Carleton
UNIVERSITY

# Consistent Query Answering in Databases Under Cardinality-Based Repair Semantics


Leopoldo Bertossi

Carleton University

School of Computer Science

Ottawa, Canada

**Joint work with: Andrei Lopatenko (Google, Zurich)**

# Introduction

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

- DBMS has no mechanism to maintain the ICs
- Data of different sources are integrated

  Even when the independent data sources are (locally) consistent, integrated data may be inconsistent (wrt global ICs)
- 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"

Which is the data that is still consistent in the DB?

Which are the semantically correct answer to queries posed to the inconsistent database?

How can we obtain them?

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

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

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

  That is, true in all minimally repaired versions of the DB

- Mechanism based on FO query rewriting for computing them (for certain classes of queries and ICs)

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

An instance $D$, seen as a set of ground atoms, maybe inconsistent wrt $IC$:

A repair $D'$ of $D$ is an instance that satisfies $IC$ and makes $\Delta(D, D')$ minimal under set inclusion

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

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:

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

- Repairs are used as an auxiliary concept
- Try to compute consistent answers avoiding repair computation as much as possible

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

Example 2: (continued)

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

Query: $Employee(x, y)$?

Consistent answers can be obtained by means of the transformed query

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

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

Pose the rewritten query to the original, inconsistent database

FO query rewriting works for limited classes 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 query (and beyond)

CQA using these programs gets an upper bound on data complexity of $\Pi_2^P$

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

[Bertossi. Sigmod Record, June 2006]

- Complete analysis of complexity of CQA for scalar 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

  $\Pi_2^P$-complete cases identified (and below)

  Even undecidability (cyclic referential ICs)

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

- Conjunctive queries, FDs, and KDs: Broad and tight tractable classes identified

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

# Other Repair Semantics

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

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:

A repair $D'$ minimizes the cardinality of the set of whole tuples by which it differs from the original DB $D$

$|\Delta(D, D')|$ is minimized [Arenas, Bertossi, Chomicki. 2003]

Example 3: DB schema $P(X, Y, Z)$, $FD$: $X \rightarrow Y$

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

$D$ 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:

Repairs are obtained by fixing attribute values [Wijsen; 2003]

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

Usually the number of changes is minimized

The distance function may be more general

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

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

Example 4: Denial constraints

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

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

$D$:

| **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) repair $D'$ with $cost = 1^2 + 2^2 + 1^2 + 2^2 = 10$

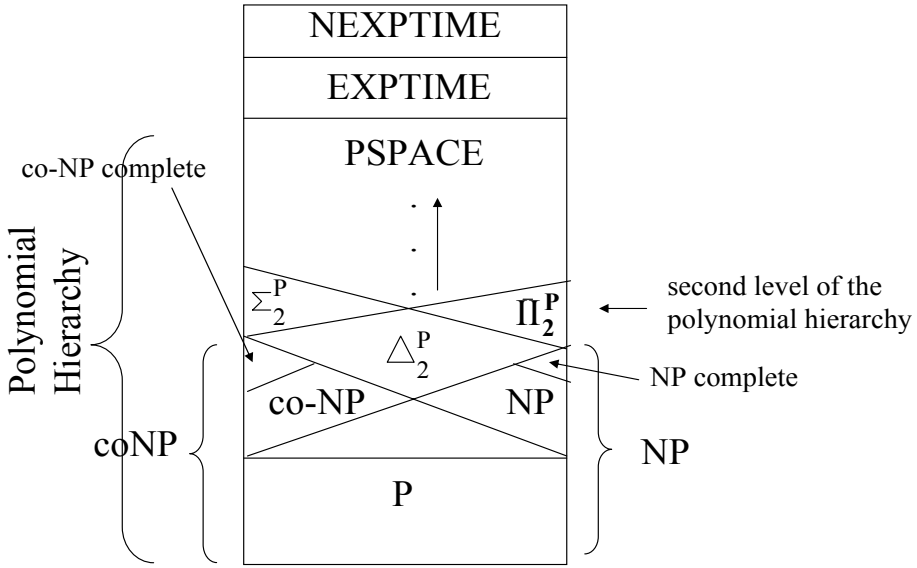
| **Client'** | **ID** | **A** | **M** |
|---|---|---|---|
| | 1 | 15 | ~~52~~ 50 |
| | 2 | 16 | ~~51~~ 50 |
| | 3 | 60 | 900 |
| **Buy'** | **ID** | **I** | **P** |
| | 1 | CD | ~~27~~ 25 |
| | 1 | DVD | ~~26~~ 25 |
| | 3 | DVD | 40 |

A repair $D''$ 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 |

The complexity of deciding the existence of a repair close enough to the original instance is $NP$-complete

CQA for ground atomic queries is $P^{NP}$-hard ($P^{NP}$ aka $\Delta_2^P$)



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

The repair semantics of choice may depend on:

- The application domain and databases at hand

  For example, the C-repair semantics may be better when a few tuples are in conflict with many others

- Properties of the semantics

  For example, C-repairs are a subset of S-repairs, so more consistent answers can be obtained

- Complexity issues ...

$Rep(D, IC, \mathcal{S})$: 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

Interesting per se, but also on the way:

Formulation in graph-theoretic terms allows us to find interesting relationships for this semantics between:

- CQA (a *certain semantics*), and
- The *possible semantics*
  (an answer is consistently true when true in some repair)

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

Dynamic aspects of CQA have been largely ignored

More research on incremental properties of- and algorithms for CQA is necessary to make ideas and techniques applicable

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

Assumption: DB is consistent wrt the given ICs before the updates

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 $D$?

We started this research motivated by the analysis of incremental complexity

C-repairs seemed to have better properties from this point of view (see later ...)

Then we decided to investigate also the "static" complexity of CQA under C-repairs

Which had not been investigated before ...

We concentrate mostly on denial ICs

Example 5: DB schema $P(X, Y, Z)$, $FD: X \rightarrow Y$

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

For the update operation $U: \ insert(P(a, f, d))$, $U(D_1)$ becomes inconsistent

The only C-repair of $U(D_1)$ is $D_1$ itself

Thus, CQA from $U(D_1)$ 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 $U(D_2)$ is different from classic query answering from $D_2$ (two repairs to consider)

# Conflict Graphs for C-Repairs

Given a set of denial constraints $IC$ and a DB instance $D$:

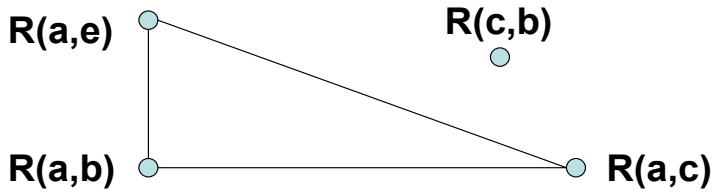
Conflict hypergraph:

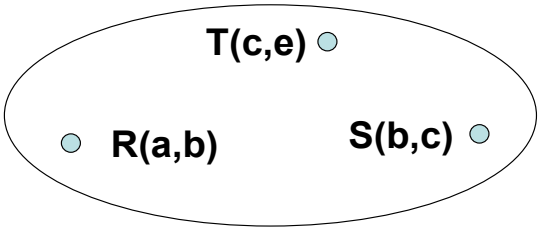
- Vertices are the DB tuples
- Hyperedges are formed by DB tuples that simultaneously violate the same ground instance of one of the ICs

[Arenas, Bertossi, Chomicki. ICDT 2001], [Chomicki, Marcinkowski. I&C 2005]

Repairs are obtained by tuple deletions only

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




$$\forall xyzw\neg(R(x,y) \wedge S(y,z) \wedge T(z,w))$$

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)

An independent set in a hypergraph is a set of vertices that does not contain any hyperedge

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

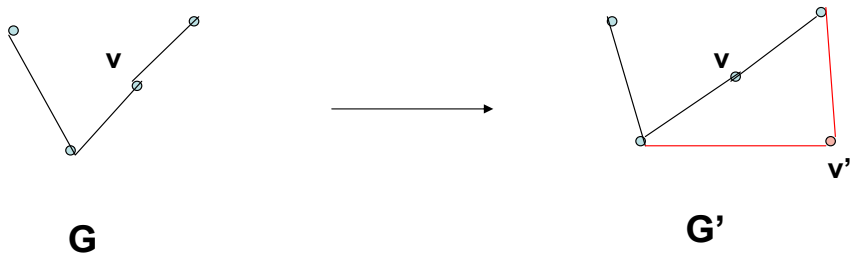
Every tuple in $D$ belongs to an S-repair, but not necessarily to a C-repair

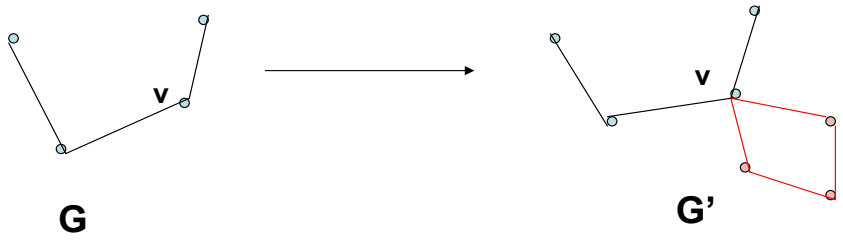
So, testing membership of vertices to some (or all) MIS becomes relevant

Useful polynomial time self-reducibility properties for MIS:

$v \in$ some MIS of $G \iff v \in$ all MIS of $G'$

$\iff |MIS(G)|,\ |MIS(G')|$ differ by one




$v \in$ all MIS of $G \iff v \in$ some MIS of $G'$

We formulate the results in terms of (conflict) graphs, but they carry over to (conflict) hypergraphs

As a consequence:

The problems of CQA under the *certain* and *possible* C-repair semantics are polynomially reducible to each other
(which is not true for the S-repair semantics)

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 $P^{NP(log(n))}$

Obtained using as a subroutine an algorithm for computing the size of a maximum clique in a graph, which belongs to $FP^{NP(log(n))}$ [Krentel. 1988]

Thus:

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

What about hardness?

Before that, a useful representation theorem:

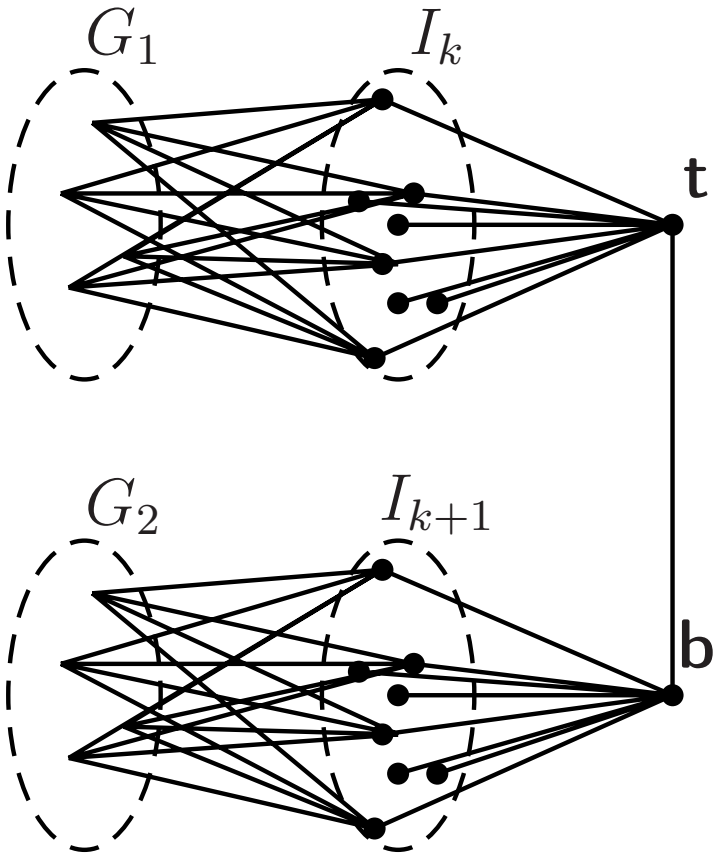
Lemma: There is a fixed database schema $\mathcal{D}$ and a denial constraint, such that for every graph $G$, there is an instance $D$ over $\mathcal{D}$, whose C-repairs are in one-to-one correspondence with the MISs of $G$

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

- Introduce a relation for vertices and a relation for labelled edges
- Padding the edges relation with many copies of the labels makes certain potential repairs too costly

For hardness, we establish first an auxiliary poly-time graph construction: the block $B_k(G, \mathbf{t})$

- $G_1, G_2$ are copies of a given graph $G$
- $k$ 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:

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

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

Can be proved by reduction from the following $P^{NP(log(n))}$-complete [Krentel. Op. cit.]:

Given a graph $G$ and an integer $k$, is the size of a maximum clique in $G$ equivalent to $0 \ mod \ k$?

The complement graph of $G$ is reduced to a graph $G'$ that is built by combining a number of versions of the block construction, so that counting sizes of maximum independent sets can be represented

By the representation lemma, the graph in the theorem above can be represented as (reduced to) a database consistency problem

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

However:

For the S-repair semantics: [Chomicki, Marcinkowski. 2005]

- This problem can be solved in polynomial time
- For conjunctive boolean queries and denial ICs, CQA is $coNP$-complete

# Incremental CQA and Parameterized Complexity

Given:

- $D$, $IC$, with $D \models IC$

- $U: U_1, \ldots, U_m$, with $m < c \cdot |D|$
  Each $U_i$ an insertion, deletion or change of attribute value

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

The problem has several dimensions:

- The repair semantics
- The kind of ICs
- The kind of update actions in $U$

We have results for several combinations of these dimensions

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 size of the update sequence?

A naive algorithm in the proof 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 question about *fixed parameter tractability*:

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

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 $O(log(m) \times (1{,}2852^m + m \cdot n))$

- Algorithm calls a subroutine for Vertex Cover
- Vertex Cover belongs to FPT

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 $k$ (FPT)

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

For FDs we have conflict graphs: two database atoms per IC

For denial ICs we have hypergraphs ...

The FPT of incremental CQA still holds for denial ICs:

- Since they are fixed, the maximum number $d$ of atoms in them is fixed
  This bounds the size of hyperedges

- We can use the FPT of the $d$-Hitting Set: Finding the size of a minimum hitting set for a hypergraph with hyperedges bounded in size by $d$

# Incremental CQA: S-Repair Semantics

Under the C-repair semantics: (see above)

- For conjunctive boolean queries and denial ICs, static CQA is $P^{NP(log(n))}$-complete
- For conjunctive queries and denial ICs, incremental CQA is in $PTIME$

However, under the S-repair semantics:

For boolean conjunctive queries and denial ICs, incremental CQA is $coNP$-complete

This can be obtained by reduction from the static case for the S-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 6: 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 $(R(\bar{t}), A, newValue)$

- $R(\bar{t}) \in D$
- $A$ is an attribute of $R$, and
- $newValue$ 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 $w(R(\bar{t}), A, newValue)$

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

Or, as in the example with numerical attributes:

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

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: [Krentel. Op.cit.]

Given a propositional formula $\psi(X_1, \cdots, X_n)$ in 3CNF:

Is the last variable $X_n$ equal to $1$ in the lexicographically maximum satisfying assignment? (answer is $No$ if $\psi$ is not satisfiable)

For the incremental part:

For denial constraints, deletions as update actions 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 2005]

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; 1999], [Weber, Schwentick; 2005]
  - *Incremental complexity*
    [Miltersen, Subramanian, Vitter, Tamassia; 1994]

  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. 2005]