Repair Checking in Inconsistent Databases: Algorithms and Complexity[*]

Foto N. Afrati[†‡]
National Technical University of Athens, Greece
afrati@softlab.ece.ntua.gr

Phokion G. Kolaitis[§]
UC Santa Cruz and IBM Almaden
kolaitis@cs.ucsc.edu

ABSTRACT

Managing inconsistency in databases has long been recognized as an important problem. One of the most promising approaches to coping with inconsistency in databases is the framework of database repairs, which has been the topic of an extensive investigation over the past several years. Intuitively, a repair of an inconsistent database is a consistent database that differs from the given inconsistent database in a minimal way. So far, most of the work in this area has addressed the problem of obtaining the consistent answers to a query posed on an inconsistent database. Repair checking is the following decision problem: given two databases r and r', is r' a repair of r? Although repair checking is a fundamental algorithmic problem about inconsistent databases, it has not received as much attention as consistent query answering. In this paper, we give a polynomial-time algorithm for subset-repair checking under integrity constraints that are the union of a weakly acyclic set of local-as-view (LAV) tuple-generating dependencies and a set of equality-generating dependencies. This result significantly generalizes earlier work for subset-repair checking when the integrity constraints are the union of an acyclic set of inclusion dependencies and a set of functional dependencies. We also give a polynomial-time algorithm for symmetric-difference repair checking, when the integrity constraints form a weakly acyclic set of LAV tgds. After this, we establish a number of complexity-theoretic results that delineate the boundary between tractability and intractability for the repair-checking problem. Specifically, we show that the aforementioned tractability results are optimal; in particular, subset-repair checking for arbitrary weakly acyclic sets of tuple-generating dependencies is a coNP-complete problem. We also study cardinality-based repairs and show that cardinality-repair checking is coNP-complete for various classes of integrity constraints encountered in database design and data exchange.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; F.1.2 [**Theory of Computation**]: Complexity Measures and Classes—*Reducibility and completeness*

General Terms

Theory, Algorithms

Keywords

Inconsistent databases, database repairs, repair checking, consistent query answering, tuple-generating dependencies, equality-generating dependencies, weakly acyclic set, polynomial time, coNP-complete problem

1. Introduction

Managing inconsistency in databases has long been recognized as an important problem. It is well understood that inconsistent databases arise in a variety of contexts and for several different reasons. For example, inconsistent databases arise because current database management systems may not support a particular type of integrity constraints. Furthermore, inconsistent databases arise in data integration, data warehousing, and other critical data inter-operability applications in which heterogeneous and distributed data have to be integrated or materialized in a new format and have to obey potentially different integrity constraints. Data cleaning is an area of research that addresses data quality by considering inconsistent databases and extracting “desirable" consistent databases according to some criterion of goodness. Some of the work done so far focuses on elimination of duplicates (e.g., [2, 19, 16, 23]) using a variety of different methods and algorithms or on more elaborate data value manipulations as in [26], which integrates transformation and discrepancy detection. In addition, application-specific integrity constraints have been considered to model the semantics of the data [7, 14, 25]. It has been argued, however, that much

[*] A preliminary version of this paper has been presented at the Workshop on Logic in Databases - LID ’08.

[†] Work was done partly while this author was visiting at IBM Almaden and at Stanford University.

[‡] This research project, no 03ED176, is co-financed by E.U.-European Social Fund (80%) and the Greek Ministry of Development-GSRT (20%).

[§] Research of this author partially supported by NSF Grant IIS-0430994.

ICDT 2009, March 23–25, 2009, Saint Petersburg, Russia.
Copyright 2009 ACM 978-1-60558-423-2/09/0003 ...$5.00

remains to be done in the area of data cleaning as regards to well-justified solutions [12].

Database Repairs In 1999, Arenas, Bertossi and Chomicki [1] introduced and developed a framework for extracting consistent data from an inconsistent database. Their framework is based on the concept of a *repair* of an inconsistent database. Intuitively, a repair of an inconsistent database is a database over the same schema that satisfies the integrity constraints at hand and differs from the given inconsistent database in some *minimal* way. This framework turned out to be quite influential and gave rise to numerous subsequent investigations of various notions of repairs based on different minimality criteria. In particular, work in this direction has considered repairs obtained by tuple deletions and insertions [1, 10, 22, 27] or by attribute value changes [4, 5, 6, 8, 28]; see also the survey [3] and the keynote paper [9].

The minimality criterion adopted in [1] was that the symmetric difference $r \oplus r'$ between the inconsistent database r and a consistent database r' is minimal with respect to set inclusion. This gives rise to the concept of *symmetric-difference repairs* or, in short, *$\oplus$-repairs*. Several other notions of repairs obtained by deletion and/or insertion of tuples have also been considered in recent years. Specifically, *subset-repairs* [10] are the special case of $\oplus$-repairs in which the repair r' is also a subinstance of the inconsistent database r. In general, a $\oplus$-repair need not be a subset-repair; it is easy to see, however, that if all constraints at hand are functional dependencies, then $\oplus$-repairs and subset-repairs coincide. Finally, *cardinality-repairs* or, in short *C-repairs*, [22] are consistent databases such that the cardinality $|r \oplus r'|$ of the symmetric difference $r \oplus r'$ is minimized. Every C-repair is a $\oplus$-repair, but not vice versa.

Repair Checking There are two fundamental algorithmic problems concerning repairs of inconsistent databases. The first and most extensively studied is the problem of computing the *consistent answers* of a query over an inconsistent database. The set of all repairs of an inconsistent database is viewed as a set of possible worlds. The *consistent answers* of a query over an inconsistent database are then defined to be the certain answers of the query over the set of all repairs, i.e., the tuples that are an answer to the query in every repair. Consistent query answering has been the focus of numerous investigations aiming to discover the boundary between tractability and intractability for this problem by taking into account two parameters: the type of repairs and the class of integrity constraints. Instead of giving a comprehensive list of earlier work on this topic, we refer the reader to the overviews [3] and [9].

The second fundamental algorithmic problem concerning repairs is *repair checking*: given two databases r and r' over the same schema, is r' a repair of r? This paper is devoted to the study of the repair checking problem. As with consistent query answering, repair checking is parameterized by the type of repairs and by the class of integrity constraints considered. Repair checking is *the model checking problem for repairs*. As such, it is a natural data-cleaning problem. Moreover, it underlies consistent query answering, because, for many classes of integrity constraints, repair checking is log-space reducible to consistent query answering (but not the other way around) [10].

So far, repair checking has not been investigated in the same depth as consistent query answering. The first study of repair checking in its own right was carried out by Chomicki and Marcinkowski [10], who investigated this problem for subset-repairs and for various classes of integrity constraints. In particular, they showed that if Σ is the union of an acyclic set of inclusion dependencies and a set of functional dependencies, then subset-repair checking with respect to Σ is in PTIME. The assumption of acyclicity turns out to be crucial, since they also showed that, for arbitrary sets of inclusion dependencies and functional dependencies, subset-repair checking is coNP-complete. Staworko [27] showed that if Σ is a set of full tuple-generating dependencies (full tgds), then $\oplus$-repair checking is in PTIME (hence, subset-repair checking is also in PTIME). Prior to this, Wijsen [28] had established that, for a certain type of attribute-based repairs, the repair checking problem for sets of full tgds and equality-generating dependencies (egds) is in PTIME. Finally, it follows from results in Lopatenko and Bertossi [22] that there is a set Σ of denial constraints for which the C-repair checking problem is coNP-complete (denial constraints form a broad class of constraints that include as a special case; the precise definition of a *denial constraint* will be given in the next section).

Summary of Results Acyclic sets of inclusion dependencies and sets of full tgds are important special cases of *weakly acyclic* sets of tuple-generating dependencies (tgds). Weakly acyclic sets of tgds have been extensively studied in the context of data exchange, where it was shown that they have tractable behavior as regards the key algorithmic problems encountered there, including the existence of solutions, the computation of universal solutions and conjunctive-query answering [13], and also the computation of the core of universal solutions [17]. Thus it is natural to ask: does the tractable behavior of weakly acyclic sets of tgds extend to the repair checking problem?

The main tractability results established in this paper are as follows. First, if Σ is the union of a weakly acyclic set of LAV (local-as-view) tgds and a set of equality-generating dependencies (egds), then the subset-repair checking problem w.r.t. Σ is in PTIME; in fact, it is in LOGSPACE. This tractability result significantly extends the result of Chomicki and Marcinkowski [10] about acyclic sets of inclusion dependencies and functional dependencies, because every acyclic set of inclusion dependencies is a weakly acyclic set of LAV tgds (but not vice versa). Second, we show that if Σ is a weakly acyclic set of LAV tgds, then the $\oplus$-checking problem w.r.t. Σ is in PTIME; in fact, it is in LOGSPACE. We also show that there is a set of full tgds for which the subset-repair checking problem is PTIME-complete, hence highly unlikely to be in LOGSPACE.

After this, we obtain a number of complexity-theoretic results that delineate the boundary between tractability and intractability for the repair checking problem. To begin with, we show that our two main tractability results are, in a sense, optimal. First, we construct a weakly acyclic set of non-LAV tgds for which the subset-repair checking problem is coNP-complete. To the best of our knowledge, subset-repair checking is the first example of a basic algorithmic problem that is tractable both for sets of full tgds and for acyclic sets of inclusion dependencies (in fact, it is tractable for weakly acyclic sets of LAV tgds), but can be intractable for weakly acyclic sets of tgds. We also construct a set that is the union of a weakly acyclic set of LAV tgds with a set of egds for which the $\oplus$-repair checking problem is coNP-complete.

Finally, we investigate cardinality-based repairs. In fact, we consider two different versions of cardinality-based repairs. The first is the aforementioned notion of a C-repair studied in [22] in which the aim is to minimize the cardinality $|r \oplus r'|$ of the symmetric difference $r \oplus r'$ between a given inconsistent database r and a consistent database r'. The second is a new notion of *component cardinality repair*, denoted by CC*-repair*, which we introduce in this paper. This new notion has a flavor of *Pareto optimality*, as the aim is to simultaneously minimize the cardinalities $|P^r \oplus P^{r'}|$ of the symmetric differences $P^r \oplus P^{r'}$, for every relation symbol P in the database schema. Every C-repair is a CC-repair, and every CC-repair is a $\oplus$-repair, but not vice versa. Depending on the application at hand, cardinality-based repairs may be preferable to other types of repairs since they amount to "repairing" an inconsistent database with a minimum number of insertions and deletions. We show, however, that the repair-checking problem for cardinality-based repairs is intractable even for classes of constraints for which the $\oplus$-repair checking problem is tractable. Specifically, we construct a set Σ of constraints that is the union of an acyclic set of inclusion dependencies and a set of functional dependencies and has the property that the C-repair checking problem w.r.t. Σ is coNP-complete. We also construct a set of full tgds for which the C-repair problem is coNP-complete. Finally, we show that the CC-repair checking problem can be coNP-complete for sets of denial constraints, for sets of full tgds, and for acyclic sets of inclusion dependencies. This leaves subset-repairs and $\oplus$-repairs as the only type of repairs based on tuple insertion and deletion for which the repair-checking problem is tractable for considerably broad classes of constraints.

Table 1 summarizes the emerging picture of the complexity of repair checking for various types of repairs and for various classes of integrity constraints. In this table, the annotation ‡ in an entry indicates a new result established in this paper, while the annotation † indicates that PTIME-hardness was established in this paper, while membership in PTIME was proved in [27]. Entries with no annotation are results obtained earlier in [10] and [22]. Finally, IND stands for inclusion dependencies.

Complete proofs of the results reported here will appear in the full version of the paper.

2. Preliminaries and Basic Facts

We consider instances over some fixed relational schema $\mathbf{S}$. If P is a relation symbol of $\mathbf{S}$ and r is an instance over $\mathbf{S}$, then P^r denotes the interpretation of P on r. We write $|B|$ to denote the cardinality of a set B. If r is an instance over $\mathbf{S}$ and $\mathbf{t}$ is a tuple such that $\mathbf{t} \in P^r$ for some relation symbol P of $\mathbf{S}$, then we say that $P^r(\mathbf{t})$ is a *fact* of r. An instance r can be identified with the set of all its facts.

We now give the precise definitions and several examples of the different types of integrity constraints that we will consider in this paper. They include the main constraints studied in classical dependency theory and, more recently, in data exchange and data integration (see the surveys [20, 21]).

DEFINITION 1. *Let* $\mathbf{S}$ *be a relational schema.*

1. *An* equality-generating dependency (egd) *is a first-order formula of the form*

   $$\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow x_i = x_j),$$

   where $\phi(\mathbf{x})$ *is a conjunction of atomic formulas over* $\mathbf{S}$ *with variables from* $\mathbf{x}$*, each variable in* $\mathbf{x}$ *occurs in at least one atomic formula in* $\phi(\mathbf{x})$*, and* x_i, x_j *are among the variables in* $\mathbf{x}$.

   An example of an egd is the functional dependency

   $$\forall x, y, z(\mathrm{MOTHER}(z, x) \wedge \mathrm{MOTHER}(w, x) \rightarrow z = w).$$

   In fact, every functional dependency is an egd, but not vice-versa.

2. *A* denial constraint *is a first-order formula of the form* $\forall \mathbf{x} \neg(\alpha(\mathbf{x}) \wedge \beta(\mathbf{x}))$*, where* $\alpha(\mathbf{x})$ *is a non-empty conjunction of atomic formulas* $P(\mathbf{t})$ *over* $\mathbf{S}$ *such that the variables in* $\mathbf{t}$ *are among the variables in* $\mathbf{x}$*, and* $\beta(\mathbf{x})$ *is a (possibly empty) conjunction of comparison atoms* $x_i = x_j$*,* $x_i \neq x_j$*,* $x_i < x_j$*,* $x_i \leq x_j$*, where the variables* x_i *and* x_j *are among those occurring in* $\alpha(\mathbf{x})$.

   Clearly, every egd is (logically equivalent to) a denial constraint, but not vice versa. For example, the formula

   $$\forall x, y(\mathrm{MOTHER}(x, y) \rightarrow x \neq y)$$

   is (logically equivalent to) a denial constraint, but is not (logically equivalent to) any egd.

3. *A* tuple-generating dependency (tgd) *is a first-order formula of the form*

   $$\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})),$$

   where $\phi(\mathbf{x})$ *is a conjunction of atomic formulas over* $\mathbf{S}$ *with variables in* $\mathbf{x}$*, each variable in* $\mathbf{x}$ *occurs in at least one formula in* $\phi(\mathbf{x})$*, and* $\psi(\mathbf{x}, \mathbf{y})$ *is a conjunction of atomic formulas with variables in* $\mathbf{x}$ *and* $\mathbf{y}$.

   For example, the following formula is a tgd:

   $$\forall x, y, z(\mathrm{MOTHER}(z, x) \wedge \mathrm{MOTHER}(z, y) \rightarrow \exists u, v(\mathrm{FATHER}(u, x) \wedge \mathrm{FATHER}(v, y))).$$

4. *A* full tgd *is a tgd with no existential quantifiers in the right-hand side; thus, a full tgd is a formula of the form* $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \psi(\mathbf{x}))$*, where* $\phi(\mathbf{x})$ *and* $\psi(\mathbf{x})$ *are conjunctions of atomic formulas over* $\mathbf{S}$.

   For example, the formula

   $$\forall x, y, z(\mathrm{MOTHER}(z, x) \wedge \mathrm{MOTHER}(z, y) \rightarrow \mathrm{SIBLING}(x, y))$$

   is a full tgd.

5. *A* LAV (local-as-view) tgd *is a tgd in which the left-hand side is a single atom. In other words, a* LAV *tgd is a first-order formula of the form* $\forall \mathbf{x}(P(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y}))$*, where* P *is a relation symbol in* $\mathbf{S}$ *and* $\psi(\mathbf{x}, \mathbf{y})$ *is a conjunction of atomic formulas over* $\mathbf{S}$.

   For example, the formula

   $$\forall x, y(\mathrm{SIBLING}(x, y) \rightarrow \exists z(\mathrm{MOTHER}(z, x) \wedge \mathrm{MOTHER}(z, y)))$$

   is a LAV tgd. Note also that inclusion dependencies are the special case of LAV tgds in which the right-hand side has a single atom.

From now on, we will drop the universal quantifiers in front of egds, denial constraints, and tgds.

DEFINITION 2. *In what follows, we will make use of the following notation.*

Constraints \ Semantics	**Subset-repair**	**$\oplus$-repair**	**C-repair**	CC-Repair
Denial	LOGSPACE	LOGSPACE	coNP-complete	coNP-complete‡
Acyclic set of IND & egds	LOGSPACE	Open	coNP-complete‡	coNP-complete‡
Weakly acyclic set of LAV tgds & egds	LOGSPACE‡	coNP-complete‡	coNP-complete‡	coNP-complete‡
Weakly acyclic set of LAV tgds	LOGSPACE‡	LOGSPACE‡	coNP-complete‡	coNP-complete‡
Full tgds & egds	PTIME-complete†	PTIME-complete†	coNP-complete‡	coNP-complete‡
IND & egds	coNP-complete	coNP-complete	coNP-complete‡	coNP-complete‡
Weakly acyclic set of tgds & egds	coNP-complete‡	coNP-complete‡	coNP-complete‡	coNP-complete‡

Table 1: The complexity of repair checking.

- *$r \subseteq r'$ denotes that for every relation symbol P in the schema, we have that $P^r \subseteq P^{r'}$. This is the same as asserting that r is contained in r', where r and r' are identified with the sets of their facts.*
- *$r \subset r'$ denotes that $r \subseteq r'$ and there is at least one relation symbol P in the schema such that $P^r \subset P^{r'}$. This is the same as asserting that r is properly contained in r', where r and r' are identified with the sets of their facts.*
- *$r \oplus r'$ denotes the symmetric difference of r and r' as sets of facts.*
- *$|r| \leq_{cc} |r'|$ denotes that for every relation symbol P in the schema, we have that $|P^r| \leq |P^{r'}|$. Similarly, $|r| <_{cc} |r'|$ denotes that $|r| \leq_{cc} |r'|$ and there is at least one relation symbol P in the schema such that $|P^r| < |P^{r'}|$.*

Repairs are defined with respect to a fixed set of constraints Σ and with respect to minimality in some partial order between instances over the fixed relational schema $\mathbf{S}$. Here, we will focus on the following notions of repairs. From now on, we assume that all sets of constraints considered are finite.

DEFINITION 3. *Let Σ be a set of integrity constraints and let r be a database instance.*

1. *A* subset-repair of *r is a sub-instance r' of r that is a maximal consistent sub-instance of r; this means that r' satisfies Σ and there is no instance r'' such that $r' \subset r'' \subseteq r$ and r'' satisfies Σ.*
2. *A* symmetric-difference-repair of *r (or, in symbols, a* $\oplus$-repair of *r) is an instance r' that satisfies Σ and there is no instance r'' such that $r \oplus r'' \subset r \oplus r'$ and r'' satisfies Σ.*
3. *A* cardinality repair of *r (or, in symbols, a* C-repair of *r) is an instance r' that satisfies the constraints in Σ and there is no instance r'' such that $|r \oplus r''| < |r \oplus r'|$ and r'' satisfies Σ.*
4. *A* component-cardinality repair of *r (or, in symbols a* CC-repair of *r) is an instance r' that satisfies Σ and there is no instance r'' such that $|r \oplus r''| <_{cc} |r \oplus r'|$ and r'' satisfies Σ.*

As mentioned in the Introduction, the concept of a CC-repair is new, and is introduced for the first time in this paper. We believe that it is a natural concept of database repair that deserves to be studied in its own right. To give some intuition about CC-repairs, we need to define an auxiliary concept. If r and r' are two database instances over the same relational schema, then the *characteristic sequence of r' with respect to r* is the sequence with coordinates the cardinalities $|P^r \oplus P^{r'}|$, as P varies over the relation symbols of the database schema (under some fixed order of the relation symbols in the underlying database schema). The definition of CC-repair can then be restated as follows: an instance r' is a CC-repair of r if and only if r' satisfies Σ and the characteristic sequence of r' with respect to r is minimal under the coordinate-wise ordering of sequences. In other words, a CC-repair of r is a consistent database r' such that the cardinality $|P^r \oplus P^{r'}|$ of the symmetric difference between a relation P^r of r and the corresponding relation $P^{r'}$ of r' cannot be made smaller without at the same time increasing the cardinality $|Q^r \oplus Q^{r'}|$ of the symmetric difference between some other relation Q^r of r and the corresponding relation $Q^{r'}$ of r'. Thus, the characteristic sequence of a CC-repair r' of r satisfies a *Pareto optimality condition*.

The relation between C-repairs, CC-repairs, and $\oplus$-repairs is as follows.

PROPOSITION 1. *Let Σ be a set of integrity constraints and let r be a database instance.*

1. *Every C-repair of r is a CC-repair of r.*
2. *Every CC-repair of r is a $\oplus$-repair of r.*

The converses are not true, i.e., there are $\oplus$-repairs that are not CC-repairs, and there are CC-repairs that are not C-repairs. The latter is shown in the following example, which, in addition, shows that there are CC-repairs with distinct characteristic sequences.

EXAMPLE 1. *Let Σ be the set consisting of the following four tgds:*

$$P(x) \rightarrow R(x), P'(x) \rightarrow R'(x), R(x) \rightarrow R'(x), P'(x) \rightarrow Q'(x).$$

Consider the inconsistent instance $r = \{P(1), P'(1)\}$ and the four consistent instances r_1, r_2, r_3, and r_4 below. After each instance r_i, we list the cardinality $|r \oplus r_i|$ and the characteristic sequence of r_i under the order (P, P', R, R', Q'):

$$\begin{aligned} r_1 = \emptyset; &\quad 2; \quad (1,1,0,0,0) \\ r_2 = \{P(1), P'(1), R(1), R'(1), Q'(1)\}; &\quad 3; \quad (0,0,1,1,1) \\ r_3 = \{P'(1), R'(1), Q'(1)\}; &\quad 3; \quad (1,0,0,1,1) \\ r_4 = \{P(1), R(1), R'(1),\}; &\quad 3; \quad (0,1,1,1,0) \end{aligned}$$

Each of the above instances is a CC-repair of r. However, it is obvious from the cardinalities of the symmetric differences $r \oplus r_i$ that only r_1 is a C-repair of r.

In the remainder of this section, we will give several elementary, but useful, results about repairs. They have been observed by several different researchers in one form or another. We begin with the relationship between $\oplus$-repairs and subset-repairs.

PROPOSITION 2. *If $r' \subseteq r$, then the following statements are equivalent.*

1. *r' is a subset-repair of r*
2. *r' is a $\oplus$-repair of r.*

The next proposition is about denial constraints.

PROPOSITION 3. *Let Σ be a set of denial constraints.*

1. *If $r \models \Sigma$ and $r^* \subseteq r$, then $r^* \models \Sigma$.*
2. *If r' is a $\oplus$-repair of r, then r' is a subset repair of r.*

Consequently, for denial constraints, $\oplus$-repairs and subset-repairs coincide.

It should be emphasized that Proposition 3 is not true for set of full tgds. For example, let ψ be the full tgd $(E(x, y) \rightarrow E(y, x))$. Consider the database instances $r = \{E(a, b), E(b, a)\}$ and $r^* = \{E(a, b)\}$. Then $r \models \psi$ and $r^* \subseteq r$, but $r^* \not\models \psi$. Furthermore, r is a $\oplus$-repair of r^*, but not a subset-repair.

We now introduce the main algorithmic problem studied in this paper.

DEFINITION 4. *Let Σ be a set of constraints and T a type of repair. The T*-repair checking problem with respect to *Σ is the following decision problem: given two instances r and r', is r' is a T-repair of r?*

By Proposition 2, if the $\oplus$-repair checking problem w.r.t. Σ is in some complexity class, then so is the subset-repair checking problem w.r.t. Σ. Moreover, if the subset-repair checking problem w.r.t. Σ is hard for some complexity class, then so is the $\oplus$-repair checking problem w.r.t. Σ.

The following simple result yields an upper bound for all types of repairs and for all classes of constraints introduced in this section.

PROPOSITION 4. *Let Σ be a set of constraints such that checking whether an instance r satisfies Σ is in PTIME. Then the C-repair checking problem w.r.t. Σ, the CC-repair checking problem w.r.t. Σ, and the $\oplus$-repair checking problem w.r.t. Σ are in coNP. Hence, the subset-repair problem w.r.t. Σ is also in coNP.*

Note that the hypothesis in Proposition 4 is satisfied by every finite set Σ of first-order constraints. In particular, Proposition 4 applies to every finite set Σ of tgds, egds, or denial constraints.

3. Tractable Repair Checking

In this section, we give a polynomial-time algorithm for the subset-repair checking problem w.r.t. sets of constraints that are the union of a *weakly acyclic* set of LAV tgds and a set of egds. Such sets contain as a special case all sets of constraints that are the union of an acyclic set of inclusion dependencies and a set of functional dependencies. In addition, we give a polynomial-time algorithm for the $\oplus$-repair checking problem w.r.t. weakly acyclic sets of LAV tgds. To place our tractability results in the proper context, we begin by reviewing some earlier tractability results for repair checking. The first is part of the "folklore".

PROPOSITION 5. *If Σ is a set of denial constraints over some relational schema $\mathbf{S}$, then the subset-repair checking problem w.r.t. Σ (equivalently, the $\oplus$-repair checking problem w.r.t. Σ) is in LOGSPACE.*

PROOF. The basic idea is that we can test whether r' is maximal by checking one fact from $r - r'$ at a time. More formally, we first check whether $r' \models \Sigma$ and then we check whether $r' \subseteq r$. If $r' \not\subseteq r$, then, by the second part of Proposition 3, r' is not a $\oplus$-repair. If $r' \models \Sigma$ and $r' \subseteq r$, then we add facts $P(\mathbf{t})$ from r to r' one at a time and check whether $r' \cup \{R(\mathbf{t})\} \models \Sigma$. If such a fact $R(\mathbf{t})$ is found, then r' is not a repair; otherwise, it is a repair. The correctness of this algorithm follows from the first part of Proposition 3. □

Recall that a set Σ of inclusion dependencies is *acyclic* if the dependency graph of Σ has no cycles, where the *dependency graph of* Σ is the graph whose nodes are the relation symbols occurring in Σ and whose edges are pairs of the form (R, S) such that R occurs in the left-hand side of some dependency in Σ and S occurs in the right-hand side of that dependency. Chomicki and Marcinkowski [10, Theorem 4.2] obtained the following tractability result for subset-repair checking w.r.t. acyclic sets of inclusion dependencies.

THEOREM 1. *(Chomicki and Marcinkowski) If Σ is the union of an acyclic set of inclusion dependencies and a set of functional dependencies over some relational schema $\mathbf{S}$, then the subset-repair checking problem w.r.t. Σ is in PTIME.*

A perusal of the algorithm in the proof of Theorem 1 in [10] actually shows that this algorithm uses logarithmic space only. Intuitively, the reason is that the algorithm uses the fixed finite acyclic dependency graph, examines all relations in the database schema starting with the sinks of the dependency graph, and, in each step, it tests one fact at a time. Thus, if Σ is the union of an acyclic set of inclusion dependencies and a set of functional dependencies, then the subset-repair checking problem w.r.t. Σ is in LOGSPACE.

In what follows, we will significantly extend Theorem 1 from acyclic sets of inclusion dependencies and functional dependencies to weakly acyclic sets of LAV tgds and egds. As is well known, the notion of a *weakly acyclic* set of tgds was considered first in [11] and [13], and plays a key role in data exchange.

DEFINITION 5. *Let Σ be a set of tgds over a relational schema $\mathbf{S}$.*

- *The* position graph of Σ *is constructed as follows:*
    1. *There is a node for every pair (R, A), where R is a relation symbol of $\mathbf{S}$ and A is an attribute of R. We call such a pair (R, A) a* position.
    2. *Let $\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ be a tgd in Σ and let x in $\mathbf{x}$ be a variable that also occurs in $\psi(\mathbf{x}, \mathbf{y})$. For every occurrence of x in $\phi(\mathbf{x})$ in position (R, A_i), do the following:*

       (i) For every occurrence of x in $\psi(\mathbf{x}, \mathbf{y})$ in position (S, B_j), add an edge $(R, A_i) \rightarrow (S, B_j)$ (if one does not already exist);

       (ii) In addition, for every existentially quantified variable y in $\mathbf{y}$ and for every occurrence of y in $\psi(\mathbf{x}, \mathbf{y})$ in position (T, C_k), add a special edge *$(R, A_i) \xrightarrow{*} (T, C_k)$ (if one does not already exist).*

- *We say that* Σ *is* weakly acyclic *if the dependency graph has no cycle going through a special edge.*
- *We say that a tgd* θ *is* weakly acyclic *if the singleton* $\{\theta\}$ *is weakly acyclic.*

Every set of full tgds is weakly acyclic since the dependency graph contains no special edges. Furthermore, every acyclic set of inclusion dependencies is a weakly acyclic set of LAV tgds, since every cycle in the position graph induces a cycle in the dependency graph. The converse, however, is not true. For example, $\Sigma = \{D(e,m) \rightarrow M(m), M(m) \rightarrow \exists e D(e,m)\}$ is a weakly acyclic, but cyclic, set of inclusion dependencies. We are now ready to state our first main result.

THEOREM 2. *If* Σ *is the union of a weakly acyclic set of LAV tgds and a set of egds over some relational schema* $\mathbf{S}$*, then the subset-repair problem w.r.t.* Σ *is in PTIME; in fact, it is in LOGSPACE.*

Theorem 2 will be proved by giving a polynomial-time algorithm for this problem. The algorithm we will give has a simple description, but its proof of correctness is quite non-trivial. It will make essential use of both the hypothesis that every tgd in Σ is a LAV tgd and the hypothesis that the tgds in Σ form a weakly acyclic set. The main property of LAV tgds used is that only single facts (and not combinations of facts) "trigger" a LAV tgd. One important consequence of this property is that if r_1 and r_2 are two databases instances that satisfy a LAV tgd d, then their union (as sets of facts) also satisfies d; this property fails for arbitrary tgds. The main property of weakly acyclic sets of tgds used is that the *chase procedure* converges in polynomial time on such sets (see [13]). Actually, we will need a more delicate version of the chase procedure, called *solution aware chase*, which was introduced and studied in [15]. Before describing our algorithm, we present the necessary background about the solution aware chase and several technical lemmas.

Suppose we have a schema mapping specified by a set of source-to-target tgds and a set of target constraints that is the union of a weakly acyclic set of target tgds with a set of target egds. In [13], the chase procedure was used in this context to detect the existence of solutions and, if solutions exist, to construct a canonical universal solution. During a chase step, the chase procedure introduces new labeled nulls as needed to witness the existentially quantified variables of a tgd that has been triggered. Thus, databases instances produced by the chase procedure may contain constants and labeled nulls as values. Suppose now that we have a set Σ of tgds and egds, and two database instances K and K' such that K is a sub-instance of K', K does not satisfy Σ, and K' satisfies Σ. In [15], the *solution aware chase* procedure was introduced and used to chase K with Σ and in such a way that, instead of new nulls, values from K' are used to witness the existentially quantified variables in the tgds in Σ (such values exist because K' satisfies Σ). The precise notions of a *solution-aware chase step* and *solution-aware chase* are as follows.

DEFINITION 6. (Solution-aware chase step [15]) *Let* K_1 *be an instance.*

(tgd) Let d *be a tgd* $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x},\mathbf{y}))$*. Let* K *be an instance that contains* K_1 *and satisfies* d*. Let* h *be a homomorphism from* $\phi(\mathbf{x})$ *to* K_1 *such that there is no extension of* h *to a homomorphism* h' *from* $\phi(\mathbf{x}) \wedge \psi(\mathbf{x},\mathbf{y})$ *to* K_1*. We say that* d can be applied to K_1 with homomorphism h and solution K*, or simply,* d can be applied to K_1 with homomorphism h *if* K *is understood from context.*

Let h' *be an extension of* h *such that every variable in* $\mathbf{y}$ *is assigned a value in* K *and* $h' : \psi(\mathbf{x},\mathbf{y}) \rightarrow K$ *and let* $K_2 = K_1 \cup h'(\psi(\mathbf{x},\mathbf{y}))$*, where*

$$h'(\psi(\mathbf{x},\mathbf{y})) =$$

$$\{R(h'(z_1),...,h'(z_n)) : R(z_1,\ldots,z_n) \textit{ is atom of } \psi(\mathbf{x},\mathbf{y})\}$$

We say that the result of applying d to K_1 with h and solution K is K_2*, and write* $K_1 \xrightarrow{d,h,K} K_2$*. We drop* K *and write* $K_1 \xrightarrow{d,h} K_2$ *if* K *is understood from the context.*

(egd) Let d *be an egd* $\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow (x_1 = x_2))$*. Let* h *be a homomorphism from* $\phi(\mathbf{x})$ *to* K_1 *such that* $h(x_1) \neq h(x_2)$*. We say that* d can be applied to K_1 with homomorphism h*. We distinguish two cases.*

- *If both* $h(x_1)$ *and* $h(x_2)$ *are constants, then we say that* the result of applying d to K_1 with h is "failure"*, and write* $K_1 \xrightarrow{d,h} \perp$.
- *Otherwise, let* K_2 *be* K_1 *where we identify* $h(x_1)$ *and* $h(x_2)$ *as follows: if one is a constant, then the labeled null is replaced everywhere by the constant; if both are labeled nulls, then one is replaced everywhere by the other. We say that* the result of applying d to K_1 with h is K_2*, and write* $K_1 \xrightarrow{d,h} K_2$.

DEFINITION 7. **(Solution-aware chase)** *Let* Σ *be a set of tgds and egds. Let* K *be an instance and* K' *be an instance that contains* K *and satisfies the set of tgds in* Σ*.*

- *A* solution-aware chase sequence of K with Σ and K' *is a sequence (finite or infinite) of solution-aware chase steps* $K_i \xrightarrow{d_i,h_i} K_{i+1}$*, with* $i = 0, 1, ...$*, with* $K = K_0$ *and* d_i *a dependency in* Σ*.*
- *A* finite solution-aware chase of K with Σ and K' *is a finite solution-aware chase sequence* $K_i \xrightarrow{d_i,h_i} K_{i+1}$, $0 \leq i \leq m$*, with the requirement that either (a)* $K_m = \perp$ *or (b) there is no dependency* d_i *of* Σ *and there is no homomorphism* h_i *such that* d_i *can be applied to* K_m *with* h_i*. We say that* K_m *is the result of the finite solution-aware chase. We refer to case (a) as the case of a* failing finite solution-aware chase *and we refer to case (b) as the case of a* successful finite solution-aware chase.

The following lemma (which is Lemma 3.4 in [15]) asserts that if Σ is the union of a weakly acyclic set of tgds with a set of egds, then the length of every chase sequence of an instance is bounded by a polynomial in the size of the instance.

LEMMA 1. [15, Lemma 3.4] *Let* Σ *be the union of a weakly acyclic set of tgds with a set of egds over some schema. Then there exists a polynomial* $p(x)$ *having the following property: if* K *and* K' *are instances such that* K' *satisfies* Σ *and* K' *contains* K*, then the length of every solution-aware chase sequence of* K *with* Σ *and* K' *is bounded by* $p(|K|)$*, where* $|K|$ *is the size of* K*.*

The next lemma follows easily from (the proof of) Lemma 3.5 in [15].

LEMMA 2. (implicit in [15]) *Let Σ be the union of a weakly acyclic set of tgds with a set of egds over some schema. Then there exists a polynomial $p(x)$ having the following property: if K and K' are instances such that K' satisfies Σ and K' contains K, then there is an instance K^* such that K is contained in K^*, K^* is contained in K', K^* satisfies Σ, and $|K^*| \leq p(|K|)$.*

In a nutshell, Lemma 2 is proved by showing that if we chase K with Σ and K', then a successful finite solution-aware chase exists. This gives the desired instance K^* whose size, according to Lemma 1 is bounded by a polynomial in the size of K. The following result is an immediate consequence of Lemma 2.

COROLLARY 1. *Let Σ be the union of a weakly acyclic set of tgds with a set of egds over some schema. Then there exists a constant c (that depends only on Σ) having the following property: if K' is an instance that satisfies Σ and t is a fact in K', then there is an instance K^* such that $t \in K^*$, K^* is contained in K', K^* satisfies Σ, and $|K^*| \leq c$.*

PROOF. We let $c = p(1) = p(|\{t\}|)$, where $p(x)$ is the polynomial given by Lemma 2. □

Using Corollary 1 and properties of LAV tgds and egds, we can now obtain the following key technical lemma, which will be heavily used in our polynomial-time algorithm for subset-repair checking w.r.t. a set of constraints that is the union of a weakly acyclic set of LAV tgds with a set of egds. This lemma will also be used in designing our polynomial-time algorithm for $\oplus$-repair checking w.r.t. a weakly acyclic set of LAV tgds (note that in Lemma 3 below we do not assume that r' is contained in r).

LEMMA 3. *Let Σ be a set that is the union of a weakly acyclic set of LAV tgds with a set of egds. Then there is a constant c (that depends only on Σ) such that the following holds. Let r and r' be two instances and suppose that r' satisfies Σ. Let t be a fact in $r - r'$ such that there is a non-empty set $A \subseteq r - r'$ of facts such that $t \in A$ and $r' \cup A$ satisfies Σ. Then there is a set A_t of facts such that $t \in A_t$, $A_t \subseteq r - r'$, $|A_t| \leq c$, and $r' \cup A_t$ satisfies Σ.*

PROOF. Let c be the constant given by Corollary 1. Suppose that we are given instances r and r', a fact t in $r - r'$, and a set A of facts that satisfy the hypotheses of the lemma. By applying Corollary 1 to Σ, to t, and to $K' = r' \cup A$, we obtain an instance K^* such that $t \in K^*$, K^* is contained in $r' \cup A$, K^* satisfies Σ, and $|K^*| \leq c$. We now claim that the instance $r' \cup K^*$ satisfies Σ. First, $r' \cup K^*$ satisfies every tgd in Σ because all tgds in Σ are LAV tgds and both r' and K^* satisfy Σ (as mentioned earlier, LAV tgds are preserved under union of models). Furthermore, $r' \cup K^*$ satisfies every egd in Σ because $r' \cup K^*$ is contained in $r' \cup A$ and $r' \cup A$ satisfies Σ (egds are preserved under sub-instances). Let $A_t = K^* \cap (r - r')$. Then $t \in A_t$, $A_t \subseteq r - r'$, $|A_t| \leq c$, and $r' \cup A_t$ satisfies Σ, since $r' \cup A_t = r' \cup K^*$. □

We are now ready to embark on the proof of Theorem 2.

Proof of Theorem 2: We give the promised polynomial-time algorithm for the subset-repair checking problem and outline its proof of correctness. Assume that Σ is the union of a weakly acyclic set of LAV tgds and a set of egds. Let c be the constant given by Corollary 1.

Algorithm for subset-repair checking w.r.t. Σ

Input: Two instances r and r' such that $r' \subset r$, r does not satisfy Σ and r' satisfies Σ.

Output: Determine whether or not r' is a subset-repair of r w.r.t. Σ.

Test whether there is a set A^* of facts such that:

1. A^* is non-empty;
2. $|A^*| \leq c$;
3. A^* is contained in $r - r'$;
4. $r' \cup A^*$ satisfies Σ.

If such a set A^* is found, then stop, report that "*r' is not a subset-repair of r*", and exit. If no such set A^* is found, then stop, report that "*r' is a subset-repair of r*", and exit.

We now have to analyze the running time of this algorithm and prove its correctness. The algorithm runs in time polynomial in the sizes of r and r', since we have a polynomial number of tests and each step can be carried out in polynomial time. Actually, each of these tests can be done in space logarithmic in the sizes of r and r', and we can keep track of the number of tests by maintaining a counter in binary. It follows that this a log-space algorithm. Next, we turn to the correctness of the algorithm. It is obvious that if the algorithm terminates by finding such a set A^*, then r' is not a subset-repair of r, because $r' \cup A^*$ is a consistent instance that is contained in r and properly contains r'. In the other direction, assume that r' is not a subset-repair of r. Let r'' be a database instance such that $r'' \models \Sigma$ and $r' \subset r'' \subset r$. Hence, there is a non-empty set $A \subset r - r'$ such that $r'' = r' \cup A$. By applying Lemma 3, we conclude that there is a non-empty set $A^* \subseteq r - r'$ such that $|A^*| \leq c$ and $r' \cup A^* \models \Sigma$. This completes the proof of Theorem 2. ▮.

As a byproduct of the polynomial-time algorithm for subset-repair checking, we obtain a polynomial-time algorithm for finding a subset-repair of an inconsistent instance.

COROLLARY 2. *Let Σ be the union of a weakly acyclic set of LAV tgds and a set of egds over some relational schema $\mathbf{S}$. Then there is a polynomial-time algorithm for the following problem: given a database instance r such that r does not satisfy Σ, find a subset-repair of r.*

PROOF. We start with the empty instance and apply repeatedly the polynomial-time algorithm for subset-repair checking. Since at each step we get a consistent instance that properly contains the previous consistent instance, the algorithm will terminate in a polynomial-time of steps and produce a subset-repair of r. □

Our second main result is a polynomial-time algorithm for the $\oplus$-repair checking problem w.r.t. weakly acyclic sets of LAV tgds (no egds are allowed).

THEOREM 3. *If Σ is a weakly acyclic set of LAV tgds over some relational schema $\mathbf{S}$, then the $\oplus$-repair checking problem w.r.t. Σ is in PTIME; in fact, it is in LOGSPACE.*

PROOF. Assume that Σ is a weakly acyclic set of LAV tgds. Let c be the constant given by Corollary 1.

Algorithm for $\oplus$-repair checking w.r.t. Σ

Input: Two instances r and r' such that r does not satisfy Σ and r' satisfies Σ.

Output: Determine whether or not r' is a $\oplus$-repair of r w.r.t. Σ.

Step 1. Test whether there is a set A^* of facts such that:

1. A^* is non-empty;
2. $|A^*| \leq c$;
3. A^* is contained in $r - r'$;
4. $r' \cup A^*$ satisfies Σ.

If such a set A^* is found, then stop, report that "*r' is not a $\oplus$-repair of* r", and exit; else, go to Step 2.

Step 2. Test whether there is a set B^* of facts such that

1. B^* is non-empty;
2. $|B^*| \leq c$;
3. B^* is contained in $r' - r$;
4. $r' - B^*$ satisfies Σ.

If such a set B^* is found, then stop, report that "*r' is not a $\oplus$-repair of* r", and exit. If no such B^* is found, then stop, report that "*r' is a $\oplus$-repair of* r", and exit.

We now have to analyze the running time of this algorithm and prove its correctness. The algorithm runs in polynomial time, because we have a polynomial-number of tests in Steps 1 and 2, and each step can be carried out in polynomial time. In fact, this is a log-space algorithm. Next, we turn to the correctness of the algorithm. It is obvious that if the algorithm terminates by finding such a set A^*, then r' is not a $\oplus$-repair of r since $r' \cup A^*$ is a consistent instance that is "better" than r'. Similarly, if the algorithm terminates by finding such a set B^*, then r' is not a $\oplus$-repair of r since $r' - B^*$ is a consistent instance that is "better" than r'. So, it remains to show that if r' is not a $\oplus$-repair of r, then such a set A^* exists or such a set B^* exists. To establish this fact, we will need three additional lemmas. The first is an elementary property of sets.

LEMMA 4. *Let r, r', and r'' be sets such that $(r'' - r) \cup (r - r'') \subset (r' - r) \cup (r - r')$. Then there are sets A and B such that the following hold:*

1. *At least one of A and B is non-empty;*
2. *$r'' = (r' - B) \cup A$;*
3. *$A \subseteq r - r'$ and $B \subseteq r' - r$.*

Using Lemma 4, we obtain the following crucial lemma.

LEMMA 5. *Let Σ be a weakly acyclic set of LAV tgds, and let r and r' be two database instances such that r does not satisfy Σ and r' satisfies Σ. If r' is not a $\oplus$-repair of r, then there are sets A and B of facts such that the following hold:*

1. *At least one of A and B is non-empty;*
2. *$A \subseteq r - r'$ and $B \subseteq r' - r$;*
3. *Either A is empty (which implies that B is non-empty) and $r' - B$ satisfies Σ, or A is non-empty and $r' \cup A$ satisfies Σ.*

PROOF. If r' is not a $\oplus$-repair of r, then there is an instance r'' such that r'' satisfies Σ and $(r'' - r) \cup (r - r'') \subset (r' - r) \cup (r - r')$. By Lemma 4, there are sets A and B of facts that satisfy the three conditions in the conclusion of Lemma 4. In particular, A and B already satisfy the first two conditions that we had to prove in the present lemma. Note also that $r'' = (r' - B) \cup A \subseteq r' \cup A$. It remains to show that either A is non-empty and $r' \cup A$ satisfies Σ, or A is empty and $r' - B$ satisfies Σ. For this, we consider two cases: A is empty or A is non-empty.

Case 1: Suppose that A is empty. In this case, we have that $r'' = (r' - B) \cup A = r' - B$ and so $r' - B$ satisfies Σ (since r'' satisfies Σ).

Case 2: Suppose that A is non-empty. We will show that $r' \cup A$ satisfies Σ. Let d be a tgd in Σ. Then d must be a LAV tgd, which implies that only single facts can "trigger" d. So, assume that t is a fact in $r' \cup A$ that satisfies the left-hand side of d. If t is in r', then the right-hand side of d is satisfied in r' (since r' satisfies Σ) and so d is satisfied in $r' \cup A$. If $t \in A$, then $t \in r''$, hence the right-hand of d is satisfied in r'' (since r'' satisfies Σ) and so d is satisfied in $r' \cup A$ (since $r'' = (r' - B) \cup A \subseteq r' \cup A$). □

It should be pointed out that it was important in the proof of Lemma 5 that Σ contained no egds (Case 2 does not go through for egds). We now state the third and final lemma.

LEMMA 6. *Let Σ be a weakly acyclic set of LAV tgds. Then there is a constant (that depends only on Σ) such that the following holds. Suppose that r and r' are two instances such that r' satisfies Σ. Suppose also that there exists a non-empty set $B \subseteq r' - r$ such that $r' - B$ satisfies Σ. Then there is a non-empty set B_0 such that $B_0 \subseteq r' - r$, $|B_0| \leq c$, and $r' - B_0$ satisfies Σ.*

PROOF. Let c be the constant in Corollary 1. Let B_0 be a *minimal* set such that B_0 is non-empty, $B_0 \subseteq r' - r$, and $r' - B_0$ satisfies Σ (i.e., no proper subset of B_0 has these three properties). We claim that $|B_0| \leq c$. Towards a contradiction, suppose that $|B_0| > c$. Let t be a fact in B_0. By applying Corollary 1 to r' and to t, we obtain a set B' of facts such that $t \in B'$, $B' \subseteq r'$, $|B'| \leq c$, and $B' \models \Sigma$. It follows that $(r' - B_0) \cup B' \models \Sigma$. Moreover, we have that $(r' - B_0) \cup B' = r' - (B_0 - B')$. We now have that $B_0 - B'$ is a proper subset of B_0 (because $t \in B' \cap B_0$), $B_0 - B'$ is non-empty (because $|B_0| > c$ and $|B'| \leq c$), $B_0 - B' \subseteq r' - r$ (because $B_0 \subseteq r' - r$), and $r' - (B_0 - B') \models \Sigma$. This, however, violates the minimality of B_0, hence $|B_0| \leq c$. □

Using Lemmas 5 and 6, we can now prove that if r' is not a $\oplus$-repair of r, then such a set A^* exists or such a set B^* exists. For this, we distinguish two cases. The first is the case in which, by Lemma 5, a non-empty subset A of $r - r'$ exists such that $r' \cup A \models \Sigma$. In this case, by applying Corollary 1 to $r' \cup A$ and to some fact t in A, we obtain the desired set A^*. The second case is the case in which, by Lemma 5, a set non-empty subset B of $r' - r$ exists such that $r' - B \models \Sigma$. In this case, the desired set B^* is given by Lemma 6.

We note that our polynomial-time algorithm for $\oplus$-repair checking can be modified to a polynomial-time algorithm for finding a $\oplus$-repair of an inconsistent database (w.r.t. a weakly acyclic set of LAV tgds).

3.1 P-completeness for the case of full tgds

In his Ph.D. thesis, Staworko [27] identified another natural and extensively studied class of constraints for which the $\oplus$-repair checking problem is tractable.

THEOREM 4. *(Staworko) If Σ is a set of full tgds, then the $\oplus$-repair checking problem w.r.t. Σ is in PTIME.*

Our second result in this section asserts that there is a set of full tgds for which the subset-repair checking problem is PTIME-complete (consequently, the $\oplus$-repair checking problem is PTIME-complete as well).

THEOREM 5. *There is a set Σ of full tgds such that the subset-repair checking problem w.r.t. Σ is PTIME-complete.*

PROOF. *(Sketch)* By Theorem 4, for every finite set of full tgds, the subset-repair checking problem is in PTIME. We will establish PTIME-hardness via a log-space reduction from HORN 3-SAT, which is the following decision problem: given a Horn formula φ with at most three literals per clause, is φ satisfiable? (It is well known that HORN 3-SAT is PTIME-complete; see [18].)

We consider a database schema consisting of four unary relation symbols A, U, M, F, and two ternary relation symbols P and N. Let Σ be the set of the following four full tgds:

$$\begin{aligned} A(w) \wedge U(x) &\rightarrow M(x)\\ A(w) \wedge P(x,y,z) \wedge M(x) \wedge M(y) &\rightarrow M(z)\\ (w) \wedge N(x,y,z) \wedge M(x) \wedge M(y) \wedge M(z) &\rightarrow F(w)\\ M(w) &\rightarrow A(w). \end{aligned}$$

In the full version of this paper, we show that, given a Horn formula φ with at most three literals per clause, we can construct in log-space two database instances r and r' such that φ is satisfiable if and only if r' is not a subset-repair of r. □

It follows that, unless PTIME $=$ LOGSPACE, the subset-repair checking problem for sets of full tgds is not in LOGSPACE. Thus, in a precise complexity-theoretic sense, the subset-repair checking problem for sets of full tgds is harder than the subset-repair checking problem for sets of denial constraints or for sets that are the union of a weakly acyclic set of LAV tgds with a set of egds.

4. Intractable Repair Checking

4.1 Subset-repair checking and $\oplus$-repair checking

Chomicki and Marcinkowski [10, Theorem 4.4] obtained the following intractability result.

THEOREM 6. *(Chomicki and Marcinkowski) There is a set Σ consisting of one inclusion dependency and one functional dependency such that the subset-repair checking problem w.r.t. Σ is coNP-complete.*

The inclusion dependency in the proof of Theorem 6 is

$$R(x_1, x_2, x_3, x_4) \rightarrow \exists y_1, y_2, y_3 R(y_1, y_2, x_4, y_3).$$

This inclusion dependency is cyclic; as a matter of fact, it is not even weakly acyclic, since its position graph contains a self-loop with a special edge in position $(R, 4)$. By Theorem 2, if Σ is the union of a weakly acyclic set of LAV tgds and a set of egds, then the subset-repair checking problem w.r.t. Σ is in PTIME. Our first intractability result reveals that the hypothesis in Theorem 2 that all tgds are LAV is of the essence.

THEOREM 7. *There is a weakly acyclic set Σ of tgds such that the subset-repair problem w.r.t. Σ is coNP-complete. Consequently, also the $\oplus$-repair checking problem w.r.t. Σ is coNP-complete.*

PROOF. *(Sketch)* For every finite set of tgds, the subset-repair checking problem is in coNP by Corollary 2 and Proposition 4. We will show that there is a weakly acyclic set of tgds for which this problem is coNP-hard via a reduction from POSITIVE-1-IN-3-SAT. Recall that POSITIVE 1-IN-3-SAT is the following decision problem: given a Boolean formula φ in conjunctive normal form and such that each clause is a disjunction $(x \vee y \vee z)$ of three positive literals, is there a truth assignment that makes true exactly one variable in every clause?

Let Σ be the set consisting of the (non-LAV) acyclic tgd

$$\begin{aligned} A(s) \wedge P(x,y,z) \rightarrow \exists u, v, w(T(x,u) \wedge T(y,v) \wedge{}\\ T(z,w) \wedge S(u,v,w)) \end{aligned}$$

and the following two full tgds:

$$\begin{aligned} T(x,u) \wedge T(x,u') \wedge D(u,u') &\rightarrow S(u,u,u)\\ T(x,u) &\rightarrow A(u). \end{aligned}$$

Note that Σ is a weakly acyclic set of tgds; indeed, the only special edges in the dependency graph of Σ are from the positions of P to the positions of T, and no position in P has an incoming edge. In the full version of this paper, we show that, given an instance φ of POSITIVE 1-IN-3 SAT with n variables, we can construct in polynomial time two database instances r and r' such that here is a truth assignment s that makes true exactly one variable in each clause of φ if and only if r' is not a subset-repair of r. □

By Theorem 3, if Σ is a weakly acyclic set of LAV tgds, then the $\oplus$-repair checking problem w.r.t. Σ is in PTIME. Our second intractability result reveals that the absence of egds from Σ is of the essence. It is proved via a reduction from POSITIVE-1-IN-3-SAT, which is given in the full version of this paper.

THEOREM 8. *There is a weakly acyclic set Σ_1 of LAV tgds and a set Σ_2 of egds such that the $\oplus$-repair checking problem w.r.t. $\Sigma_1 \cup \Sigma_2$ is coNP-complete.*

Combined with the tractability results in Section 3, Theorems 7 and 8 yield a fairly complete characterization of the computational complexity of the subset-repair problem and the $\oplus$-repair checking problem for the various classes of constraints considered here. In fact, as depicted in Table 1, the only question that remains open is whether there is a set Σ that is the union of an acyclic set of inclusion dependencies and a set of egds such that the $\oplus$-repair checking problem w.r.t. Σ is coNP-complete.

4.2 Cardinality-based repair checking

The following result is implicit in [22]; actually, it follows from Lemma 4 in [22].

THEOREM 9. *(Lopatenko and Bertossi) There is a denial constraint φ such that the C-repair checking problem w.r.t. φ is coNP-complete.*

Theorem 9 tells that, for denial constraints, the coNP-upper bound in Proposition 4 is optimal. It should be pointed out that the denial constraint used to prove Theorem 9 involves no comparison atoms, i.e., it is an first-order formula of the form $\forall \mathbf{x} \neg \alpha(\mathbf{x})$, where $\alpha(\mathbf{x})$ is a conjunction of relational atoms. Furthermore, the proof is via a reduction from the following variant of the MAXIMUM INDEPENDENT SET problem: given a graph $G = (V, E)$ and a subset V' of the set V of the nodes of G, is V' an independent set of maximum cardinality in G?

We now adapt the main idea in the proof of Theorem 9 to show that the C-repair checking problem can be coNP-hard for full tgds. It is proved via a reduction from MAX CLIQUE.

THEOREM 10. *There is a full tgd φ such that the C-repair checking problem w.r.t. φ is coNP-complete.*

PROOF. *(Sketch)* For every finite set of tgds, the C-repair checking problem is in coNP by Proposition 4. We will show that there is a full tgd for which this problem is coNP-hard via a reduction from the following variant of the MAXIMUM CLIQUE problem: given a graph $G = (V, E)$ and a subset V' of the set V of the nodes of V, is V' a clique of maximum cardinality in G?

We consider a relational vocabulary consisting of two unary relation symbol A and N, and a ternary relation symbol F. Let φ be the following full tgd:

$$\forall x, y, w(A(x) \wedge A(y) \wedge N(w) \rightarrow F(x, y, w)).$$

Suppose that we are given a graph $G = (V, E)$ and a subset V' of V, and we wish to determine whether or not V' is a clique of maximum cardinality in G. Without loss of generality, we may also assume that V is not a clique, but V' is a clique. In the full version of this paper, we show that we can construct in polynomial time two database instances r and r' such that V' is a clique of maximum cardinality in G if and only if r' is a C-repair of r. □

Our next result shows a dramatic difference in the complexity between the subset-repair checking problem and the C-repair checking problem for acyclic sets of inclusion dependencies and functional dependencies.

THEOREM 11. *There is an acyclic set Σ of inclusion dependencies and a functional dependency d such that the C-repair checking problem w.r.t. $\Sigma \cup \{d\}$ is coNP-complete.*

Theorem 11 is established via a delicate reduction from CUBIC POSITIVE-IN-3-SAT, which is the restriction of POSITIVE-IN-3-SAT to positive 3CNF-formulas in which every variable occurs exactly three times; this problem was shown to be NP-complete in [24].

Our final result shows that the CC-repair checking problem can be coNP-hard for all classes of constraints considered here.

THEOREM 12. *The following statements are true.*

1. *There is denial constraint χ such that the CC-repair checking problem w.r.t. χ is coNP-complete.*
2. *There is a full tgd θ such that the CC-repair problem w.r.t. θ is coNP-complete.*
3. *There is a LAV acyclic tgd ψ such that the CC-repair checking problem w.r.t. ψ is coNP-complete.*
4. *There is an acyclic set Ψ of inclusion dependencies such that the CC-repair problem w.r.t. Ψ is coNP-complete.*

PROOF. *(Sketch)* For every finite set of tgds, the CC-repair checking problem is in coNP by Proposition 4. The coNP-hardness for denial constraints is proved via a reduction from MAXIMUM INDEPENDENT SET using the denial constraint

$$\forall x, y(\neg A(x) \vee \neg A(y) \vee \neg F(x, y)).$$

The coNP-hardness for full tgds is proved via a reduction from MAXIMUM CLIQUE using the full tgd

$$\forall x, y(A(x) \wedge A(y) \rightarrow F(x, y)).$$

The last two coNP-hardness results are proved via suitable reductions from POSITIVE-1-IN-3-SAT. First, let ψ be the following LAV acyclic tgd:

$$\forall x, y, z, u, v, w(P(x, y, z) \rightarrow$$
$$\exists u, v, w(T(x, u) \wedge T(y, v) \wedge T(z, w) \wedge S(u, v, w))).$$

Finally, let Ψ be the set consisting of the following five inclusion dependencies (clearly, Ψ is an acyclic set):

$$\begin{aligned} P(x, y, z) &\rightarrow \exists u, v, w Q(x, y, z, u, v, w),\\ Q(x, y, z, u, v, w) &\rightarrow S(u, v, w)\\ Q(x, y, z, u, v, w) &\rightarrow T(x, u),\\ Q(x, y, z, u, v, w) &\rightarrow T(y, v),\\ Q(x, y, z, u, v, w) &\rightarrow T(z, w). \end{aligned}$$

In the full version of this paper, we show that the complement of POSITIVE-1-IN-3-SAT can be reduced in polynomial time to both the CC-repair checking problem w.r.t. to ψ and the CC-repair checking problem w.r.t. Ψ. □

5. Concluding Remarks

In this paper, we carried out a systematic investigation of the repair-checking problem for several different types of repairs and for various classes of integrity constraint encountered in database design, data integration, and data exchange. On the side of tractability, we gave a polynomial-time algorithm for the subset-repair checking problem w.r.t. every set of constraints that is the union of a weakly acyclic set of LAV tgds and a set of egds. This result significantly extends earlier tractability results for acyclic sets of inclusion dependencies and functional dependencies. We also gave a polynomial-time algorithm for the $\oplus$-repair checking problem w.r.t. a weakly acyclic set of LAV tgds. These two tractability results turn out to be optimal because we found a weakly acyclic set of non-LAV tgds for which the subset-repair checking problem is coNP-complete. Furthermore, we found a set that is the union of a weakly acyclic set of LAV tgds with a set of egds for which the $\oplus$-repair checking problem is coNP-complete.

We also studied the repair-checking problem for cardinality-based repairs. In addition to considering C-repairs which minimize the cardinality of the symmetric difference, we introduced and studied CC-repairs, a new type of cardinality-based repairs that have a Pareto optimality character. In general, C-repairs are stricter than CC-repairs, and CC-repairs are stricter than $\oplus$-repairs. We showed that both the C-repair checking problem and the CC-repair checking problem are intractable w.r.t. classes of integrity constraints for which the subset repair problem or the $\oplus$-repair problem are tractable. Thus, our results reveal that testing that a consistent instance repairs a database by minimizing the number of insertions and deletions is generally harder than testing that it is a subset-repair or a $\oplus$-repair.

The work reported here suggests several different research directions. On the side of theory, the definitive result in the study of the repair-checking problem would be a *dichotomy theorem* to the effect that, for every set Σ of tgds and egds and for every type T of repairs, either the T-repair checking problem w.r.t. Σ is in PTIME or it is coNP-complete. We conjecture that such a dichotomy theorem holds, even though its discovery is not in sight. On the side of practice, it would be interesting to identify natural syntactic conditions on various classes of integrity constraints used in applications for which the repair checking problem is solvable efficiently by algorithms of low running time. In a different direction, the results obtained here hint at a connection between data exchange and database repairs. In particular, a version of the chase algorithm (which is conspicuous in the study of algorithmic problems in data exchange) was used in proving the correctness of our polynomial-time algorithm for the subset-repair checking problem w.r.t. a set of constraints that is the union of a weakly acyclic set of LAV tgds and a set of egds (Theorem 2). It would be interesting to pursue this connection further and perhaps unveil a deeper relationship between algorithmic problems in data exchange and algorithmic problems concerning database repairs.

Acknowledgments We thank Leo Bertossi and Andrei Lopatenko for pointing out that Lemma 4 in their paper [22] implies Theorem 9. We also thank the four reviewers for their thorough reviews and helpful comments.

6. References

[1] M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'99)*, pages 68–79, 1999.

[2] O. Benjelloun, H. Garcia-Molina, H. Gong, H. Kawai, T. E. Larson, D. Menestrina, and S. Thavisomboon. D-swoosh: A family of algorithms for generic, distributed entity resolution. In *ICDCS*, page 37, 2007.

[3] L. E. Bertossi. Consistent query answering in databases. *SIGMOD Record*, 35(2):68–76, 2006.

[4] L. E. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. Data cleansing for numerical data sets. In *SEBD*, pages 292–299, 2005.

[5] L. E. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. Fixing inconsistent databases by updating numerical attributes. In *DEXA Workshops*, pages 854–858, 2005.

[6] L. E. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. Fixing inconsistent databases by updating numerical attributes. In *DEXA Workshops*, pages 854–858, 2005.

[7] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *ICDE*, pages 746–755, 2007.

[8] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD Conference*, pages 143–154, 2005.

[9] J. Chomicki. Consistent query answering: Five easy pieces. In *ICDT*, pages 1–17, 2007.

[10] J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1/2):90–121, 2005.

[11] A. Deutsch and V. Tannen. Reformulation of XML Queries and Constraints. In *ICDT*, pages 225–241, 2003.

[12] W. Eckerson. Data quality and the bottom line: Achieving business success through a commitment to high quality data. Technical report, The Data Warehousing Institute, 2002. http://www.tdwi.org/research/display.aspx?ID=6064.

[13] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005. Preliminary version in ICDT 2003.

[14] E. Franconi, A. L. Palma, N. Leone, S. Perri, and F. Scarcello. Census data repair: a challenging application of disjunctive logic programming. In *LPAR*, pages 561–578, 2001.

[15] A. Fuxman, P. G. Kolaitis, R. J. Miller, and W. C. Tan. Peer data exchange. *ACM Trans. Database Syst.*, 31(4):1454–1498, 2006. Preliminary version in PODS 2005.

[16] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita. Improving data cleaning quality using a data lineage facility. In *DMDW*, page 3, 2001.

[17] G. Gottlob and A. Nash. Data exchange: computing cores in polynomial time. In *PODS*, pages 40–49, 2006.

[18] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, 1995.

[19] M. A. Hernández and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Min. Knowl. Discov.*, 2(1):9–37, 1998.

[20] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.

[21] M. Lenzerini. Data Integration: A Theoretical Perspective. pages 233–246, 2002.

[22] A. Lopatenko and L. E. Bertossi. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *ICDT*, pages 179–193, 2007.

[23] D. Menestrina, O. Benjelloun, and H. Garcia-Molina. Generic entity resolution with data confidences. In *CleanDB*, 2006.

[24] C. Moore and J. M. Robson. Hard tiling problems with simple tiles. *Discrete and Computational Geometry*, 26(4):573–590, 2001.

[25] E. Rahm and H. Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4), 2000.

[26] V. Raman and J. M. Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB*, pages 381–390, 2001.

[27] S. Staworko. *Declarative Inconsistency Handling in Relational and Semi-structured databases*. PhD thesis, May 2007.

[28] J. Wijsen. Database repairing using updates. *ACM Trans. Database Syst.*, 30(3):722–768, 2005.