Consistent Query Answering By Minimal-Size Repairs

Andrei Lopatenko
Free University of Bozen-Bolzano
Faculty of Computer Science
Bozen-Bolzano, Italy.
lopatenko@inf.unibz.it

Leopoldo Bertossi
Carleton University
School of Computer Science
Ottawa, Canada.
bertossi@scs.carleton.ca

Abstract

A database D may be inconsistent wrt a given set IC of integrity constraints. Consistent Query Answering is the problem of computing from D the answers to a query that are consistent wrt IC. Consistent answers have been characterized as those that are invariant under certain minimal forms of restoration of the consistency of the database. In this paper we investigate algorithmic and complexity theoretic issues of CQA under database repairs that minimally depart −wrt the cardinality of the symmetric difference− from the original database. Research on this kind of repairs had been suggested in the literature, but no systematic study had been done. Here we obtain first tight complexity bounds.

1. Introduction

Integrity constraints (ICs) are expected to be satisfied by a database in order to keep its semantic correspondence with the outside reality it is modelling. However, it is often the case that IC satisfaction cannot be guaranteed, and inconsistent database states are common, e.g. in integrated databases, census databases, legacy data, etc. [7].

Consistent Query Answering (CQA) is the problem of computing from a database those answers to a query that are consistent with respect to certain ICs that the database as a whole may fail to satisfy. Consistent answers have been characterized as those that are invariant under minimal forms of restoration of the consistency of the database [2]. From this perspective, CQA is a form of cautious reasoning from a database under integrity constraints.

The notion of minimal restoration of consistency was captured in [2] in terms of database *repairs*, i.e. new database instances that share the schema with the original database, but differ from the latter by a *minimal set of whole tuples under set inclusion* (the S-repairs below). That is, the symmetric difference $D\Delta D'$ is minimized wrt set inclusion between the original instance D and a repaired version D'. In [7, 17, 2, 9, 4, 10] complexity bounds for CQA under this repair semantics have been given (c.f. [11] for a survey). However, the semantics of CQA based on cardinality-based repairs of the original database that *minimize the number of whole tuples by which the instances differ* (C-repairs below) has received less attention.

Example 1. Consider a database schema $P(X, Y, Z)$ with the functional dependency $X \rightarrow Y$. The inconsistent instance $D = \{P(a, b, c), P(a, c, d), P(a, c, e)\}$ has two repairs wrt set inclusion, namely $\{P(a, b, c)\}$ and $\{P(a, c, d),\ P(a, c, e)\}$, because the symmetric set difference with the original instance is minimal under set inclusion. However only the latter is a cardinality-based repair, because the cardinality of the symmetric set difference becomes a minimum. □

In this paper we address the problem of obtaining complexity bounds for CQA under the semantics given by cardinality-based repairs, and we do this by introducing some graph theoretic techniques and results that, apart from being interesting by themselves, have a wider applicability in the context of CQA. Although research on cardinality- and tuple-based repairs had been proposed and started before in the context of CQA [7, 3], no detailed analysis of their complexity theoretic properties has been provided.

Our emphasis is on CQA, as opposed to computing or checking repairs under the cardinality semantics. This is because we are usually not interested in computing specific repairs (there are exceptions though, e.g. in census-like data [6]), but in characterizing and computing consistent answers to queries. However, the repair semantics we choose will have an impact on CQA.

Example 2. (example 1 continued) The query $P(x, y, x)$? has (a, c, d) and (a, c, e) as consistent answers under the cardinality semantics (the classic

answers in the only repair), but none under the set inclusion semantics (there is no classic answer shared by the two repairs). □

We extend our analysis of cardinality-based repairs by considering the natural generalization where database tuples have possibly different weights that have to be taken into account when repairing the database. Such an extension of the minimum cardinality semantics is useful for consistency handling in data integration systems [5], because we can impose different preferences on the data sources, to capture different degrees of trust among them [18]; and also in census applications [16], where preferences are imposed depending on the kinds of data at hand.

The cardinality-based CQA as studied in this paper has interesting properties that make it useful as a semantics for CQA. First of all, as illustrated in Example 1, it is clearly the case that every cardinality-based repair is also a set inclusion-based repair, but not necessarily the other way around. In consequence, the consistent query answers under cardinality repairs form a superset of the consistent answers under the set inclusion-based semantics. Actually, in situations where the latter does not give any answers (c.f. Example 2), the former does return answers. These answers could be further filtered out according to other criteria at a post-processing phase. In extreme cases, when there is only one database tuple in semantic conflict with the rest of a possible large set of other tuples, the existence of a set inclusion-based repair containing the only conflicting tuple would easily lead to an empty set of consistent answers. The cardinality-based semantics would not allow such a repair.

This feature of the cardinality-based repair semantics comes at a a price. Actually, we prove here that CQA has a higher data complexity than the classic, set inclusion-based semantics, actually $P^{NP(log(n))}$-hard vs. *PTIME* for denial constraints [10]. On the other side, the cardinality-based semantics has the interesting property that CQA, a form of cautious (or *certain*) reasoning (true in *all* repairs) and its brave (or *possible*) version, i.e. true in *some* repair, are mutually poly-time reducible and share the same complexity. This is established by first proving some useful graph theoretic lemmas about maximum independent sets. This result may not hold for classic CQA.

We concentrate on the class of denial integrity constraints, which includes most of the constraints found in applications where inconsistencies naturally arise, e.g. census-like databases [6], experimental samples databases, biological databases, etc. Complexity results refer to data complexity [1]. For complexity theoretic definitions and classic results we refer to [22], to [1] for foundations of databases. The proofs of all the results in this paper can be found in [21].

2. Repair Semantics and CQA

A relational database instance D can be identified with a finite set of ground atoms of the form $R(\bar{t})$, where R is a relation in the database schema $\mathcal{D}$, and $\bar{t}$ is a finite sequence of constants taken from the underlying database domain $\mathcal{U}$. The ground atom $R(\bar{t})$ is also called a *database tuple*.[1]

The relational schema $\mathcal{D}$ determines a first-order language $L(\mathcal{D})$ based on the relation names, the elements of $\mathcal{U}$, and extra built-in predicates. In that language, integrity constraints are sentences, and queries are formulas, usually with free variables. We assume in this paper that sets IC of ICs are always consistent in the sense that they are simultaneously satisfiable as first-order sentences. A database is *consistent* wrt to a given set of integrity constraints IC if the sentences in IC are all true in D, denoted $D \models IC$. An answer to a query $Q(\bar{x})$, with free variables $\bar{x}$, is a tuple $\bar{t}$ that makes Q true in D when the variables in $\bar{x}$ are interpreted as the corresponding values in $\bar{t}$, denoted $D \models Q(\bar{t})$.

For a database D, possibly inconsistent with respect to IC, the *consistent answers to a query Q from D wrt IC* are characterized as those answers that are invariant under all *minimal* forms of restoration of consistency for D, where minimality refers to some sort of distance between the original instance D and alternative consistent instances.

Definition 1. For a database D, integrity constraints IC and a partial order $\preceq_{D,\mathcal{S}}$ over databases depending on the original database D and a repair semantics $\mathcal{S}$, a *repair of D wrt IC under $\mathcal{S}$* is an instance D' such that: (a) D' has the same schema and domain as D; (b) $D' \models IC$; and (c) there is no D'' satisfying (a) and (b), such that $D'' \prec_{D,\mathcal{S}} D'$. The set of all repairs is denoted with $Rep(D, IC, \mathcal{S})$. □

The class $Rep(D, IC, \mathcal{S})$ depends upon the semantics $\mathcal{S}$, which determines the partial order (or distance) $\preceq$ (c.f. Definition 3) and the way repairs can be obtained, e.g. by allowing both insertions and deletions of whole tuples [2], or deletions only [10], or changes of attribute values only [23, 6, 15], etc.

Definition 2. Let D be a database, IC a set of ICs, and Q a query. (a) A ground tuple $\bar{t}$ is a *consistent answer* to Q wrt IC under semantics $\mathcal{S}$ if for every

1 We also use the term *tuple* to refer to a finite sequence $\bar{t} = (c_1, \ldots, c_n)$ of constants of the database domain, but a *database tuple* is a ground atomic sentence with predicate in $\mathcal{D}$ (excluding built-ins predicates, like comparisons).

$D' \in Rep(D, IC, \mathcal{S})$, $D' \models Q(\bar{t})$.
(b) $Cqa(Q, D, IC, \mathcal{S})$ is the set of consistent answers to Q in D wrt IC under semantics $\mathcal{S}$. If Q is a sentence (a boolean query), $Cqa(Q, D, IC, \mathcal{S}) := \{yes\}$ when $D' \models Q$ for every $D' \in Rep(D, IC, \mathcal{S})$, and $Cqa(Q, D, IC, \mathcal{S}) := \{no\}$, otherwise.
(c) $CQA(Q, IC, \mathcal{S}) := \{(D, \bar{t}) \mid \bar{t} \in Cqa(\ Q, D,\ IC,\ \mathcal{S})\}$, is the *decision problem* of CQA. □

In the literature different notions of distance have been considered, they give rise to different repair semantics. We summarize here the most common ones, those that we will investigate in this work.

Definition 3. (repair semantics) (a) *S-repair semantics* [2]: $D' \preceq D''$ is defined by $D' \triangle D \subseteq D'' \triangle D$.
(b) *C-repair semantics*: $D' \preceq D''$ is defined by $|D' \triangle D| \leq |D'' \triangle D|$.
(c) *A-repair semantics*: It minimizes a numerical aggregation function over attribute changes throughout the database. □

Particular classes of A-repairs can be found in [16, 15], where the aggregation function to be minimized is the number of all attribute changes; and in [6], where the function is the overall quadratic difference obtained from the changes in numerical attributes between the original database and the repair. Complexity theoretic results for A-repairs under denial constraints are reported in [6]. Another notion of attribute-based repair, not explored here and not included in Definition 3, would minimize, set-theoretically, the set of attribute changes.

Another dimension of the repair problem is related to the allowed repair actions. In [2] they are insertions and deletions of whole database tuples into/from relations; in [10] only deletions are allowed; and in [23, 6, 15], an A-repair semantics is used that accepts only changes of values in attributes, so that tuples cannot be deleted nor can they be inserted. We call the former *tuple-based repairs*, and the latter, *attribute-based repairs*.

In this paper, unless otherwise stated, we assume as usual that repairs are obtained through insertions or deletions of whole database tuples. In this framework, it is easy to prove that every C-repair is an S-repair; and consequently every consistent query answer under the S-repair semantics is a consistent query answer under the C-repair semantics. However, as Example 1 shows, not every S-repair is a C-repair. In that example, attribute-based repairs could be $\{P(a, c, c), P(a, c, d),\ P(a, c, e)\}$, suggesting that we made a mistake in the second argument of the first tuple, but also $\{P(a, b, c), P(a, b, d),\ P(a, b, e)\}$. If the aggregate function in Definition 3(c) is the number of changes in attribute values, the former would be a repair, but not the latter. These instances are neither S- nor C-repairs if the changes of attribute values have to be simulated via deletions followed by insertions.

Integrity constraints may be any first-order sentences written in language $L(\mathcal{D})$, but most of our results refer to denial constraints only.

Definition 4. *Denial constraints* are integrity constraints of the form $\forall \bar{x} \neg(A_1 \wedge \ldots \wedge A_m \wedge \gamma)$, where each A_i is a database atom and γ is a conjunction of comparison atoms. *Binary denial constraints* are denial constraints with only two database atoms. □

Notice that functional dependencies, e.g. $\forall x \forall y \forall z \neg(R(x, y) \wedge R(x, z) \wedge y \neq z)$, are binary denial constraints; and range constraints are one-database atom denials. For denial ICs, tuple-based repairs are obtained by tuple deletions only [10].

3. Complexity of CQA under C-Repairs

CQA under the C-repair semantics (*mcss*-repair semantics in Definition 3(b)) has received less attention in the literature than the same problem under the S-repair semantics. An exception is [3], where C-repairs were specified using logic programs with non-prioritized weak constraints under the skeptical stable model semantics [8]. As a consequence, from results in [8] (c.f. also [20]), we obtain that an upper bound on the data complexity of CQA under the C-repair semantics is the class $\Delta_3^P(log(n))$ of decision problems that can be solved by a polynomial time machine that makes a logarithmic number of calls to an oracle in Σ_2^P. Also [16, 15] considers minimum cardinality-based repairs, however not under a tuple-based semantics, but under attribute-based changes. In [16] no complexity issues are investigated; and [15] studies only aggregation constraints that impose restrictions on the values taken by aggregation functions. In this section we investigate the static complexity of tuple-based CQA under the C-repair semantics.

In [4], *conflict graphs* were first introduced to study the complexity of CQA for aggregate queries wrt FDs. They have as vertices the database tuples, and edges connect two tuples that simultaneously violate a FD. There is a one-to-one correspondence between S-repairs of the database and the set-theoretically maximal independent sets (simply called *maximal independent sets*) in the conflict graph. Similarly, there is a one-to-one correspondence between C-repairs and *maximum independent sets* in the same graph (but now they are maximum in cardinality).

Conflict graphs for databases wrt general denial constraints become *conflict hypergraphs* [10] that have as vertices the database tuples, and as hyperedges the (set

theoretically minimal) collections of tuples that simultaneously violate one of the denial constraints.

The correspondence for conflict graphs between repairs and independent sets -maximum or maximal depending on the semantics- still holds for hypergraphs, where an independent set in an hypergraph is a set of vertices that does not contain any hyperedges [10].

Notice that every tuple in the original database belongs to an S-repair[2], but not necessarily to some C-repair. In consequence, testing membership of vertices to some maximum independent set becomes a problem that is relevant to address. In order to deal with this problem, we will use some graph theoretic constructions and lemmas about maximum independent sets in them. The proofs of these results use in general a self-reducibility property of independent sets that can be expressed as follows: For any graph G and vertex v, every maximum independent set that contains v (meaning a maximum independent set that, in addition, contains v) consist of vertex v together with a maximum independent set of the graph G' that is obtained from G by deleting all vertices adjacent to v.

In order to make the presentation simpler, we will concentrate mostly on conflicts graphs and FDs. However, the results obtained carry over to denial constraints (and their hypergraphs).

Lemma 1. Given a graph G and a vertex v in it, there is a graph G' extending G, obtained by adding a new vertex v' that is connected only to the neighbors of v, such that in G' the following properties are equivalent: (a) There is a maximum independent set I of G containing v. (b) v belongs to every maximum independent set of G'. (c) The sizes of maximum independent sets in G and G' differ by one. □

Lemma 2. For every graph G and vertex v there is a graph G' extending G that can be constructed in polynomial time in the size of G, such that v belongs to all maximum independent sets of G iff v belongs to some maximum independent set of G'. □

From the lemmas and the membership to $FP^{NP(log(n))}$ [22] of computing the size of a maximum clique in a graph [19], we obtain

Lemma 3. The problems of deciding for a vertex in a graph if it belongs to some maximum independent set and if it belongs to all maximum independent sets are both in $P^{NP(log(n))}$. □

Since a ground atomic query is consistently true when it belongs, as a database tuple, i.e. as a vertex in the conflict graph, to all the maximum independent sets of the conflict graph, we obtain

Theorem 1. For functional dependencies and ground atomic queries, CQA under the C-repair semantics belongs to $P^{NP(log(n))}$. □

Lemmas 1, 2 and 3 still hold for hypergraphs, and in consequence the polynomial time mutual reducibility between the *certain* and *possible* semantics for CQA still holds for denial constraints and ground atomic queries.

Theorem 2. For denial constraints and non-existentially quantified conjunctive queries, CQA under the C-semantics belongs to $P^{NP(log(n))}$. □

Lemma 4. Deciding if a vertex belongs to all maximum independent sets of a graph is $P^{NP(log(n))}$-hard. □

Theorem 3. For denial constraints, CQA for ground atomic queries under the C-semantics is $P^{NP(log(n))}$-hard. □

This theorem is interesting, because CQA under denial constraints but S-repair semantics is in *PTIME* for arbitrary ground atomic queries [10]; and also because query answering under the S-repair semantics in the context of belief revision/update is more complex than the same problem for S-repair semantics (assuming the polynomial hierarchy does not collapse); more precisely Winslett's framework [12] (based on set inclusion) is Π_2^P-complete, while Dalal's [13] (based on set cardinality) is $P^{NP(log(n))}$-complete [14].

Connections between CQA and belief revision were already established in [2]. Notice that our complexity results do not follow, at least not straightforwardly, from the results for belief revision presented in [14]. They apply in the propositional setting, in combined complexity (as opposed to data complexity), and the revision formulas (in our case the constraints) and the query do not necessarily satisfy our conditions.

The C-repair semantics can be extended in order to consider weights on the tuples (in the original, non-weighted case all of them would have weight 1).

Definition 5. Assume that every database tuple $R(\bar{t})$ in D has an associated numerical cost $w(R(\bar{t}))$. D' is a repair of D under the *weighted minimum cardinality set semantics* if the order relation used in Definition 1 is given by $D_1 \preceq_{D,w} D_2 :\Longleftrightarrow |D \triangle D_1|_w \leq |D \triangle D_2|_w$, where $|S|_w$ for a set of tuples S is the sum of the weights of the elements of S. □

Theorem 4. CQA for ground atomic queries wrt denial constraints under the weighted minimum cardinality set semantics belongs to P^{NP}. □

2 Except when an IC forces one particular tuple not to belong to the database, but we do not consider in this work such *non generic* ICs [7].

We have provided tight complexity bounds for consistent query answering under the (non-weighted) C-repair semantics. With these results we have a much more clear picture of the complexity of CQA under the three main alternative semantics for database repairs, namely those based on sets of tuples inclusion, cardinality of set of tuples (this work), and changes of attribute values (c.f. [21, 6] for more details). The repairs, mainly under the first two approaches, depend on the structure of the database. Recent and incipient work that could lead to a better understanding of CQA under database normalization conditions is reported in [24].

Acknowledgments: Research supported by NSERC, and EU projects: Knowledge Web, Interop and Tones. L. Bertossi is Faculty Fellow of IBM Center for Advanced Studies (Toronto Lab.). Part of this research was done while L. Bertossi visited the University of Bolzano during the summer 2005; he appreciates the hospitality and support of Enrico Franconi and the KRDB group. We appreciate the useful feedback from the anonymous reviewers.

References

[1] Abiteboul, S., Hull, R. and Vianu, V. *Foundations of Databases*. Addison-Wesley, 1995.

[2] Arenas, M., Bertossi, L. and Chomicki, J. Consistent Query Answers in Inconsistent Databases. *Proc. ACM Symposium on Principles of Database Systems (PODS 99)*, 1999.

[3] Arenas, M., Bertossi, L. and Chomicki, J. Answer Sets for Consistent Query Answering in Inconsistent Databases. *Theory and Practice of Logic Programming*, 2003, 3(4-5):393-424.

[4] Arenas, M., Bertossi, L., Chomicki, J., He, X., Raghavan, V. and Spinrad, J. Scalar Aggregation in Inconsistent Databases. *Theoretical Computer Science*, 2003, 296:405–434.

[5] Bertossi, L. and Bravo, L. Consistent Query Answers in Virtual Data Integration Systems. In *Inconsistency Tolerance*, Springer LNCS 3300, 2004, pp. 42-83.

[6] Bertossi, L., Bravo, L., Franconi, E. and Lopatenko, A. Fixing Numerical Attributes under Integrity Constraints. *Proc. Tenth International Symposium on Database Programming Languages (DBPL 05)*, Springer LNCS 3774, 2005, pp. 262-278.

[7] Bertossi, L. and Chomicki, J. Query Answering in Inconsistent Databases. In *Logics for Emerging Applications of Databases*. Springer, 2003, pp. 43-83.

[8] Buccafurri, F., Leone, N. and Rullo, P. Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering*, 2000, 12(5):845-860.

[9] Calì, A., Lembo, D. and Rosati, R. Complexity of Query Answering over Inconsistent and Incomplete Databases. *Proc. ACM Symposium on Principles of Database Systems (PODS 03)*, 2003, pp. 260-271.

[10] Chomicki, J. and Marcinkowski, J. Minimal-Change Integrity Maintenance using Tuple Deletions. *Information and Computation*, 2005, 197(1-2):90-121.

[11] Chomicki, J. and Marcinkowski, J. On the Computational Complexity of Minimal-Change Integrity Maintenance in Relational Databases. In *Integrity Tolerance*, L. Bertossi, A. Hunter, T. Schaub (eds.), Springer LNCS 3300, 2004, pp. 119-150.

[12] Chou, T. and Winslett, M. A Model-Based Belief Revision System. *J. Automated Reasoning*, 1994, 12:157-208.

[13] Dalal, M. Investigations into a Theory of Knowledge Base Revision: preliminary report. *Proc. Seventh National Conference on Artificial Intelligence (AAAI 88)*, 1988, pp. 475–479.

[14] Eiter, T. and Gottlob, G. On the Complexity of Propositional Knowledge Base Revision, Updates, and Counterfactuals. *Artificial Intelligence*, 1992, 57(2-3):227-270.

[15] Flesca, S., Furfaro, F. Parisi, F. Consistent Query Answers on Numerical Databases under Aggregate Constraints. *Proc. Tenth International Symposium on Database Programming Languages (DBPL 05)*, Springer LNCS 3774, 2005, pp. 279-294.

[16] Franconi, E., Laureti Palma, A., Leone, N., Perri, S. and Scarcello, F. Census Data Repair: a Challenging Application of Disjunctive Logic Programming. In *Proc. Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 01)*, Springer LNCS 2250, 2001, pp. 561-578.

[17] Fuxman, A. and Miller, R. First-Order Query Rewriting for Inconsistent Databases. *Proc. International Conference on Database Theory (ICDT 05)*, Springer LNCS 3363, 2004, pp. 337-351.

[18] Greco, G. and Lembo, D. Data Integration with Preferences Among Sources. *Proc. International Conference on Conceptual Modeling (ER 04)*, Springer LNCS 3288, 2004, pp. 231-244.

[19] Krentel, M. The Complexity of Optimization Problems. *J. Computer and Systems Sciences*, 1988, 36:490-509.

[20] Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S. and Scarcello, F. The DLV System for Knowledge Representation and Reasoning. To appear in *ACM Transactions on Computational Logic*.

[21] Lopatenko, A. and Bertossi, L. Complexity of Consistent Query Answering in Databases under Cardinality-Based and Incremental Repair Semantics. Corr Archiv paper cs.DB/0604002. Posted April 02, 2006.

[22] Papadimitriou, C. *Computational Complexity*. Addison-Wesley, 1994.

[23] Wijsen, J. Condensed Representation of Database Repairs for Consistent Query Answering. *Proc. International Conference on Database Theory (ICDT 03)*, Springer LNCS 2572, 2003, pp. 378-393.

[24] Wijsen, J. Project-Join-Repair: An Approach to Consistent Query Answering Under Functional Dependencies. *Proc. Int. Conf. on Flexible Query Answering Systems (FQAS 06)*, Springer LNAI 4027, 2006, pp. 1-12.