

Complexity and Approximation of Fixing Numerical Attributes in Databases Under Integrity Constraints^{*}

Leopoldo Bertossi Loreto Bravo

Carleton University, School of Computer Science, Ottawa, Canada.
{bertossi,lbravo}@scs.carleton.ca

Enrico Franconi Andrei Lopatenko¹

Free University of Bozen–Bolzano, Faculty of Computer Science, Italy.
{franconi,lopatenko}@inf.unibz.it

Abstract

Consistent query answering is the problem of computing the answers from a database that are consistent with respect to certain integrity constraints that the database as a whole may fail to satisfy. Those answers are characterized as those that are invariant under minimal forms of restoring the consistency of the database. In this context, we study the problem of repairing databases by fixing integer numerical values at the attribute level with respect to denial and aggregation constraints. We introduce a quantitative definition of database fix, and investigate the complexity of several decision and optimization problems, including *DFP*, i.e. the existence of fixes within a given distance from the original instance, and *CQA*, i.e. deciding consistency of answers to aggregate conjunctive queries under different semantics. We provide sharp complexity bounds, identify relevant tractable cases; and introduce approximation algorithms for some of those that are intractable. More specifically, we obtain results like undecidability of existence of fixes for aggregation constraints; *MAXSNP*-hardness of *DFP*, but a good approximation algorithm for a relevant special case; and intractability but good approximation for *CQA* for aggregate queries for one database atom denials (plus built-ins).

Key words: Integrity constraints, Inconsistent databases, Consistent query answering, Database repairs

^{*} Dedicated to the memory of Alberto Mendelzon. Our research on this topic started with conversations with him. Alberto was always generous with his time, advice and ideas.

¹ Also: University of Manchester, Department of Computer Science, UK.

1 Introduction

Integrity constraints (ICs) are used to impose semantics on a database with the purpose of making the database an accurate model of an application domain. Database management systems or application programs enforce the satisfaction of the ICs by rejecting undesirable updates or executing additional compensating actions. However, there are many situations where we need to interact with databases that are inconsistent in the sense that they do not satisfy certain desirable ICs. In this context, an important problem in database research consists in characterizing and retrieving consistent data from inconsistent databases [4], in particular consistent answers to queries. From the logical point of view, consistently answering a query posed to an inconsistent database amounts to evaluating the truth of a formula against a particular class of first-order structures [2], as opposed to the usual process of truth evaluation in a single structure (the relational database).

Certain database applications, like census, demographic, financial, and experimental data, contain quantitative data, usually associated to nominal or qualitative data, e.g. number of children associated to a household identification code (or address); or measurements associated to a sample identification code. Usually this kind of data contains errors or mistakes with respect to certain semantic constraints. For example, a census form for a particular household may be considered incorrect if the number of children exceeds 20; or if the age of a parent is less than 10. These restrictions can be expressed with denial integrity constraints, that prevent some attributes from taking certain combinations of values [11]. Other restrictions may be expressed with aggregation ICs, e.g. the maximum concentration of certain toxin in a sample may not exceed a certain specified amount; or the number of married men and married women must be the same. Inconsistencies in numerical data can be resolved by changing individual attribute values, while keeping values in the keys, e.g. without changing the household code, the number of children is decreased considering the admissible values.

We consider the problem of fixing integer numerical data wrt certain constraints while (a) keeping the values for the attributes in the keys of the relations, and (b) minimizing the quantitative global distance between the original and modified instances. Since the problem may admit several global solutions, each of them involving possibly many individual changes, we are interested in characterizing and computing data and properties that remain invariant under any of these fixing processes. We concentrate on denial and aggregation constraints; and conjunctive queries, with or without aggregation.

Database repairs have been studied in the context of consistent query answering (CQA), i.e. the process of obtaining the answers to a query that are consistent wrt a given set of ICs [2] (cf. [4] for a survey). There, consistent

data is characterized as invariant under all minimal forms of restoring consistency, i.e. as data that is present in all minimally repaired versions of the original instance (the *repairs*). Thus, an answer to a query is consistent if it can be obtained as a standard answer to the query from *every possible* repair. In most of the research on CQA, a repair is a new instance that satisfies the given ICs, but differs from the original instance by a minimal set, under set inclusion, of (completely) deleted or inserted tuples. Changing the value of a particular attribute can be modelled as a deletion followed by an insertion, but this may not correspond to a minimal repair. However, in certain applications it may make more sense to correct (update) numerical values only in certain attributes. This requires a new definition of repair that considers: (a) the quantitative nature of individual changes, (b) the association of the numerical values to other key values; and (c) a quantitative distance between database instances.

Example 1 Consider a network traffic database D storing flow measurements of links in a network. This network has two types of links, labelled 0 and 1, with maximum capacities 1000 and 1500, resp. The following database D is inconsistent wrt this IC.

<i>Traffic</i>	<i>Time</i>	<i>Link</i>	<i>Type</i>	<i>Flow</i>
	1.1	a	0	1100
	1.1	b	1	900
	1.3	b	1	850

Under the tuple and set oriented semantics of repairs [2], there is a unique repair, namely deleting tuple $Traffic(1.1, a, 0, 1100)$ $Traffic(1.1, a, 0, 1100)$. However, we have two options that may make more sense than deleting the flow measurement, namely updating the violating tuple to $Traffic(1.1, a, 0, 1000)$ or to $Traffic(1.1, a, 1, 1100)$; satisfying an implicit requirement that the numbers should not change too much. \square

Update-based repairs for restoring consistency are studied in [26]; where changing values in attributes in a tuple is made a primitive repair action; and semantic and computational problems around CQA are analyzed from this perspective. However, peculiarities of changing numerical attributes are not considered, and more importantly, the distance between databases instances used in [26,27] is based on set-theoretic homomorphisms, but is not quantitative, as in this paper.

In [26] the repaired instances are called *fixes*, a term that we keep here (instead of *repairs*), because our basic repair actions are also changes of (numerical) attribute values. In this paper we consider fixable attributes that take integer values and the quadratic, Euclidean distance L_2 between database instances. Specific fixes and approximations may be different under other distance functions, e.g. the “city distance” L_1 (the sum of absolute differences), but the general (in)tractability and approximation results remain. Moving to the case of

real numbers will certainly bring new issues that require different approaches; they are left for ongoing and future research. Actually it would be natural to investigate them in the richer context of constraint databases [18].

The problem of attribute-based correction of census data forms is addressed in [11] using disjunctive logic programs with stable model semantics. Several underlying and implicit assumptions that are necessary for that approach to work are made explicit and used here, extending the semantic framework of [11].

We provide semantic foundations for fixes that are based on changes on numerical attributes in the presence of key dependencies and wrt denial and aggregate ICs, while keeping the numerical distance to the original database to a minimum. This framework introduces new challenging decision and optimization problems, and many algorithmic and complexity theoretic issues. We concentrate in particular on the “Database Fix Problem” (*DFP*), of determining the existence of a fix at a distance not bigger than a given bound, in particular considering the problems of construction and verification of such a fix. These problems are highly relevant for large inconsistent databases. For example, solving *DFP* can help us find the minimum distance from a fix to the original instance; information that can be used to prune impossible branches in the process of materialization of a fix. The *CQA* problem of deciding the consistency of query answers is studied wrt decidability, complexity, and approximation under several alternative semantics.

We prove that *DFP* and *CQA* become undecidable in the presence of aggregation constraints. However, *DFP* is *NP*-complete for linear denials, which are enough to capture census like applications. *CQA* belongs to Π_2^P and becomes Δ_2^P -hard, but for a relevant class of denials we get tractability of *CQA* for non-aggregate queries, which is again lost with aggregation. Wrt approximation, we prove that *DFP* is *MAXSNP*-hard in general, and for a relevant subclass of denials we provide an approximation within a constant factor that depends on the number of atoms in them. All the algorithmic and complexity results, unless otherwise stated, refer to data complexity [1], i.e. to the size of the database that here includes a binary representation for numbers. For complexity theoretic definitions and classical results we refer to [21].

This paper is structured as follows. Section 2 introduces basic definitions. Section 3 presents the notion of database fix, several notions of consistent answer to a query; and some relevant decision problems. Section 4 investigates their complexity. In Section 5 approximations for the problem of finding the minimum distance to a fix are studied, obtaining negative results for the general case, but good approximation for the class of local denial constraints. Section 6 investigates tractability of *CQA* for conjunctive queries and denial constraints containing one database atom plus built-ins. Section 8 presents some conclusions and refers to related work. Some auxiliary, technical results

can be found in Appendix A, and they are used in the proofs of some of the results presented in the main body of this paper.²

2 Preliminaries

Consider a relational schema $\Sigma = (\mathcal{U}, \mathcal{R}, \mathcal{B}, \mathcal{A})$, with domain \mathcal{U} that includes \mathbb{Z} ,³ \mathcal{R} a set of database predicates, \mathcal{B} a set of built-in predicates, and \mathcal{A} a set of attributes. A database instance is a finite collection D of *database tuples*, i.e. of ground atoms $P(\bar{c})$, with $P \in \mathcal{R}$ and \bar{c} a tuple of constants in \mathcal{U} . There is a set $\mathcal{F} \subseteq \mathcal{A}$ of all the *fixable* attributes, those that take values in \mathbb{Z} and are allowed to be fixed. Attributes outside \mathcal{F} are called *rigid*. \mathcal{F} need not contain all the numerical attributes, that is we may also have rigid numerical attributes.

We also have a set \mathcal{K} of key constraints expressing that relations $R \in \mathcal{R}$ have a primary key K_R , $K_R \subseteq (\mathcal{A} \setminus \mathcal{F})$. Later on (cf. Definition 2), we will assume that \mathcal{K} is satisfied both by the initial instance D , denoted $D \models \mathcal{K}$, and its fixes. Since $\mathcal{F} \cap K_R = \emptyset$, values in key attributes cannot be changed in a fixing process; so the constraints in \mathcal{K} are *hard*. In addition, there may be a separate set IC of *flexible* ICs that may be violated, and it is the job of a fix to restore consistency wrt them (while still satisfying \mathcal{K}).

A *linear denial constraint* [18] has the form $\forall \bar{x} \neg (A_1 \wedge \dots \wedge A_m)$, where the A_i are database atoms (i.e. with predicate in \mathcal{R}), or built-in atoms of the form $x\theta c$, where x is a variable, c is a constant and $\theta \in \{=, \neq, <, >, \leq, \geq\}$, or $x = y$. If $x \neq y$ is allowed, we call them *extended* linear denials. We assume that all the constants appearing in ICs belong to the database domain.

Example 2 The following are linear denials (we replace \wedge by a comma): (a) No customer is younger than 21: $\forall Id, Age, Income, Status \neg (Customer(Id, Age, Income, Status), Age < 21)$. (b) No customer with income less than 60000 has “silver” status: $\forall Id, Age, Income, Status \neg (Customer(Id, Age, Income, Status), Income < 60000, Status = silver)$. (c) The constraints in Example 1, e.g. $\forall T, L, Type, Flow \neg (Traffic(T, L, Type, Flow), Type = 0, Flow > 1000)$. \square

We consider aggregation constraints (ACs) [23] and aggregate queries with *sum*, *count*, *average*. *Filtering* ACs impose conditions on the tuples over which aggregation is applied, e.g. $sum(A_1 : A_2 = 3) > 5$ is a sum over A_1 of tuples with $A_2 = 3$. *Multi-attribute* ACs allow arithmetical combinations of attributes as arguments for *sum*, e.g. $sum(A_1 + A_2) > 5$ and $sum(A_1 \times A_2) > 100$. If an AC has attributes from more than one relation, it is *multi-relation*, e.g. $sum_{R_1}(A_1) = sum_{R_2}(A_1)$, otherwise it is *single-relation*.

² The results in Appendix A. are numbered with an “A” preceding the number, e.g. Lemma A.30.

³ With simple denial constraints, numbers can be restricted to, e.g. \mathbb{N} or $\{0, 1\}$.

An *aggregate conjunctive query* has the form $q(x_1, \dots, x_m; \text{agg}(z)) \leftarrow B(x_1, \dots, x_m, z, y_1, \dots, y_n)$, where agg is an aggregation function and its *non-aggregate matrix* (NAM) given by $q'(x_1, \dots, x_m) \leftarrow B(x_1, \dots, x_m, z, y_1, \dots, y_n)$ is a usual first-order (FO) conjunctive query with built-in atoms, such that the *aggregation attribute* z does not appear among the x_i . Here we use the set semantics. An aggregate conjunctive query is *cyclic* (*acyclic*) if its NAM is cyclic (acyclic) [1].

Example 3 $q(x, y, \text{sum}(z)) \leftarrow R(x, y), Q(y, z, w), w \neq 3$ is an aggregate conjunctive query, with aggregation attribute z . Each answer (x, y) to its NAM, i.e. to $q(x, y) \leftarrow R(x, y), Q(y, z, w), w \neq 3$, is expanded to $(x, y, \text{sum}(z))$ as an answer to the aggregate query. $\text{sum}(z)$ is the sum of all the values for z having a w , such that (x, y, z, w) makes $R(x, y), Q(y, z, w), w \neq 3$ true. In the database instance $D = \{R(1, 2), R(2, 3), Q(2, 5, 9), Q(2, 6, 7), Q(3, 1, 1), Q(3, 1, 5), Q(3, 8, 3)\}$ the answer set for the aggregate query is $\{(1, 2, 5 + 6), (2, 3, 1 + 1)\}$. \square

An *aggregate comparison query* is a sentence of the form $q(\text{agg}(z)) \wedge \text{agg}(z) \theta k$, where $q(\text{agg}(z))$ is the head of a *scalar* aggregate conjunctive query (i.e. with no free variables, or equivalently, without *group-by*), θ is a comparison operator, and k is an integer number. For example, the following is an aggregate comparison query asking whether the aggregated value obtained via $q(\text{sum}(z))$ is bigger than 5: $Q: q(\text{sum}(z)) \wedge \text{sum}(z) > 5$, with $q(\text{sum}(z)) \leftarrow R(x, y), Q(y, z, w), w \neq 3$. We can see that aggregate comparison queries are boolean, i.e. they have a *true* or *false* answer in a database instance.

3 Least Squares Fixes

When we update numerical values to restore consistency, it is desirable to make the smallest overall variation of the original values, while considering the relative relevance or specific scale of each of the fixable attributes. Since the original instance and a fix will share the same key values (cf. Definition 2), we can use them to compute variations in the numerical values. For a tuple \bar{k} of values for the key K_R of relation R in an 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} . To each attribute $A \in \mathcal{F}$ a fixed numerical weight α_A is assigned.

Definition 1 For instances D and D' over schema Σ with the same set $\text{val}(K_R)$ of tuples of key values for each relation $R \in \mathcal{R}$, their *square distance* is

$$\Delta_{\bar{\alpha}}(D, D') = \sum_{\substack{R \in \mathcal{R}, A \in \mathcal{F} \\ \bar{k} \in \text{val}(K_R)}} \alpha_A (\pi_A(\bar{t}(\bar{k}, R, D)) - \pi_A(\bar{t}(\bar{k}, R, D')))^2,$$

where π_A is the projection on attribute A and $\bar{\alpha} = (\alpha_A)_{A \in \mathcal{F}}$. \square

Definition 2 For an instance D , a set of fixable attributes \mathcal{F} , a set of key dependencies \mathcal{K} , such that $D \models \mathcal{K}$, and a set of flexible ICs IC : A *fix* for D

wrt IC is an instance D' such that: (a) D' has the same schema and domain as D ; (b) D' has the same values as D in the attributes in $\mathcal{A} \setminus \mathcal{F}$; (c) $D' \models \mathcal{K}$; and (d) $D' \models IC$. A *least squares fix* (LS-fix) for D is a fix D' that minimizes the square distance $\Delta_{\bar{\alpha}}(D, D')$ over all the instances that satisfy (a) - (d). \square

In general we are interested in LS-fixes, but (non-necessarily minimal) fixes will be useful auxiliary instances.

Example 4 (example 1 cont.) $\mathcal{R} = \{Traffic\}$, $\mathcal{A} = \{Time, Link, Type, Flow\}$, $K_{Traffic} = \{Time, Link\}$, $\mathcal{F} = \{Type, Flow\}$, with weights $\bar{\alpha} = (10^{-5}, 1)$, resp. For original instance D , $val(K_{Traffic}) = \{(1.1, a), (1.1, b), (1.3, b)\}$, $\bar{t}((1.1, a), Traffic, D) = (1.1, a, 0, 1100)$, etc. Fixes are $D_1 = \{(1.1, a, 0, 1000), (1.1, b, 1, 900), (1.3, b, 1, 850)\}$ and $D_2 = \{(1.1, a, 1, 1100), (1.1, b, 1, 900), (1.3, b, 1, 850)\}$, with distances $\Delta_{\bar{\alpha}}(D, D_1) = 100^2 \times 10^{-5} = 10^{-1}$ and $\Delta_{\bar{\alpha}}(D, D_2) = 1^2 \times 1$, resp. Therefore, D_1 is the only LS-fix. \square

The coefficients α_A can be chosen in many different ways depending on factors like relative relevance of attributes, actual distribution of data, measurement scales, etc. In the rest of this paper we will assume, for simplification, that $\alpha_A = 1$ for all $A \in \mathcal{F}$ and $\Delta_{\bar{\alpha}}(D, D')$ will be simply denoted by $\Delta(D, D')$.

Example 5 Database D has tables $Client(ID, A, C)$, with attributes for identification (the key), age and credit line of the client; and $Buy(ID, I, P)$, with key $\{ID, I\}$ and containing clients buying items at certain prices. There are two denial ICs $IC_1 : \forall ID, P, A, C \neg (Buy(ID, I, P), Client(ID, A, C), A < 18, P > 25)$ and $IC_2 : \forall ID, A, C \neg (Client(ID, A, C), A < 18, C > 50)$, requiring that people younger than 18 cannot spend more than 25 on one item nor have a credit line higher than 50 in the store. The following tables show the database contents; and we added an extra column with values used to refer to each tuple.

<i>Client</i>	<i>ID</i>	<i>A</i>	<i>C</i>	
	1	15	52	t_1
	2	16	51	t_2
	3	60	900	t_3

<i>Buy</i>	<i>ID</i>	<i>I</i>	<i>P</i>	
	1	CD	27	t_4
	1	DVD	26	t_5
	3	DVD	40	t_6

D' :

<i>Client</i>	<i>ID</i>	<i>A</i>	<i>C</i>	
	1	15	50	t'_1
	2	16	50	t'_2
	3	60	900	t_3

<i>Buy</i>	<i>ID</i>	<i>I</i>	<i>P</i>	
	1	CD	25	t'_4
	1	DVD	25	t'_5
	3	DVD	40	t_6

D'' :

<i>Client</i>	<i>ID</i>	<i>A</i>	<i>C</i>	
	1	18	52	t''_1
	2	16	50	t''_2
	3	60	900	t_3

<i>Buy</i>	<i>ID</i>	<i>I</i>	<i>P</i>	
	1	CD	27	t_4
	1	DVD	26	t_5
	3	DVD	40	t_6

We can see that a global fix may not be the result of applying “local” minimal fixes to tuples. \square

The built-in atoms in linear denials determine a solution space for fixes as an intersection of semi-spaces, and LS-fixes can be found at its “borders” (cf. previous example and Proposition A.1 in Appendix A.). It is easy to construct examples with an exponential number of fixes. For the kind of fixes and ICs we are considering, it is possible that no fix exists, in contrast to [2,3], where, if the set of ICs is consistent as a set of logical sentences, a fix for a database always exist.

Example 6 $R(X, Y)$ has key X and fixable Y . $IC_1 = \{\forall X_1 X_2 Y \neg(R(X_1, Y), R(X_2, Y), X_1 = 1, X_2 = 2), \forall X_1 X_2 Y \neg(R(X_1, Y), R(X_2, Y), X_1 = 1, X_2 = 3), \forall X_1 X_2 Y \neg(R(X_1, Y), R(X_2, Y), X_1 = 2, X_2 = 3), \forall XY \neg(R(X, Y), Y > 3), \forall XY \neg(R(X, Y), Y < 2)\}$ is consistent. The first three ICs force Y to be different in every tuple. The last two ICs require $2 \leq Y \leq 3$. The inconsistent database $R = \{(1, -1), (2, 1), (3, 5)\}$ has no fix. Now, for IC_2 with $\forall X, Y \neg(R(X, Y), Y > 1)$ and $sum(Y) = 10$, any database with less than 10 tuples has no fixes. \square

Proposition 1 If D has a fix wrt IC , then it also has an LS-fix wrt IC .

Proof: Let ρ be the square distance between D and D' in Definition 1. The circle of radius ρ around D intersects the non empty “consistent” region that contains the database instances with the same schema and key values as D and satisfy IC . Since the circle has a finite number of instances, the distance takes a minimum in the consistent region. \square

4 Decidability and Complexity

In applications where fixes are based on changes of numerical values, computing concrete fixes is a relevant problem. In databases containing census forms, correcting the latter before doing statistical processing is a common problem [11]. In databases with experimental samples, we can fix certain erroneous quantities as specified by linear ICs. In these cases, the fixes are relevant objects to compute explicitly, which contrasts with CQA [2], where the main motivation for introducing repairs is to formally characterize the notion of a consistent answer to a query as an answer that remains under all possible repairs. In consequence, we now consider some decision problems related to existence and verification of LS-fixes, and to CQA under different semantics.

Definition 3 For an instance D and a set IC of ICs:

- (a) $Fix(D, IC) := \{D' \mid D' \text{ is an LS-fix of } D \text{ wrt } IC\}$, the *fix checking problem*.
- (b) $Fix(IC) := \{(D, D') \mid D' \in Fix(D, IC)\}$.
- (c) $NE(IC) := \{D \mid Fix(D, IC) \neq \emptyset\}$, for *non-empty* set of fixes, i.e. the prob-

lem of *checking existence of LS-fixes*.

(d) $NE := \{(D, IC) \mid Fix(D, IC) \neq \emptyset\}$.

(e) $DFP(IC) := \{(D, k) \mid \text{there is } D' \in Fix(D, IC) \text{ with } \Delta(D, D') \leq k\}$, the *database fix problem*, i.e. the problem of checking existence of LS-fixes within a given positive distance k .

(f) $DFOP(IC)$ is the optimization problem of finding the minimum distance from an LS-fix wrt IC to a given input instance. \square

Definition 4 Let D be a database, IC a set of ICs, and Q a conjunctive query.⁴ (a) A ground tuple \bar{t} is a *consistent answer* to $Q(\bar{x})$ under the: (a1) *skeptical semantics* if for every $D' \in Fix(D, IC)$, $D' \models Q(\bar{t})$. (a2) *brave semantics* if there exists $D' \in Fix(D, IC)$ with $D' \models Q(\bar{t})$. (a3) *majority semantics* if $|\{D' \mid D' \in Fix(D, IC) \text{ and } D' \models Q(\bar{t})\}| > |\{D' \mid D' \in Fix(D, IC) \text{ and } D' \not\models Q(\bar{t})\}|$.

(b) That \bar{t} is a consistent answer to Q in D under semantics \mathcal{S} is denoted by $D \models_{\mathcal{S}} Q[\bar{t}]$. If Q is boolean (a sentence) and $D \models_{\mathcal{S}} Q$, we say that *yes* is a consistent answer, meaning that Q is true in the fixes of D as prescribed by semantics \mathcal{S} . $CA(Q, D, IC, \mathcal{S})$ is the set of consistent answers to Q in D wrt IC under semantics \mathcal{S} . For a boolean Q , if $CA(Q, D, IC, \mathcal{S}) \neq \{yes\}$, $CA(Q, D, IC, \mathcal{S}) := \{no\}$.

(c) $CQA(Q, IC, \mathcal{S}) := \{(D, \bar{t}) \mid \bar{t} \in CA(Q, D, IC, \mathcal{S})\}$ is the *decision problem of consistent query answering*, of checking consistent answers. \square

Proposition 2 $NE(IC)$ can be reduced in polynomial time to the complements of $CQA(False, IC, Skeptical)$ and $CQA(True, IC, Majority)$, where *False*, *True* are ground queries that are always false, resp. true.

Proof: First for the skeptical semantics. Given a database instance D , consider the instance (D, no) for $CQA(False, IC, Sk)$, corresponding to the question “Is there an LS-fix of D wrt IC that does not satisfy *False*?” has answer *yes* iff the class of LS-fixes of D is empty. For the majority semantics, for the instance (D, no) for $CQA(True, IC, Maj)$, corresponding to the question “Is it not the case that the majority of the LS-fixes satisfy *True*?”, we get answer *yes* iff the set of LS-fixes is empty. \square

In Proposition 2, it suffices for queries *False*, *True* to be false, resp. true, in all instances that share the key values with the input database. Then, they can be represented by $\exists YR(\bar{c}, Y)$, where \bar{c} are not (for *False*), or are (for *True*) key values in the original instance.

Theorem 1 Under extended linear denials and complex, filtering, multi-attribute, single-relation, aggregation constraints, the problems NE of existence of LS-fixes, and CQA under skeptical or majority semantics are undecidable.

⁴ Whenever we say “conjunctive query”, we mean a non-aggregate query.

Proof: Hilbert’s 10th problem on existence of integer solutions to diophantine equations can be reduced to *NE*. More precisely, given a diophantine equation, it is possible to construct a database D and a set of ICs IC such that the existence of an LS-fix for D wrt IC implies the existence of a solution to the equation, and viceversa. An example can be found in Appendix C. For CQA, apply Proposition 2. \square

In Theorem 1 we have the original database and the set of ICs as input parameters. In the following we will be interested in data complexity, when only the input database varies and the set of ICs is fixed [1].

Theorem 2 For a fixed set IC of linear denials: (a) Deciding if for an instance D there is an instance D' (with the same key values as D) that satisfies IC with $\Delta(D, D') \leq k$, with positive integer k that is part of the input, is in *NP*. (b) $DFP(IC)$ is *NP*-complete.

Proof: (a) First of all, we notice that a linear denial with implicit equalities, i.e. occurrences of a same variable in two different database atoms, e.g. $\forall X, Y, Z \neg(R(X, Y), Q(Y, Z), Z > 3)$, can be replaced by its *explicit version* with explicit equalities, e.g. $\forall X, Y, Z, W \neg(R(X, Y), Q(W, Z), Y = W, Z > 3)$.

Let n be the number of tuples in the database, and l be the number of attributes which participate in IC . They are those that appear in built-in predicates in the explicit versions of the ICs that do not belong to a key or are equal to a key (because they are not allowed to change). For example, given the denial $\neg(P(X, Y), Q(X, Z), Y > 2)$, since its explicit version is $\neg(P(X, Y), Q(W, Z), Y > 2, X = W)$, the number l is 1 (for Y) if X is a key for P or Q , and 3 if X is not a key (for Y, X, W).

If there exists an LS-fix D' with $\Delta(D, D') \leq k$, then no value in a fixable attribute in D' differs from its corresponding value (through the key value) in D by more than \sqrt{k} . In consequence, the size of an LS-fix may not differ from the original instance by more than $l \times n \times \text{bin}(k)/2$, where $\text{bin}(k)$ is the size of the binary representation of k . Thus, the size of an LS-fix is polynomially bounded by the size of D and k . Since we can determine in polynomial time if D' satisfies the ICs and if the distance is smaller than k , we obtain the result.

(b) Membership: According to Proposition 1, there is an LS-fix at a square distance $\leq k$ iff there is an instance D' with the same key values that satisfies IC at a square distance $\leq k$. We use Proposition 2.

Hardness: We can reduce Vertex Cover (VC) to $DFP(IC_0)$ for a fixed set of denials IC_0 . Given a graph instance $(\mathcal{V}, \mathcal{E})$, k for VC, consider a database D with a relation $E(X, Y)$ and key $\{X, Y\}$ for the edges of the graph, and a relation $V(X, Chosen)$ for the vertices, where X is the key and attribute $Chosen$, the only fixable attribute, is initially set to 0. The constraint $IC : \forall X, Y, C_1, C_2 \neg(E(X, Y) \wedge V(X, C_1) \wedge V(Y, C_2) \wedge C_1 < 1 \wedge C_2 < 1)$ expresses

that for any edge, at least one of the incident vertices is covered. A vertex cover of size k exists iff there exists an LS-fix of D wrt IC at a distance $\leq k$. The encoding is polynomial in the size of the original graph. \square

By Proposition 1, there is a fix for D wrt IC at a distance $\leq k$ iff there is an LS-fix at a distance $\leq k$. By Theorem 2(a), if there is a fix at a distance $\leq k$, the minimum distance to D for a fix can be found by binary search in $\log(k)$ steps. Actually, if an LS-fix exists, its square distance to D is polynomially bounded by the size of D (cf. proof of Theorem 3). Since D and a fix have the same number of tuples, only the size of their values in a fix matter, and they are constrained by a fixed set of linear denials and the condition of minimality.

Theorem 3 For a fixed set IC of extended linear denials: (a) The problem $NE(IC)$ of deciding if an instance has an LS-fix wrt IC is NP -complete, and (b) CQA under the skeptical and the majority semantics is $coNP$ -hard.

Proof: (a) For hardness, linear denials are good enough. We reduce the graph 3-colorability problem to $NE(IC_0)$, for a fixed set IC_0 of ICs. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with set of vertices \mathcal{V} and set of edges \mathcal{E} . Consider the following database schema, instance D , and set IC_0 of ICs:

1. Relation $Vertex(Id, Red, Green, Blue)$ with key Id and domain \mathbb{N} for the last three attributes, actually the only three fixable attributes in the database; they can be subject to changes. For each $v \in \mathcal{V}$, we have the tuple $(v, 0, 0, 0)$ in $Vertex$ (and nothing else).

2. Relation $Edge(id_1, id_2)$. For each $e = (v_1, v_2) \in \mathcal{E}$, there are the tuples $(v_1, v_2), (v_2, v_1)$ in $Edge$. This relation is not subject to fixes.

3. Relation $Tester(Red, Green, Blue)$, with extension $(1, 0, 0), (0, 1, 0), (0, 0, 1)$. This relation is not subject to fixes.

4. Integrity constraints:

$\forall xyz \neg (Vertex(i, x, y, z), x < 1, y < 1, z < 1)$;

$\forall xyz \neg (Vertex(i, x, y, z), x > 1)$ (the same for y, z);

$\forall xyz \neg (Vertex(i, x, y, z), x = 1, y = 1, z = 1)$;

$\forall xyz \neg (Vertex(i, x, y, z), x = 1, y = 1)$; etc.

$\forall ijxyz \neg (Vertex(i, x, y, z), Vertex(j, x, y, z), Edge(i, j), Tester(x, y, z))$.

The graph is 3-colorable iff the database has an LS-fix wrt IC_0 . The reduction is polynomial in the size of the graph. If there is an LS-fix of the generated instance, then the graph is 3-colorable. If the graph is colorable, then there is a consistent instance with the same key values as the original instance; then, by Proposition 1, there is an LS-fix.

For membership, it suffices to prove that if an LS-fix exists, then its square distance to D is polynomially bounded by the size of D , considering both the number of tuples and the values taken by the fixable attributes.

We will show that if an LS-fix D' exists, then all the values in its fixable attributes are bounded above by the maximum of $n_1 + n + 1$ and $n_2 + n + 1$, where n is the number of tuples in the database, n_1 is the maximum absolute value in a fixable attribute in D , and n_2 is the maximum absolute value of a constant appearing in the ICs.

The set of denial ICs put in disjunctive form gives us a representation for all the ways we have to restore the consistency of the database. So, we have a constraint of the form $\varphi_1 \wedge \varphi_2 \cdots \varphi_m$, where each φ_i is a disjunction of negated database atoms and inequalities, e.g. something like $\neg P(X, Y, Z) \vee \neg R(X_1, Y_1) \vee X \leq c_1 \vee Y \leq c_2 \vee Z \neq Y_1$. Since fixes can be obtained by changing values of non key attributes, each tuple in a fix is determined by a set of constraints, each of which is a disjunction of atoms of the form $X_i \theta_i c_m$ or $X_i \neq Y_j$, where θ_i is an inequality of the form $\leq, \geq, <, >$. E.g. from $\neg P(X, Y, Z) \vee \neg R(X_1, Y_1) \vee X \leq c_1 \vee Y \leq c_2 \vee Z \neq Y_1$, we get $X \leq c_1 \vee Y \leq c_2 \vee Z \neq Y_1$, which for a specific tuple becomes $Y \leq c_2 \vee Z \neq Y_1$ if X is part of the key and its specific value for the tuple at hand does not satisfy $X \leq c_1$ (otherwise we drop the constraint for that tuple). In any case, every tuple in a fix can take values in a space S that is the intersection of the half-spaces defined by inequalities of the form $X_i \theta_i c_m$ minus the set of points determined by the non-equalities $X_i \neq Y_j$.

If there is a set of values that satisfies the resulting constraints, i.e. if there is an instance with the same key values that satisfies the ICs, then we can find an LS-fix at the right distance: if the difference between any value and $\max(c_1, \dots, c_l)$ is more than $n+1$ (the most we need to be sure the inequalities $X_i \neq Y_j$ are satisfied), then we systematically change values by 1, making them closer to the borders of the half-spaces, but still keeping the points within S .

(b) *coNP*-hardness follows from Proposition 2 and part (a). \square

For hardness in (a), (b) in Theorem 3, linear denials suffice. Membership in (a) can be obtained for any fixed finite set of extended denials.

Theorem 4 For a fixed set IC of linear denials: (a) The problem $Fix(IC)$ of checking if an instance is an LS-fix is *coNP*-complete, and (b) CQA under skeptical semantics is in Π_2^P , and, for ground atomic queries, Δ_2^P -hard.

Proof: (a) We reduce 3-SAT's complement to LS-fix checking for a fixed schema and set of denials IC . We have a table $Lit(l, \bar{l})$ storing complementary literals (only), e.g. $(p, \neg p)$ if p is one of the variables in the instance for SAT. Also a table Cl storing tuples of the form (φ, l, k) , where φ is a clause (we assume all the clauses have exactly 3 literals, which can be simulated by adding extra literals with unchangeable value 0 if necessary), l is a literal in the clause, and k takes value 0 or 1 (the truth value of l in φ). The first two arguments are the key of C . Finally, we have a table $Aux(K, N)$, with key K and fixable numerical attribute N , and a table $Num(N)$ with a rigid numerical attribute

N .

Given an instance $\Phi = \varphi_1 \wedge \dots \wedge \varphi_m$ for 3-SAT, we produce an initial extension D for the relations in the obvious manner, assigning arbitrary truth values to the literals, but making sure that the same literal takes the same truth value in every clause, and complementary literals take complementary truth values. Aux contains $(0, 0)$ as its only tuple; and Num contains $(s + 1)$, where s is the number of different propositional variables in Φ .

Consider now the following set of denial constraints:

- (a) $\neg(Cl(\varphi, L, U), U > 1); \neg(Cl(\varphi, L, U), U < 0)$ (possible truth values).
- (b) $\neg(Cl(\varphi, L, U), Cl(\psi, L, V), U \neq V)$ (same value for a literal everywhere).
- (c) $\neg(Cl(\varphi, L, U), Cl(\psi, L', V), Lit(L, L'), U = V)$ (complementary literals).
- (d) $\neg(Cl(\varphi, L, U), Cl(\varphi, L', V), Cl(\varphi, L'', W), U = V = W = 0, L \neq L', \dots, Aux(K, N), N = 0)$ (each clause becomes true).
- (e) $\neg(Num(Z), Aux(K, N), N \neq 0, N \neq Z)$ (possible values).

It holds that the formula is unsatisfiable iff the instance D' that coincides with D except for Aux , that now has the only tuple $(0, s + 1)$, is an LS-fix of D wrt IC . Thus, checking D' for LS-fix suffices to check unsatisfiability.

For membership to $coNP$, for an initial instance D , instances D' in the complement of $Fix(IC)$ have witnesses D'' that can be checked in polynomial time, namely instances D'' that have the same key values as D , satisfy the ICs, but $\Delta(D, D'') < \Delta(D, D')$.

(b) For the first claim on CQA, let IC and a query Q be given. The complement of CQA is in NP^{coNP} : Given an instance D , non deterministically choose an instance D' with $D' \not\models Q$ and D' a fix of D . The latter test can be done in $coNP$ (by part (a)). But $NP^{coNP} = NP^{\Sigma_1^P} = \Sigma_2^P$. In consequence, CQA belongs to $co\Sigma_2^P = \Pi_2^P$.

For the second claim, we prove hardness of CQA by a *LOGSPACE*-reduction from the following problem [17, Theo. 3.4]: Given a Boolean formula in 3CNF $\psi(X_1, \dots, X_n)$, decide if the last variable X_n is equal to 1 in the lexicographically maximum satisfying assignment (the answer is *No* if ψ is not satisfiable).

Given the clauses C_1, \dots, C_m in ψ , we create a database D with relations $Var(id, tr, fa, weight)$, $Cl(id, var_1, val_1, var_2, val_2, var_3, val_3)$ and constraints:

- (1) $\forall id, tr, fa \neg(Var(id, tr, fa, _) \wedge tr \leq 0 \wedge fa \leq 0)$
- (2) $\forall id, tr, fa \neg(Var(id, tr, fa, _) \wedge tr \geq 1 \wedge fa \geq 1)$
- (3) $\forall id, tr, fa, w \neg(Var(id, tr, fa, w) \wedge fa = 1 \wedge w > 0)$
- (4) $\forall id, v_1, x_1, v_2, x_2, v_3, x_3 \neg(Cl(id, v_1, x_1, v_2, x_2, v_3, x_3) \wedge Var(_, v_1, x'_1) \wedge Var(_, v_2, x'_2) \wedge Var(_, v_3, x'_3) \wedge x_1 \neq x'_1 \wedge x_2 \neq x'_2 \wedge v_3 \neq x'_3)$.

The extended denial constraint in 4. could be replaced by eight non-extended denial constraints.

For each variable X_i , insert a tuple $(X_i, 0, 0, 2^{n-i})$ into Var . In binary encoding, the values 2^{n-i} are polynomial in the size of original formula. For each clause $C_i = l_{i_1} \vee l_{i_2} \vee l_{i_3}$, insert a tuple $(C_i, X_{i_1}, \tilde{l}_{i_1}, X_{i_2}, \tilde{l}_{i_2}, X_{i_3}, \tilde{l}_{i_3})$ into Cl , where \tilde{l}_{i_j} is equal to 1 in case of positive occurrence of variable X_{i_j} in C_i and equal to 0 for negative occurrence. For example, for $C_6 = X_6 \vee \neg X_9 \vee X_{12}$, we insert $(C_6, X_6, 1, X_9, 0, X_{12}, 1)$.

It is easy to see that all the fixes D represent satisfying assignments for ψ , such that in case a tuple $(X_i, 1, 0, _)$ is in a fix, then value *true* must be assigned to variable X_i ; and value *false* must be assigned to X_i in case $(X_i, 0, 1, _)$ is in the fix. If ψ is unsatisfiable, then there are no fixes.

Let us now consider the cost of a repair. Assume that $S_1 = x_{i_1^1}, \dots, x_{i_{m_1}^1}$ and $S_2 = x_{i_1^2}, \dots, x_{i_{m_2}^2}$ are satisfying assignments with $S_1 \prec S_2$ under lexicographical order. Since $S_1 \prec S_2$, there exists an integer m such that $i_m^1 < i_m^2$, while for all $j < m$, $i_j^1 = i_j^2$, by definition of lexicographical order. The cost of repair S is equal to $2^{n-i_1} + 2^{n-i_2} + \dots + 2^{n-i_m} + n$, because (a) we have to update attribute *weight* for each variable that is assigned value *true*, and (b) for each tuple in relation Var , attribute *tr* or attribute *fa* is changed by 1.

Because of (a), there exists a term $2^{n-i_m^1}$ in the cost of S_1 that is bigger than the sum of all terms in S_2 , with index $\geq i_m^2$. So, the cost of the fix representing S_1 is greater than the cost of the fix representing S_2 . In consequence, the minimal fix would be the maximum according to the lexicographical order of satisfying assignments. The answer to the ground atomic query $Var(X_n, 1, 0, 1)$ is *true* iff X_n takes the value 1 in the lexicographically greatest assignment. \square

Theorem 5 For aggregate comparison queries using *sum*, *CQA* under linear denials and brave semantics is *coNP*-hard.

Proof: The reduction can be established with a fixed set IC_0 of ICs. Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consider a database with relations $V(X, Z)$, $E(U, W)$, where X is a key and Z is the only fixable attribute and takes values in $\{0, 1\}$ (which can be enforced by including in IC_0 the linear denials $\forall X \forall Z \neg(V(X, Z), Z > 1)$, $\forall X \forall Z \neg(V(X, Z), Z < 0)$). Intuitively, Z indicates with 1 if the vertex X is in the cover, and with 0 otherwise. Attributes U, V are vertices and then, non numerical.

In the original database D we have the tuples $V(e, 0)$, with $e \in \mathcal{V}$; and also the tuples $E(e_1, e_2)$ for $(e_1, e_2) \in \mathcal{E}$. Putting into IC_0 the linear constraint $\forall X_1 Z_1 X_2 Z_2 \neg(V(X_1, Z_1), V(X_2, Z_2), E(X_1, X_2), Z_1 = 0, Z_2 = 0)$, the LS-fixes of the database are in one-to-one correspondence with the vertex covers of minimal cardinality.

For the query $Q^{(k)} : q(\text{sum}(Z)) \wedge \text{sum}(Z) < k$, with $q(\text{sum}(Z)) \leftarrow V(X, Z)$, the instance (D, yes) for consistent query answering under brave semantics has answer *No*, (i.e. $Q^{(k)}$ is false in all LS-fixes) only for every k smaller than

the minimum cardinality c of a vertex cover. □

5 Approximation for the Database Fix Problem

We consider the problem of finding a good approximation for the general optimization problem $DFOP(IC)$.

Proposition 3 For a fixed set IC of linear denials, $DFOP(IC)$ is *MAXSNP*-hard.

Proof: By reduction from the *MAXSNP*-hard problem *B-Minimum Vertex Cover* (BMVC), which asks for a minimum vertex cover in a graph whose nodes have a bounded degree [16, chap. 10]. We start by encoding the graph as in the proof of Theorem 5. We also use the same initial database D . Every LS-fix D' of D corresponds to a minimum vertex cover \mathcal{V}' for \mathcal{G} and vice versa, and it holds $|\mathcal{V}'| = \Delta(D, D')$. This gives us an L -reduction from BMVC to *DFP* [21]. □

As an immediate consequence [21], we obtain that $DFOP(IC)$ cannot be uniformly approximated within arbitrarily small constant factors.

Corollary 1 Unless $P = NP$, there is no *Polynomial Time Approximation Schema* for $DFOP$. □

This negative result does not preclude the possibility of finding an efficient algorithm for approximation within a constant factor for $DFOP$. Actually, in the following we do this for a restricted but still useful class of denial constraints.

5.1 Local denials

Definition 5 A set of linear denials IC is *local* if: (a) Attributes participating in equality atoms between attributes or in joins are all rigid; (b) There is a built-in atom with a fixable attribute in each element of IC ; (c) No attribute A appears in IC both in comparisons of the form $A < c_1$ and $A > c_2$.⁵ □

In Example 5, IC is local. In Example 6, IC_1 is not local. Local constraints have the property that by doing local fixes, no new inconsistencies are generated, and there is always an LS-fix wrt to them (cf. Proposition A.2). Locality is a sufficient, but not necessary condition for existence of LS-fixes as can be seen from the database $\{P(a, 2)\}$, with the first attribute as a key and non-local denials $\neg(P(x, y), y < 3)$, $\neg(P(x, y), y > 5)$, that has the LS-fix $\{P(a, 3)\}$.

⁵ To check condition (c), $x \leq c$, $x \geq c$, $x \neq c$ have to be expressed using $<$, $>$, e.g. $x \leq c$ by $x < c + 1$.

Proposition 4 For the class of local denials, DFP is NP -complete, and $DFOP$ is $MAXSNP$ -hard.

Proof: For the first claim, membership follows from Theorem 2(b); and for hardness, we can do the same reduction as in Theorem 2(b), because the ICs used there are local denials. For the second claim, it is not difficult to see that the non-local denials in the proof of Proposition 3 can be eliminated. \square

This proposition tells us that the problem of finding good approximations in the case of local denials is still relevant. Local constraints do not make our decision problems easier. For example, Theorem 5 still holds for them, because in its proof the first two ICs in IC_0 can be eliminated, and the third one is local (cf. Definition 5).

Definition 6 A set I of database tuples from D is a *violation set* for $ic \in IC$ if $I \not\models ic$, and for every $I' \subsetneq I$, $I' \models ic$. $\mathcal{I}(D, ic, t)$ denotes the set of violation sets for ic that contain tuple t . \square

A violation set I for ic is a minimal set of database tuples that simultaneously participate in the violation of ic .

Definition 7 Given an instance D and a set IC of ICs, a *local fix* for $t \in D$, is a tuple t' with: (a) the same values for the rigid attributes as t ; (b) $S(t, t') := \{I \mid \text{there is } ic \in IC, I \in \mathcal{I}(D, ic, t) \text{ and } ((I \setminus \{t\}) \cup \{t'\}) \models ic\} \neq \emptyset$; and (c) there is no tuple t'' that simultaneously satisfies (a), $S(t, t'') = S(t, t')$, and $\Delta(\{t\}, \{t''\}) \leq \Delta(\{t\}, \{t'\})$, where Δ denotes quadratic distance. \square

$S(t, t')$ contains the violation sets that include t and are solved by replacing t' for t . A local fix t' solves some of the violations due to t and minimizes the distance to t .

5.2 Database fix problem as a set cover problem

For a fixed set IC of local denials, we can solve an instance of $DFOP$ by transforming it into an instance of the *Minimum Weighted Set Cover Optimization Problem (MWSCP)*. This problem is $MAXSNP$ -hard [20,21], and its general approximation algorithms are within a logarithmic factor [20,9]. By concentrating on local denials, we will be able to generate a version of the $MWSCP$ that can be approximated within a constant factor (cf. Section 5.3).

Definition 8 For a database D and a set IC of local denials, $\mathcal{G}(D, IC) = (T, H)$ denotes the *conflict hypergraph* for D wrt IC [8], which has in the set T of vertices the database tuples in D , and in the set H of hyperedges, the violation sets for elements $ic \in IC$. \square

Hyperedges in H can be labelled with the corresponding ic , so that different hyperedges may contain the same tuples. Now we build an instance of $MWSCP$.

Definition 9 For a database D and a set IC of local denials, the instance (U, \mathcal{S}, w) for the $MWSCP$, where U is the underlying set, \mathcal{S} is the set collection, and w is the weight function, is given by: (a) $U := H$, the set of hyperedges of $\mathcal{G}(D, IC)$. (b) \mathcal{S} contains the $S(t, t')$, where t' is a local fix for a tuple $t \in D$. (c) $w(S(t, t')) := \Delta(\{t\}, \{t'\})$. \square

It can be proved that the $S(t, t')$ in this construction are non empty, and that \mathcal{S} covers U (cf. Proposition A.2).

If for the instance (U, \mathcal{S}, w) of $MWSCP$ we find a minimum weight cover \mathcal{C} , we could think of constructing a fix by replacing each inconsistent tuple $t \in D$ by a local fix t' with $S(t, t') \in \mathcal{C}$. The problem is that there might be more than one t' and the key dependencies would not be respected. Fortunately, this problem can be circumvented.

Definition 10 Let \mathcal{C} be a cover for instance (U, \mathcal{S}, w) of the $MWSCP$ associated to D, IC . (a) \mathcal{C}^* is obtained from \mathcal{C} as follows: For each tuple t with local fixes t_1, \dots, t_n , $n > 1$, such that $S(t, t_i) \in \mathcal{C}$, replace in \mathcal{C} all the $S(t, t_i)$ by a single $S(t, t^*)$, where t^* is such that $S(t, t^*) = \bigcup_{i=1}^n S(t, t_i)$. (b) $D(\mathcal{C})$ is the database instance obtained from D by replacing t by t' if $S(t, t') \in \mathcal{C}^*$. \square

It holds (cf. Proposition A.3) that such an $S(t, t^*) \in \mathcal{S}$ exists in part (a) of Definition 10. Notice that there, tuple t could have other $S(t, t')$ outside \mathcal{C} . Now we can show that the reduction to $MWSCP$ keeps the value of the objective function. From Propositions A.3 and A.4, we obtain

Proposition 5 If \mathcal{C} is an optimal cover for instance (U, \mathcal{S}, w) of the $MWSCP$ associated to D, IC , then $D(\mathcal{C})$ is an LS-fix of D wrt IC , and $\Delta(D, D(\mathcal{C})) = w(\mathcal{C}) = w(\mathcal{C}^*)$. \square

Proposition 6 For every LS-fix D' of D wrt a set of local denials IC , there exists an optimal cover \mathcal{C} for the associated instance (U, \mathcal{S}, w) of the $MWSCP$, such that $D' = D(\mathcal{C})$.

Proof: We construct the optimal cover. Let $\mathcal{C} = \{S(t, t') \mid t' \in (D' \setminus D)\}$. By definition, $\mathcal{C}' = \mathcal{C}$ and $D(\mathcal{C}) = D'$. We need to prove that \mathcal{C} is an optimal cover. Since D' is consistent, all the violation sets were solved and therefore, \mathcal{C} is a cover. Also, since $\Delta(D, D') = \Delta(D, D(\mathcal{C})) = w$ and $\Delta(D, D')$ is minimum, \mathcal{C} minimizes the weight and therefore is an optimal cover. \square

Proposition 7 The transformation of $DFOP$ into $MWSCP$, and the construction of database instance $D(\mathcal{C})$ from a cover \mathcal{C} for (U, \mathcal{S}, w) can be done in polynomial time in the size of D .

Proof: We have to establish that the transformation of $DFOP$ into $MWSCP$ given above is an L -reduction [21]. So, it remains to verify that the reduction can be done in polynomial time in the size of instance D for $DFP(IC)$, i.e. that \mathcal{G} can be computed in polynomial time in n , the number of tuples in D . Notice

that if m_i the number of database atoms in $ic_i \in IC$, and m the maximum value of m_i there are at most n^{m_i} hyperedges associated to $ic_i \in IC$, each of them having between 1 to m tuples. We can check that the number of sets $S(t, t')$ and their weights are polynomially bounded by the size of D . There is one $S(t, t')$ for each local fix. Each tuple may have no more than $|\mathcal{F}| \times |IC|$ local fixes, where \mathcal{F} is the set of fixable attributes.

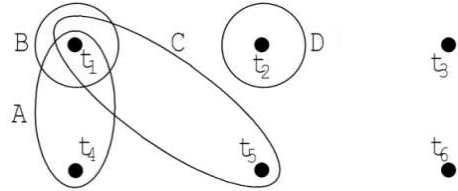
The weight of each $S(t, t')$ is polynomially bounded by the maximum absolute value in an attribute in the database and the maximum absolute value of a constant appearing in IC (by an argument similar to the one given in the proof of Proposition 2).

With respect to $D(\mathcal{C})$, the number of sets in \mathcal{S} is polynomially bounded by the size of D , and since $\mathcal{C} \subseteq \mathcal{S}$, \mathcal{C} is also polynomially bounded by the size of D . To generate \mathcal{C}' it is necessary to search through \mathcal{S} . Finally, in order to replace t in D for each tuple t' such that $S(t, t') \in \mathcal{C}$ we need to search through D . \square

We have established that the transformation of $DFOP$ into $MWSCP$ is an L -reduction [21]. Proposition 7 establishes, in particular, that the number of violation sets $S(t, t')$ is polynomially bounded by the size of the original database D .

Example 7 (example 5 continued) We illustrate the reduction from $DFOP$ to $MWSCP$. The violation sets are $\{t_1, t_4\}$ and $\{t_1, t_5\}$ for IC_1 and $\{t_1\}$ and $\{t_2\}$ for IC_2 . The figure shows the hypergraph. For the $MWSCP$ instance, we need the local fixes. Tuple t_1 has two local fixes $t'_1 = (1, 15, 50)$, that solves the violation set $\{t_1\}$ of IC_2 (hyperedge B), and $t''_1 = (1, 18, 52)$, that solves the violation sets $\{t_1, t_4\}$ and $\{t_1, t_5\}$ of IC_1 , and $\{t_1\}$ of IC_2 (hyperedges A, B, C), with weights 4 and 9, resp. t_2, t_4 and t_5 have one local fix each corresponding to: $(2, 16, 50)$, $(1, CD, 25)$ and $(1, DVD, 25)$, resp. The consistent tuple t_3 has no local fix.

Set	S_1	S_2	S_3	S_4	S_5
Local Fix	t'_1	t''_1	t'_2	t'_4	t'_5
Weight	4	9	1	4	1
Hyperedge A	0	1	0	1	0
Hyperedge B	1	1	0	0	0
Hyperedge C	0	1	0	0	1
Hyperedge D	0	0	1	0	0



The $MWSCP$ instance is shown in the table, where the elements are rows and the sets (e.g. $S_1 = S(t_1, t'_1)$), columns. An entry 1 means that the set contains the corresponding element; and a 0, otherwise. There are two minimal covers, both with weight 10: $\mathcal{C}_1 = \{S_2, S_3\}$ and $\mathcal{C}_2 = \{S_1, S_3, S_4, S_5\}$. $D(\mathcal{C}_1)$ and $D(\mathcal{C}_2)$ are the two fixes for this problem. \square

If we apply the transformation to Example 6, that had non-local set of ICs and no repairs, we will find that instance $D(\mathcal{C})$, for \mathcal{C} a set cover, can be

constructed as above, but it does not satisfy the flexible ICs, because changing inconsistent tuples by their local fixes solves only the initial inconsistencies, but new inconsistencies are introduced.

5.3 Approximation via set cover optimization

Now that we have transformed the database fix problem into a weighted set cover problem, we can apply approximation algorithms for the latter. We know, for example, that using a greedy algorithm, *MWSCP* can be approximated within a factor $\log(N)$, where N is the size of the underlying set U [9]. The approximation algorithm returns not only an approximation \hat{w} to the optimal weight w^o , but also a -non necessarily optimal- cover $\hat{\mathcal{C}}$ for problem (U, \mathcal{S}, w) . As in Definition 10, $\hat{\mathcal{C}}$ can be used to generate via $(\hat{\mathcal{C}})^*$, a fix $D(\hat{\mathcal{C}})$ for D that may not be LS-minimal.

Example 8 (examples 5 and 7 continued) We show how to compute a solution to this particular instance of *DFOP* using the greedy approximation algorithm for *MWSCP* presented in [9]. We start with $\hat{\mathcal{C}} := \emptyset$, $S_i^0 := S_i$; and we add to \mathcal{C} the S_i such that S_i^0 has the maximum *contribution ratio* $|S_i^0|/w(S_i^0)$. The alternatives are $|S_1|/w(S_1) = 1/4$, $|S_2|/w(S_2) = 3/9$, $|S_3|/w(S_3) = 1$, $|S_4|/w(S_4) = 1/4$ and $|S_5|/w(S_5) = 1$. The ratio is maximum for S_3 and S_5 , so we can add any of them to $\hat{\mathcal{C}}$. If we choose the first, we get $\hat{\mathcal{C}} = \{S_3\}$. Now we compute the $S_i^1 := S_i^0 \setminus S_3^0$, and choose again an S_i for $\hat{\mathcal{C}}$ such that S_i^1 maximizes the contribution ratio. Now S_5 is added to $\hat{\mathcal{C}}$, because S_5^1 gives the maximum.

By repeating this process until we get all the elements of U covered, i.e. all the S_i^k become empty at some iteration point k , we finally obtain $\hat{\mathcal{C}} = \{S_3, S_5, S_1, S_4\}$. In this case $\hat{\mathcal{C}}$ is an optimal cover and therefore, $D(\hat{\mathcal{C}})$ is exactly an LS-fix, namely D' in Example 5. Since this is an approximation algorithm, in other examples the cover obtained might not be optimal. \square

Proposition 8 Given database instance D with local ICs IC , the database instance $D(\hat{\mathcal{C}})$ obtained from the approximate cover $\hat{\mathcal{C}}$ is a fix and it holds $\Delta(D, D(\hat{\mathcal{C}})) \leq \log(N) \times \Delta(D, D')$, where D' is any LS-fix of D wrt IC and N is the number of violation sets for D wrt IC .

Proof: Using the same arguments as in the proof of Proposition A.4 we have that since $\hat{\mathcal{C}}$ is a cover then $D(\hat{\mathcal{C}})$ is a fix of D wrt IC . We need to prove that $\Delta(D, D(\hat{\mathcal{C}})) \leq \log(N) \times \Delta(D, D')$. We know that $\Delta(D, D(\hat{\mathcal{C}})) = \sum_{t \in D} \Delta(\{t\}, \{t'\}) = \sum_{S(t,t') \in \hat{\mathcal{C}}^*} w_{S(t,t')}$. As described in definition 10, $\hat{\mathcal{C}}^*$ is obtained from $\hat{\mathcal{C}}$ by replacing, for each t , all the sets $S(t, t_i) \in \hat{\mathcal{C}}$ by a unique set $S(t, t^*)$ such that $S(t, t^*) = \cup_i S(t, t_i)$. Since we are using Euclidian distance to calculate the local fixes, $\Delta(\{t\}, \{t^*\}) \leq \sum_i \Delta(\{t\}, \{t_i\})$. Then, $\Delta(D, D(\hat{\mathcal{C}})) = \sum_{S(t,t') \in \hat{\mathcal{C}}^*} w_{S(t,t')} \leq \sum_{S(t,t') \in \mathcal{C}} w_{S(t,t')} = \hat{w}$. Thus, $\Delta(D, D(\hat{\mathcal{C}})) \leq \hat{w} \leq \log(N) \times w^o = \log(N) \times \Delta(D, D')$, for every LS-fix D' of D . \square

As a consequence, we obtain that, for any set IC of local denials, there is a polynomial time approximation algorithm that solves $DFOP(IC)$ within an $O(\log(N))$ factor, where N is the number of violation sets for D wrt IC . As mentioned before, this number N , the number of hyperedges in \mathcal{G} , is polynomially bounded by $|D|$ (cf. Proposition 7). N may be small if the number of inconsistencies or the number of database atoms in the ICs are small, which is likely the case in real applications.

However, in our case we can get even better approximations via a cover $\hat{\mathcal{C}}$ obtained with an approximation algorithms for the special case of the $MWSCP$ where the number of occurrences of an element of U in elements of \mathcal{S} is bounded by a constant. For this case of the $MWSCP$ there are approximations within a constant factor based on “linear relaxation” [16, Chapter 3]. This is clearly the case in our application, being $m \times |\mathcal{F}| \times |IC|$ a constant bound (independent from $|D|$) on the *frequency* of each element of U (i.e. the number of elements of \mathcal{S} covering the element of U), where m is the maximum number of database atoms in an IC.

Theorem 6 There is an approximation algorithm that, for a given database instance D with local ICs IC , returns a fix $D(\hat{\mathcal{C}})$ such that $\Delta(D, D(\hat{\mathcal{C}})) \leq c \times \Delta(D, D')$, where c is a constant and D' is any LS-fix of D . \square

6 One Atoms Denials and Conjunctive Queries

In this section we concentrate on the common case of *one database atom denials* (1AD), i.e. of the form $\forall \neg(A, B)$, where atom A has a predicate in \mathcal{R} , and B is a conjunction of built-in atoms. They capture range constraints; and census data is usually stored in single relation schemas [11].

For 1ADs, we can identify tractable cases for CQA under LS-fixes by reduction to CQA for (tuple and set-theoretic) repairs of the form introduced in [2] for key constraints. This is because each violation set (cf. Definition 6) contains one tuple, maybe with several local fixes, but all sharing the same key values; and then the problem consists in choosing one from different tuples with the same key values (cf. proof of Theorem 7). The transformation preserves consistent answers to both ground and open queries.

The “classic” -tuple and set oriented- repair problem as introduced in [2] has been studied in detail for key dependencies in [8,12]. In particular, for tractability of CQA in our setting, we can use results and algorithms obtained in [12] for the classical framework.

The *join graph* $\mathcal{G}(Q)$ [12] of a conjunctive query Q is a directed graph, whose vertices are the database atoms in Q . There is an arc from L to L' if $L \neq L'$ and there is a variable w that occurs at the position of a non-key attribute in L and also occurs in L' . Furthermore, there is a self-loop at L if there is a

variable that occurs at the position of a non-key attribute in L , and at least twice in L .

When Q does not have repeated relations symbols, we write $Q \in \mathcal{C}_{Tree}$ if $\mathcal{G}(Q)$ is a forest and every non-key to key join of Q is full i.e. involves the whole key. Classical CQA is tractable for queries in \mathcal{C}_{Tree} [12].

Theorem 7 For a fixed set of 1ADs and queries in \mathcal{C}_{Tree} , consistent query answering under LS-fixes is in $PTIME$.

Proof: Based on the tractability results in [12], it suffices to show that the LS-fixes for a database D are in one-to-one and polynomial time correspondence with the repairs using tuple deletions [2,8] for a database D' wrt a set of key dependencies.

Since we have 1ADs, the violation sets will have a single element, then, for an inconsistent tuple t wrt a constraint $ic \in IC$, it holds $\mathcal{I}(D, ic, t) = \{t\}$. Since all the violation sets are independent, in order to compute an LS-fix for D , we have to generate independently all the local fixes t' for all inconsistent tuples t such that $(\{t\}, ic) \in S(t, t')$, with $ic \in IC$; and then combine them in all possible ways.

Those local fixes can be found by considering all the *candidate* fixes (not necessarily LS-minimal) that can be obtained by combining all the possible borders for each attribute provided by the ICs (cf. Proposition A.1); and then checking which of them satisfy IC , and finally choosing those that minimize $\Delta(\{t\}, \{t'\})$. There are at most $2^{|\mathcal{F}|}$ possible candidate fixes, where \mathcal{F} is the set of fixable attributes.

Let us now define a database D' consisting of the consistent tuples in D together with all the local fixes of the inconsistent tuples. By construction, D and D' share the same keys. Since each inconsistent tuple in D may have more than one local fix, D' may become inconsistent wrt its key constraints. Each repair for D' , obtained by tuple deletions, will choose one local fix for each inconsistent tuple t of D , and therefore will determine an LS-fix of D wrt IC . \square

We may define that a aggregate conjunctive query belongs to \mathcal{C}_{Tree} if its underlying non-aggregate conjunctive query, i.e. its NAM (cf. Section 2) belongs to \mathcal{C}'_{Tree} . Similarly, an aggregate comparison query of the form $q(agg(X)) \wedge agg(X)\theta c$ belongs to this class if the NAM of the conjunctive query that defines $q(agg(X))$ does. Even for 1ADs, with simple comparison aggregate queries with *sum*, tractability is lost under the brave semantics.

Proposition 9 For a fixed set of 1ADs, and for aggregate comparison queries with *sum* that belong to \mathcal{C}_{Tree} or are acyclic, CQA is NP -hard under the brave semantics.

Proof: The NP -complete *PARTITION* problem [13] can be reduced to this case for a fixed set of 1ADs. Let a A be a finite set, whose elements a have integer sizes $s(a)$. We need to determine if there exists a subset S of A , such that $\sum_{a \in S} s(a) = n := (\sum_{a \in A} s(a))/2$.

We use two tables: $Set(Element, Weight)$, with key $\{Element, Weight\}$, containing the tuples $(a, s(a))$ for $a \in A$; and $Selection(Element, X, Y)$, with key $Element$, fixable numerical attributes X, Y (the partition of A) taking values 0 or 1 (which can be specified with 1ADs), and initially containing the tuples $(a, 0, 0)$ for $a \in A$. Finally, we have the 1AD $\forall E, X, Y \neg (Selection(E, X, Y), X < 1, Y < 1)$.

There is a one-to-one correspondence between LS-repairs of the original database and partitions X, Y of A (collecting the elements with value 1 in either X or Y). Then, there is a partition with the desired property iff the comparison query $q(sum(W)) \wedge sum(W) = n$, with $q(sum(W)) \leftarrow Set(E, W), Selection(E, X, Y), X = 1$, has answer *yes* under the brave semantics. The query used in this proof is acyclic and belongs to the class \mathcal{C}_{Tree} . \square

For queries Q returning numerical values, which is common in our framework, it is natural to use the *range semantics* for *CQA*, introduced in [3] for scalar aggregate queries and functional dependencies under classical repairs. Under this semantics, a consistent answer is the pair consisting of the *min-max* and *max-min* answers, i.e. the supremum and the infimum, resp., of the set of answers to Q obtained from LS-fixes. The *CQA* decision problems under the range semantics consist in determining if a numerical query Q , e.g. an aggregate query, has its answer $\leq k_1$ in every fix (*min-max* case), or $\geq k_2$ in every fix (*max-min* case). For all the common scalar aggregate functions (no group-by), we exhibit an aggregate query for which computing the range semantics becomes NP -complete. For example, for *sum* the query computes the cubes of the numbers of vertices of different color in (the database representation of) a graph.

Theorem 8 For each of the aggregation functions *sum*, *count*, *count distinct*, and *average*, there is a fixed set of 1ADs and a fixed aggregate acyclic conjunctive query with one occurrence of the function, such that *CQA* under the range semantics is NP -complete.

Proof: The membership to NP in all the cases is a consequence of the existence of a polynomially bounded and polynomial-time verifiable certificate as established in Theorem 2. Now we consider hardness.

(a) For *sum*: By reduction from a variation of *Independent Set*, for graphs whose vertices have all the same degree. It remains NP -hard as a special case of *Independence Set for Cubic Planar Graphs* [14]. Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with degree 3, and a minimum bound k for the size of the maximal independent set, we create a relation $Ver(V, C_1, C_2)$, where the key

V is a vertex and C_1, C_2 are fixable and may take values 0 or 1, but are all equal to 0 in the initial instance D . This relation is subject to the denial $IC : \forall V, C_1, C_2 \neg (Ver(V, C_1, C_2), C_1 < 1, C_2 < 1)$. D is inconsistent wrt this constraint, and, in any of its LS-fix, each vertex v will have associated a tuple $Ver(v, 1, 0)$ or $Ver(v, 0, 1)$, but not both.

Each LS-fix of the database defines a partition of \mathcal{V} into two subsets: S with $(v, 1, 0)$ and S' with $(v, 0, 1)$, where clearly $S \cup S' = \mathcal{V}$ and $S \cap S' = \emptyset$. Let us define a second relation $Edge(V_1, V_2, W)$, with rigid attributes only, that contains the tuples $(v_1, v_2, 1)$ for $(v_1, v_2) \in \mathcal{E}$ or $(v_2, v_1) \in \mathcal{E}$. Every vertex v appears in each argument in exactly 3 tuples.

Consider the ground aggregate conjunctive query Q :

$$q(sum(W_0)) \leftarrow Ver(V_1, C_{11}, C_{12}), C_{11} = 1, Edge(V_1, V_2, W_0), Ver(V_2, C_{21}, C_{22}), \\ C_{21} = 0, Edge(V_1, V_3, W_1), Ver(V_3, C_{31}, C_{32}), C_{31} = 0, \\ Edge(V_1, V_4, W_2), Ver(V_4, C_{41}, C_{42}), C_{41} = 0.$$

Given an LS-fix as a partition a set of vertices into two subsets S and S' , for each vertex $v \in S$, the body of Q will be satisfied m^3 times, where m is the number of vertices of S' that are adjacent to elements in S . This happens because the query has the last three pairs of predicates $Edge$, Ver , and each predicate can be satisfied m times and they are independent from each other (there are no equality, non-equality or relation predicates connecting them), then whole body of the query will be satisfied m^3 times. Each satisfying assignment of the body of the query brings value 1 to the multiset (W_0) which is under the sum aggregation function. That is, query Q , computes the function from graphs to nonnegative numbers $F(\mathcal{G}, S)$ that gives the sum of cubes of the number of vertices of S' adjacent to vertices in S . More precisely, $F(\mathcal{G}, S) = Q(D')$, for $D' \in Fix(D, IC)$ and $S = \{v \mid Ver(v, 1, 0) \in D'\}$.

Since this function is nonnegative and its value is zero for $S = \emptyset$ and $S = \mathcal{V}$, we have that its minimum value in the fixes is zero. We are interested in the maximum value for Q in $Fix(D, IC)$, i.e. the *min-max answer* introduced in [3].

From Lemma A.4 we have that the answer to query Q is at most $3^3 \times |I|$ with I a maximum independent set. In consequence, the min-max answer for Q is $3^3 \times m$, with m the cardinality of the maximum independent set. Thus, there is an independent set of size at least k iff min-max answer to $Q \geq k \times 3^3$.

(b) For *count*: Basically the same proof as for (a) applies. Only the query has to be changed to:

$$q(count) \leftarrow Ver(V_1, C_{11}, C_{12}), C_{11} = 1, Edge(V_1, V_2, W_0), Ver(V_2, C_{21}, C_{22}), \\ C_{21} = 0, Edge(V_1, V_3, W_1), Ver(V_3, C_{31}, C_{32}), C_{31} = 0, \\ Edge(V_1, V_4, W_2), Ver(V_4, C_{41}, C_{42}), C_{41} = 0, W_0 = 1.$$

(c) For *count distinct*: By reduction from *MAXSAT*. Assume that an instance

for *MAXSAT* is given, consisting of a set U of propositional variables, a collection C of clauses over U and a positive integer k . The question is whether at least k clauses can be satisfied simultaneously, which will get answer *Yes* exactly when a question of the form $countd \leq (k - 1)$, with $countd$ defined by an aggregate query over a database instance (both of them to be constructed below), gets answer *No* under the min-max semantics.

Define a relation $Var(u, c_1, c_2)$, with (rigid) first key attribute, and the second and third fixable (the denial below and the minimality condition will make them take values 0 or 1). The initial database has a tuple $(u, 0, 0)$ for every $u \in U$. Another relation $Clause(u, c, s)$, has no fixable attributes and contains, for every occurrence of variable $u \in U$ in a clause $c \in C$, a tuple (u, c, s) with s an assignment for u satisfying clause c . The IC is $\forall u, c_1, c_2 \neg (Var(u, c_1, c_2), c_1 < 1, c_2 < 1)$. The acyclic query is $q(countd(c)) \leftarrow Var(u, c_1, c_2), Clause(u, c, s), c_1 = s$, where $countd$ denotes the “count distinct” aggregation function. Its answer tells us how many clauses are satisfied in a given LS-fix. The *max* value taken on a LS-fix, i.e. the min-max answer, will be the *max* number of clauses which may be satisfied for *MAXSAT*.

(d) For *average*: By reduction from *3SAT*. We use the same table $Var(u, c_1, c_2)$ and IC as in (a). Now, we encode clauses as tuples in a fixed relation $Clause(val, var_1, val_1, var_2, val_2, var_3, val_3)$, where var_1, var_2, var_3 are the variables in the clause (in any order), val_1, val_2, val_3 all possible combinations of truth assignments to variables (at most eight combinations per clause). And val is the corresponding truth value for the clause (0 or 1). Now, consider the acyclic query

$$q(avg(Val)) \leftarrow Clause(Val, Var_1, Val_1, Var_2, Val_2, Var_3, Val_3), \\ Var(Var_1, Val_1, Val'_1), Var(Var_2, Val_2, Val'_2), Var(Var_3, Val_3, Val'_3).$$

Then value of q is maximum in a LS-fix, taking value 1, i.e. the min-max answer to q is 1, iff the formula satisfiable. \square

Notice that for the four aggregation functions one 1AD suffices, and that for *sum* and *count*, we use a reduction from the *Independent Set Problem* with bounded degree 3 [14]. The general *Independent Set Problem* has bad approximation properties [16, Chapter 10]. The *Bounded Degree Independent Set* has efficient approximations within a constant factor that depends on the degree [15].

Theorem 9 For any set of 1ADs and conjunctive query with *sum* over a non-negative attribute, there is a polynomial time approximation algorithm with a constant factor for *CQA* under *min-max* range semantics. \square

The factor in this theorem depends upon the ICs and the query, but not on the size of the database. The acyclicity of the query is not required. The algorithm is based on a reduction of our problem to satisfying a subsystem

with maximum weight of a system of weighted algebraic equations over the Galois field with two elements $GF[2]$ (a generalization of problems in [13,25]), for which a polynomial time approximation similar to the one for *MAXSAT* can be given [25]. The long proof of this theorem is given in Appendix B.

7 Extensions

7.1 Minimum distribution-variation fixes

Fixing a database under minimization of the square distance to the original database may not preserve the statistical or aggregate properties of the original data, which in some applications could be relevant, like in census data. Given that there may be several LS-fixes, we may prefer those that preserve the data distribution.

Definition 11 Let D and D' be instances over the schema $\Sigma = (\mathcal{U}, \mathcal{R}, \mathcal{B}, \mathcal{A})$, and $R \in \mathcal{R}$. (a) The *distribution distance between D and D' wrt attribute A of R* is $\Delta_d^{R.A}(D, D') = \sum_{a \in \text{Dom}(A)} (\text{count}(a, R.A, D) - \text{count}(a, R.A, D'))^2$, where $\text{count}(a, R.A, D)$ gives the number of occurrences of value a in A of R in instance D .

(b) The *distribution distance $\Delta_d(D, D')$* between D and D' is the maximum of the distribution distances over all relations and their attributes.

(c) Given set of ICs IC , D' is a *minimum distribution variation fix* (MDV-fix) of D wrt IC if D' is an LS-fix that also minimizes $\Delta_d(D, D')$. \square

Example 9 Consider the IC $\forall N, E, S \neg(\text{Emp}(N, E, S), E < 5, S > 5)$, which requires that no employee with experience shorter than 5 years gets a salary higher than 5 thousand. The inconsistent instance $D = \{\text{Emp}(\text{Ann}, 4, 6), \text{Emp}(\text{Bill}, 3, 7), \text{Emp}(\text{Chris}, 2, 2), \text{Emp}(\text{Dan}, 6, 6)\}$ has the following LS-fixes:

$$D_1 = \{\text{Emp}(\text{Ann}, 4, 5), \text{Emp}(\text{Bill}, 3, 5), \text{Emp}(\text{Chris}, 2, 2), \text{Emp}(\text{Dan}, 6, 6)\},$$

$$D_2 = \{\text{Emp}(\text{Ann}, 4, 5), \text{Emp}(\text{Bill}, 5, 7), \text{Emp}(\text{Chris}, 2, 2), \text{Emp}(\text{Dan}, 6, 6)\},$$

$$D_3 = \{\text{Emp}(\text{Ann}, 5, 6), \text{Emp}(\text{Bill}, 3, 5), \text{Emp}(\text{Chris}, 2, 2), \text{Emp}(\text{Dan}, 6, 6)\},$$

$$D_4 = \{\text{Emp}(\text{Ann}, 5, 6), \text{Emp}(\text{Bill}, 5, 7), \text{Emp}(\text{Chris}, 2, 2), \text{Emp}(\text{Dan}, 6, 6)\}.$$

The distance is $\Delta(D, D_i) = 1^2 + 2^2 = 5$ for $i = 1, 2, 3, 4$.

$\Delta_d(D, D_1) = \text{Max}\{\Delta_d^N(D, D_1), \Delta_d^E(D, D_1), \Delta_d^S(D, D_1)\}$, with $\Delta_d^N(D, D_1) = 0$, $\Delta_d^E(D, D_1) = 0$, and $\Delta_d^S(D, D_1) = (\text{count}(5, S, D) - \text{count}(5, S, D_1))^2 + (\text{count}(6, S, D) - \text{count}(6, S, D_1))^2 + (\text{count}(7, S, D) - \text{count}(7, S, D_1))^2 = 2^2 + 1^2 + 1^2 = 6$. Thus, $\Delta_d(D, D_1) = 6$.

$\Delta_d(D, D_2) = \text{Max}\{\Delta_d^N(D, D_2), \Delta_d^E(D, D_2), \Delta_d^S(D, D_2)\}$, with $\Delta_d^N(D, D_2) = 0$, $\Delta_d^E(D, D_2) = (\text{count}(3, S, D) - \text{count}(3, S, D_2))^2 + (\text{count}(5, S, D) - \text{count}(5, S, D_2))^2 = 1^2 + 1^2$, and $\Delta_d^S(D, D_2) = (\text{count}(5, S) - \text{count}(5, S, D_2))^2 + (\text{count}(6, S, D) - \text{count}(6, S, D_2))^2 = 1^2 + 1^2$. Thus, $\Delta_d(D, D_2) = 2$.

This shows that fix D_1 has a bigger impact over the distribution than D_2 . Then, if we want to keep the statistical properties of the database we will prefer D_2 better than D_1 . Analogously, we can obtain $\Delta_d(D, D_3) = 2$ and $\Delta_d(D, D_4) = 6$. Then, $\Delta_d(D, D_2) = \Delta_d(D, D_3) < \Delta_d(D, D_1) = \Delta_d(D, D_4)$, and the MDV-fixes are D_2 and D_3 . \square

From Theorem 3 and the fact that for a database there is an LS-fix if and only if there is a MDV-fix, we obtain

Proposition 10 The problem of existence of MDV-fixes under linear constraints is *NP*-complete. \square

7.2 Other distances

Our results for the square distance rely basically on the additivity of the distance wrt the tuples of the database and the application of a monotonically increasing function over the absolute values of the differences between values for the same attribute. In particular, they apply to the “city” distance (or L_1 distance) given by the sum of those absolute differences. The property of polynomial-time computability of the distance function is also required.

Of course, if instead of the quadratic (or L_2) distance, the L_1 distance is used, we may get for the same database a different set of fixes. The actual approximations obtained in this paper change too. However, the general complexity and approximability results still hold.

The *edit distance* (ED) between two strings is the minimum number of substitutions, deletions and insertions of characters that are needed to transform one string into the other. The *Hamming distance* between two strings of the same length is the number of positions that have different characters. The Hamming distance is an upper bound for the edit distance. For example, $HD(234, 345) = 3$, but $ED(234, 345) = 2$. These distances are used in data editing, but they are more appropriate for strings of characters, and not for numerical data. Actually, our results would not apply to the edit distance or the Hamming distance, because they do not monotonically increase over the absolute value of the difference between two attributes. For example, for the numbers 21, 30 and 31, $ED(21, 30) = 2 > ED(21, 31) = 1$. However, since $|21 - 31| > |21 - 30|$, the edit distance does not monotonically increase over the absolute value of the difference. The same example can be used for the Hamming distance, because $HD(21, 30) = ED(21, 30)$ and $HD(21, 31) = ED(21, 31)$.

Another problem with the edit and hamming distance is that there are too many possible fixes to consider. For example, the database $D = \{P(a, 150)\}$, where the first attribute is the primary key and the second attribute is fixable, is inconsistent wrt $P(x, y) \rightarrow y \geq 200$. There exists only one LS-fix (under

the quadratic distance): $D' = \{P(a, 200)\}$. If instead, we consider the edit distance, there are more than 40 fixes at distance one. For example: $D_1 = \{P(a, 250)\}$, $D_2 = \{P(a, 350)\}$, $D_3 = \{P(a, 1050)\}$, $D_4 = \{P(a, 1530)\}$, $D_5 = \{P(a, 1590)\}$, $D_6 = \{P(a, 6150)\}$, $D_7 = \{P(a, 1507)\}$, etc.

Another possible semantics could accept an epsilon of error in the distance, in such a way that if, for example, the distance of a fix is 5 and the distance to another fix is 5.001, we could take both of them as (minimal) LS-fixes.

8 Conclusions

We have shown that fixing numerical values in databases poses many new computational challenges that had not been addressed before in the context of consistent query answering. These problems are particularly relevant in census like applications, where the problem of *statistical data editing* [6,24] is a common and difficult task. Also our concentration on aggregate queries is particularly relevant for this kind of statistical applications. In this paper we have just started to investigate some of the many problems that appear in this context, and several extensions are in development.

We concentrated on integer numerical values, which provide a useful and challenging domain. Considering real numbers in fixable attributes opens many new issues, requires different approaches; and is a subject of ongoing research. Some of the results presented here carry over to the case of real number. However, apart from the technical problems, the main complication is to come up with a right repair semantics in the presence of real numbers, in particular in comparisons of attributes. For example, if two attributes A, B take the same value, but a constraint prevents this from happening, it is not clear what new values have to be given to them in order to restore consistency. Most likely making them differ by an infinitesimal quantity would not make much sense in most of the applications.

The framework established in this paper could be applied to qualitative attributes with an implicit linear order given by the application; and the numerical distances, like the ones introduced here, could be applied to domains other than numerical if their elements can be naturally mapped to numbers. The result we have presented for fixable attributes that are all equally relevant ($\alpha_A = 1$ in Definitions 1 and 2) should carry over without much difficulty to the general case of arbitrary weighted fixes. We have shown how to extend our approach in order to consider *minimum distribution variation* LS-fixes that keep the overall statistical properties of the database. We have also developed (but not reported here) optimizations of the approximation algorithm presented in Section 5; and its implementation and experiments are ongoing efforts. More research on the impact of aggregation constraints on LS-fixes is needed.

Other open problems refer to cases of polynomial complexity for linear denials with more than one database atom; approximation algorithms for the *DFOP* for non-local cases; and approximations to CQA for other aggregate queries.

For related work, we refer to the literature on consistent query answering (cf. [4] for a survey and references). Papers [26] and [11] are the closest to our work, because changes in attribute values are basic repair actions, but the peculiarities of numerical values and quantitative distances between databases are not investigated. Under the set-theoretic, tuple-based semantics, [8,7,12] report on complexity issues for conjunctive queries, functional dependencies and foreign key constraints. A majority semantics was studied in [19] for database merging. Quite recent papers, but under semantics different than ours, report research on fixing numerical values under aggregation constraints [10]; and heuristic construction of repairs based on attribute values changes [5].

Acknowledgments: Research supported by NSERC, CITO/IBM-CAS Student Internship Program, and EU projects: Sewasie, Knowledge Web, and Interop. L. Bertossi is Faculty Fellow of IBM Center for Advanced Studies (Toronto Lab.).

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. In *Proc. ACM Symposium on Principles of Database Systems (PODS 99)*, 1999, pp. 68-79.
- [3] Arenas, M, Bertossi, L. and Chomicki, J., He, X., Raghavan, V., and Spinrad, J. Scalar aggregation in inconsistent databases. *Theoretical Computer Science*, 2003, 296:405–434.
- [4] Bertossi, L. and Chomicki, J. Query Answering in Inconsistent Databases. In *Logics for Emerging Applications of Databases*, J. Chomicki, G. Saake and R. van der Meyden (eds.), Springer, 2003.
- [5] Bohannon, P., Michael, F., Fan, F. and Rastogi, R. A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification. In *Proc. ACM International Conference on Management of Data (SIGMOD 05)*, 2005, pp. 143-154.
- [6] Boskovitz, A., Goré, R. and Wong, P. Data Editing and Logic. *Proc. Work Session on Statistical Data Editing*, United Nations Statistical Commission and Economic Commission for Europe, Conference of European Statisticians, Ottawa, Canada, May 2005.
<http://www.unece.org/stats/documents/2005.05.sde.htm>

- [7] Cali, A., Lembo, D., Rosati, R. On the Decidability and Complexity of Query Answering over Inconsistent and Incomplete Databases. In *Proc. ACM Symposium on Principles of Database Systems (PODS 03)*, 2003, pp. 260-271.
- [8] Chomicki, J. and Marcinkowski, J. Minimal-Change Integrity Maintenance Using Tuple Deletions. *Information and Computation*, 2005, 197(1-2):90-121.
- [9] Chvatal, V. A Greedy Heuristic for the Set Covering Problem. *Mathematics of Operations Research*, 1979, 4:233-235.
- [10] Flesca, S., Furfaro, F. and Parisi, F. Consistent Query Answers on Numerical Databases under Aggregate Constraints. In *Proc. Tenth International Symposium on Database Programming Languages (DBPL 05)*, 2005.
- [11] 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.
- [12] Fuxman, A. and Miller, R. First-Order Query Rewriting for Inconsistent Databases. In *Proc. International Conference on Database Theory (ICDT 05)*, Springer LNCS 3363, 2004, pp. 337-354.
- [13] Garey, M.R. and Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., 1979.
- [14] Garey, M., Johnson, D. and Stockmeyer, L. Some Simplified NP-Complete Graph Problems. *Theoretical Computer Science*, 1976, 1(3):237-267.
- [15] Halldorsson, M. and Radhakrishnan, J. Greed is Good: Approximating Independent Sets in Sparse and Bounded-degree Graphs. In *Proc. ACM Symposium on Theory of Computing (SToC 94)*, 1994, pp. 439-448.
- [16] Hochbaum, D.(ed.) *Approximation Algorithms for NP-Hard Problems*. PWS, 1997.
- [17] Krentel, M. The Complexity of Optimization Problems. *J. Computer and Systems Sciences*, 1988, 36:490-509.
- [18] Kuper, G., Libkin, L. and Paredaens, J.(eds.) *Constraint Databases*. Springer, 2000.
- [19] Lin, J. and Mendelzon, A.O. Merging Databases under Constraints. *International Journal of Cooperative Information Systems*, 1996, 7(1):55-76.
- [20] Lund, C. and Yannakakis, M. On the Hardness of Approximating Minimization Problems. *J. of the Association for Computing Machinery*, 1994, 45(5):960-981.
- [21] Papadimitriou, Ch. *Computational Complexity*. Addison-Wesley, 1994.
- [22] Papadimitriou, Ch. and Yannakakis, M. Optimization, Approximation and Complexity Classes. *J. Computer and Systems Sciences*, 1991, 43:425-440.

- [23] Ross, K., Srivastava, D., Stuckey, P., and Sudarshan, S.. Foundations of Aggregation Constraints. *Theoretical Computer Science*, 1998, 193(1-2):149–179.
- [24] United Nations Statistical Commission and Economic Commission for Europe. *Glossary of Terms on Statistical Data Editing*. Conference of European Statisticians, Methodological Material, United Nations, Geneva, 2000, 18 pp. <http://www.unece.org/stats/publications/editingglossary.pdf>
- [25] Vazirani, V. *Approximation Algorithms*. Springer, 2001.
- [26] Wijzen, J. Condensed Representation of Database Repairs for Consistent Query Answering. In *Proc. International Conference on Database Theory (ICDT 03)*, Springer LNCS 2572, 2003, pp. 378-393.
- [27] Wijzen, J. Making More Out of an Inconsistent Database. In *Proc. East-European Conference on Advances in Databases and Information Systems (ADBIS 04)*, Springer LNCS 3255, 2004, pp. 291-305.

Appendix A: Some technical results

Those auxiliary technical results and definitions that are stated in this appendix, but not in the main body of the paper, are numbered in the form **A.n**, e.g. Lemma A.1.

First we establish that if the set of constraints is consistent and there is at least one fixable attribute in each integrity constraint, and a tuple is involved in an inconsistency, then there always exists a local fix (see Definition 7) for it. The condition of locality (cf. Definition 5) implies the hypothesis of the following lemma.

Lemma A.1. Consider a database D and a consistent set of linear denial constraints IC , where each constraint contains at least one built-in involving a flexible constraint and there are equalities or joins only between rigid attributes. For every tuple t with at least one fixable attribute and at least one ic in IC such that $\mathcal{I}(D, ic, t) \neq \emptyset$, there exists a local fix t' (cf. Definition 7). \square

Proof: Each constraint $ic \in IC$ has the form $\forall \bar{x} \neg (P_1(\bar{x}), \dots, P_n(\bar{x}), A_i < c_i, A_j \geq c_j, A_k = c_k, A_l \neq c_l, \dots)$ and can be rewritten as a FO clause containing only $<$, $>$ and $=$ as follows

$$\forall \bar{x} (\neg P_1(\bar{x}) \vee \dots \vee \neg P_n(\bar{x}) \vee A_i \geq c_i \vee A_j < c_j \vee A_k < c_k \vee A_k > c_k \vee A_l = c_l \vee \dots) \quad (1)$$

Since the repairs are obtained by attributes changes, this formula shows that the only way we have to solve an inconsistency is by fixing at least one of the values of a fixable attribute. Let ic be a constraint in IC such that $\mathcal{I}(D, ic, t) \neq \emptyset$ and I be a violation set $I \in \mathcal{I}(D, ic, t)$. Now, since $ic \in IC$, ic is logically

consistent. Then, for each fixable attribute A in ic , we are able to derive an interval $[c_l, c_u]$, such that if we pick up the value of A from it, we would restore the consistency of I . For example, if we have a constraint in form of equation (1), with $A \leq 5$, then, if we want to restore consistency by modifying A , we would need to have $A \in (-\infty, 5]$. If the constraint had also $A \geq 1$, the interval would be $[1, 5]$. Since t has at least one fixable attribute and each fixable attribute has an interval, it is always possible to adjust the value of that fixable attribute to a value in the interval $[c_l, c_u]$ and restore consistency. By finding the adjustment that minimizes the distance from the original tuple we have find a local fix for the tuple t . \square

The *borders* of an attribute in an extended linear denial correspond to the surfaces of the semi-spaces determined by the built-in atoms in it.

Proposition A.1. Given a database D and a set of linear denials IC , where equalities and joins can only exist between rigid attributes, the value in every fixable attribute in a local fix t' of a tuple $t \in D$ corresponds to the original value in t or to a border of a constraint in IC . Furthermore, the values in every attributes of a tuple $t' \in D'$ corresponds to the original value of the attribute in the tuple in D or to a border of a constraint in IC . \square

Proof: First we will replace in all the constraints $X \leq c$ by $X < (c + 1)$, $X \geq c$ by $X > (c - 1)$ and $X = c$ by $(X > (c - 1) \wedge X < (c + 1))$. We can do this because we are dealing with integer values. Then, a constraint ic will have the form $\forall \bar{x} \neg (P_1(\bar{x}), \dots, P_n(\bar{x}), A_i < c_i, A_j > c_j, A_k \neq c_k, \dots)$, and can be rewritten as

$$\forall \bar{x} (\neg P_1(\bar{x}) \vee \dots \vee \neg P_n(\bar{x}) \vee A_i \geq c_i \vee A_j \leq c_j \vee A_k = c_k \vee \dots). \quad (2)$$

Since the repairs are obtained by attribute changes, this formula shows that the only way we have of solving an inconsistency is by fixing at least one of the values of a fixable attribute. This would imply changing the value of a fixable attribute A_i to something equal or greater than c_i , to change the value of a fixable attribute A_j to a value equal or smaller than c_j or to change the value of attribute A_k to c_k .

If D is consistent wrt IC , then there is a unique LS-fix $D' = D$, and all the values are the same as the original ones and therefore, the proposition holds. If D is inconsistent wrt IC , then there exists a tuple t with at least one fixable attribute and a set $IC_t \subseteq IC$, such that, for every $ic \in IC_t$, it holds $\mathcal{I}(D, ic, t) \neq \emptyset$. If IC_t is an inconsistent set of constraints, then there exists no local fix and the proposition holds. If IC_t is consistent, but there is at least one constraint with no fixable attributes involved, then, since it is not possible to modify any attribute in order to satisfy the constraint, there is no local fix and the proposition holds.

So it is only missing to prove the proposition for IC_t consistent and with at least one fixable attributes for each ic in IC_t . From Lemma A.1, we know

that there exists a local fix for t . Also, since IC_t is consistent, using the same arguments as in proof of Lemma A.1, it is possible to define for each fixable attribute A an interval, such that, if the value of A is in it, we can restore the consistency of the violation sets for constraints in IC_t involving t . Then, we need to prove that, if a value of an attribute, say A , of a local fix t' of t is different from the one in t , then the value corresponds to one of the closed limits of the interval for A .

Let us assume that an attribute A is restricted by the constraints to an interval $[c_l, c_u]$, and that the local fix t' takes for attribute A a value strictly smaller than c_u and strictly greater than c_l . Without loss of generality, we will assume that the value of attribute A in t is bigger than c_u . Let t'' be a tuple with the same values as t' , except that attribute A is set to c_u . t'' will have the same values in the rigid attributes as t , and also $S(t, t') = S(t, t'')$, because the value of A in t'' is still in the interval. We also have that $\Delta(\{t\}, \{t''\}) \leq \Delta(\{t\}, \{t'\})$. This implies that t' is not a local fix and we have a contradiction.

For the second part of the proposition, the proof of the first part can be easily extended to prove that the values in D' correspond to borders of a constraint in IC , because the LS-fixes are combination of local fixes. \square

Lemma A.2. Given a database D and a set of consistent local denials IC , there will always exist an LS-fix D' of D wrt IC . \square

Proof: As shown in proof of Lemma A.1, for every fixable attribute in F , it is possible to define, using the ICs in IC , an interval $[c_l, c_u]$, such that, if the value of attribute A is in that interval, there is no constraint $ic \in IC$ with a built-in involving A with $\mathcal{I}(D, ic, t) \neq \emptyset$. Let D'' be a database constructed in the following way: for every tuple $t \in D$ such that the value of a fixable attribute does not belong to its interval, replace its value by any value in the interval. Clearly D'' will be a fix, but will not necessarily be an LS-fix. By Proposition 1, we know that there exists an LS-fix D' for D wrt IC . \square

Definition A.1. Given a database D and a set of ICs IC , a local fix t' for a tuple t *does not generate new violations* if $\bigcup_{ic \in IC} (\bigcup_{l \in D'} \mathcal{I}(D', ic, l) \setminus \bigcup_{l \in D} \mathcal{I}(D, ic, l)) = \emptyset$ for $D' = (D \setminus \{t\}) \cup \{t'\}$. \square

Lemma A.3. For a set IC of local denials, if t' is a local fix of a tuple t , then t' does not generate new violations in database D wrt IC . Furthermore, this holds also for a “relaxed” local fix t' whose distance to t is not necessarily minimal. \square

Proof: Tuple t' can only differ from t in the value of fixable attributes. Let us assume that one of the modified values is for attribute A . Since we have local constraints, attribute A can only be in the constraints related either to $<$ and \leq or to $>$ and \geq , but not both. Without loss of generality, we will assume that the constraint is written as in equation (1) and that A is related only to

$>$ and \geq . Since t' is a local fix, $S(t, t')$ is not empty, and there is a set IC_t of constraints for which t' solves the inconsistency in which t' was involved. There is an interval $[c_l, +\infty)$ for A that can be obtained by the limits given in IC_t that show the possible values for A that would force the satisfaction of the constraints in IC_t that have attribute A in an inequality. This shows that the value of attribute A in t' is bigger than the value of A in t .

For $D' = (D \setminus \{t\}) \cup \{t'\}$ we need to prove that $\bigcup_{ic \in IC} (\bigcup_{l \in D'} \mathcal{I}(D', ic, l) \setminus \bigcup_{l \in D} \mathcal{I}(D, ic, l)) = \emptyset$. By contradiction, let us assume that for a constraint $ic \in IC$, there exists a violation set I such that $I \in \bigcup_{l \in D'} \mathcal{I}(D', ic, l)$ and $I \notin \bigcup_{l \in D} \mathcal{I}(D, ic, l)$. There are two cases to consider (with (I, ic) we indicate that I is a violation set for IC ic):

- $(I, ic) \in S(t, t')$. Then $I \in \mathcal{I}(D, ic, t)$, but since we want an $I \notin \bigcup_{l \in D} \mathcal{I}(D, ic, l)$, this is not possible.
- $(I, ic) \notin S(t, t')$. Then we have two possibilities $I \notin \mathcal{I}(D, ic, t)$ or $((I \setminus \{t\}) \cup \{t'\}) \not\models ic$.
 - Let us consider first that $I \notin \mathcal{I}(D, ic, t)$. We have that $I \in \bigcup_{l \in D'} \mathcal{I}(D', ic, l)$ and, since t' is the only difference between D and D' , it holds $I \in \mathcal{I}(D', ic, t')$. Since all the constraints can only have attribute A with $>$ or \geq , we know that in particular ic does. Since $I \notin \mathcal{I}(D, ic, t)$, we know that A satisfied the condition in ic , and since we know that t' has a bigger value than in t , it is not possible to generate an inconsistency in D' . We have a contradiction.
 - Let us consider $((I \setminus \{t\}) \cup \{t'\}) \not\models ic$. In this case, $I \in \mathcal{I}(D', ic, t')$. From our assumption, it follows $I \notin \bigcup_{l \in D} \mathcal{I}(D, ic, l)$. This is the same situation analyzed in the previous item.

In all the cases we have reached a contradiction and therefore, the proposition is proved. Since we never used the property of minimal distance between t' and t , the second part of the lemma is also established. \square

Proposition A.2. For local denials it always exists an LS-fix for a database D ; and for every LS-fix D' , $D' \setminus D$ is a set of local fixes. Furthermore, for each violation set (I, ic) , there is a tuple $t \in I$ and a local fix t' for t , such that $(I, ic) \in S(t, t')$. \square

Proof: Since each attribute A can only be associated to $<$ or $>$ built-ins, but not both, it is clear that set of local denials is always consistent. By Lemma A.2, there always exists an LS-fix D' . Now we need to prove that $D' \setminus D$ is a set of local fixes. By contradiction, assume that $t' \in (D' \setminus D)$ is not a local fix of the tuple t . This can happen in the following situations:

- t was consistent. From Lemma A.3 we know that no new inconsistencies can be added by the modifications done to the other tuples, and therefore, t is not related to any inconsistency. Then $D^* = D' \setminus \{t'\} \cup \{t\}$ is also consistent, and $\Delta(D, D^*) < \Delta(D, D')$. But D' is an LS-fix, so this is not possible.

- t is involved at least in one violation set. If $S(t, t') = \emptyset$, then t' is not solving any violation set, and therefore, $D^* = D' \setminus \{t'\} \cup \{t\}$ is also consistent, and $\Delta(D, D^*) < \Delta(D, D')$. But D' is an LS-fix, so this is not possible. Now, if $S(t, t') \neq \emptyset$, from Lemma A.2, considering $D = \{t\}$ and $IC = \{ic | (I, ic) \in S(t, t')\}$, there exists an LS-fix D' of D , i.e. there exists a local fix t'' , such that $S(t, t'') = S(t, t')$. Since t'' is a local fix, we know that $\Delta(\{t\}, \{t''\}) \leq \Delta(\{t\}, \{t'\})$. They cannot be equal, which would imply that t' is a local fix and it is not. Then $D^* = D' \setminus \{t'\} \cup \{t\}$ is also consistent, and $\Delta(D, D^*) < \Delta(D, D')$. Again, this is not possible because D' is an LS-fix.

The second part of the proposition can be proved using Lemma A.2 and considering a database D equal to I and the set of constraints $IC = \{ic\}$. \square

Proposition A.3. For a database D and a set of local denial constraints IC :

- (1) For a set of local fixes $\{t_1, \dots, t_n\}$ of a tuple t , there always exists a local fix t^* such that $S(t, t^*) = \bigcup_{i=1}^n S(t, t_i)$.
- (2) For local fixes t' , t'' and t''' of a tuple t , with $S(t, t''') = S(t, t') \cup S(t, t'')$, it holds that $\Delta(\{t\}, \{t'''\}) \leq \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$. \square

Proof: First we prove (1). Let $IC_t = \{ic | \mathcal{I}(D, ic, t) \neq \emptyset \text{ and } IC_S(t, t') = \{ic | (I, ic) \in S(t, t')\}$. From Lemma A.2, considering $D = \{t\}$ and IC any subset of IC_t , there always exists an LS-fix D' of D . This LS-fix is a local fix of tuple t , with $IC_S(t, t') = IC$. Since we can find a local fix for any $IC \subseteq S_t$, the lemma is obtained.

Now we prove (2). If the fixable attributes that were modified in t' and t'' are disjoint, and t''' is obtained by combining the changes, we get $\Delta(\{t\}, \{t'''\}) = \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$. Now, we consider that t' and t'' have at least one fixable attribute, say A , that is modified by both local fixes. In this case t''' , will have a value in A that solves the inconsistencies solved by t' and t'' . This value will in fact correspond to the value of A in t' or t'' and therefore, we have that $\Delta(\{t\}, \{t'''\}) < \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$. Let M be the set of attributes that are modified both by t' and t'' , we can express the relation as follows: $\Delta(\{t\}, \{t'''\}) = \sum_{A \in \mathcal{F}} (\pi_A(t) - \pi_A(t'''))^2 = \sum_{A \in \mathcal{F}} (\pi_A(t) - \pi_A(t'))^2 + \sum_{A \in \mathcal{F}} (\pi_A(t) - \pi_A(t''))^2 - \sum_{A \in \mathcal{M}} \text{Min}\{(\pi_A(t) - \pi_A(t'))^2, (\pi_A(t) - \pi_A(t''))^2\}$. \square

Proposition A.4. If an optimal cover \mathcal{C} for the instance (U, \mathcal{S}) of *MWSCP* has more than one $S(t, t')$ for a tuple t , then there exists another optimal cover \mathcal{C}' for (U, \mathcal{S}) with the same total weight as \mathcal{C} , but with only one t' such that $S(t, t') \in \mathcal{C}$. Furthermore, $D(\mathcal{C})$ is an LS-fix of D wrt IC with $\Delta(D, D(\mathcal{C}))$ equal to the total weight of the cover \mathcal{C} . \square

Proof: To prove the first part, let us assume that $S(t, t'), S(t, t'') \in \mathcal{C}$. By Proposition A.3, there exists an $S(t, t''') \in \mathcal{S}$ such that $S(t, t''') = S(t, t') \cup S(t, t'')$, i.e that covers the same elements as $S(t, t')$ and $S(t, t'')$. By Proposition A.3, $\Delta(\{t\}, \{t'''\}) \leq \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$, and therefore, the weight of $S(t, t''')$ is smaller than or equal to the sum of the weights of the two original

sets. If $\Delta(\{t\}, \{t'''\}) < \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$, we would have that \mathcal{C} is not an optimal solution; so this is not possible. Then, $\Delta(\{t\}, \{t'''\}) = \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$. Thus, if we define $\mathcal{C}' = (\mathcal{C} \setminus \{S(t, t'), S(t, t'')\}) \cup \{S(t, t''')\}$, we will cover all the elements and we will have the same optimal weight.

Now we need to prove that $D(\mathcal{C})$ is an LS-fix. $D(\mathcal{C})$ is obtained by first computing \mathcal{C}' , and therefore, we have an optimal cover with at most one $S(t, t')$ for each tuple t . Then, $D(\mathcal{C})$ is obtained by replacing t by t' for each $S(t, t') \in \mathcal{C}$. It is direct that $D(\mathcal{C})$ has the same schema as D and that it satisfies the key constraints. Now, since \mathcal{C}' covers all the elements, all the inconsistencies in D are solved in $D(\mathcal{C})$. By Lemma A.3, the local fixes t' do not add any new violations, and therefore, $D(\mathcal{C}) \models IC$ and $D(\mathcal{C})$ is a fix. We still have to prove that $D(\mathcal{C})$ minimizes the distance from D . Clearly, $\Delta(D, D(\mathcal{C})) = \sum_{t \in D} \Delta(\{t\}, \{t'\}) = \sum_{S(t, t') \in \mathcal{C}'} w_{S(t, t')} = \sum_{S(t, t') \in \mathcal{C}} w_{S(t, t')} = w$. So, since the optimal solution minimizes w , $\Delta(D, D(\mathcal{C}))$ is minimum and $D(\mathcal{C})$ is an LS-fix. \square

For the proof of Theorem 8 we need some preliminaries. Let us define a function F , with domain $\mathcal{G} \times S$, where $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is a graph and S is a subset of vertices of graph \mathcal{G} , and the range is the set of non-negative integers. The function is defined as the summation over all the vertices $v \in S$, of cubes of the number of edges connecting v to vertices in the complement of S .

Lemma A.4. Given a fixed regular undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of degree 3 (then, by regularity, all the vertices have degree 3), and for sets of vertices $S \subseteq \mathcal{V}$, let us define a function F by

- $F^l(S, v) = |T(S, v)|^3$ where $T(S, v) = \begin{cases} \{v' | v' \in (\mathcal{V} \setminus S) \wedge (v, v') \in \mathcal{E}\}, & v \in S \\ \emptyset, & v \notin S \end{cases}$
- $F(\mathcal{G}, S) = \sum_{v \in S} F^l(S, v)$.

The maximum value of $F(\mathcal{G}, S)$ over all possible sets $S \subseteq V$ is $(3^3 \times |I|)$, for I a maximal (wrt set inclusion) independent set. \square

Proof: Let us first assume that S is an independent set, not necessarily maximal. In this case the value $F(\mathcal{G}, S)$ is $3^3 \times |S|$, because each element $v \in S$ is connected to three vertices in $\mathcal{V} \setminus S$. Then, among independent sets, the maximum value for $F(\mathcal{G}, S)$ is $3^3 \times m$, where m is the maximum cardinality of an independent set.

Let $\mathcal{G}[S] = \mathcal{G}(S, \mathcal{E}_S)$, where \mathcal{E}_S are all the edges $(v, v') \in \mathcal{E}$ such that $v, v' \in S$. Now, if S is not an independent set, there exists a maximum independent set I_S of $\mathcal{G}[S]$. Every $v \in (\mathcal{V} \setminus S)$ is adjacent to at least one vertex in I_S , otherwise $I_S \cup \{v\}$ would be an independent set contained in S and with more vertices than I_S , contradicting our choice of I_S . Now let us define $F_{ext}(S, v) = (F^l(S, v) + \sum_{(v, v') \in \mathcal{E}} F^l(S, v'))$. Since every edge $v' \in (S \setminus I_S)$ is adjacent to I_S , it is easy to see that:

$$F(\mathcal{G}, S) \leq \sum_{v \in I} F_{ext}(S, v). \quad (3)$$

We want to prove that $F(\mathcal{G}, S) \leq F(\mathcal{G}, I_S)$. This, combined with equation (3), shows that it suffices to prove that $\sum_{v \in I_S} F_{ext}(S, v) \leq F(\mathcal{G}, I_S)$. Since $F(\mathcal{G}, I_S) = \sum_{v \in I_S} F^l(I_S, v)$, we need to prove that $\sum_{v \in I_S} F_{ext}(S, v) \leq \sum_{v \in I_S} F^l(I_S, v)$. It is sufficient to prove that $F_{ext}(S, v) \leq F^l(I_S, v)$ is true for every $v \in I_S$. For $v \in I_S$ and $S' = (S \setminus I_S)$, we have the following cases:

- (1) If v is adjacent to one vertex in S' , then $F_{ext}(S, v) \leq 2^3 + 2^3$, and $F^l(I_S, v) = 3^3$. In consequence, $F_{ext}(S, v) \leq (F^l(I_S, v) - 11)$.
- (2) If v is adjacent to two vertices in S' , analogously to (1), we get $F_{ext}(S, v) \leq (F^l(I_S, v) - 10)$.
- (3) If v is adjacent to three vertices in S' , analogously to (1), we get $F_{ext}(S, v) \leq (F^l(I_S, v) - 3)$.

Then, we have proved that $F_{ext}(S, v) \leq F^l(I_S, v)$, and therefore, that $F(\mathcal{G}, S) \leq F(\mathcal{G}, I_S)$. We also know that, since I_S is an independent set, that $F(\mathcal{G}, S) \leq F(\mathcal{G}, I_S) \leq 3^3 \times m$. \square

Appendix B: Approximation algorithm for *sum* (proof of Thm. 9)

First we reduce *CQA* under range semantics for aggregate queries with *sum* to *RWAE2*, a restricted weighted version of the problem of solving algebraic equations over $GF[2]$, the field with two elements. Next, we prove that such an algebraic problem can be solved within constant approximation factor.

(A). *Reduction to RWAE2.* In order to define polynomial equations, we need variables. We introduce a set \mathcal{V} of variables $X_{k,i}^R$, taking values in $GF[2]$, for every tuple t_i in an LS-fix corresponding to a tuple t (a ground database atom in the database) with key k in a relation R in the original database, i.e. t_i belongs to some LS-fix and t_i t share the key values k . For example if the tuple t is consistent or admits only one local fix (one attribute can be changed and in only one way), only one variable is introduced due to t . Denote with $bag(t)$ the set of variables introduced due to a same initial tuple t .

Consider a conjunctive query $Q(sum(z)) \leftarrow R_1(\bar{x}), \dots, R_m(\bar{x})$. Throughout the a proof ψ is the body of the query as a conjunction of atoms, m is the number of database predicates in ψ , n is the number of tuples in the database, k is the maximal number of attribute comparisons in the ICs (and the maximal number of fixes of a given tuple).

We may consider all the possible assignments β from database atoms in the query to grounds tuples in fixes that satisfy ψ . The number of assignments is polynomial in the size of the database, actually $\leq n^m$. Notice that the number of LS-fixes of a database may be exponential, but the number of local fixes of each original tuple is restricted by the number of attributes of the tuple. So, the number of all possible LS-fixes of tuples is polynomial in the size of the original database (even linear). Here we are using the fact that we have 1ADs.

Now we build a system \mathcal{E} of weighted algebraic equations. Each such assignment β is associated with a combination of tuples $t_{k_1, i_1}^{R_1}, \dots, t_{k_m, i_m}^{R_m}$ satisfying ψ . For each combination, we put the following equation E^β over $GF[2]$ into \mathcal{E} :

$$\overbrace{X_{k_1, i_1}^{R_1} \cdots X_{k_m, i_m}^{R_m}}^{\text{selected}} \cdot \underbrace{\prod_{i \neq i_1} (1 - X_{k_1, i}^{R_1}) \cdots \prod_{i \neq i_m} (1 - X_{k_m, i}^{R_m})}_{\text{non-selected}} = 1. \quad (4)$$

The first product in (4), before the first \prod , contains the variables corresponding to the tuples selected by β . The rest of the product contains variables for those tuples that were not selected, i.e. if t_1 appears in the first product, with $t_1 \in \text{bag}(t)$, and $t_2 \in \text{bag}(t)$, with $t_1 \neq t_2$, then the variable X_2 corresponding to t_2 appears as $(1 - X_2)$ in the second part of the product. This captures the restriction that no two different tuples from the same bag can be used (because they share the key values). For each combination β of tuples in LS-fixes, there is no more than one equation, which in turn has a polynomial number of factors.

Equation (4) gets weight $w(E^\beta)$, that is equal to the value of aggregation attribute z in β .

In this way we have an instance of the *RWAE2* problem, that requires to find the maximum weight for a subsystem of \mathcal{E} that can be (simultaneously) satisfied in $GF[2]$. Here, the weight of the subsystem is the sum of the weights of the individual equations. Of course, this problem also has a version as a decision problem, so as *CQA* under the range semantics.

Claim: The maximal weight of a satisfied subsystem of \mathcal{E} is the same as the maximal value of $Q(\text{sum}z)$ over all possible LS-fixes of D .

(\geq) Assume that query Q takes a maximum value over all possible LS-fixes of D on an LS-fix D' . Under 1ADs, an LS-fix D' is a union of local fixes, with one local fix selected for every original tuple. Consider an assignment A defined on \mathcal{V} that maps variables corresponding a selected local fix to 1 and all other variables to 0.

Consider all sets of local fixes which simultaneously satisfy ψ . If local fixes t_1, \dots, t_m satisfy ψ , then there exists exactly one equation e for that given set of local fixes. The equation e will be satisfied because variables corresponding to the selected local fixes have value 1, and “non-selected” variables have value 0. So, for every set of local fixes satisfying the query body, there would be a satisfied equation with weight equal to the value of aggregated attribute. This means that a solution to the algebraic equation problem is bigger or equal to the maximal query answer (*min-max* answer).

(\leq) Consider an assignment A which is a solution of the algebraic equation problem. It maps elements of \mathcal{V} to $\{0, 1\}$, in such a way that the the weight

of satisfied equations of \mathcal{E} is maximum over all possible assignment for \mathcal{V} .

First we prove that if there exists a bag B such that more than one of its variables is mapped to 1, then there exists an assignment A' with the same weight of satisfied equations of \mathcal{E} as A , but B contains no more than one variable mapped to 1.

Assume that for a bag B , more than two variables (let us say X_i, X_j) are mapped to 1. This means that every equation which contains variables from B will be unsatisfied, because it contains either $(1 - X_i)$ or $(1 - X_j)$ as factors in the equation. If we change a value of one of the variables (say X_i) to 0, then no satisfied equation become unsatisfied, because satisfied equations do not contain X_i . No unsatisfied equation becomes satisfied, due to the assumption of maximality of the weight of the satisfied subset of E for A .

In a second step, we prove that if A is a maximal assignment and there exists a bag B such that all of its variables are mapped to 0, then there exists an assignment A' that satisfies the same subset of \mathcal{E} as A , but at least one variable from that B is mapped to 1.

If all variables from a bag B are mapped to 0, then all equations which contain variables from B are unsatisfied. If we change a value of one variable to 1, then no satisfied equation becomes unsatisfied, because all satisfied equations do not contain variables from B . No unsatisfied equation becomes satisfied due to the maximality assumption about the weight of the satisfied equation for A . Taking step by step all the bags from \mathcal{V} , for a given maximum assignment A , we produce an assignment A' , which has exactly one variable from each bag mapped to 1.

Now, we construct a database D' which is the set of local fixes corresponding to variables mapped to 1. It is obviously a LS-fix, and $w(E(A)) \leq Q(D')$.

(B). *A deterministic approximation algorithm for RWAE2.* The construction and approximation factor obtained are similar those in the approximation of *MAXSAT* [25,21]. In two steps, first a randomized algorithm is produced, that is next de-randomized.

(B1). *Randomized approximation algorithm.* Assume that from each bag we select one variable with probability $1/k$, where k is the number of variables in the bag. We map the selected variable to 1 and all other variables in the bag to 0. For each equation e , random variable W_e denotes the weight contributed by e to the total weight W . Thus, $W = \sum_{e \in \mathcal{E}} W_e$ and $\mathbf{E}[W_e] = w_e \cdot \mathbf{Pr}[e \text{ is satisfied}]$, where \mathbf{E} is the mathematical expectation and \mathbf{Pr} is the probability.

If the query contains m predicates, then each equation contains no more than m variables from different bags (never two different variables from the same

bag), then $\mathbf{E}(W_e) \geq k^{-m}w_e$. Now, by linearity of expectation,

$$\mathbf{E}[W] = \sum_{e \in \mathcal{E}} \mathbf{E}[W_e] \geq k^{-m} \sum_{e \in \mathcal{E}} w_e \geq k^{-m} \cdot OPT.$$

(B2). *De-randomization via conditional expectation.* We first establish

Claim: The *RWAE2* problem is self-reducible [25, chapter A.5].

In fact, assume A' is a partial assignment from \mathcal{V} , such that the variables X_1, \dots, X_i are mapped to $\{0, 1\}$. Let \mathcal{E}^s be the set of equations satisfied by A' with total weight $W[\mathcal{E}^s]$, and \mathcal{E}^u is the set of equations which cannot be satisfied under A' . Let E'' be a set of equations from $\mathcal{E} \setminus (\mathcal{E}^s \cup \mathcal{E}^u)$, such that the variables from X_1, \dots, X_i are replaced by their values. By additivity of the weight function and the independence of the variables, the maximal weight of satisfied equations under an assignment which extends A' is $W[\mathcal{E}^s] + \max W[E'']$, where $W[E'']$ is a solution of the *RWAE2* problem restricted to E'' . It is good enough to consider the self-reducibility trees T such only one variable from each bag gets value 1 along any path in the tree. This establishes our claim.

Assume that a self-reducibility tree T is given, with each node in it corresponding to a step of the self-reduction. Each node v of T is labelled with $X_1 = a_1, \dots, X_i = a_i$, a partial assignment of values to variables $X_1, \dots, X_i \in \mathcal{V}$ associated to the step for v of the self-reduction. Since this is a partial assignment, some of the equations in \mathcal{E} become immediately satisfied, other unsatisfied, and some other undetermined. The latter become a set of equations E' associated to v on variables $\mathcal{V} \setminus \{X_1, \dots, X_i\}$, obtained from \mathcal{E} by giving to the variables X_1, \dots, X_i their values a_1, \dots, a_i . By construction, these equations inherit the weight of the corresponding equations in \mathcal{E} .

For example, if the set of equations consists of: (1), $yp(1-x) = 1$, (2) $2xz(1-y) = 1$, (3) $3xw(1-y) = 1$, with variables x, y, z, p, w , and the partial assignment, at some step of self-reduction for v is $x = 1, y = 0, w = 1$, then equation (1) becomes unsatisfiable, (2) is not satisfied but possibly satisfiable with an appropriate value for z ; and (3) satisfied. So, E' contains equation (2), but with x, y replaced by their values 1, 0, resp.

The conditional expectation of any node v in T can be computed via its sets of equations E' we just described. Clearly, the expected weight of satisfied equations of E' under a random assignment of values in $GF[2]$ to $\mathcal{V} \setminus X_1, \dots, X_i$ can be computed in polynomial time. Adding to this the weight of the equations in \mathcal{E} already satisfied by the partial assignment $X_1 = a_1, \dots, X_i = a_i$ gives the conditional expectation.

Then we compute in polynomial time a path from the root to a leaf, such that the conditional expectation of each node on this path is $\geq \mathbf{E}[W]$. This can be done as in the construction in [25, theorem 16.4].

In consequence, we can find a deterministic approximate solution to the *RWAE2* problem in polynomial time. It approximates the optimum solution with a factor greater than k^{-m} . It means that we can approximate the maximal value of the aggregate conjunctive query within a factor k^{-m} , which depends on the ICs and the query, but not on the size of the database. This ends the proof.

For example, the query with *sum* used in the proof of the *NP*-hardness in Theorem 8 has $m = 4, k = 2$, then it can be approximated within the factor 2^{-4} .

Appendix C: An example for Thm. 1

Consider the diophantine equation $2x^3y^2 + 3xy + 105 = x^2y^3 + y^2$. Each term t in it will be represented by a relation $R(t)$ with 8 attributes taking values in \mathbb{N} : three, X_1, X_2, X_3 , for the maximum exponent of x , three, Y_1, Y_2, Y_3 , for the maximum exponent of y , one, C , for the constant terms, plus a last one, K , for a key. Value 0 for a non-key attribute indicates that the term appears in t , otherwise it gets value 1. We introduce as many tuples in $R(t)$ as the coefficient of the term; they differ only in the key value. We will see that only the 0 values will be subject to fixes. These are the relations and their ICs:

$R(2x^3y^2)$	X_1	X_2	X_3	Y_1	Y_2	Y_3	C	K
	0	0	0	1	0	0	1	1
	0	0	0	1	0	0	1	2

For this table we have the following set, $IC(2x^3y^2)$, of ICs:

$\forall x_1 \cdots x_8 \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge x_1 \neq x_2), \forall x_1 \cdots x_8 \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge x_2 \neq x_3), \forall x_1 \cdots x_8 \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge x_5 \neq x_6), \forall x_1 \cdots x_8 \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge x_4 \neq 1), \forall x_1 \cdots x_{16} \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge R(2x^3y^2)(x_9, \dots, x_{16}) \wedge x_1 \neq x_9), \forall x_1 \cdots x_{16} \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge R(2x^3y^2)(x_9, \dots, x_{16}) \wedge x_5 \neq x_{13})$.

$R(3xy)$	X_1	X_2	X_3	Y_1	Y_2	Y_3	C	K
	1	1	0	1	1	0	1	3
	1	1	0	1	1	0	1	4
	1	1	0	1	1	0	1	5

$IC(3xy): \forall x_1 \cdots x_{16} \neg (R(3xy)(x_1, \dots, x_8) \wedge R(3xy)(x_9, \dots, x_{16}) \wedge x_3 \neq x_{11}), \forall x_1 \cdots x_{16} \neg (R(3xy)(x_1, \dots, x_8) \wedge R(3xy)(x_9, \dots, x_{16}) \wedge x_6 \neq x_{14}), \forall x_1 \cdots x_8 \neg (R(3xy)(x_1, \dots, x_8) \wedge x_1 \neq 1), \forall x_1 \cdots x_8 \neg (R(3xy)(x_1, \dots, x_8) \wedge x_2 \neq 1), \forall x_1 \cdots x_8 \neg (R(3xy)(x_1, \dots, x_8) \wedge x_4 \neq 1), \forall x_1 \cdots x_8 \neg (R(3xy)(x_1, \dots, x_8) \wedge x_5 \neq 1)$.

$R(105)$	X_1	X_2	X_3	Y_1	Y_2	Y_3	C	K
	1	1	1	1	1	1	105	6

$IC(105): \forall x_1 \cdots x_8 \neg (R(105)(x_1, \dots, x_8) \wedge x_1 \neq 1), \forall x_1 \cdots x_8 \neg (R(105)(x_1, \dots, x_8) \wedge x_2 \neq 1), \forall x_1 \cdots x_8 \neg (R(105)(x_1, \dots, x_8) \wedge x_3 \neq 1), \forall x_1 \cdots x_8 \neg (R(105)(x_1, \dots, x_8) \wedge x_5 \neq 1)$.

$\wedge x_4 \neq 1$), $\forall x_1 \cdots x_8 \neg (R(105)(x_1, \dots, x_8) \wedge x_5 \neq 1)$, $\forall x_1 \cdots x_8 \neg (R(105)(x_1, \dots, x_8) \wedge x_6 \neq 1)$, $\forall x_1 \cdots x_6 \neg (R(105)(x_1, \dots, x_6) \wedge x_7 \neq 105)$.

Similar tables $R(x^2y^3)$ and $R(y^2)$ and corresponding sets of ICs are generated for the terms on the RHS of the original equation.

Next we need ICs that are responsible for making equal all x 's and y 's in all terms of the equation: $\forall x_1 \cdots x_{16} \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge R(3xy)(x_9, \dots, x_{16}) \wedge x_1 \neq x_{11})$, $\forall x_1 \cdots x_{16} \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge R(3xy)(x_9, \dots, x_{16}) \wedge x_5 \neq x_{13})$, $\forall x_1 \cdots x_{16} \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge R(x^2y^3)(x_9, \dots, x_{16}) \wedge x_1 \neq x_{10})$, $\forall x_1 \cdots x_{16} \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge R(x^2y^3)(x_9, \dots, x_{16}) \wedge x_5 \neq x_{12})$, $\forall x_1 \cdots x_{16} \neg (R(2x^3y^2)(x_1, \dots, x_8) \wedge R(y^2)(x_9, \dots, x_{16}) \wedge x_5 \neq x_{13})$.

Now we construct a single table $R(equ)$ that represents the original equation by appending the previous tables:

$R(equ)$	X_1	X_2	X_3	Y_1	Y_2	Y_3	C	K
	0	0	0	1	0	0	1	1
	0	0	0	1	0	0	1	2
	1	1	0	1	1	0	1	3
	1	1	0	1	1	0	1	4
	1	1	0	1	1	0	1	5
	1	1	1	1	1	1	105	6
	1	0	0	0	0	0	1	7
	1	1	1	1	0	0	1	8

We need ICs stating the correspondence between the terms in the tables $R(t)$ and table $R(equ)$: $\forall x_1 \cdots x_{16} \neg (R(equ)(x_1, \dots, x_8) \wedge R(2x^3y^2)(x_9, \dots, x_{16}) \wedge x_8 = x_{16} \wedge x_1 \neq x_9)$, $\forall x_1 \cdots x_{16} \neg (R(equ)(x_1, \dots, x_8) \wedge R(2x^3y^2)(x_9, \dots, x_{16}) \wedge x_8 = x_{16} \wedge x_2 \neq x_{10})$, \dots , $\forall x_1 \cdots x_{16} \neg (R(equ)(x_1, \dots, x_6) \wedge R(y^2)(x_7 \cdots x_{16}) \wedge x_8 = x_{16} \wedge x_7 \neq x_{15})$.

Finally, we have one aggregate constraint that is responsible for making equal the LHS and RHS of the equation:

$$sum_{R(equ)}(x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 : x_6 < 7) = sum_{R(equ)}(x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 : x_6 > 6).$$

If the database has an LS-fix, then there is an integer solution to the diophantine equation. If the equation has a solution s , then there is an instance $R(equ)'$ corresponding to s that satisfies the ICs. By Proposition 1, there is an LS-fix of the database.

The reduction could be done with the table $R(equ)$ alone, making all the ICs above to refer to this table, but the presentation would be harder to follow.