Distance-Bounded Consistent Query Answering*

Andreas Pfandler[1,2] **and Emanuel Sallinger**[1]
lastname@dbai.tuwien.ac.at
[1]Vienna University of Technology, Austria
[2]University of Siegen, Germany

Abstract

The ability to perform reasoning on inconsistent data is a central problem both for AI and database research. One approach to deal with this situation is consistent query answering, where queries are answered over all possible repairs of the database. In general, the repair may be very distant from the original database. In this work we present a new approach where this distance is bounded and analyze its computational complexity. Our results show that in many (but not all) cases the complexity drops.

1 Introduction

The ability to perform reasoning on inconsistent data is a central problem both for AI and database research. An important scenario in this area is when a database is augmented with a set of integrity constraints (ICs), typically specified in some logical formalism. In this scenario, it might well happen that the database becomes inconsistent w.r.t. the constraints.

Following Bertossi [2006], there are several reasons for a database to become inconsistent w.r.t. a given set of integrity constraints: (i) The database system may not be able to maintain a given class of ICs. (ii) Adding constraints may lead to inconsistency in existing data. (iii) The constraints are not intended to be hard constraints, but rather as soft constraints that are only considered when a user makes a query. (iv) Data from different sources is integrated into a single database.

This shows that, although it would be desirable to limit the attention to consistent databases only, there is a need for methods to give reasonable answers to queries even on inconsistent databases. One approach to deal with inconsistent databases is *consistent query answering* introduced by Arenas *et al.* [1999]. In this approach, queries are evaluated over all *repairs* of the database, which gives the *consistent answers*. A repair is a database that is consistent with the constraints and has a subset minimal edit distance to the original database. For surveys on this line of research we refer to the work of Bertossi [2006], Chomicki [2007], the monograph of Bertossi [2011], and for recent work to [Arenas and Bertossi, 2010; Kolaitis and Pema, 2012; Fontaine, 2013; Greco *et al.*, 2014; Wijsen, 2014]. Although we will focus on database-related constraint classes, we note that similar concepts have gained recent attention in the area of description logics (see e.g. [Ortiz, 2013]).

For consistent query answering, two fundamental problems arise: In REPAIR CHECKING (RC), we want to check whether a given database R is indeed a repair of a given database D w.r.t. the constraints C. In CONSISTENT QUERY ANSWERING (CQA), we want to evaluate whether a conjunctive query Q holds in all repairs R. The computational complexity of these problems has been intensively studied [Calì *et al.*, 2003a; 2003b; Staworko and Chomicki, 2010; ten Cate *et al.*, 2012; Arming *et al.*, 2014]. In these studies it turned out that for many interesting constraint languages the problem of CQA is of high computational complexity (e.g., Π_2^{P}-completeness for functional dependencies) or even undecidable (e.g., for the well-known class of existential rules – also called tgds), which imposes a severe obstacle to practical applications. It turns out that the reason for this high complexity is in many cases related to the fact that the distance to the original database may grow arbitrarily large.

In this work, we introduce a bounded version of RC and CQA, where an additional bound on the distance (i.e., number of changes) to the original database is given. This step is motivated by the fact that inconsistencies are often limited to a small portion of the data in the typically large database. In addition, this new approach is interesting because it usually gives lower complexity bounds on other CQA formalisms. For the bounded case all considered constraint classes become decidable and in many cases the complexity drops. There are, however, possibly surprising cases where the complexity increases. We illustrate the usefulness of our approach in the following example.

Example. Let the database contain the facts

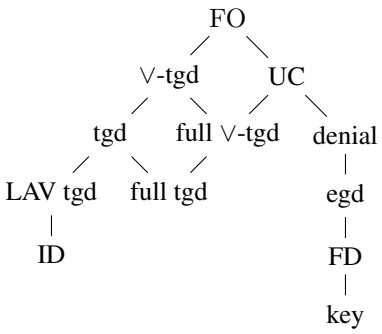
$$D = \{Comp(c_1), Emp(e_1, d_1, c_1), \ldots, Emp(e_l, d_1, c_1), Emp(e_{l+1}, d_2, c_1), \ldots, Emp(e_m, d_2, c_1)\}.$$

The constraints for this example are given by

$$C = \{Comp(c) \wedge Emp(e, d, c) \rightarrow Dep(d, c)\},$$

which states that whenever there is an employee working at a company in a certain department, this department must also be associated to this company.

*Supported by the Austrian Science Fund (FWF): P25518, P25207, and Y698, and the German Research Foundation (DFG): ER 738/2-1.



IC $\mathcal{L}$	b-RC		RC*	b-CQA		CQA*
FO	PSPACE	▲▼3	PSPACE	PSPACE	▲▼10	undec
∨-tgd	DP_2	▼5	Π_3^P	Π_3^P	▼11	undec
tgd	DP_2	▲6	Π_3^P	Π_3^P	▲12	undec
LAV tgd	DP	▲▼7	DP	Π_2^P	▼13	NP
ID	in P	▼8	in P	Π_2^P	▲14	NP
UC	DP	▼4	Π_2^P	DP-Π_2^P	▼15	coNEXP-P^{NE}
full ∨-tgd	DP		Π_2^P	DP-Π_2^P		coNEXP-P^{NE}
full tgd	DP	▲9	DP	DP-Π_2^P	▲16	EXP-coNEXP
denial	DP		DP	DP-Π_2^P		Π_2^P
egd	DP	▲9	DP	DP-Π_2^P	▲16	Π_2^P
FD	in P	▼8	in P	NP	▼17	Π_2^P
key	in P		in P	NP	▲18	Π_2^P

Figure 1: *Left:* Hasse diagram of the hierarchy of IC languages. *Right:* Complexity of b-RC and b-CQA shown in this paper and known results for RC and CQA marked by * (cf. [Arming *et al.*, 2014]). All results denote completeness results, except when given in the form *lower - upper* bound. Triangles refer to results for lower (▲) and upper (▼) bounds shown in this paper.

Observe that $D \not\models C$ and that there are three repairs for this instance. First, we can delete all employees, resulting in $R_1 = \{Comp(c_1)\}$. Second, we can create two departments, which yields $R_2 = D \cup \{Dep(d_1, c_1), Dep(d_2, c_1)\}$. Third, we can delete the company, resulting in $R = D \setminus \{Comp(c_1)\}$. A closer look at the repairs reveals that their nature is quite different. There are applications where R_1, i.e., deleting all employees, is too radical. Yet, in the classical setting of CQA, R_1 is considered equally good as R_2, i.e., solving the inconsistency by inserting two departments.

Notice that deleting the whole database always fulfills the constraints (for all classes of constraints considered in this work). Hence, for many instances there is a repair that deletes large fractions of the database in order to obtain consistency. This can, however, be an undesired repair, as a large part of the information of the original database is lost. This changes also the behavior for queries: Asking whether there is at least one employee in any repair is answered with no for CQA, but with yes if we admit only repairs with at most two changes.

One might argue that a way to overcome these problems is to consider cardinality minimal repairs only. There are, however, instances for which this restriction is too strong and excludes reasonable repairs. In the above example, this restriction would only allow repair R_3, i.e., deleting the company. Yet, this is rather short-sighted as in many scenarios repair R_2, i.e., adding two departments, is also a reasonable repair. Also considering only the top-k repairs, can be disadvantageous, as even small values for k can include repairs with a very high distance to the original database.

Furthermore, notice that there is a problematic asymmetry between insertions and deletions: While the number of possible deletions is bounded by the input size, the number of insertions needed to obtain consistency might be arbitrarily large. This effect is also the root for various undecidability results known for CQA.

Contributions. We introduce a natural version of RC and CQA where the distance to the original database is bounded. This new notion leads to more natural answers in certain scenarios, as discussed in the example above, whereas it preserves the nice properties of CQA.

We give a detailed complexity analysis (of the combined complexity) for twelve important constraint classes and show that in many cases the complexity drops. For instance, the complexity drops from undecidable to Π_3^P for disjunctive tgds and from Π_2^P to NP for functional dependencies. This shows that the bound captures an important source of complexity. Interestingly, there is the notable case of LAV tgds and inclusion dependencies where the complexity increases from NP to Π_2^P. For an overview on the results see Figure 1.

2 Preliminaries

We assume familiarity with the basics of complexity theory and logic. Recall that a Boolean conjunctive query is a first-order formula of the form $\exists\overline{x}\, R_1(\overline{x}_1) \wedge \cdots \wedge R_n(\overline{x}_n)$ where $\overline{x}_i \subseteq \overline{x}$ and R_i are (not necessarily distinct) relation symbols. Furthermore, similarly to the class DP, the class DP_2 contains all problems that consist of the conjunction of two independent problems from Σ_2^P and Π_2^P. Unless stated otherwise, we use n to denote the size of the input.

Consistent query answering. Given a set of ICs C and a database instance D, a *repair* of D w.r.t. C is a database instance R which satisfies C and which *differs minimally* from D. Difference and minimality can be defined in several ways. We follow the approach of [Arenas *et al.*, 1999] where repairs are obtained from the original database by insertion and deletion of tuples and minimality means that the symmetric set difference $\Delta(D, R)$ is minimal w.r.t. subset inclusion. More formally, let $\Delta(D, R) = (D \setminus R) \cup (R \setminus D)$. Then R is a repair of D w.r.t. C if $R \models C$ and there is no instance R' with $R' \models C$ and $\Delta(D, R') \subsetneq \Delta(D, R)$.

The idea of consistent query answers is that even from an inconsistent database instance D, we can derive consistent information, namely those answers to a query that one would obtain over every repair R of D. More precisely, the set of consistent answers to a query Q for a given database D and ICs C is defined as $\bigcap\{Q(R) \mid R \text{ is a repair of } D \text{ w.r.t. } C\}$.

The following two decision problems are crucial to deal with inconsistent data. In the REPAIR CHECKING (RC) prob-

lem, we are given databases D and R and a set of constraints C, and the question is whether R is a repair of D w.r.t. C. In the CONSISTENT QUERY ANSWERING (CQA) problem, we are given a database D, a set of constraints C, and a Boolean conjunctive query Q, and the question is whether Q is true in all repairs of D w.r.t. C.

Constraint classes. We now introduce the considered IC languages. The left part of Figure 1 shows their relationship.

Besides *domain independent first order (FO)* sentences, all languages studied in this paper arise from restrictions on formulas of the form $\forall\overline{x}\,(\varphi(\overline{x}) \wedge \beta(\overline{x}) \rightarrow \bigvee_{i=1}^{n} \exists\overline{y_i}\,\psi_i(\overline{x},\overline{y_i}))$ where φ, ψ_i are conjunctions of database atoms and β a quantifier-free formula using only built-in (i.e. comparison) predicates. To ensure safety, we require that every variable in $\overline{x}$ must occur in some atom in φ.

We call such a constraint *full*, or a *universal constraint (UC)*, if it contains no existential quantifiers. A *disjunctive tuple-generating dependency ($\vee$-tgd)* has empty β, while a *tuple-generating dependency (tgd)* additionally has $n = 1$. A *local-as-view (LAV) tgd* is a tgd where φ is a single atom, and an *inclusion dependency (ID)* is a LAV tgd where also ψ_1 is a single atom. A *denial constraint* is of the form $\forall\overline{x}\,\neg(\varphi(\overline{x}) \wedge \beta(\overline{x}))$ and can be thought of as a universal constraint with empty right hand side. An *equality-generating dependency (egd)* is a denial constraint where β is a single inequality. A *functional dependency (FD)* is an egd where φ consists of two atoms over the same relation symbol. We recall that *key constraints* are a special case of FDs.

Connections to OBQA. Before introducing our new approach to CQA in the next section, we want to illustrate the connection between CQA and *ontology-based query answering* (OBQA), which has seen a lot of attention in the description logics (DL) community over the last decade (see, e.g., [Ortiz, 2013]). In OBQA, a query is evaluated over a knowledgebase – that is a database (i.e., an ABox) together with an ontology (i.e., a TBox). Similarly to the database setting, the problem of inconsistency also arises in OBQA. In fact, inspired by the work on CQA, the concept of AR-semantics was introduced by Lembo and Ruzzi [2007] (and later given the name *AR-semantics* by Lembo *et al.* [2010]) to deal with inconsistencies. This topic has found a lot of attention as witnessed by a number of publications related to DLs [Bienvenu, 2012; Bienvenu and Rosati, 2013; Bienvenu *et al.*, 2014; Lembo *et al.*, 2012] and recently also related to existential rules (i.e., tgds) [Lukasiewicz *et al.*, 2015].

Technically, there are a lot of similarities between CQA and AR-based OBQA, but also subtle yet important differences. The key difference is in what the term "inconsistency" means and how inconsistencies can be repaired: In CQA, a database D is inconsistent w.r.t. a set of constraints C, if database D violates at least one constraint from C. Such an inconsistency can be repaired by adding or deleting tuples from D. A query Q is now directly evaluated over all repairs.

In OBQA with AR-semantics, "inconsistency" means that the knowledgebase, that is the ABox together with the TBox, does not have a model. Through modifying the ABox, such an inconsistency can only be repaired by removing tuples, not by adding tuples. Note that adding a tuple to the ABox only decreases the number of possible models, but may never make an inconsistent knowledgebase consistent. Thus we now have a number of repaired knowledgebases, consisting of modified ABoxes and unmodified TBoxes. A query Q is now evaluated over these repaired knowledgebases as typical in OBQA, i.e., by checking whether the ABox together with the TBox entails query Q. Implicit in this step is the addition of tuples as forced by the TBox.

Altogether, both CQA and AR-based OBQA consider additions and deletions of tuples, but in different phases. CQA considers additions and deletions when computing repairs, whereas AR-based OBQA considers only deletions when computing repairs, but implicitly allows additions during query evaluation. A subtle but crucial difference between both settings arises from the required minimality of changes. That is, it can make a difference whether minimality is required of additions and deletions at the same time (as in CQA) or for deletions alone (as in OBQA with AR-semantics).

Yet, if we restrict ourselves to certain classes of *purely negative constraints*, these two formalisms coincide. In this work, we consider several important kinds of negative constraints, such as denial constraints, equality-generating dependencies (egds) and functional dependencies (cf. the hierarchy of constraints shown in Figure 1). Note that while egds and functional dependencies are usually given with equalities in the conclusion (and thus, syntactically, do not look like negative constraints), they can be equivalently formulated with inequalities in the antecedent and $\bot$ as the conclusion.

3 New Approach and Overview of Results

Motivated by the discussion presented in the introduction, we now introduce bounded variants of RC and CQA. The only modification compared to the original version is that $|\Delta(D, R)|$ is bounded by a fixed b.

b-REPAIR CHECKING (b-RC)
Instance: Databases D and R and a set of constraints C.
Question: Is R a repair of D w.r.t. C s.t. $|\Delta(D, R)| \leq b$?

b-CONSISTENT QUERY ANSWERING (b-CQA)
Instance: A database D, a set of constraints C, and a Boolean conjunctive query Q.
Question: Is Q true in all repairs R of D w.r.t. C where $|\Delta(D, R)| \leq b$?

While there is always at least one repair for CQA, this is not necessarily the case for b-CQA (as the repair candidate might exceed the bound b). In case that there are no repairs, we fix the answer to b-CQA as "no".

Note that there are several complexity measures that can be investigated for these problems (as database, constraints and/or query can be considered to be fixed). In this work, we focus on the combined complexity, as it gives an upper bound for all of these complexity measures.

Before we move on to the complexity analysis of particular classes of constraints, we present two results on the general relationship of the bounded and unbounded case.

Proposition 1. *b-RC and b-CQA for a class of constraints $\mathcal{C}$ is at least as hard as model-checking for class $\mathcal{C}$.*

Proof. To see this, let $b = 0$. Hence, there is only one possible repair $R = D$. For bounded repair checking, it is easy to see that the answer is yes in case that $D \models C$ and no in case that $D \not\models C$. That is, it is exactly the model checking problem for D and C.

For b-CQA, we first note that since the only possible repair is $R = D$, the minimality check of b-CQA is trivially true. It only remains to be checked whether $D \models C$ and $D \models Q$. Query Q can be constructed to be trivially fulfilled, i.e., we add to database D an additional atom $R(c)$ where R is a relation symbol not yet occurring in D, the value c is arbitrary, and we ask the query $\exists x\, R(x)$. Thus also b-CQA comes down to the model checking problem of D and C. □

Proposition 2. *b-RC for a class of constraints $\mathcal{C}$ is at most as hard as RC for class $\mathcal{C}$.*

Proof. We show this result by presenting a reduction from b-RC to RC. First we check whether $|\Delta(D, R)| \leq b$. In this case, it is sufficient to pass the b-RC instance to the RC problem. Otherwise, we return a trivial no-instance as the size bound is violated. Observe that this procedure can be carried out in LOGSPACE. □

These two results will turn out to be very useful as they allow us to directly build upon the results known for CQA and RC.

Overview of results. In the sections that follow, we will present a complexity analysis (of the combined complexity) of the b-RC and b-CQA problem for a wide variety of classes of integrity constraints. For an overview we refer to Figure 1. If we compare b-RC with RC, it turns out that b-RC is never harder than RC. In contrast, if we compare b-CQA with CQA, the picture is different. For FO constraints, $\vee$-tgds, and tgds CQA is undecidable, but moving to b-CQA, we obtain PSPACE-completeness for FO, and Π_3^{P}-completeness for $\vee$-tgds and tgds. The reason for this change is that in these cases, hardness is caused by a very large (in general, unboundedly large) difference between the repair and the original database.

The case of LAV tgds and ID constraints is possibly surprising, as the complexity increases from NP to Π_2^{P} when we move from CQA to b-CQA. This is because for CQA, it is possible to limit the attention to a unique repair R_u, which is obtained through deletions only. Since for CQA a query Q is true in all repairs if and only if it is true in R_u, both problems can be solved in NP (implicit in [Arming *et al.*, 2014]). However, for b-CQA this trick does not always work, since it might be that R_u exceeds the bound b, and thus is no repair at all.

In all other cases the complexity stays the same or drops. Notice that in the cases where a tight classification for CQA was open [Arming *et al.*, 2014], we do not have matching upper and lower bounds for b-CQA either, but we can show that the complexity is guaranteed to drop.

4 Bounded Repair Checking

We start our complexity analysis by studying b-RC. From Proposition 2 we know that b-RC is at most as hard as RC, but we will show that in many cases the complexity actually drops.

Proposition 3. *b-RC for FO is* PSPACE*-complete.*

Proof. Note that in the work of Arming *et al.* [2014] it was shown that RC is PSPACE-complete. Hence, by Proposition 1 (taking into account that model-checking for FO is well known to be in PSPACE) and by Proposition 2 we obtain that b-RC is also PSPACE-complete. □

We now move from first-order constraints to universal constraints and show that repair checking can be solved by two calls to an NP-oracle.

Theorem 4. *b-RC for UCs is in* DP.

Proof. We show this by reduction to SAT/UNSAT which is known to be DP-complete. We need to show that we can encode an arbitrary b-RC instance into a SAT/UNSAT instance, i.e., a pair of formulas (φ, ψ) such that $\varphi \not\models \bot$ and $\psi \models \bot$ if and only if the b-RC instance is a yes-instance.

Encoding the check whether $R \models C$ into ψ is straightforward as model-checking for UCs is coNP-complete (cf. [Pichler and Skritek, 2011]). Let the formula encoding this check be denoted by ψ_C. What remains to be shown is how to encode the minimality check of R. More precisely, we need to show that $\Delta(D, R)$ is subset minimal, i.e., there is no other R' with $R' \models C$ and $\Delta(D, R') \subsetneq \Delta(D, R)$. To this end, we consider instances R_i, which are induced by a strict subset of $\Delta(D, R)$. Because $\Delta(D, R)$ is of size at most b there are at most $2^b - 1$ such instances. Hence we can enumerate these $R_1, \ldots, R_{2^b-1}$ in polynomial time (as b is fixed). Recall that model-checking for UCs is coNP-complete. Therefore, $R_i \not\models C$ can be decided in NP. For each R_i, we create a formula φ_i that is satisfiable if and only if $R_i \not\models C$. W.l.o.g. we assume that the variables of all φ_i, with $1 \leq i \leq 2^b - 1$, are disjoint. Taken together, the instance for SAT/UNSAT is given by $(\varphi \coloneqq \bigwedge_{i=1}^{2^b-1} \varphi_i, \psi \coloneqq \psi_C)$. □

Next, we show that for $\vee$-tgds and tgds the complexity increases by one level.

Theorem 5. *b-RC for $\vee$-tgds is in* DP_2.

Proof. The membership proof for this problem is similar to the proof of Theorem 4. The important difference is, however, that model-checking for $\vee$-tgds is Π_2^{P}-complete (cf. [Pichler and Skritek, 2011]). Thus the formula needed to encode whether $R \models C$ is an $\forall\exists$-QBF, namely $\forall \overline{x}\, \exists \overline{y}\, \psi'(\overline{x}, \overline{y})$. For each smaller repair R_i, we need to verify that $R_i \not\models C$. This done by creating a $\exists\forall$-QBF, $\exists \overline{x}_i\, \forall \overline{y}_i\, \varphi_i(\overline{x}_i, \overline{y}_i)$, that is valid if and only if $R_i \not\models C$ (again assuming w.l.o.g. that the formulas φ_i do not share any variables.) We obtain an $\exists$-QSAT$_2$/$\forall$-QSAT$_2$ instance (which is clearly DP_2-complete) by putting the QBF together accordingly: $(\varphi \coloneqq \exists \overline{x}_1 \cdots \overline{x}_{2^b-1}\, \forall \overline{y}_1 \cdots \overline{y}_{2^b-1} \bigwedge_{i=1}^{2^b-1} \varphi_i(\overline{x}_i, \overline{y}_i), \psi \coloneqq \forall \overline{x}\, \exists \overline{y}\, \psi'(\overline{x}, \overline{y}))$. □

Theorem 6. *b-RC for tgds is DP_2-hard.*

Proof. We show this by a reduction from the $\exists$-QSAT$_2$/$\forall$-QSAT$_2$ problem, which is clearly DP_2-complete. Since model-checking for tgds is Π_2^{P}-complete (cf. [Pichler and Skritek, 2011]), we will actually reduce from the problem where we

are given two databases D_1, D_2 and two sets of tgds C_1, C_2. The question is whether $D_1 \not\models C_1$ *and* $D_2 \models C_2$. W.l.o.g. we assume that D_1 and D_2 as well as C_1 and C_2 do not share any symbols. We construct the b-RC instance for $b = 1$ as follows. For C_1 (C_2) we create a set of tgds C_1' (C_2') that is obtained from C_1 (C_2) by adding g' (g) to the body of each tgd. Let now $D := D_1 \cup D_2 \cup \{g, g'\}$, $R := D \setminus \{g'\}$, and $C := C_1' \cup C_2'$. It is now easy to verify that R is a repair if and only if $D_2 \models C_2'$ (otherwise $R \models C$ would not hold) and $D_1 \not\models C_1'$ (otherwise $R' = C$ would also be a repair and as a consequence R would not be minimal). Furthermore, notice that $|\Delta(D, R)| = 1 \leq b$. □

The remaining three results can be shown by building upon results from the literature.

Proposition 7. *b-RC for LAV tgds is* DP*-complete.*

Proof. Note that membership can again be obtained by Proposition 2 and the result for RC [Arming *et al.*, 2014]. Hardness can be shown similarly to the hardness proof of Arming *et al.* [2014] (cf. the full version of the paper). □

Proposition 8. *b-RC for FDs and IDs is in* P*.*

Proof. As RC for FD and ID constraints is in P [Arming *et al.*, 2014] this holds by Proposition 2 also for b-RC. □

Proposition 9. *b-RC for full tgds and egds is* DP*-hard.*

Proof. Hardness can be shown similarly to the hardness proofs of Arming *et al.* [2014] (cf. the full version of the paper). □

5 Bounded Consistent Query Answering

We now move to the study of b-CQA. Here we will see that in many cases the complexity drops compared to CQA. In particular, the undecidable cases become decidable. Yet interestingly, for LAV tgds and IDs, the complexity actually increases.

For first-order constraints, we see that the complexity for b-CQA and RC coincides.

Theorem 10. *b-CQA for FO is* PSPACE*-complete.*

Proof. As mentioned earlier, model-checking for FO is well known to be PSPACE-complete. Hence hardness for PSPACE follows by Proposition 1. Membership in PSPACE can be seen as follows. First, notice that the representation of a single addition/deletion is always of polynomial size (observe that the maximum arity is trivially bounded by the size of the input). Hence, the largest set $\Delta(D, R)$ is of size at most $O(b \cdot n)$. Therefore, also each repair R is of size polynomial in the input. We can now use the following procedure to decide the problem. We guess a repair candidate and verify whether it is a repair, i.e., whether it fulfills the FO constraints and that there is no "smaller" repair. Notice that the number of smaller repairs is also polynomial as b is fixed. For each of the repairs obtained in this way, we ensure that the query Q is fulfilled.

In more detail, we iterate over all repair candidates R (of polynomial size) and verify whether the FO constraints C are satisfied. If this is the case, we look for a "smaller" repair candidate R' such that $\Delta(D, R') \subsetneq \Delta(D, R)$ and $R' \models C$. In case R does not satisfy the constraints or there is a smaller repair candidate R', we drop R and continue with the next repair candidate until all candidates have been considered. Otherwise, we check whether repair R satisfies the query Q (this can be done in PSPACE, as model checking for conjunctive queries is known to be in NP). If no repair exists or a repair $R \not\models Q$, the procedure returns no. Otherwise, if no choice of R yields such a counterexample, the procedure returns yes. □

Note that in contrast to the above result, CQA is undecidable for first-order constraints. The same holds for $\vee$-tgds and tgds, where the complexity even drops to the third level of the polynomial hierarchy.

Theorem 11. *b-CQA for $\vee$-tgds is in Π_3^{P}.*

Proof. This follows from a naive guess and check algorithm, together with the fact that model-checking for $\vee$-tgds is Π_2^{P}-complete. We can simply guess a repair candidate R, because, as argued in the proof of Theorem 10, the largest set $\Delta(D, R)$ is of size at most $O(b \cdot n)$. For each candidate we check whether it fulfills constraints C, which can be done by a call to a Π_2^{P}-oracle. Then, we verify that R is indeed minimal and hence a repair, by iterating over all smaller repair candidates R' with $\Delta(D, R') \subsetneq \Delta(D, R)$ (at most $2^b - 1$ many) and verify that $R' \not\models C$ by a call to a Σ_2^{P}-oracle. Finally, it remains to be checked whether $R \models Q$, which is done by a single call to an NP-oracle (as model checking for conjunctive queries is known to be in NP). If this is the case, the procedure continues until all repair candidates have been considered, which means that it can terminate with answer yes. Otherwise, or if no repair exists, the procedure terminates with no. □

In some of the following proofs, we make use of the following notation. Let φ be the 3CNF formula $\bigwedge_i (l_{i1} \vee l_{i2} \vee l_{i3})$. We denote by φ^* the conjunction $\bigwedge_i c(l_{i1}^*, l_{i2}^*, l_{i3}^*)$ where $l_{ij}^* = x$ if l_{ij} is the positive literal x and $l_{ij}^* = \overline{x}$, if l_{ij} is the negative literal $\neg x$. For example, $[(x \vee \neg z \vee y) \wedge (\neg z \vee y \vee \neg y)]^* = c(x, \overline{z}, y) \wedge c(\overline{z}, y, \overline{y})$. Furthermore, let $\hat{c} := \{c(x, y, z) \mid x, y, z \in \{0, 1\}\} \setminus \{c(0, 0, 0)\}$.

Theorem 12. *b-CQA for tgds is Π_3^{P}-hard.*

Proof. We proceed by a reduction from $\exists$QSAT$_3$ to the co-problem of b-CQA. Let the instance of $\exists$QSAT$_3$ be a QBF of the form $\varphi = \exists x_1 \cdots x_k \, \forall y_1 \cdots y_l \, \exists z_1 \cdots z_m \, \psi$, where ψ is a propositional formula over $x_1, \ldots, x_k, y_1, \ldots, y_l, z_1, \ldots, z_m$ in 3CNF. We set $b = 2$ and construct the instance of the co-problem of b-CQA as follows.

$$
\begin{aligned}
D = \; & \hat{c} \cup \{\mathit{True}(1), \mathit{False}(0), V(0), V(1), \mathit{Neg}(0,1), \\
& \mathit{Neg}(1,0)\} \cup \{\mathit{Neg}_i(0,1), \mathit{Neg}_i(1,0) \mid 1 \leq i \leq 3\} \cup \\
& \{\mathit{True}_i(1), \mathit{False}_i(0), V_i(0), V_i(1) \mid 1 \leq i \leq 3\} \\
Q = \; & \mathit{False}(f) \wedge c(f, f, f)
\end{aligned}
$$

Let C be given as the set of the following formulas:

1. $\mathit{True}(v) \rightarrow \exists x_1 \cdots x_k X(x_1, \ldots, x_k)$
2. $X(x_1, \ldots, x_k) \wedge \bigwedge_{1 \leq i \leq k} \mathit{Neg}(x_i, \overline{x}_i) \wedge \bigwedge_{1 \leq i \leq l} \mathit{Neg}(y_i, \overline{y}_i) \rightarrow \exists z_1 \cdots z_m \bigwedge_{1 \leq i \leq m} \mathit{Neg}(z_i, \overline{z}_i) \wedge \psi^*$

3. $\{X(x_1, \ldots, x_i, \ldots, x_k) \rightarrow V(x_i) \mid 1 \leq i \leq k\}$
4. $\{Neg_i(x, y) \rightarrow Neg(x, y), Neg(x, y) \rightarrow Neg_i(x, y),$
   $\mathcal{R}_i(x) \rightarrow \mathcal{R}(x), \mathcal{R}(x) \rightarrow \mathcal{R}_i(x) \mid$
   $1 \leq i \leq 3, \mathcal{R} \in \{True, False, V\}\}$

First, notice that due to the constraints in block (4) of C (together with the distance bound b) it is not possible to insert or delete any facts over the relations $True$, $False$, V, and Neg, as fulfilling these constraints would immediately exceed the bound b. Also, removing facts from the relation $c(\cdot, \cdot, \cdot)$ is of no help, as this relation occurs only in the head of a tgd.

Taking a closer look at the constraints, we notice that there are only two types of repair candidates. Type (i): A single fact $X(x_1, \ldots, x_k)$ is added, where $x_1, \ldots, x_k \in \{0, 1\}$ by block (3) of C; or Type (ii): Where type (i) is extended by adding the fact $c(0, 0, 0)$, which is the only missing fact of this relation as $\hat{c}$ is already present in D.

We now argue why we only need to consider these two types of repair candidates. Notice that a fact over relation X must be inserted in any repair, as this is enforced by the constraint (1) in C (and the fact that $True \neq \emptyset$ always holds, as discussed before). Furthermore, a repair will only contain the insertion of a single fact over relation X. Otherwise this would contradict the minimality of the repair, as constraint (1) is already satisfied by the first fact over X and adding second fact over X makes blocks (2) and (3) of constraints harder to satisfy rather than easier. For type (ii), observe that adding the fact $c(0, 0, 0)$ is optional and provides a backdoor to satisfy the constraint (2) vacuously. A repair candidate of type (ii) will only be minimal if for the given fact over relation X the second constraint cannot be fulfilled without this backdoor. We now claim that φ is true if and only if there is a repair R that does not contain $c(0, 0, 0)$, and as a consequence $R \not\models Q$.

Below, we give the intuition why this correspondence indeed holds. The intended meaning of the constraint (2) is to model the QBF φ unless the backdoor fact $c(0, 0, 0)$ is inserted in the repair. That is, for an assignment to the x_i variables (represented by the X relation in the repair) and for any assignment to the y_i variables (in the body of this constraint) there is an assignment to the z_i variables such that ψ (represented by ψ^*) is satisfied. The constructed instance of the co-problem of b-CQA is a no-instance (i.e., for any repair R it holds that $R \models Q$) if $c(0, 0, 0)$ is contained in all repairs. Therefore, we know that for any assignment to the x_i variables (represented by the X relation), there exists some assignment to the y_i variables such that there was no way to satisfy the formula ψ since otherwise $c(0, 0, 0)$ would have been dispensable in some repair. Thus, φ must be false.

Conversely, assume that φ is false. Then we know that for all assignments to the x_i variables there is an assignment to the y_i variables such that any assignment to the z_i variables makes ψ false. Therefore, for any assignment to the X relation in the repair, the repair will violate constraint (2) unless $c(0, 0, 0)$ is also contained in the repair to enable the backdoor. As a consequence, for any repair R, it holds that $R \models Q$, and hence the constructed instance of the co-problem of b-CQA is indeed a no-instance. □

In the next two results, we see the only cases where the complexity increases when we move from CQA to b-CQA.

Proposition 13. *b-CQA for LAV tgds is in Π_2^{P}.*

Proof. The membership proof is analogous to the proof of Theorem 11, with the difference that model-checking for LAV tgds is coNP-complete. This is the reason why the complexity drops by one level in the polynomial hierarchy. □

Intuitively, an explanation for this increase of complexity is the following. In the case of CQA, it suffices to limit the search to a single unique repair R_u, which can be obtained by using deletions only. Then we only need to check whether Q is true in R_u, to decide whether a query Q is true in all repairs. However, this trick does not work for b-CQA as R_u might be excluded from the repairs for exceeding the bound b, and as a consequence make the inspection of further repairs necessary.

Theorem 14. *b-CQA for IDs is Π_2^{P}-hard.*

Proof. We establish this result by a reduction from $\exists\text{QSAT}_2$, which is well-known to be Π_2^{P}-complete, to the co-problem of b-CQA. An instance of $\exists\text{QSAT}_2$ is given by a QBF of the form $\varphi = \exists x_1 \cdots x_k \forall y_1 \cdots y_l \, \psi$, where ψ is a propositional formula in 3DNF over $x_1, \ldots, x_k, y_1, \ldots, y_l$. Let $b = 1$ and $\chi \equiv \neg\psi$, which is known to be in 3CNF since ψ is in 3DNF. The instance of the co-problem of b-CQA is given as follows.

$$
\begin{aligned}
D &= \hat{c} \cup \{Neg(0, 1), Neg(1, 0), V(0), V(1)\} \\
C &= \{ V(v) \rightarrow \exists x_1 \cdots x_k X(x_1, \ldots, x_k), \\
&\qquad Neg(x, y) \rightarrow V(x), Neg(x, y) \rightarrow V(y), \\
&\qquad X(x_1, \ldots, x_i, \ldots, x_k) \rightarrow V(x_i) \mid 1 \leq i \leq k\} \\
Q &= \chi^* \wedge \bigwedge_{1 \leq i \leq l} Neg(y_i, \overline{y}_i)
\end{aligned}
$$

Notice that, since $b = 1$, it is not possible to delete both $Neg(0, 1)$ and $Neg(1, 0)$ from D. Therefore, it is ensured by the constraints with the body $Neg(X, Y)$ that only an insertion can be used to obtain a repair. This insertion will be a fact of the form $X(x_1, \ldots, x_k)$ where $x_1, \ldots, x_k \in \{0, 1\}$, which is ensured by the first constraint together with the last group of constraints. Now observe that there is a repair R such that $R \not\models Q$ if and only if there is an assignment – expressed by the inserted fact of the form $X(x_1, \ldots, x_k)$ – such that there is no assignment to the variables $y_1, \ldots, y_l$ where χ is satisfied, or equivalently, ψ is unsatisfied. In other words, (D, C, Q) is a no-instance (witnessed by a repair R such that $R \not\models Q$) if and only if φ is a yes-instance. □

We continue our investigation with universal constraints.

Proposition 15. *b-CQA for UCs is in Π_2^{P}.*

Proof. Since model-checking for UCs is coNP-complete, membership can be seen by analogous arguments as in the proof of Proposition 13. □

The next result shows a lower complexity bound, i.e., DP-hardness, for egds and full tgds.

Theorem 16. *b-CQA for egds and full tgds is DP-hard.*

Proof. We first show the result for egds. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two undirected graphs. Since 3-COLORABILITY is NP-complete, it is DP-complete to decide whether G_1 is 3-colorable and G_2 is not. We now present a reduction to b-CQA for egds. To this end, we construct a b-CQA instance for $b = 0$ as follows.

$$D = \{C(1,2), C(1,3), C(2,1), C(3,1), C(3,2), \\ \mathit{True}(1), \mathit{False}(0)\}$$

$$C = \{\mathit{True}(t) \wedge \mathit{False}(f) \wedge \bigwedge_{(i,j)\in E_2} C(i,j) \rightarrow t = f\}$$

$$Q = \bigwedge_{(i,j)\in E_1} C(i,j)$$

Because $b = 0$, the only repair candidate is $R = D$. Since the head of the only constraint in C can never be satisfied, the constraint can only be fulfilled if its body is not satisfied for any assignment to the variables. This is, however, only the case if G_2 is not 3-colorable. Similarly, the query is only satisfied by the repair R if G_1 is 3-colorable. For full tgds, it suffices to replace the head $t = f$ in C by $X(t)$. □

Finally, we investigate the complexity of functional dependencies and key constraints, yielding the lowest complexity results in our study of b-CQA.

Theorem 17. *b-CQA for FDs is in* NP.

Proof. We show this by reduction to SAT. The basic idea of this result is to solve parts of the problem already as a preprocessing step in the transformation phase. Observe that whenever a set of FDs is violated, it never helps to add facts. Hence, the only way to become consistent with FDs is to delete facts. The number of database instances that can be obtained by deleting facts from the input database D is bounded by $\sum_{i=1}^{b} \binom{n}{i} \leq b \cdot n^b$, which is polynomial since b is fixed. Thus, also the number of repair candidates that need to be considered is bounded, say by the integer u. Therefore, we can enumerate all repair candidates (including the original database) $\mathcal{R} = \{D, R_1, \ldots, R_u\}$ in polynomial time and check whether the FDs are satisfied by the elements of $\mathcal{R}$. As we are interested in the set of subset-minimal repairs only, we construct the set $\mathcal{R}_{min} \subseteq \mathcal{R}$ of minimal repairs. Notice that this can also be done in polynomial time. It remains to check for each repair $R_{min} \in \mathcal{R}_{min}$ whether $R_{min} \models Q$, where Q is the conjunctive query that has to hold for all repairs.

After this preprocessing step, we create a propositional formula that is satisfiable if and only if Q is satisfied by all subset-minimal repairs in $\mathcal{R}_{min}$. Since, as discussed earlier, model-checking for conjunctive queries is NP-complete, we have that for any repair R and any conjunctive query Q, there is a propositional formula $[\![R \models Q]\!]$ that is satisfiable if and only if $R \models Q$. Therefore, we can construct a single propositional formula $\varphi := \bigwedge_{R_{min}\in\mathcal{R}_{min}} [\![R_{min} \models Q]\!]$ to perform the check. Taken together, we can decide the problem after polynomial preprocessing, by deciding a single SAT instance. □

Proposition 18. *b-CQA for key constraints is* NP*-hard.*

Proof. Recall that model-checking for conjunctive queries is NP-complete. Hence, even if $C = \emptyset$ and $b = 0$, the check whether $D \models Q$ holds, dominates the complexity and leads to hardness for NP. □

6 Conclusion

In this work, we have introduced natural variants of RC and CQA, called b-RC and b-CQA, where the distance of a repair to the original instance is bounded by some fixed b. As discussed in the introduction, there are settings where answering queries over repairs of bounded distance can lead to more natural solutions. In our complexity analysis, we have shown that complexity drops in many, but not all cases, and that all problems are decidable even if they were undecidable for CQA. Thus, our approach is not only natural, but also has better computational properties than CQA in many cases. We think that these theoretical results, in which we pinpoint the sources of hardness, can support the development of tools for RC and CQA. Furthermore, these results can inspire the study of related formalisms, e.g., CQA where only a certain fraction (e.g., one quarter) of the database may change.

Extending this study to related formalisms and to perform a parameterized complexity analysis are interesting tasks for future work. In this work, we have considered a fixed bound b on the distance to the original database as this is the strongest restriction, and therefore most likely to give tractability or at least results for lower complexity classes. Nevertheless, it would be interesting to consider weakening this restriction. Exploring other notions of complexity such as data complexity is also on top of our agenda. Furthermore, it would be desirable to obtain matching complexity bounds for all considered constraint classes.

References

[Arenas and Bertossi, 2010] Marcelo Arenas and Leopoldo Bertossi. On the decidability of consistent query answering. In *AMW 2010*, volume 619 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[Arenas *et al.*, 1999] Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *PODS 1999*, pages 68–79. ACM Press, 1999.

[Arming *et al.*, 2014] Sebastian Arming, Reinhard Pichler, and Emanuel Sallinger. Combined complexity of repair checking and consistent query answering. In *AMW 2014*, volume 1189 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

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

[Bertossi, 2011] Leopoldo Bertossi. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[Bienvenu and Rosati, 2013] Meghyn Bienvenu and Riccardo Rosati. Tractable approximations of consistent query answering for robust ontology-based data access. In *IJCAI 2013*, pages 775–781. AAAI Press, 2013.

[Bienvenu *et al.*, 2014] Meghyn Bienvenu, Camille Bourgaux, and François Goasdoué. Querying inconsistent description logic knowledge bases under preferred repair semantics. In *AAAI 2014*, pages 996–1002. AAAI Press, 2014.

[Bienvenu, 2012] Meghyn Bienvenu. On the complexity of consistent query answering in the presence of simple ontologies. In *AAAI 2012*, pages 705–711. AAAI Press, 2012.

[Calì *et al.*, 2003a] Andrea Calì, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *PODS 2003*, pages 260–271. ACM Press, 2003.

[Calì *et al.*, 2003b] Andrea Calì, Domenico Lembo, and Riccardo Rosati. Query rewriting and answering under constraints in data integration systems. In *IJCAI 2003*, pages 16–21. Morgan Kaufmann, 2003.

[Chomicki, 2007] Jan Chomicki. Consistent Query Answering: Five Easy Pieces. In *ICDT 2007*, volume 4353 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2007.

[Fontaine, 2013] Gaëlle Fontaine. Why is it hard to obtain a dichotomy for consistent query answering? In *LICS 2013*, pages 550–559. IEEE Computer Society, 2013.

[Greco *et al.*, 2014] Sergio Greco, Fabian Pijcke, and Jef Wijsen. Certain query answering in partially consistent databases. *PVLDB 2014*, 7(5):353–364, 2014.

[Kolaitis and Pema, 2012] Phokion G. Kolaitis and Enela Pema. A dichotomy in the complexity of consistent query answering for queries with two atoms. *Information Processing Letters*, 112(3):77–85, 2012.

[Lembo and Ruzzi, 2007] Domenico Lembo and Marco Ruzzi. Consistent query answering over description logic ontologies. In *RR 2007*, volume 4524 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2007.

[Lembo *et al.*, 2010] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico F. Savo. Inconsistency-tolerant semantics for description logics. In *RR 2010*, volume 6333 of *Lecture Notes in Computer Science*, pages 103–117. Springer, 2010.

[Lembo *et al.*, 2012] Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati, Marco Ruzzi, and Domenico F. Savo. Inconsistency-tolerant first-order rewritability of DL-Lite with identification and denial assertions. In *DL 2012*, volume 846 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.

[Lukasiewicz *et al.*, 2015] Thomas Lukasiewicz, Maria Vanina Martinez, Andreas Pieris, and Gerardo I. Simari. From classical to consistent query answering under existential rules. In *AAAI 2015*, pages 1546–1552. AAAI Press, 2015.

[Ortiz, 2013] Magdalena Ortiz. Ontology based query answering: The story so far. In *AMW 2013*, volume 1087 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[Pichler and Skritek, 2011] Reinhard Pichler and Sebastian Skritek. The complexity of evaluating tuple generating dependencies. In *ICDT 2011*, pages 244–255. ACM Press, 2011.

[Staworko and Chomicki, 2010] Sawomir Staworko and Jan Chomicki. Consistent query answers in the presence of universal constraints. *Information Systems*, 35(1):1–22, 2010.

[ten Cate *et al.*, 2012] Balder ten Cate, Gaëlle Fontaine, and Phokion G. Kolaitis. On the data complexity of consistent query answering. In *ICDT 2012*, pages 22–33. ACM Press, 2012.

[Wijsen, 2014] Jef Wijsen. A survey of the data complexity of consistent query answering under key constraints. In *FoIKS 2014*, volume 8367 of *Lecture Notes in Computer Science*, pages 62–78. Springer, 2014.