $ic \in IC$ if $I \not\models ic$, and for every $I' \subsetneq I$, $I' \models ic$, i.e. a minimal set of tuples that participate in the violation of an IC

   $$U := \text{set of violation sets for } D, IC$$

2. $\mathcal{I}(D, ic, t)$ denotes the set of violation sets for ic in D that contain tuple t

   $$S(t, t') := \{I \mid \text{ exists } ic \in IC,\ I \in \mathcal{I}(D, ic, t) \text{ and } ((I \setminus \{t\}) \cup \{t'\}) \models ic\},$$

   i.e. the violations sets containing tuple t that are solved by changing t to t'

$\mathcal{S}$:= collection of $S(t, t')$'s such that t' is a *local fix* of t

I.e.

(a) t, t' share same values in non-fixable attributes

(b) $S(t, t') \neq \emptyset$ (some inconsistency is solved)

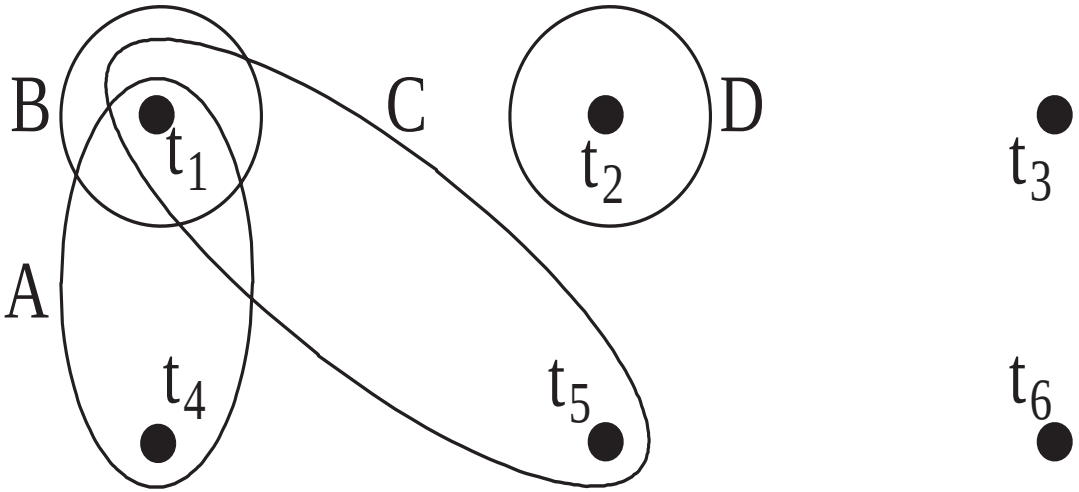
(c) $\Delta(\{t\}, \{t''\})$ is minimum (relative to (a), (b))

3. Finally, $w(S(t, t')) := \Delta(\{t\}, \{t''\})$

Properties of the Reduction:

- Local fixes can be obtained in polynomial time
- One-to-one correspondence between solutions to $DFOP$ and solutions to $WSCP$ that keeps the optimum values
- Each set cover corresponds to a consistent instance
- Transformation is in polynomial time
- Correspondence may be lost if applied to non-local LDCs (LDCs may not be satisfied by resulting instances)

Example 3: (cont.)



Client	ID	A	M	
	1	15	52	t_1
	2	16	51	t_2
	3	60	900	t_3
Buy	**ID**	**I**	**P**	
	1	CD	27	t_4
	1	DVD	26	t_5
	3	DVD	40	t_6

Violation sets: $\{t_1, t_4\}$, $\{t_1, t_5\}$ for IC_1; $\{t_1\}$, $\{t_2\}$ for IC_2

Tuple t_1: One local fix wrt IC_1, one wrt IC_2

Tuple t_2: One local fix Tuples t_4, t_5: One local fix each

Conflict (Hyper)Graphs: (Arenas, Bertossi, Chomicki; ICDT 01),
(Chomicki, Marcinkowski; Information and Computation 05)

Tuples, their local fixes, and weights ...

Set cover els.	$S(t_1, t_1')$	$S(t_1, t_1'')$	$S(t_2, t_2')$	$S(t_4, t_4')$	$S(t_5, t_5')$
Local Fix	t_1'	t_1''	t_2'	t_4'	t_5'
Weight	4	9	1	4	1
Violation set A	0	1	0	1	0
Violation set B	1	1	0	0	0
Violation set C	0	1	0	0	1
Violation set D	0	0	1	0	0

1 and 0 at the bottom indicate if the violation set belongs or not to $S(t, t')$

Approximating $DFOP$

We approximate a solution to $DFOP$ by approximating $WSCP$

In general, $WSCP$ can be approximated within a factor $O(log(n))$ by greedy algorithms (Chvatal)

In our case, the frequency (number of elements of $\mathcal{S}$ covering an element of U) is bounded by the maximum number of atoms in the ICs (small in general)

In this case $WSCP$ can be efficiently approximated within a constant factor given by the maximum frequency
(Hochbaum; 1997)

The approximation algorithm always returns a cover, from which we can compute an instance, that turns out to satisfy the LDCs, but may not be optimal (in example 2 we get D')

Consistent Query Answering

We have several results on data complexity for CQA, that is a decision or optimization problem depending on the semantics

Some involving 1-atom denial constraints (1AD), a common case in census-like applications (one DB atom plus built-ins)

Usual semantics for CQA:

- Certain: Answer true in every LS-fix (default)
- Possible: True in some LS-fix
- Range: Shortest numerical interval where all answers from LS-fixes can be found (for numerical, mainly aggregate queries)
  (Arenas, Bertossi, Chomicki; ICDT 2001)

  Two optimization problems: find extremes of the interval

Some results for CQA:

- Non aggregate queries, certain semantics:
  - 1ADs; atomic ground query (and a wide class of boolean conjunctive queries): $PTIME$
  - Arbitrary extended LDCs; atomic query: P^{NP}-hard and in Π_2^P
- Aggregate queries (with one of *sum*, *count distinct*, *average*) and possible semantics (boolean query true in some fix: comparison of the aggregation to a constant) or range semantics
  - 1ADs; acyclic conjunctive query: $coNP$-hard

  E.g. 1AD: $\forall u, c_1, c_2 \neg(Var(u, c_1, c_2) \wedge c_1 < 1 \wedge c_2 < 1)$
  Query: $q(count(distinct\ c)) \leftarrow Var(u, c_1, c_2), Clause(u, c, s), c_1 = s$

Approximating CQA for Aggregate Queries

Even for simple constraints, and simple aggregate queries defined on top of also simple conjunctive queries, CQA becomes hard

Finding the minimum/maximum values for an aggregate query among the LS-fixes become optimization problems

They are $MAXSNP$-hard for *sum*

For 1ADs and conjunctive aggregate query with *sum*, the maximum value for CQA (i.e. range semantics) can be efficiently approximated within a constant factor

II. Cardinality-Based Repairs

In most of the research on CQA, 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

Today we have a quite clear picture of tractable/intractable cases for CQA under this repair semantics

- 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

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 they differ from the original DB [Arenas, Bertossi, Chomicki. TPLP 2003]

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

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

Two repairs wrt set inclusion, i.e. that 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

Cardinality-based repairs form a subset of the set-oriented repairs

They may return consistent answers when the latter does not

- 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]

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 wrt integrity constraints IC, and a repair semantics $\mathcal{S}$

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

The decision problem of CQA

We are interested in data complexity

Motivation for this Research

• 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: (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 hypergraph:

- 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 MISs:

- Given a graph G and a vertex v in it, there is an extension 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 an extension 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]

From this 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

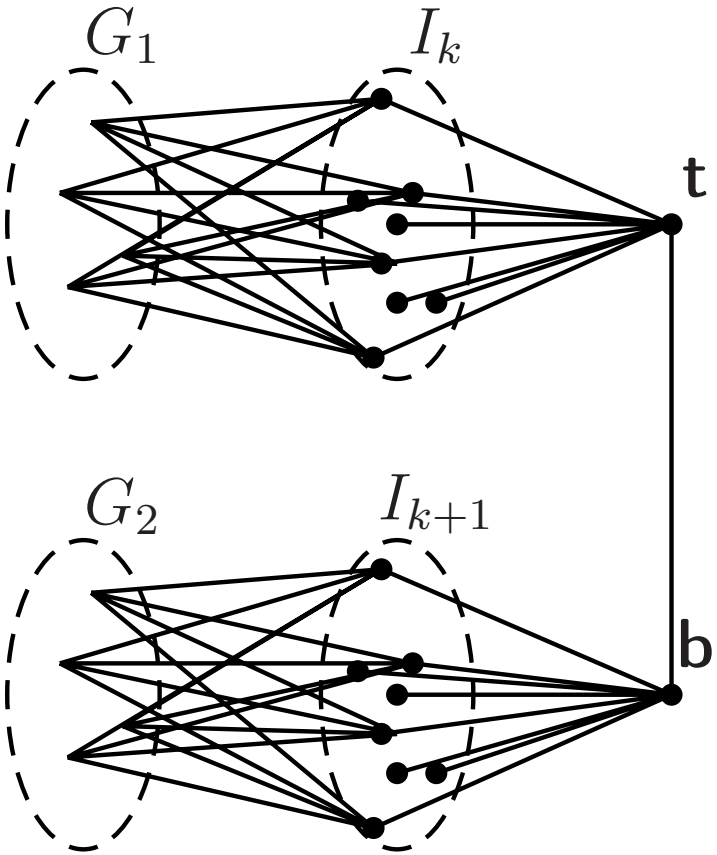
From the $P^{NP(log(n))}$-completeness of determining the size of a maximum clique [Krentel. Op. cit.], we obtain:

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 problem: Given a graph G and an integer k, is the size of a maximum clique in G equivalent to $0 \ mod \ k$? [Krentel. Op. cit.]

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

Represent G' as database repair problem (previous lemma)

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

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

Incremental CQA

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*:

$CQA^p(Q, IC, \mathcal{S}) := \{(D, \bar{t}, U) \mid U$ 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

Use 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 $G' \setminus \{v\}$ are the same

What about denial constraints? For them 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 hypergraph with hyperedges bounded in size by d

Incremental CQA: Other Repair Semantics

Theorem: 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 5: 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

A-repair semantics?: We know already

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

What about the incremental part?

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

Theorem: Incremental CQA wrt denial constraints and atomic queries under the A-repair semantics is P^{NP}-hard
(by reduction from static the static case)

Here one tuple insertion is good enough for the update part

Attribute values changes are used to restore consistency

Incremental CQA under A-repair semantics but more general ICs?

Now update sequences may contain deletions

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 $E(X, Y), Coloring(X, C), Colors(C)$
- 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 A-repairs iff the graph is colorable with 3 colors

Then, the query $Colors(C)$ is (trivially) consistently true iff there are no A-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

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]

Appendix: The Gist for COUNT DISTINCT Aggregation

CQA under range semantics for COUNT DISTINCT acyclic conjunctive queries and one 1AD is $coNP$-complete

By reduction from MAX-SAT with instance $P = \langle U, C, K \rangle$; U a set of variables, C collection of clauses over U, K a positive integer

- Introduce relation $Var(\underline{u}, C_1, C_2)$ with tuple $(u, 0, 0)$ for every variable in $u \in U$; C_1, C_2 fixable, taking values 0 or 1
- Introduce non-fixable relation $Clause(u, c, s)$, with tuple (u, c, s) for every occurrence of $u \in U$ in clause $c \in C$, s is assignment (0 or 1) for u satisfying clause c

- 1AD: $\forall u, c_1, c_2 \neg(Var(u, c_1, c_2) \wedge c_1 < 1 \wedge c_2 < 1)$

Query: $q(count(distinct\ c)) \leftarrow Var(u, c_1, c_2), Clause(u, c, s), c_1 = s$
asks for how many clauses are satisfied in a given fix

Max value in a fix is max number of clauses that can be satisfied for P

Appendix: On $MAXSNP$

- $MAXSNP$: Class of optimization problems that are L-reducible to a problem in $MAXSNP_0$
- $MAXSNP_0$ is the class of optimization problems that are definable by certain logical formulas, i.e. of the form

  $max_{S \subseteq V^r} |\{(x_1, \ldots, x_k) \in V^k \mid \varphi(G_1, \ldots, G_m, S, x_1, \ldots, x_n)\}|$

  Here the input relations are $G_1, \ldots, G_m$ over a finite universe V
- Problems in $MAXSNP$ are all approximable within a constant factor $\epsilon < 1$
- Most likely, the problems complete for the class $MAXSNP$ do not have polynomial time approximation schemas, i.e. not arbitrarily close approximation (all problems in the class would have such an approximation schema)

- $MAXSNP$-complete problems: $MAX3SAT$, the problem of finding a truth assignment that maximizes the number of clauses that are simultaneously satisfied in a 3CNF formula
- Other $MAXSNP$-complete problems
  - 3-Occurrence-$MAX3SAT$ (each variable appears at most three times in the clauses)
  - 4-Degree Independent Set
  - 4-Degree Node Cover
  - 5-Occurrence-$MAX2SAT$
- If an $MAXSNP$-complete problem has a polynomial time approximation schema, then $P = NP$