On the Data Complexity of Consistent Query Answering over Graph Databases

Pablo Barceló[a,*], Gaëlle Fontaine[a]

[a] *Center for Semantic Web Research & Department of Computer Science, University of Chile Beaucheff 851, Santiago, Chile, 837-0456*

Abstract

Applications of graph databases are prone to inconsistency due to interoperability issues. This raises the need for studying query answering over inconsistent graph databases in a simple but general framework. We follow the approach of consistent query answering (CQA), and study its data complexity over graph databases for conjunctive regular-path queries (CRPQs) and conjunctive regular-path constraints (CRPCs). We deal with subset, superset and symmetric-difference repairs. Without restrictions, CQA is undecidable for the semantics of superset- and symmetric-difference repairs, and Π_2^{P}-complete for subset-repairs. However, we identify restrictions on CRPCs and databases that lead to decidability, and even tractability of CQA.

Keywords: graph databases; regular path queries; consistent query answering; description logics; rewrite systems

1. Introduction

Query languages for graph databases are typically navigational, in the sense that they allow for recursively traversing the labeled edges while checking for the existence of a path whose label satisfies a particular regular condition (see, e.g., [42, 5]). The basic building block for navigational languages over graph databases is the class of *regular path queries*, or RPQs [15]. Each

*Corresponding author

Email addresses: pbarcelo@dcc.uchile.cl (Pablo Barceló), gaelle@dcc.uchile.cl (Gaëlle Fontaine)

This is the full version of [6].

RPQ is a regular expression L, and its evaluation $L(G)$ over a graph database G corresponds to a binary relation that contains all pairs of nodes in G that are linked by some path whose label matches L. The RPQs are often extended with the ability to traverse edges in both directions and expressing joins and projections. This gives rise to the class of *conjunctive two-way* RPQs, or C2RPQs [14]. The evaluation problem for C2RPQs can be solved in NLOGSPACE in *data complexity*; i.e., when the C2RPQ is fixed (cf., [5]).

Although graph databases are schema-less, it is possible to enforce data consistency over them using *path constraints* [1, 10]. These constraints have been used in several scenarios that are based on the graph database paradigm, e.g., to express local knowledge about semi-structured data [1]; to enforce restrictions over object-oriented databases, XML and RDF [38, 13, 24, 3, 32]; and to capture ontological hierarchies in the context of description logics (DLs) [17, 18, 19]. The simplest class of path constraints are those based on RPQs. They were introduced by Abiteboul and Vianu under the name of *regular path constraints*, or RPCs [1]. These are expressions of the form $L_1 \subseteq L_2$, where L_1 and L_2 are RPQs. In the graph database and DL contexts, a graph database G satisfies $L_1 \subseteq L_2$ if and only if $L_1(G) \subseteq L_2(G)$ [28, 17, 18] (but we also consider a more restrictive semantics for RPCs, motivated by their application in semistructured data, following the original proposal of Abiteboul and Vianu). RPCs are a constituent part of the semantics of graph data and can be used in the query optimization process [1].

RPCs are rather limited in expressive power, and thus have been recently extended with the ability to traverse edges in both directions [19]. We concentrate here an even more expressive constraint language that is based on the class of C2RPQs. In particular, we study the constraint language of C2RPCs, which are expressions of the form $q \subseteq q'$, for CRPQs q and q'.

An important problem when dealing with dependencies is that databases might be *inconsistent*, i.e., they might fail to satisfy its integrity constraints. In the case of graph database applications, high levels of inconsistency appear due to interoperability and distribution; e.g., in RDF and social/scientific networks [29, 43]. As an example, inconsistency might arise while integrating several sources into a single RDF graph, or while performing statistical inference on a scientific or social network. This raises the need for developing an *inconsistency-tolerant* semantics in a simple yet general framework that abstracts away from its many implementations. In order to tackle this problem, we use the widespread approach of *consistent query answering* (as first introduced in the seminal work of Arenas et al [2]), which we now describe.

The approach is based on the notion of *repair*, which represents a possible minimal way in which consistency over the data could be restored. More formally, a repair is a database that satisfies the constraints and "minimally differs" from the original database. In general, a database does not admit a unique repair. This leads to an inconsistency-intolerant semantics based on the *consistent answers* of a query, i.e., the answers that hold in every possible repair. The problem of computing consistent answers to a query is known as *consistent query answering* (CQA).

Here we study the data complexity of CQA over graph databases in the scenario in which queries are C2RPQs and constraints are C2RPCs. That is, we study the complexity of evaluating consistent answers over an inconsistent graph database G for a *fixed* C2RPQ L under a *fixed* set Γ of C2RPCs. We explain next the context and main contributions of our work.

The context. The complexity of CQA has received considerable attention over different data models, notions of repairs and classes of constraints. Under the relational model, for instance, this problem has been studied for set-based [2, 20, 26], cardinality-based [31], and attribute-based repairs [40]; for constraints expressed as traditional functional dependencies, inclusion dependencies and denial constraints (see, e.g., [12, 21, 27, 41, 30, 26]); and for constraints expressed as tuple-generating dependencies (tgds) and equality-generating dependencies (egds) that arise in the context of data integration and data exchange [20].

CQA has also been studied in depth in the DL context, starting from the work of Lembo and Ruzzi [33]. The DL semantics is *open-world* in nature, meaning that the non-presence of a fact is not sufficient to ensure that the negation of the fact holds. This implies that in the DL context the only meaningful set-based repairs are the *subset* repairs; i.e., those that allow to restore consistency with respect to the DL constraints (i.e., the ones in the *TBox*) only by *deleting* facts from the database (i.e., the *ABox*). It is worth mentioning that the notion of DL repair is slightly different to its relational counterpart: A DL repair is not a subinstance that satisfies the constraints in the TBox, but one that does not lead to a contradiction in conjunction with those constraints [33, 34].

Some applications of graph databases, such as RDF, are open-world in nature. However, graph databases are not tied to this interpretation and may also accommodate closed-world applications. Therefore, there should be no a priori restriction on the class of set-based repairs one allows in this context.

We thus study CQA for graph databases under the three usual notions of set-based repairs: *subset*, *superset*, and *symmetric difference* repairs [20].

Our contributions. We start by looking at the data complexity of CQA over graph databases under subset repairs. The problem is in Π_2^{P} in general. We prove that:

- It is complete for this class in restricted cases; namely, for RPQs under a simple class of RPCs known as *word constraints*. These are expressions of the form $w_1 \subseteq w_2$, for words w_1 and w_2 [1].
- It remains intractable even by forcing RPCs to be read from a particular node in the graph database, called the *origin* (which corresponds to the original semantics Abiteboul and Vianu defined for these constraints).

In order to deal with this high complexity, we provide tractable cases by restricting the class of RPCs or the class of graph databases allowed.

- In the first case, we prove that the problem is tractable if C2RPCs are in *LAV* form, i.e., if they are of the form $a \subseteq q$, where a is a single letter and q is an arbitrary C2RPQ. This expresses that every time that two nodes are linked by an edge labeled a, these two nodes also satisfy the C2RPQ q.
- We show that the previous result is, in a sense, tight, since allowing a single RPC of the form $ab \subseteq c$, for symbols a, b and c, leads to intractability.
- In the case of restrictions on graph databases, by applying a deep result of Courcelle [22] we obtain that the data complexity of CQA under subset repairs is tractable over graph databases of *bounded treewidth*.

We then move to study CQA under superset and symmetric difference repairs. As a main contribution:

- We prove that in both cases the problem is undecidable. This again holds even for RPQs under word constraints.

On the other hand, we obtain decidability by restricting the class of C2RPCs allowed or their semantics. We start with the restrictions on C2RPCs.

- We prove that if C2RPCs are in LAV form then the data complexity of CQA is tractable when dealing with symmetric difference repairs.

  We leave open whether CQA is also decidable in this case under the semantics of superset repairs, but prove that at least tractability in data complexity is not preserved.

- We then prove that if C2RPCs are in *GAV* form, i.e., if they are of the form $q(x, y) \subseteq (x, a, y)$, where a is a single letter, our CQA problem is tractable under the semantics of superset repairs, but intractable under symmetric difference repairs.

With respect to restrictions on the semantics of RPCs, as a main contribution:

- We prove that by forcing RPCs to be read from the origin, the problem can be solved in coNP under superset repairs.

Comparison with previous results. The main difference between the query and constraint languages we study here (C2RPQs and C2RPCs) and the ones studied in the relational context, is that our languages allow recursion while CQA has been studied in the relational world mostly in the absence of it. Interestingly, our results show that recursion does not add to the complexity of the problems studied. In fact, almost all of our lower bounds hold in the restricted setting in which queries are defined as RPQs that do not mention the Kleene-star and RPCs correspond to word constraints.

If we interpret graph databases as relational databases, word constraints can be represented as tgds. The tgds corresponding to word constraints have a special restricted structure and are called *chain* tgds. On the other hand, CQA under tgds has been intensively studied [20], and it is tempting to think that lower bounds obtained in such setting could be adapted to work in our scenario. This is not the case, however, as those proofs do not apply to the class of chain tgds. Actually, the proofs of our lower bounds are considerably more involved than the ones for arbitrary tgds. As a side-effect, we obtain that several of our lower bounds extend to CQA in the relational case for queries defined as unions of CQs under chain tgds.

DL databases are (essentially) graph databases. In such scenario, Π_2^{P}-hardness results have been obtained by Rosati for the data complexity of CQA under subset repairs, in particular, for the case when queries are unions

of CQs and constraints are expressed in the logic $\mathcal{ALC}$ [37]. However, constraints in this logic cannot be directly expressed as C2RPCs due to the presence of negation. Furthermore, their notion of repair is different to ours.

Organization of the paper. Preliminaries are in Section 2. Results about the subset repair semantics are in Section 3, and those about the semantics of superset and symmetric difference repairs are in Section 4. Comparison with previous work is in Section 5 and conclusions in Section 6.

2. Preliminaries

Graph databases. As it is customary in the graph database literature [15, 16, 42, 5], we consider graph databases to be finite, edge-labeled and directed graphs. Formally, let Σ be a finite alphabet. A *graph database* $G = (V, E)$ over Σ consists of a finite set V of nodes and a set of labeled edges $E \subseteq V \times \Sigma \times V$. We interpret each tuple $(u, a, v) \in E$, for $u, v \in V$ and $a \in \Sigma$, as an edge from node u to node v whose label is a. If $G = (V, E)$ and $G' = (V', E')$ are graph databases over Σ, we write $G \subseteq G'$ to denote that $V \subseteq V'$ and $E \subseteq E'$. If, in addition, it is not the case that $G' = G$, we write $G \subsetneq G'$. Moreover, given a graph database $G = (V, E)$ and a letter $a \in \Sigma$, we denote by a^G the relation:

$$\{(u, v) : (u, a, v) \in E\}.$$

If G is clear from the context, we may speak of a as a relation.

Conjunctive regular path queries. Navigational languages for graph databases express properties of *paths*. Formally, a path π in $G = (V, E)$ from node v_0 to node v_m is a sequence of the form:

$$(v_0, a_1, v_1)(v_1, a_2, v_2) \dots (v_{m-1}, a_m, v_m),$$

for $m \geq 0$, where (v_{i-1}, a_i, v_i) is an edge in E, for each $1 \leq i \leq m$. The *label* of π, denoted $\lambda(\pi)$, is the string $a_1 a_2 \dots a_m \in \Sigma^*$. When $m = 0$, i.e., when $\pi = v$, for some $v \in V$, the label $\lambda(\pi)$ is the empty string ε.

The simplest navigational language for graph databases are *regular path queries*, or RPQs [15]. An RPQ is a regular expression L over Σ. The evaluation $L(G)$ of L over a graph database $G = (V, E)$ is the set of pairs (u, v) of nodes in V for which there is a path π in G from u to v such that

$\lambda(\pi)$ satisfies L. If L does not mention the Kleene-star (i.e., if L defines a finite language) then we say that L is *non-recursive*.

RPQs are typically extended with the ability to traverse edges in both directions, yielding the class of *two-way* RPQs, or 2RPQs [15]. Formally, let Σ be a finite alphabet. We denote by $\Sigma^{\pm}$ the language that extends Σ with the *inverse* a^- of each symbol $a \in \Sigma$. A 2RPQ L over Σ is then nothing else than an RPQ over $\Sigma^{\pm}$. Let G be a graph database over Σ. The evaluation $L(G)$ of the 2RPQ L over G corresponds then to the evaluation of L (now seen as an RPQ over $\Sigma^{\pm}$) over the *completion* $G^{\pm}$ of G. The latter is the graph database that is obtained from G by adding all edges of the form (u, a^-, v), for (v, a, u) an edge in G.

The class of *conjunctive* 2RPQs (C2RPQs) is obtained by closing 2RPQs under joins and existential quantification [14]. More formally, a C2RPQ over Σ is a formula q of the form:

$$\exists \bar{z} \left((x_1, L_1, y_1) \wedge \cdots \wedge (x_m, L_m, y_m) \right), \tag{1}$$

where for each $1 \leq i \leq m$ we have that the x_i's and y_i's, are (not necessarily distinct) variables and L_i is a 2RPQ over Σ. Moreover, $\bar{z}$ is a tuple of variables among the x_i's and y_i's. If each L_i is an RPQ, then q is a CRPQ. We typically write this formula as $q(\bar{x})$ to denote that $\bar{x}$ are the *free variables* of q, i.e., those that are not existentially quantified in $\bar{z}$.

As usual, we define the semantics of C2RPQ in terms of homomorphisms. Assume that $q(\bar{x})$ is a C2RPQ of the form (1) and G is a graph database over Σ. A homomorphism from q to G is a mapping from the variables mentioned in q to the nodes of G such that for each $1 \leq i \leq m$ it is the case that $(h(x_i), h(y_i)) \in L_i(G)$. The evaluation $q(G)$ of $q(\bar{x})$ over G corresponds then to the set of all tuples of the form $h(\bar{x})$, for h a homomorphism from q to G.

The following result is considered to be folklore (cf., [5]):

Proposition 1. *The problem of evaluating C2RPQs is in* NLogspace *in data complexity. More formally, let q be a fixed C2RPQ. The problem of checking if $\bar{a} \in q(G)$, given a graph database G and a tuple $\bar{a}$ of nodes in G, is in* NLogspace.

Regular path constraints. Graph database constraints based on the class of RPQs are known as *regular path constraints* [1, 28], or RPCs. Formally, an RPC over Σ is an expression of the form $L_1 \subseteq L_2$, where L_1 and L_2 are RPQs over Σ. A *word constraint* is an RPC in which both L_1 and L_2 are words.

An RPC $L_1 \subseteq L_2$ expresses that the evaluation of the RPQ L_1 is contained in the evaluation of the RPQ L_2 [28]. Formally, a graph database G *satisfies* $L_1 \subseteq L_2$, denoted $G \models L_1 \subseteq L_2$, if and only if $L_1(G) \subseteq L_2(G)$. If Γ is a finite set of constraints, we write $G \models \Gamma$ to denote that for each RPC $L_1 \subseteq L_2$ in Γ it is the case that $G \models L_1 \subseteq L_2$.

Example 1. Let Γ be the following set of RPCs:

1. $\texttt{child\_of} \subseteq \texttt{son\_of} \cup \texttt{daughter\_of}$.
2. $\texttt{brother\_of} \cdot (\texttt{brother\_of} \cup \texttt{sister\_of}) \subseteq \texttt{brother\_of}$.
3. $\texttt{sister\_of} \cdot (\texttt{brother\_of} \cup \texttt{sister\_of}) \subseteq \texttt{sister\_of}$.
4. $\texttt{child\_of} \cdot (\texttt{brother\_of} \cup \texttt{sister\_of}) \subseteq \texttt{nephew\_of} \cup \texttt{niece\_of}$.

Intuitively, the first RPC expresses that if u is a child of v, then u is a son or a daughter of v. The second and third RPCs express that if u is a brother (resp., sister) of v and v is a sibling of w, then v is a brother (resp., sister) of w. The forth RPC states that each child of a person v is the niece or nephew of every sibling of v. □

The extension of RPCs with inverses has been recently studied in the literature [19]. Formally, a 2RPC is an expression of the form $L_1 \subseteq L_2$, where both L_1 and L_2 are 2RPQs. Similarly, it is natural to define *conjunctive* 2RPCs, or C2RPCs, which are expressions of the form $q_1(\bar{x}) \subseteq q_2(\bar{x})$, where q_1 and q_2 are C2RPQs with the same tuple $\bar{x}$ of free variables. The semantics of these extensions is naturally inherited from the previous definitions. It is clear that allowing such extensions increases the ability to express constraints in an important way.

The next proposition easily follows from Proposition 1:

Proposition 2. *Let Γ be a fixed set of C2RPCs. The problem of checking whether $G \models \Gamma$, for a given graph database G, is in* NLogspace.

Proof. NLogspace is closed under complement by Immerman-Szelepcsényi's Theorem. It is thus sufficient to show that it is possible to check $G \not\models \Gamma$ in NLogspace. The logarithmic size witness corresponds to a C2RPC $q_1(\bar{x}) \subseteq q_2(\bar{x})$ in Γ and a tuple $\bar{a}$ in G such that $\bar{a} \in q_1(\bar{x})$ and $\bar{a} \notin q_2(\bar{x})$. The latter can be checked in NLogspace from Proposition 1 and the fact that NLogspace is closed under complement. □

Repairs. A *repair* of a database D under a set of constraints Γ is a database D' that satisfies Γ but "minimally" differs from D [2]. We formalize this idea for graph databases and RPCs below, following closely its formalization in the relational context [2].

The *symmetric difference* between two relational databases D and D' is defined as a database $D \oplus D'$ that contains those "facts" that belong to (a) D but not to D', or (b) to D' but not to D. We can analogously define the symmetric difference between two graph databases $G = (V, E)$ and $G' = (V', E')$ over the same alphabet Σ, as the graph database:

$$G \oplus G' \ := \ \big(V'', (E \oplus E')\big),$$

where $E \oplus E' := (E \setminus E') \cup (E' \setminus E)$ and V'' are the nodes mentioned in $E \oplus E'$. That is, the edges of $G \oplus G'$ are those edges that appear in either G or G' but not in both. Notice that $G \oplus G'$ is also a graph database over Σ.

Three notions of set-based repairs have been studied in the literature [20]: the *subset*, *superset* and *symmetric difference* repairs ($\subseteq$–, $\supseteq$–, and $\oplus$–repairs, respectively). We introduce them below in the context of graph databases. Let $G = (V, E)$ and $G' = (V', E')$ be two graph databases over Σ, and assume that Γ is a finite set of RPCs over Σ. Then:

1. G' is a $\oplus$-repair (i.e., symmetric difference repair) of G under Γ, if (1) $G' \models \Gamma$, and (2) there is no graph database G'' over Σ such that $G'' \models \Gamma$ and $G \oplus G'' \subsetneq G \oplus G'$.
2. G' is a $\subseteq$-repair (i.e., subset repair) of G under Γ, if $G' \subseteq G$ and G' is a $\oplus$-repair of G. Equivalently, if (1) $G' \subseteq G$, (2) $G' \models \Gamma$, and (3) there is no graph database G'' over Σ such that $G'' \models \Gamma$ and $G' \subsetneq G'' \subseteq G$.
3. G' is a $\supseteq$-repair (i.e., superset repair) of G under Γ, if $G \subseteq G'$ and G' is a $\oplus$-repair of G. Equivalently, if (1) $G \subseteq G'$, (2) $G' \models \Gamma$, and (3) there is no graph database G'' over Σ such that $G'' \models \Gamma$ and $G \subseteq G'' \subsetneq G'$.

Example 2. *(Example 1 cont.)* Consider a graph database G whose set of edges is:

$$\{(a, \texttt{child\_of}, b), (b, \texttt{sister\_of}, c), (c, \texttt{brother\_of}, d)\}.$$

Then G has two $\subseteq$-repairs under Γ:

1. $\{(b, \texttt{sister\_of}, c)\}$, and
2. $\{(c, \texttt{brother\_of}, d)\}$.

On the other hand, G has eight $\supseteq$-repairs under Γ. One of them is the one that extends G with edges:

$$\{(a, \texttt{son\_of}, b), (b, \texttt{sister\_of}, d), (a, \texttt{nephew\_of}, c), (a, \texttt{nephew\_of}, d)\}.$$

Finally, G has seven $\oplus$-repairs under Γ that are neither $\subseteq$-repairs nor $\supseteq$-repairs. One of them is:

$$\{(b, \texttt{sister\_of}, c), (c, \texttt{brother\_of}, d), (b, \texttt{sister\_of}, d)\}. \quad \square$$

Repairs might not exist in some situations. Consider an RPC of the form $L \subseteq \varepsilon$, where ε is the empty word, and a graph database G that consists of nodes u and v linked by a path labeled in L. Assume that G has a $\supseteq$-repair H. Then it must be the case that $u = v$ in H, which is impossible. As the the next lemma shows, on the other hand, $\otimes$- and $\supseteq$-repairs exist in all possible situations, while for $\subseteq$-repairs it is sufficient to forbid C2RPCs $q_1 \subseteq q_2$ that contain an atom of the form (x, ε, y) in q_2 (this, in particular, forbids all RPCs of the form $L \subseteq \varepsilon$).

Lemma 1. *Let $G = (V, E)$ be a graph database and Γ a finite set of C2RPCs over Σ. Then:*

1. *There is a $\subseteq$-repair and a $\oplus$-repair of G under Γ.*
2. *If Γ contains no C2RPC $q_1 \subseteq q_2$ such that q_2 contains an atom of the form (x, ε, y), then there is a $\supseteq$-repair of G under Γ.*

Proof. Let $G = (V, E)$ be a graph database and Γ a finite set of C2RPCs. The empty graph database $G_\emptyset = (\emptyset, \emptyset)$ satisfies Γ. Hence, there is an $\subseteq$-repair H of G under Γ such that $G_\emptyset \subseteq H \subseteq G$. By definition, H is also a $\oplus$-repair of G under Γ.

Assume now that Γ contains no C2RPC $q_1 \subseteq q_2$ such that q_2 contains an atom of the form (x, ε, y). Consider the graph database $G_c = (V, V \times \Sigma \times V)$ that extends G with all possible edges between nodes. Therefore, if w is a nonempty word over Σ and (u, v) is a pair of nodes in V, there is a path from u to v in G_c labeled w. It is easy to see that G_c satisfies Γ. In fact, by construction $\bar{a} \in q_2(G_c)$ for every tuple $\bar{a}$ of nodes in G of the same arity than $\bar{x}$. Since $G \subseteq G_c$, there is a $\supseteq$-repair H of G under Γ such that $G \subseteq H \subseteq G_c$. $\square$

Consistent query answering. We are now ready to define our most important notion, that of a *consistent answer* to an C2RPQ. The consistent answers are the pairs of nodes that belong to the evaluation of the C2RPQ over every single repair of the original graph database.

Definition 1 (Consistent answers)**.** Assume that $\star \in \{\oplus, \subseteq, \supseteq\}$. Let $G = (V, E)$ be a graph database, Γ a set of C2RPCs, and q a C2RPQ, all of them over Σ. We define the set $\star$-Cons(G, q, Γ) of *$\star$-consistent answers of q over G under Γ* as follows:

$$\star\text{-Cons}(G, q, \Gamma) \;=\; \bigcap \{q(G') \,:\, G' \text{ is a } \star\text{-repair of } G \text{ under } \Gamma\}. \quad \square$$

Example 3. *(Example 2 cont.)* Consider the RPQ $L = \texttt{child\_of}\cdot\texttt{sister\_of}$. The pair (a, d) belongs to $\supseteq$-Cons(G,L,Γ). This pair also belongs to $\supseteq$-Cons(G,L',Γ), for $L' = \texttt{nephew\_of} \cup \texttt{niece\_of}$.

On the other hand, the only way in which a pair (u, v) can belong to $\subseteq$-Cons(G,L'',Γ), for an RPQ L'', is when both u and v correspond to the constant c. This is because the only element of G that belongs to both $\subseteq$-repairs of G under Γ is c. Notice that this can only happen if $L'' = \varepsilon$. $\square$

Here we study the data complexity of the problem of computing certain answers. We formalize this decision problem as follows. Assume that q is a C2RPQ and Γ is a finite set of C2RPCs over Σ. We denote by $\star$-CQA(q,Γ) the problem of, given a graph database $G = (V, E)$ over Σ and a tuple $\bar{a}$ of nodes in V, checking whether $\bar{a} \in \star$-Cons(G, q, Γ).

3. CQA under Subset Repairs

We start by proving that under the subset repair semantics our CQA problem is Π_2^{P}-complete. This holds even in the case in which all RPCs are word constraints and the query is a non-recursive RPQ, i.e., an RPQ that does not mention the Kleene-star.

Theorem 1.
1. *For each C2RPQ q and finite set Γ of C2RPCs over the same alphabet Σ, it is the case that $\subseteq$-CQA(q,Γ) is in Π_2^P.*
2. *There exist a finite alphabet Σ, a non-recursive RPQ L, and a finite set Γ of word constraints over Σ, such that $\subseteq$-CQA(L,Γ) is Π_2^P-complete.*

Proof. We start with the first part of the theorem, i.e., the upper bound. Fix a C2RPQ q and a set Γ of C2RPCs. Let G be a graph database over Σ and

$\bar{a}$ a tuple of nodes in G. By definition, $\bar{a}$ belongs to $\subseteq$-Cons(G, q, Γ) if and only if for each graph database H that is contained in G:

$$\bar{a} \in q(H) \text{ or there exists } H' \text{ such that } H \subsetneq H' \subseteq G \text{ and } H' \models \Gamma.$$

Then the following Σ_2^P-algorithm checks if $\bar{a} \notin \subseteq$-Cons(G, q, Γ): It guesses a graph database H' that is contained in G, and then it checks that $\bar{a} \notin q(H)$ – which can be solved in polynomial time from Proposition 1 – and that there is no H' such that $H \subsetneq H' \subseteq G$ and $H' \models \Gamma$. The latter is in coNP since H' is of polynomial size and checking whether $H' \models \Gamma$ is in polynomial time from Proposition 2. We conclude that $\subseteq$-CQA(q, Γ) is in Π_2^{P}.

Now we prove the second part of the theorem, i.e., the lower bound. That is, we prove that there are an RPQ L_0 and a set Γ_0 of RPCs such that $\subseteq$-CQA(L_0, Γ_0) is Π_2^{P}-hard. We do so by providing a reduction from the quantified boolean satisfaction problem for Π_2^{P} to $\subseteq$-CQA(L_0, Γ_0).

But before doing so, we state the following lemma, which will be used several times in this and other proofs.

Lemma 2. *Let Γ be a set of RPCs. Suppose that H is a $\star$-repair of a graph database $G = (V, E)$ with respect to Γ, where $\star \in \{\subseteq, \supseteq, \oplus\}$. Then:*

- *If $a \in \Sigma$ is such that for all constraints $L_1 \subseteq L_2$ in Γ the symbol a does not occur in L_2, then $a^H \subseteq a^G$.*
- *If $a \in \Sigma$ is such that for all constraints $L_1 \subseteq L_2$ in Γ the symbol a does not occur in L_1, then $a^G \subseteq a^H$.*

Proof. We only prove the first item, the second one is similar. Let us assume that for each $L_1 \subseteq L_2$ in Γ the symbol a does not occur in L_2. Let H' be the database obtained from H by removing the edges with labels a from u to v, where (u, a, v) does not belong to E. Since for all constraints $L_1 \subseteq L_2$ in Γ the symbol a does not occur in L_2, this implies that if Γ is true in H, then Γ remains true in H'. Since $H' \oplus G \subseteq H \oplus G$, it follows from the minimality condition of the repair H that $H = H'$. In particular, $a^H \subseteq a^G$. □

Let ϕ be a quantified boolean formula of the form $\forall X_0 \exists Y_0 \psi$, where X_0, Y_0 are disjoint sets of variables and ψ is of the form:

$$(z_{11} \vee z_{12} \vee z_{13}) \wedge \cdots \wedge (z_{m1} \vee z_{m2} \vee z_{m3}),$$

where $z_{ij} \in \{x, \neg x, y, \neg y : x \in X_0, y \in Y_0\}$ $(1 \leq i \leq m,\ 1 \leq j \leq 3)$. For all i, j, we define u_{ij} to be the propositional variable that is naturally associated with the literal z_{ij}, i.e.,

$$u_{ij} \ = \ \begin{cases} z_{ij} & \text{if } z_{ij} \in X_0 \cup Y_0 \\ u & \text{if } z_{ij} \text{ is of the form } \neg u. \end{cases}$$

Without loss of generality, we may assume that in each clause $C_i := z_{i1} \vee z_{i2} \vee z_{i3}$, for $1 \leq i \leq m$, there is at least one variable in Y_0. Suppose for example that the clause C_1 of ψ contains only variables in X_0. We construct a new formula ψ' defined by $(z_{11} \vee z_{12} \vee y) \wedge (\neg y \vee z_{12} \vee z_{13}) \wedge C_2 \wedge \cdots \wedge \ldots C_n$, where y is a new fresh variable. We define Y_0' as $Y_0 \cup \{y\}$. Then $\forall X_0\ \exists Y_0\ \psi$ is satisfiable iff $\forall X_0\ \exists Y_0'\ \psi$ is satisfiable.

We will define the RPQ L_0, the constraints Γ_0, and associate a graph database G_ϕ with ϕ in such a way that:

$$\phi \text{ is true} \quad \text{iff} \quad (n_s, n_s) \in \subseteq\text{-Cons}(G_\phi, L_0, \Gamma_0),$$

where n_s is a distinguished node of G_ϕ. The set V of nodes of the graph G_ϕ is the following:

$$\{n_{ij} : 1 \leq i \leq m, 1 \leq j \leq 3\} \cup \{n_t, n_f, n_s\}.$$

We associate the node n_{ij}, for $1 \leq i \leq m$ and $1 \leq j \leq 3$, with the occurrence of propositional variable u_{ij} in literal z_{ij}. Observe that even if $u_{ij} = u_{kl}$ (with $(i, j) \neq (k, l)$), the associated nodes n_{ij} and n_{kl}, respectively, are distinct. We also have three distinguished nodes n_s, n_t and n_f.

For the definition of the graph database G_ϕ, we consider available an arbitrary strict linear order on each equivalence class defined by the relation $\{(n_{ij}, n_{kl}) \mid u_{ij} = u_{kl}\}$ over the set $V \setminus \{n_s, n_t, n_f\}$. In other words, we assume the existence of an arbitrary linear order over each maximal subset of $\{n_{ij} : 1 \leq i \leq m, 1 \leq j \leq 3\}$ that is associated with the same propositional variable from $X_0 \cup Y_0$. Thus, we can naturally refer to a node in any such set S as being the *successor* of another node in S.

The graph database G_ϕ. We define now the graph database G_ϕ in the the following way:

- There is an empty relation e and a total relation s, i.e.:

$$e^{G_\phi} = \emptyset \text{ and } s^{G_\phi} = V \times V.$$

- There are loops labeled t and f over every node of the form $n_{ij} \in V$, that is:

$$t^{G_\phi} = f^{G_\phi} = \{(n_{ij}, n_{ij}) : 1 \leq i \leq m, 1 \leq j \leq 3\}.$$

- There are loops labeled t_0 and t_0' on node n_t and loops labeled f_0 and f_0' on node n_f:

$$t_0^{G_\phi} = (t_0')^{G_\phi} = \{(n_t, n_t)\} \text{ and } f_0^{G_\phi} = (f_0')^{G_\phi} = \{(n_f, n_f)\}.$$

- There is a loop labeled g on n_s, i.e.:

$$g^{G_\phi} = \{(n_s, n_s)\}.$$

- There are edges labeled w from n_s to n_t and n_f:

$$w^{G_\phi} = \{(n_s, n_t), (n_s, n_f)\}.$$

- There is a loop labeled w' on n_s:

$$(w')^{G_\phi} = \{(n_s, n_s)\}.$$

- There is an edge labeled y from n_s to each node representing an occurrence of a variable in Y_0; that is:

$$y^{G_\phi} = \{(n_s, n_{ij}) : u_{ij} \in Y_0\}.$$

- There are edges labeled z from n_t and n_f to each node representing an occurrence of a variable in Y_0; that is:

$$z^{G_\phi} = \{(n_t, n_{ij}), (n_f, n_{ij}) : u_{ij} \in Y_0\}.$$

- There is a loop labeled y' on each node representing an occurrence of variable in Y_0:

$$(y')^{G_\phi} = \{(n_{ij}, n_{ij}) : u_{ij} \in Y_0\}.$$

- For each pair (n_{ij}, n_{kl}) of nodes associated with the same propositional variable such that n_{kl} is the succesor of n_{ij}, there is an edge labeled d from n_{ij} to n_{kl} and an edge labeled d' from n_{kl} to n_{ij}:

$$d^{G_\phi} = \{(n_{ij}, n_{kl}) : u_{ij} = u_{kl} \text{ and } n_{kl} \text{ is the successor of } n_{ij}\}$$
$$\text{and } (d')^{G_\phi} = (d^{G_\phi})^{-1}.$$

- For each pair of nodes representing consecutive literals in a clause of ψ, there is an edge from the first to the second one representing the polarity of such literals. Formally, we use symbols r_{++}, r_{+-}, r_{-+}, and r_{--} such that:

$$\begin{aligned}
r_{++}^{G_\phi} &= \{(n_{ij}, n_{i(j+1)}) : z_{ij} = u_{ij}, z_{i(j+1)} = u_{i(j+1)}, 1 \leq i \leq m, 1 \leq j \leq 2\},\\
r_{+-}^{G_\phi} &= \{(n_{ij}, n_{i(j+1)}) : z_{ij} = u_{ij}, z_{i(j+1)} = \neg u_{i(j+1)}, 1 \leq i \leq m, 1 \leq j \leq 2\},\\
r_{-+}^{G_\phi} &= \{(n_{ij}, n_{i(j+1)}) : z_{ij} = \neg u_{ij}, z_{i(j+1)} = u_{i(j+1)}, 1 \leq i \leq m, 1 \leq j \leq 2\},\\
r_{--}^{G_\phi} &= \{(n_{ij}, n_{i(j+1)}) : z_{ij} = \neg u_{ij}, z_{i(j+1)} = \neg u_{i(j+1)}, 1 \leq i \leq m, 1 \leq j \leq 2\}.
\end{aligned}$$

- Each sink node of an edge labeled r_{ab}, for $a, b \in \{+, -\}$, has a loop labeled r'_{ab}. Formally, we use symbols r'_{ab}, for $a, b \in \{+, -\}$, such that:

$$\begin{aligned}
(r'_{ab})^{G_\phi} = \{(n_{i(j+1)}, n_{i(j+1)}) :\\
(n_{ij}, n_{i(j+1)}) \in r_{ab}^{G_\phi} \text{ for } 1 \leq i \leq m \text{ and } 1 \leq j \leq 2\}.
\end{aligned}$$

The intuition behind G_ϕ is the following: Relation e is empty and s is the full relation $V \times V$. Therefore, e remains empty in each $\subseteq$-repair. Moreover, the constraints in Γ_0 will not mention s, and thus s remains being the full relation in each such repair. For relations t and f, each node n_{ij} associated with a variable from ϕ has loops labeled t and f. The $\subseteq$-repairs will be such that each such node has at most one loop, either with label t or f. This allows us to define a partial map associating a truth value with each node n_{ij} (true or $\top$ if the loop has label t, and false or $\bot$ if the loop has label f).

The special node n_t has a loop with label t_0 and the special node n_f has a loop with label f_0. The repairs will be such that if one of those two loops "disappears", then the answer of the RPQ L_0 will trivially contain (n_s, n_s).

The relations r_{++}, r_{-+}, r_{+-} and r_{--} express which variables occur in the same clause and whether each variable occur positively or negatively. The relation d specifies which nodes correspond to the same variable. The constraints will be such that in a $\subseteq$-repair, two nodes linked by d (thus corresponding to the same variable) have loops with the same label (t or f). Hence, the partial map associating a truth value to each node ($\top$ if the loop has label t and $\bot$ if the loop has label f), corresponds to a partial valuation over the variables in $X_0 \cup Y_0$.

The special node n_s has a loop with label g. Such loop acts as a "witness". If such witness appears in a $\subseteq$-repair, we say that n_s gets *activated*. When

n_s is activated in a repair, the idea is that each node n_{ij} associated with a variable in Y_0 has a loop with label t or f. That is, each such node receives a truth value ($\top$ if the loop has label t and $\bot$ otherwise). Conversely, if in a $\subseteq$-repair, one node associated with a variable in Y_0 has a loop with label t or f, then the presence of such loop "activates" the witness on n_s (i.e. n_s has a loop with label g). As a consequence, all nodes associated with variables in Y_0 will have a loop. Basically, the node n_s guarantees that either all the nodes associated with variables in Y_0 have a loop, or none of them has a loop.

In order to encode in the graph database G_ϕ which variables belong to Y_0, we add an edge with label y from the distinguished node n_s to each node of the form n_{ij}, for $u_{ij} \in Y_0$. In such case, we also add edges labeled z from the distinguished nodes n_t and n_f to n_{ij}.

We use the nodes n_t and n_f in the following way. Recall that in a given $\subseteq$-repair, the (possible) truth value of a node is given by the label (either t or f) of its loop. Now, for the nodes associated with a variable in Y_0 we have an extra way of encoding the truth value. If the truth value of a node n_{ij} (with $u_{ij} \in Y_0$) is true, this will also be witnessed by an arrow with label z from the special node n_t to n_{ij} (while the other arrow with label z from n_f to n_{ij} is deleted). If the truth value is false, this will be witnessed by an arrow with label z from n_f to n_{ij}.

Finally, we also have a set of relations of the form r' with $r \in \mathcal{R}$ and

$$\mathcal{R} := \{d, y, w, t_0, f_0, r_{++}, r_{-+}, r_{+-}, r_{--}\}.$$

The role of those relations is as follows. Basically, each relation $r \in \mathcal{R}$ encodes a type of information that we do not want to lose in a repair. The idea is that if in a $\subseteq$-repair H we "lose" one edge with label r, then we make sure that (n_s, n_s) immediately belongs to the answer of the RPQ L_0 in the repair. Such a repair is called *irrelevant*; that is, a repair H such that $r^H \neq r^{G_\phi}$ for some $r \in \mathcal{R}$. The RPCs in Γ_0 will be defined in such a way that for each $\subseteq$-repair H of G_ϕ and symbol $r \in \mathcal{R}$, it is the case that $r^H = r^{G_\phi}$ iff $(r')^H$ is empty. Or to say it differently, a repair is irrelevant iff it contains a loop with label r' for some $r \in \mathcal{R}$.

The constraints in Γ_0. We have six sets of constraints defined as follows:

1. The first set of constraints (C1) is given by:

$$\begin{aligned} dd' \subseteq e, \quad & t_0 t'_0 \subseteq e, \\ f_0 f'_0 \subseteq e, \quad & yy' \subseteq e, \\ r_{ab} r'_{ab} \subseteq e, \quad & w'w \subseteq e, \end{aligned}$$

   where $a, b \in \{+, -\}$. Basically, (C1) contains all the constraints ensuring that a repair is irrelevant iff it contains an edge with label r' for some $r \in \mathcal{R}$. If H is a $\subseteq$-repair of G_ϕ and $r^H = r^{G_\phi}$, then using the constraints (C1) and the fact that e is empty, we can show that $(r')^H$ must be empty. Moreover, if $r^H \neq r^{G_\phi}$, using the minimality property of repairs we can show that H must contain an edge with label r'.
2. The second set of constraints (C2) is given by:

$$tf \subseteq e.$$

   Since e is empty, it says that a node cannot have both a loop with label t and f in a $\subseteq$-repair.
3. The third set of constraints (C3) is given by:

$$td \subseteq dt \text{ and } fd \subseteq df.$$

   It expresses that if two nodes are linked by d (hence, they are associated with the same variable), then they must have a loop with the same label (in case they have a loop). This guarantees that the map associating a truth value to each node (depending on the label of the loop), can be transformed into a valuation over the variables.
4. The fourth set of constraints (C4) is given by:

$$F(a) r_{ab} F(b) r_{bc} F(c) \subseteq e,$$

   where $a, b, c \in \{+, -\}$, $F(+) = f$ and $F(-) = t$. In a relevant $\subseteq$-repair of G_ϕ, such constraints state that the formula ψ is not false under the valuation associated with the repair.
5. The fifth set of constraints (C5) corresponds to:

$$t_0 z \subseteq zt, \quad f_0 z \subseteq zf.$$

   They express that if n_t is linked to a node n_{ij} associated with a variable in Y_0, then n_{ij} has a loop with label t. Similarly, if n_f is linked to such a node n_{ij}, then it has a loop with label f.

Indeed, let us look for example at the constraint $t_0 z \subseteq zt$. Suppose that there is a path with label $t_0 z$ from a node u to a node v. Since in G_ϕ the interpretation of t_0 only contains the pair (n_t, n_t) and the interpretation of z contains all pairs of the form n_{ij}, for $u_{ij} \in Y_0$, it follows that $u = n_t$ and v is a node of the form n_{ij} (with $u_{ij} \in Y_0$). Now, since $t_0 z \subseteq zt$, this implies that n_{ij} has a loop with label t. In combination with constraint (C6b) below, this means that if n_s is activated, then each node n_{ij} associated with a variable in Y_0 has a loop with label t or f.

6. The sixth set of constraints is divided as (C6a) and (C6b). The constraints in (C6a) are given by:

$$yt \subseteq gy \text{ and } yf \subseteq gy,$$

while the constraint (C6b) is given by:

$$gy \subseteq wz.$$

The constraints (C6a) express that if one node associated with a variable in Y_0 has a loop with label t or f, then the node n_s is activated (i.e. it has a loop with label g). Indeed, if there is a path with label yt from a node u to a node v, then u must be n_s and v a node of the form n_{ij} with $u_{ij} \in Y_0$. Since $yt \subseteq gy$, the presence of the loop with label t and the fact that u_{ij} is in Y_0, imply that there is an edge with label g. That is, n_s is activated.
The constraint (C6b) expresses that if the node n_s is activated (i.e., it has a loop with label g), then each node n_{ij} associated with a variable in Y_0 is linked via the inverse of z to either to the node n_t or to the node n_f. Together with the constraints (C5), this means that if n_s is activated, then each node n_{ij} associated with a variable in Y_0 has a loop with label t or f.

The RPQ L_0. Finally, the RPQ L_0 is defined as:

$$s \cdot (y' + d' + r'_{++} + r'_{-+} + r'_{+-} + r'_{--} + t' + f' + w' + g) \cdot s.$$

We can think of L_0 as the union of the RPQ sgs and the RPQ L'_0 given by:

$$s \cdot (y' + d' + r'_{++} + r'_{-+} + r'_{+-} + r'_{--} + t'_0 + f'_0 + w') \cdot s.$$

The answer of L_0' is $V \times V$ in all the irrelevant repairs, that is, the repairs admitting an edge with label r' for some $r \in \mathcal{R}$. The answer of sgs is $V \times V$ in the repairs admitting an edge with label g, that is, the repairs in which n_s is activated. Recall that this means that the partial valuation associated with the repair assigns a truth value to all the variables in Y_0.

The proof. We prove next that:

$$\phi \text{ is true} \quad \text{iff} \quad (n_s, n_s) \in \subseteq\text{-Cons}(G_\phi, L, \Gamma_0). \tag{2}$$

Implication from left to right. Suppose that ϕ holds and H is an arbitrary $\subseteq$-repair of G_ϕ with respect to Γ_0. We have to prove that (n_s, n_s) belongs to the answer of L_0 in H. Suppose for contradiction that this is not the case. In order to derive a contradiction, we proceed in the following way.

(a) We first prove a preliminary result establishing that for each r in:

$$\mathcal{R} = \{d, y, r_{++}, r_{-+}, r_{+-}, r_{--}, w, t_0, f_0\},$$

it is the case that $r^H = r^{G_\phi}$.

(b) Next we show that there is a valuation V_X defined over the variables in X_0, such that for each $u_{ij} \in X_0$ it is the case that $V_X(u_{ij}) = \top$ (resp. $V_X(u_{ij}) = \bot$) iff in H there is a loop labeled t (resp., f) on n_{ij}.

(c) Using the fact that ϕ is true, we know that there must be a valuation V_Y for the variables in Y_0 such that ψ is satisfied by the valuation $V_X \cup V_Y$.

(d) We use the valuation V_Y to define a graph database H_Y such that (1) $H \subsetneq H_Y \subseteq G_\phi$, and (2) $H_Y \models \Gamma_0$. This contradicts the fact that H is a $\subseteq$-repair of G_ϕ.

We start by proving (a).

Claim 1. *For each relation r in $\mathcal{R} = \{d, y, r_{++}, r_{-+}, r_{+-}, r_{--}, w, f_0, t_0\}$, it is the case that $r^H = r^{G_\phi}$.*

Proof. We only prove the claim for the relation y. The other proofs are similar. Intuitively, the claim holds for the following reason. If $y^H \subsetneq y^{G^\phi}$, then using the minimality of H and the constraints (C1), we can show that $(y')^H$ is not empty. This would imply that (n_s, n_s) belongs to the answer of L_0 in H, which is a contradiction.

Formally, suppose for contradiction that $y^H \neq y^{G_\phi}$. That is, there is a pair (n_s, n_{ij}) that belongs to y^{G_ϕ} but not to y^H. We let H' be the database obtained from H by adding a loop with label y' to the node n_{ij}. Using the facts that Γ_0 holds in H and (n_s, n_{ij}) does not belong to y^H, we can prove that Γ_0 remains true in H'. Indeed, this is clear for all the constraints, except for the only constraint that mentions y', i.e. the following RPC in (C1):

$$yy' \subseteq e.$$

We added an edge with label y' from n_{ij} to n_{ij} to obtain H'. However, the constraint above remains true, as the path (n_s, y, n_{ij}) does not belong to H'.

Since H is a $\subseteq$-repair of G_ϕ and $H \subseteq H' \subseteq G^\phi$, we have $H = H'$. Moreover, the RPCs in Γ_0 do not mention the symbol s, and therefore $s^H = s^{G_\phi} = V \times V$. In particular, the path $(n_s, s, n_{ij})(n_{ij}, y', n_{ij})(n_{ij}, s, n_s)$ belongs to H. This implies that (n_s, n_s) belongs to the answer of L_0 in H, which is a contradiction. □

We prove (b). To show that such a valuation V_X exists, we have to prove that for all $u_{ij}, u_{kl} \in X_0$:

(i) n_{ij} has exactly one loop, either with label t or with label f,

(ii) if $u_{ij} = u_{kl}$, then n_{ij} has a loop labeled t iff n_{kl} has a loop labeled t.

We prove (i). Since e is empty in G_ϕ and H is a subset of G_ϕ, we have that $e^H = \emptyset$. Since H is a $\subseteq$-repair of G_ϕ, the constraint $tf \subseteq e$ in (C2) holds in H. As e is empty in H, this means that no node n_{ij} representing an occurrence of a variable u_{ij} in X_0 has loops labeled t and f, respectively. Moreover, using the maximality condition of repairs, we can show that each node has at least one loop with label t or f.

Now we prove (ii). Intuitively, this comes from the fact that the constraints $td \subseteq dt$ and $fd \subseteq df$ hold in H. Take any pair of nodes n_{ij} and n_{kl} such that $u_{ij} = u_{kl}$ and n_{kl} is the successor of n_{ij}. We prove that:

$$\text{If } n_{ij} \text{ has a loop with label } t, \text{ then } n_{kl} \text{ has a loop with label } t. \tag{3}$$

Similarly, we can show that:

$$\text{If } n_{ij} \text{ has a loop with label } f, \text{ so does } n_{kl}. \tag{4}$$

Notice that (3) and (4) are sufficient to prove that (ii) holds. Indeed, (3) establishes the direction from left to right of (ii). For the direction from right

to left, suppose that n_{kl} has a loop with label t. Suppose for contradiction that n_{ij} does not have a loop with label t. By (i), this implies that n_{ij} has a loop with label f. Hence, from (4), n_{kl} has a loop with label f. This contradicts (i) and the fact that n_{kl} has a loop with label t.

We only prove (3), as (4) is analogous. It follows from Claim 1 that $d^H = d^{G_\phi}$. Since $u_{ij} = u_{kl}$ and n_{kl} is the successor of n_{ij}, this implies that there is an edge with label d from n_{ij} to n_{kl} in H. Since n_{ij} has a loop with label t in H, it follows that there is a path with label td from n_{ij} to n_{kl}. Recall that the constraint $td \subseteq dt$ in (C3) holds in H. Hence, there must be a path with label dt from n_{ij} to n_{kl}. Since H is a subset of G_ϕ and given the definition of G_ϕ, the existence of such a path implies that n_{kl} has a loop with label t. This finishes the proof of (ii) and (b).

Using the fact that ϕ is true, we know that exists a valuation V_Y over the variables in Y_0 such that ψ is satisfied by the valuation $V_X \cup V_Y$. This establishes (c).

Now we establish (d); that is, we construct a graph database H_Y such that $H \subsetneq H_Y \subseteq G_\phi$ and $H_Y \models \Gamma_0$. We construct H_Y in the following way. For each relation $r \notin \{g, t, f, z\}$, we define r^{H_Y} as r^H. The relations in $\{g, t, f, z\}$ are as follows:

$$\begin{aligned}
g^{H_Y} &= \{(n_s, n_s)\},\\
t^{H_Y} &= \{(n_{ij}, n_{ij}) : V_X(u_{ij}) = \top \text{ or } V_Y(u_{ij}) = \top\},\\
f^{H_Y} &= \{(n_{ij}, n_{ij}) : V_X(u_{ij}) = \bot \text{ or } V_Y(u_{ij}) = \bot\},\\
z^{H_Y} &= \{(n_t, n_{ij}) : V_Y(u_{ij}) = \top\} \cup \{(n_f, n_{ij}) : V_Y(u_{ij}) = \bot\}.
\end{aligned}$$

We start by checking that the constraints in Γ_0 hold in H_Y. The constraints in (C1) are true because they were true in H and our changes do not affect them. The constraint $tf \subseteq e$ in (C2) holds by definition of t and f and because V_X and V_Y are valuations (i.e., no node has loops labeled both t and f). The constraints in (C3) are true because nodes associated with occurrences of the same variable have the same loops by definition of t and f and because V_X and V_Y are valuations.

The constraints in (C4), i.e., those of the form $F(a)r_{ab}F(b)r_{bc}F(c) \subseteq e$, where $a, b, c \in \{+, -\}$, $F(+) = f$ and $F(-) = t$, also hold in H_Y. This is by definition of t and f and because ψ is satisfied by the valuation $V_X \cup V_Y$ (that is, at least one literal in each clause has a loop with label t under $V_X \cup V_Y$). The constraints in (C5), i.e., $t_0 z \subseteq zt$ and $f_0 z \subseteq zf$ hold in H_Y by definition of t, f, t_0, f_0 and z. The constraints $yt \subseteq gy$ and $yf \subseteq gy$ in (C6a) hold

because in H_Y the node n_s has a loop labeled g. Finally, the constraint $gy \subseteq wz$ in (C6b) holds because by definition of H_Y, if $u_{ij} \in Y_0$ then there is a path with label z from n_t or n_f to n_{ij}, and there are edges with label w from n_s to n_t and n_f.

Since $H_Y \subseteq G_\phi$ by definition, in order to prove (d) it only remains to show that H is a strict subset of H_Y. Clearly, $H \not\subseteq H_Y$ since H does not have a loop labeled g in n_s; otherwise, there would be a path labeled sgs from n_s to itself in H, implying that (n_s, n_s) belongs to $L_0(H)$. This is a contradiction. Therefore, we only need to prove that $H \subseteq H_Y$. By definition, for each $r \notin \{g, t, f, z\}$ we have that $r^H \subseteq r^{H_Y}$. Moreover, since $g^{H_Y} = g^{G_\phi}$, we also have $g^H \subseteq g^{H_Y}$. In the following we prove this for $r \in \{z, t, f\}$.

To start with, we prove the following statement:

(†) For each $u_{ij} \in Y_0$, it is the case that n_{ij} does not have loops in H.

Intuitively, this comes from the fact that we are assuming for contradiction that (n_s, n_s) does not belong to the answer of L_0 in H. In fact, this implies that the "witness loop" with label g does not appear in H. Hence, no node associated with a variable in Y_0 gets a value. We formalize this below.

Let u_{ij} be a variable in Y_0. Assume for the sake of contradiction that n_{ij} has a loop labeled t in H (the case when the loop is labeled f is analogous). Notice that $y^H = y^{G_\phi}$ by (a). In particular, there is an edge with label y in H from n_s to n_{ij}. Together with the fact that that n_{ij} has a loop labeled t, this means that there is a path with label yt from n_s to n_{ij}. Since the constraint $yt \subseteq gy$ holds in H, there is path from n_s to n_{ij} with label gy. In particular, there is a loop labeled g on n_s. Therefore, there is a path from n_s to itself labeled sgs, i.e., (n_s, n_s) belongs to the answer of L_0 in H. This is a contradiction. This finishes the proof of (†).

To finish the proof it is sufficient to show that:

(I) $z^H = \emptyset$,

(II) $t^H = \{(n_{ij}, n_{ij}) : V_X(u_{ij}) = \top\}$, and

(III) $f^H = \{(n_{ij}, n_{ij}) : V_X(u_{ij}) = \bot\}$.

We prove (I), that is $z^H = \emptyset$. Assume for the sake of contradiction that there is a pair in z^H. Since $H \subseteq G_\phi$, this pair must be of the form (v, n_{ij}), where $v = n_t$ or $v = n_f$ and u_{ij} belongs to Y_0. We treat the case $v = n_t$, as the other case is analogous. By (a), the node n_t has a loop with label t_0

in H. Hence, there is a path with label $t_0 z$ from n_t to n_{ij}. Now, since the constraint $t_0 z \subseteq zt$ holds in H, there is a path with label zt from n_t to n_{ij}. By definition of G_ϕ, if there is an outgoing edge from n_{ij} with label t, this can only happen if n_{ij} has a loop labeled t. This contradicts (†) and the fact that u_{ij} belongs to Y_0.

Next we prove (II), the proof of (III) is similar. The inclusion from right to left follows from (b). For the inclusion from left to right, consider an arbitrary pair in t^H. Since $H \subseteq G_\phi$, this pair must be of the form (n_{ij}, n_{ij}). By (†), we have that u_{ij} does not belong to Y_0, i.e., it belongs to X_0. From (b) we conclude that $V_X(u_{ij}) = \top$. This finishes the proof of (II) and the proof of (d). It also finishes the left-to-right direction of the proof.

Implication from right to left. Suppose that for every $\subseteq$-repair H of G_ϕ it is the case that (n_s, n_s) belongs to the answer of L_0 in H. We prove that ϕ is satisfiable. Let V_X be an arbitrary valuation defined over the variables in X_0. We will define a valuation V_Y over the variables in Y_0 such that ψ is true under the valuation $V_X \cup V_Y$.

The idea is as follows:

(a) Using the valuation V_X, we define a graph database H_X satisfying the constraints Γ_0.

(b) We pick an arbitrary $\subseteq$-repair H'_X of G_ϕ such that $H_X \subseteq H'_X \subseteq G_\phi$.

(c) By assumption, (n_s, n_s) belongs to the answer of L_0 in H'_X. We prove that this implies that g is not empty in H'_X.

(d) Using the fact that g is not empty in H'_X, we define a valuation V_Y such that ψ is true under the valuation $V_X \cup V_Y$.

We start by defining the graph database H_X as follows. For each:

$$r \in \{e, g, t'_0, f'_0, y', d', w', r'_{++}, r'_{-+}, r'_{+-}, r'_{--}, z\},$$

we define r^{H_X} as the empty set. Further, for each:

$$r \in \{s, y, d, w, r_{++}, r_{-+}, r_{+-}, r_{--}, t_0, f_0\},$$

we define r^{H_X} as r^{G_ϕ}. For the remaining relations, namely, t and f, we have the following:

$$\begin{aligned} t^{H_X} &= \{(n_{ij}, n_{ij}) : V_X(u_{ij}) = \top\} \\ f^{H_X} &= \{(n_{ij}, n_{ij}) : V_X(u_{ij}) = \bot\}. \end{aligned}$$

We can check that all constraints in Γ_0 hold in H_X. Indeed, the constraints in (C1) vacuously hold because the relations d', t'_0 f'_0, y', w', and r'_{ab}, for $a, b \in \{+, -\}$, are empty. The constraint $tf \subseteq e$ in (C2) and the constraints $td \subseteq dt$ and $fd \subseteq df$ in (C3) hold because V_X is a valuation. In addition, the constraints in (C4) of the form $F(a)r_{ab}F(b)r_{bc}F(c) \subseteq e$ vacuously hold because we have assumed that every clause of ψ contains at least one variable from Y_0 and relations t and f in H_X are not defined in nodes representing occurrences of variables in Y_0. The constraints $t_0 z \subseteq zt$ and $f_0 z \subseteq zf$ in (C5) vacuously hold because z is empty in H_X. The constraints $yt \subseteq gy$ and $yf \subseteq gy$ in (C6a) hold because no node n_{ij}, with $u_{ij} \in Y_0$, has loops. Finally, the constraint $gy \subseteq ww$ in (C6b) holds because g is empty. In conclusion, $H_X \models \Gamma_0$.

Hence, there is a $\subseteq$-repair H'_X of G_ϕ such that $H_X \subseteq H'_X \subseteq G_\phi$. By assumption, (n_s, n_s) belongs to the answer of L_0 in H'_X. We prove that g is not empty in H'_X. We start by proving the following preliminary result.

Claim 2. *For each $r \in \{y', d', t'_0, f'_0, w', r'_{++}, r'_{-+}, r'_{+-}, r'_{--}\}$, it is the case that $r^{H'_X}$ is empty.*

Proof. We only prove the claim for y', the other proofs are similar. Suppose for contradiction that there is an edge with label y' from n_{ij} to n_{ij} in H'_X. By definition of y', it is the case that $u_{ij} \in Y_0$. Since $y^{H_X} = y^{G_\phi}$ and $H_X \subseteq H'_X \subseteq G_\phi$, we also have $y^{H'_X} = y^{G_\phi}$. In particular, there is an edge with label y from n_s to n_{ij} . Together with the fact that n_{ij} has a loop with label y', this implies that there is a path with label yy' from n_s to n_{ij}. But H'_X satisfies the constraint $yy' \subseteq e$ in (C1), and thus $(n_s, n_{ij}) \in e$. This contradicts the fact that $e^{H'_X} \subseteq e^{G_\phi} = \emptyset$. □

Recall now that (n_s, n_s) belongs to the answer of the RPQ:

$$L_0 \;=\; s \cdot (y' + d' + t'_0 + f'_0 + w' + r'_{++} + r'_{-+} + r'_{+-} + r'_{--} + g) \cdot s.$$

Together with the previous claim, this implies that there is an edge with label g in H'_X. This finishes the proof of (c).

We define now a valuation V_Y using the database H'_X. The valuation V_Y is such that for all $u_{ij} \in Y_0$:

$$n_{ij} \text{ has a loop with label } t \text{ (resp. } f) \quad \text{iff} \quad V_Y(u_{ij}) = \top \text{ (resp. } \bot). \qquad (5)$$

Intuitively, there is such a valuation because there is a "witness" loop with label g in H'_X. The existence of such loop, together with the constraint (C6b),

guarantee that for each node n_{ij}, with $u_{ij} \in Y_0$, there is an edge labeled z from n_t or n_f to n_{ij}. Using the constraints in (C5), we can see that this implies the existence of a loop with label t or f at the node n_{ij}. A node cannot have both because of constraint (C2). Moreover, nodes associated with the same variable have loops with the same labels because of (C3).

Formally, in order to prove that such a valuation V_Y exists, it is enough to show that for all $u_{ij}, u_{kl} \in Y_0$, the following holds:

(α) n_{ij} has a loop labeled t or one labeled f,

(β) if $u_{ij} = u_{kl}$, the node n_{ij} has a loop with label t (resp. f) iff n_{kl} has a loop with label t (resp. f).

We prove (α). By definition of y^{H_X} and since g is not empty in H'_X, there is a path labeled gy from n_s to n_{ij}. Since $gy \subseteq wz$ holds in H'_X, there is an edge labeled z from n_t or n_f to the node n_{ij}. Suppose that the edge starts at the node n_t, the other case is analogous. Since n_t has a loop with label t_0 in H'_X, there is a path with label $t_0 z$ from n_t to n_{ij} in H'_X. As the constraint $t_0 z \subseteq zt$ holds in H'_X, there is a path with label zt from n_t to n_{ij}. This implies that n_{ij} has a loop with label t in H'_X. Thus, in order to prove (α), it remains to show that n_{ij} cannot have both loops with label t and f. This follows easily from the fact that $tf \subseteq e$ holds in H'_X and e is empty in H'_X.

The proof that (β) is true comes from the fact that the constraints $td \subseteq dt$ and $fd \subseteq df$ are true. We do not give details here as it is essentially the same proof as the proof of (b)(ii) in the implication from left to right. This finishes the proof that there is a valuation V_Y satisfying (5) for all $u_{ij} \in Y_0$.

It remains to prove that ψ is true under $V_X \cup V_Y$. This basically comes from the following facts: (1) if u_{ij} is true (resp. false) under $V_X \cup V_Y$, then n_{ij} has a loop with label t (resp. f) in H'_X, and (2) the constraint $F(a) r_{ab} F(b) r_{bc} F(c) \subseteq e$ holds in H'_X, for each $a, b, c \in \{+, -\}$ where $F(+) = f$ and $F(-) = t$.

Formally let $z_{i1} \vee z_{i2} \vee z_{i3}$ be a clause in ψ. We only prove it for the case when the clause is of the form $u_{i1} \vee \neg u_{i2} \vee \neg u_{i3}$; the other cases are analogous. Suppose for the sake of contradiction that this clause is not satisfied by $V_X \cup V_Y$. That is, u_{i1} is false and both u_{i2} and u_{i3} are true under $V_X \cup V_Y$. If u_{i1} belongs to Y_0, we have by (5) that n_{i1} has a loop with label f in H'_X. If u_{i1} belongs to X_0, by definition of H_X we have that n_{i1} has a loop with label f in H_X. Since $H_X \subseteq H'_X$, it is the case that n_{i1} has a loop

with label f in H'_X, regardless of whether u_{ij} belongs to X_0 or Y_0. Similarly, we can show that n_{i2} and n_{i3} have loops with label t in H'_X.

Now, since $u_{i1} \vee \neg u_{i2} \vee \neg u_{i3}$ is a clause, we have by definition of r_{+-} and r_{--} that the path $(n_{i1}, r_{+-}, n_{i2})(n_{i2}, r_{--}, n_{i3})$ belongs to H'_X. Together with the facts that n_{i1} has a loop with label f and n_{i2} and n_{i3} have loops with label t, we obtain that $fr_{+-}tr_{--}t$ is the label of a path from n_{i1} to n_{i3} in H'_X. Since e is empty, this contradicts the fact that the constraints (C4) are true in H'_X. This finishes the proof that ψ is satisfied by $V_X \cup V_Y$. □

3.1. Interpreting RPCs from the origin

In the original proposal of Abiteboul and Vianu, RPQs, and therefore RPCs, are only evaluated from a particular node called the *origin*. This is motivated by their application over semistructured data. Such interpretation has also been studied recently for 2RPCs in [19]. Formally, let o be a fixed node id that we identify as the origin. A graph database $G = (V, E)$ satisfies the 2RPC $L_1 \subseteq L_2$ *from the origin*, denoted $G \models_o L_1 \subseteq L_2$, if and only if the origin o belongs to V and the following holds:

$$\{v \in V : (o, v) \in L_1(G)\} \subseteq \{v \in V : (o, v) \in L_2(G)\}.$$

If Γ is a set of 2RPCs, we write $G \models_o \Gamma$ if $G \models_o L_1 \subseteq L_2$ for each 2RPC $L_1 \subseteq L_2$ in Γ. (We restrict our attention to constraints given as 2RPCs only, as the origin semantics is specially tailored for them).

We can now modify the definition of repairs and consistent answers with respect to the restricted $\models_o$ interpretation of 2RPCs. Assume $\star \in \{\subseteq, \supseteq, \oplus\}$. An $\{o, \star\}$-*repair* of a graph database G under a set of 2RPCs Γ is defined exactly as an $\star$-repair of G under Γ, except that now the satisfaction of the 2RPCs in Γ is defined with respect to the relation $\models_o$. (For safety reasons, we assume that G contains the origin o in this case). For instance, G' is a $\{o, \subseteq\}$-repair of G under Γ, if (1) $G' \subseteq G$, (2) $G' \models_o \Gamma$, and (3) there is no graph database G'' such that $G' \subsetneq G'' \subseteq G$ and $G'' \models_o \Gamma$.

Furthermore, if q is a C2RPQ and Γ is a finite set of 2RPCs, we define $\{o, \star\}$-CQA(q,Γ) as the problem of, given a graph database $G = (V, E)$ and a tuple $\bar{a}$ of nodes in V, checking whether $\bar{a}$ is an $\{o, \star\}$-*consistent answer of* q *over* G *under* Γ, i.e., if $\bar{a} \in q(G')$, for each $\{o, \star\}$-repair G' of G under Γ.

We show next that interpreting RPCs under the origin semantics does not help lowering the complexity of the CQA problem under $\subseteq$-repairs. This is relevant, since we show on the other hand in Section 4.1 that the origin semantics does help when computing certain answers under $\supseteq$-repairs.

In particular, we establish the following:

Proposition 3. 1. *For each C2RPQ q and finite set Γ of 2RPCs over the same alphabet Σ, it is the case that $\{o, \subseteq\}$-CQA(q,Γ) is in Π_2^P.*

2. *There exist a finite alphabet Σ, a non-recursive RPQ L, and a finite set Γ of word constraints over Σ, such that checking whether (o, o) is an $\{o, \star\}$-consistent answer of L over G under Γ is Π_2^P-complete.*

Proof. The upper bound is obtained in exactly the same way than the upper bound in Theorem 1. We prove the lower bound next, i.e., there are an RPQ L_1 and a set Γ_1 of word constraints such that $\subseteq$-CQA(L_1, Γ_1) is Π_2^{P}-hard. In particular, we define an RPQ L_1, a set Γ_1 of word constraints, and construct in polynomial time a graph database G'_ϕ from ϕ in such a way that:

$$\phi \text{ is satisfiable} \iff (o, o) \in \{o, \subseteq\}\text{-Cons}(G'_\phi, L_1, \Gamma_1). \tag{6}$$

(Recall that ϕ is a quantified boolean formula of the form $\forall X_0 \exists Y_0 \psi$, where X_0, Y_0 are disjoint sets of variables and ψ is of the form:

$$(z_{11} \vee z_{12} \vee z_{13}) \wedge \cdots \wedge (z_{m1} \vee z_{m2} \vee z_{m3}),$$

where $z_{ij} \in \{x, \neg x, y, \neg y : x \in X_0, y \in Y_0\}$, for $1 \leq i \leq m$ and $1 \leq j \leq 3$).

To do this, we provide a slight modification of the reduction used in the proof of the lower bound of Theorem 1. Recall that in such proof we defined an RPQ L_0, word constraints Γ_0, and constructed in polynomial time a graph database G_ϕ from ϕ in such a way that:

$$\phi \text{ is satisfiable} \iff (n_s, n_s) \in \subseteq\text{-Cons}(G_\phi, L_0, \Gamma_0). \tag{7}$$

Our modification is as follows:

- Recall that V is the set of nodes in G_ϕ. The graph database G'_ϕ extends the graph database G_ϕ in the following way: We add an extra node, namely, the origin o. We add edges with label b from o to all nodes in V, and edges with label b' from all nodes in V to o. Finally, we extend relation s with all pairs of nodes in $(V \cup \{o\}) \times (V \cup \{o\})$.

- The set Γ_1 of constraints modifies Γ_0 as follows. We replace each word constraint of the form $w_1 \subseteq w_2$ in Γ_0 with the word constraint:

  $$bw_1 \subseteq bw_2.$$

  Also, we add the word constraint:

  $$bb' \subseteq e.$$

- Finally, we replace the RPQ L_0 by the RPQ:

$$L_1 \ := \ L_0 + sb's.$$

The following claim is the crux of our proof:

Claim 3. 1. *Let $G \subseteq G_\phi$ such that $G \models \Gamma_0$. Assume that H is the extension of G that adds all edges labeled b in G'_ϕ (i.e., all edges from the origin o to the nodes in V). Then $H \models_o \Gamma_1$.*

2. *Let $G \subseteq G'_\phi$ such that $G \models_o \Gamma_1$ and G contains all edges labeled b from G'_ϕ (i.e., all edges from the origin o to the nodes in V). Assume that H is the graph database that is obtained by removing the origin o (and thus all edges labeled b) from G. Then $H \models \Gamma_0$.*

Proof. We start with (1). Take first an arbitrary word constraint of the form $bw_1 \subseteq bw_2$ in Γ_1 such that $w_1 \subseteq w_2$ is in Γ_0. By definition of Γ_0, the word w_1 must be nonempty. Assume then that there is a path labeled bw_1 in H. This path starts in o and then follows a nonempty path π labeled w_1 from a node u to a node v. By construction of H, the path π must belong to G. But $G \models \Gamma_0$, and, therefore, $G \models w_1 \subseteq w_2$. Thus the nodes u and v are also linked by a path labeled w_2 in G. Then by the way H is constructed from G, there is a path labeled bw_2 from o to v in H. This proves that $H \models_o bw_1 \subseteq bw_2$. Moreover, it is also the case that $H \models_o bb' \subseteq e$ since there is no edge labeled b' in H. We conclude that $H \models_o \Gamma_1$.

Now we prove (2). There are two classes of word constraints in Γ_0: Those in (C1), (C2), and (C4), which are of the form $w_1 \subseteq e$, and those in (C3), (C5), and (C6), which are of the form $w_1 \subseteq w_2$, for w_2 a word different from e. We show first that each constraint of the form $w_1 \subseteq e$ in (C1), (C2), or (C4) holds in H. By assumption, $G \models_o \Gamma_1$, and thus $G \models_o bw_1 \subseteq be$. This implies that if a pair of the form (o, v) is linked by a path labeled bw_1 in G, it is also linked by a path labeled be. But e is empty in G'_ϕ, and thus there is no path labeled be in G. We conclude then that there is no path labeled bw_1 in G. By construction, this can only happen if there is no path labeled w_1 in H. Therefore, $H \models w_1 \subseteq e$.

Consider now the constraints in (C3), (C5), and (C6). We only consider the word constraint $gy \subseteq wz$ in (C6b), since all other cases are proved using exactly the same reasoning. Assume then that there is a pair of nodes in H that is linked by a path labeled gy. By definition of G_ϕ, this path must

start in node n_s, where it witnesses the loop g, and then continue via an edge labeled y to a node of the form n_{ij} such that $u_{ij} \in Y_0$. By assumption, $G \models_o \Gamma_1$, and thus $G \models_o bgy \subseteq bwz$. Since there is a path labeled bgy from o to n_{ij} in G, it must be the case then that there is a path labeled bwz from o to n_{ij} in G. But by definition of G_ϕ and G'_ϕ, this path must go from o to n_s via an edge labeled b, and then continue from n_s to n_{ij} via a path labeled wz. Therefore, there is a path labeled wz from n_s to n_{ij} in H. We conclude then that $H \models gy \subseteq wz$. □

We now prove (6); that is:

$$\phi \text{ is satisfiable} \iff (o,o) \in \{o, \subseteq\}\text{-Cons}(G'_\phi, L_1, \Gamma_1).$$

Implication from right to left. Assume that $(o,o) \in \{o, \subseteq\}$-Cons$(G'_\phi, L_1, \Gamma_1)$. We prove next that $(n_s, n_s) \in \subseteq$-Cons(G_ϕ, L_0, Γ_0). This implies that ϕ is satisfiable from (7).

In fact, assume for contradiction that $(n_s, n_s) \notin \subseteq$-Cons$(G_\phi, L_0, \Gamma_0)$. Then there is a $\subseteq$-repair H of G_ϕ under Γ_0 such that $(n_s, n_s) \notin L_0(H)$. Let H' be the graph database that extends H with all edges labeled b connecting the origin o with the nodes in V. We prove next that H' is an $\{o, \subseteq\}$-repair of G'_ϕ under Γ_1 such that $(o,o) \notin L_1(H')$. This contradicts the fact that $(o,o) \in \{o, \subseteq\}$-Cons$(G'_\phi, L_1, \Gamma_1)$, thus finishing the proof.

We start by proving that H' is an $\{o, \subseteq\}$-repair of G'_ϕ under Γ_1. It follows from the first part of Claim 3 that $H' \models_o \Gamma_1$. This is because $H \subseteq G_\phi$, it is the case that $H \models \Gamma_0$, and H' is the extension of H with all edges labeled b from G'_ϕ. Assume now for the sake of contradiction that there is an H'' such that $H' \subsetneq H'' \subseteq G'_\phi$ and $H'' \models_o \Gamma_1$. Then all edges in H'' that do not belong to H' must belong to G_ϕ. Assume otherwise. Then H'' extends H' with (i) an edge labeled s, (ii) an edge labeled b, or (iii) an edge labeled b'. Case (i) is impossible. This is because H' is an $\{o, \subseteq\}$-repair of G'_ϕ under Γ_1, and Γ_1 does not mention s. Therefore, $s^{H'} = s^{G'_\phi}$. Case (ii) is also impossible, since all edges labeled b are already in H' by definition. Finally, case (3) is impossible. Otherwise $H'' \not\models_o bb' \subseteq e$, which is a contradiction.

Therefore, the restriction H_0 of H'' to the edges in G_ϕ satisfies that $H \subsetneq H_0 \subseteq G_\phi$. Moreover, $H_0 \models \Gamma_0$. This follows directly from the second part of Claim 3, since $H'' \models_0 \Gamma_1$ and H'' contains all edges labeled b from G'_ϕ. We conclude that H is not a $\subseteq$-repair of G_ϕ under Γ_0, which is a contradiction.

We finally prove that $(o,o) \notin L_1(H')$. Assume otherwise. Then it must be the case that there is a path labeled L_1 in H'. But H' contains no

edge labeled b', which implies that there is a path labeled L_0 in H'. By construction, this path is labeled sas, where a is a symbol different from b and s. This means that there is an edge labeled a also in H. Moreover, since Γ_0 does not mention the symbol s and H is a $\subseteq$-repair of G_ϕ under Γ_0, it is the case that $s^H = s^{G_\phi} = V \times V$. We conclude that there is a path labeled sas from n_s to n_s in H, and thus that $(n_s, n_s) \in L_0(H)$. This is a contradiction.

Implication from left to right. Assume that ϕ is satisfiable, and for the sake of contradiction that $(o, o) \notin \{o, \subseteq\}\text{-Cons}(G'_\phi, L_1, \Gamma_1)$. We prove next that $(n_s, n_s) \notin \subseteq\text{-Cons}(G_\phi, L_0, \Gamma_0)$, thus contradicting (7).

Since $(o, o) \notin \{o, \subseteq\}\text{-Cons}(G'_\phi, L_1, \Gamma_1)$, there is an $\{o, \subseteq\}$-repair H' of G'_ϕ under Γ_1 such that $(o, o) \notin L_1(H')$. Therefore, $H' \subseteq G'_\phi$ and $H' \models_o \Gamma_1$. Moreover, H' contains all edges labeled b from G_ϕ. Assume otherwise, i.e., there is a node $v \in V$ such that there is no edge labeled b from o to v. By the maximality property of $\subseteq$-repairs, it must be the case then that there is an edge labeled b' from v to o (as, otherwise, we could extend H' with such an edge without violating Γ_1). Moreover, since Γ_1 does not mention the symbol s and H' is an $\{o, \subseteq\}$-repair of G'_ϕ under Γ_1, it is the case that $s^{H'} = s^{G'_\phi} = (V \cup \{o\}) \times (V \cup \{o\})$. We conclude that there is a path labeled $sb's$ from o to o in H', and thus that $(o, o) \in L_1(H')$. This is a contradiction.

We obtain then from the previous paragraph and the second part of Claim 3, that removing the origin o from H' (and thus all nodes labeled b) yields an H such that $H \subseteq G_\phi$ and $H \models \Gamma_0$. We show next that H is a $\subseteq$-repair of G_ϕ under Γ_0 such that $(n_s, n_s) \notin L_0(H)$. This is our desired contradiction.

Assume first, for the sake of contradiction, that H is not a $\subseteq$-repair of G_ϕ under Γ_0. Since $H \models \Gamma_0$, it must be the case that there is an H_0 such that $H \subsetneq H_0 \subseteq G_\phi$ and $H_0 \models \Gamma_0$. Let H'_0 be the extension of H_0 with all edges labeled b from the origin o to the nodes in V. Then $H'_0 \models_o \Gamma_1$ from the first part of Claim 3. Moreover, since $H \subsetneq H_0 \subseteq G_\phi$ it must be the case by construction that $H' \subsetneq H'_0 \subseteq G'_\phi$. We conclude that H' is not an $\{o, \subseteq\}$-repair of G'_ϕ under Γ_1, which is a contradiction.

We finally prove that $(n_s, n_s) \notin L_0(H)$. Assume otherwise. Then it must be the case that there is a path labeled L_0 in H. By definition of L_0, this path is labeled sas, where a is a symbol mentioned in G_ϕ. Now, since Γ_1 does not mention the symbol s and H' is an $\{o, \subseteq\}$-repair of G'_ϕ under Γ_1, it is the case that $s^{H'} = s^{G'_\phi} = (V \cup \{o\}) \times (V \cup \{o\})$. Therefore, there is also a path labeled sas from o to o in H'. We conclude that $(o, o) \in L_0(H')$, which

implies that $(o, o) \in L_1(H')$. This is a contradiction. □

3.2. Tractable restrictions

Due to the inherent high complexity of our CQA problem under the subset repair semantics, it is important to look for meaningful restrictions leading to tractability. We provide two such restrictions in this section. The first one is based on the class of LAV RPCs, and the second one on the class of graph databases of bounded treewidth.

Restricting RPCs. In the relational case, the data complexity of CQA for CQs under the class of *LAV tgds* (i.e., tgds with a single atom in the left-hand side [35]) is tractable. This actually holds for any of the three repair semantics [20]. The direct analogue of LAV tgds in our setting is the class of *LAV* C2RPCs, which are C2RPCs with a single symbol in the left-hand side. Formally, a LAV C2RPC over Σ is a C2RPC of the form $(x, a, y) \subseteq q(x, y)$, where a is a symbol in Σ or the inverse of a symbol in Σ, and q is a C2RPQ over Σ with free variables x, y. Correspondingly, a LAV RPC is a constraint of the form $a \subseteq L$, for $a \in \Sigma$ and L a RPQ. We can leverage the techniques used to study CQA under LAV tgds to prove tractability in data complexity for the CQA problem for C2RPQs under LAV C2RPCs.

Theorem 2. *For each C2RPQ q and finite set Γ of LAV C2RPCs over the same alphabet Σ, it is the case that $\subseteq$-CQA(q, Γ) is in* NLogspace.

Proof. The proof is an immediate adaption of the proof of Theorem 4.4 in [20]. We still provide it, as it is short. Let Γ be a set of constraints of the form $(x, a, y) \subseteq q'(x, y)$, where $a \in \Sigma^{\pm}$ and q' is a C2RPQ. The result follows from the fact that in this case any graph database G admits a unique $\subseteq$-repair H_0 which can be computed in NLogspace. We can then compute the certain answers to q by evaluating q over H_0. This can be done in NLogspace from Proposition 1. The whole process can be carried out in NLogspace since NLogspace computable functions are closed under composition.

We show how to compute H_0 and its unicity. We define the following procedure P_0. Given a graph database $G = (V, E)$, for each constraint $(x, a, y) \subseteq q(x, y)$ (resp., $(x, a^-, y) \subseteq q'(x, y)$) in Γ, where $a \in \Sigma$, we do the following: For each edge (u, a, v) in G (resp., edge (v, a, u) in G), if (u, v) does not belong to $q'(G)$, remove (u, a, v) from G (resp., (v, a, u) from G).

Starting with G, we respectively apply the procedure P_0 until we stop removing tuples. We let H_0 be the obtained graph database. Clearly, H_0

can be computed in NLOGSPACE. This is because the edges in $E \cup E^-$ can be enumerated in logarithmic space and C2RPQs can be evaluated in NLOGSPACE. The claim then follows from the fact that NLOGSPACE computable functions are closed under composition. We show next that H_0 is the unique $\subseteq$-repair of G. First, we observe that Γ holds in H_0. This follows from the fact that when applying the procedure P_0 to H_0, we obtain H_0.

Next we prove that for all repairs H, we have $H = H_0$. Let H be a $\subseteq$-repair of G. We can inductively show that after each application of P_0 over G, the repair H is a subset of the obtained database. In particular, H is a subset of H_0. Since $H \subseteq H_0$, the graph H is a $\subseteq$-repair, and Γ holds in H_0, it follows from the maximality condition of $\subseteq$-repairs that $H = H_0$. $\square$

It is interesting to also consider C2RPCs based on the class of *GAV* tgds [35], that only allow for one symbol on the right-hand side. That is, a GAV C2RPC over Σ is of the form $q(x, y) \subseteq (x, a, y)$, for q a C2RPQ over Σ and a a symbol in Σ or the inverse of a symbol in Σ. Correspondingly, a GAV RPC is of the form $L \subseteq a$, for $a \in \Sigma$ and L and RPQ. While this restriction improves the complexity of the CQA problem, it does not lead to tractability.

Proposition 4. 1. *For each C2RPQ q and finite set Γ of GAV C2RPCs over the same alphabet Σ, it is the case that $\subseteq$-CQA(L,Γ) is in* coNP.
2. *There exist a finite alphabet Σ, a non-recursive RPQ L over Σ, and a single GAV RPC γ of the form $ab \subseteq c$, where $a, b, c \in \Sigma$, such that $\subseteq$-CQA(L,γ) is* coNP*-complete.*

Proof. We show that for each C2RPQ q and finite set Γ of GAV C2RPCs over the same alphabet Σ, it is the case that $\subseteq$-CQA(q,Γ) is in coNP. Let q be a C2RPQ and Γ a set of constraints of the form $q'(x, y) \subseteq (x, a, y)$, where q' is an C2RPQ and a belongs to $\Sigma^{\pm}$. Let also G be a graph database. First, we recall that the *subset repair checking problem* for a fixed set of GAV tgds over relational databases is in polynomial time [39]. If we fix a set of GAV tgds Γ', the subset repair checking problem is the following problem: given two relational databases D' and D, check whether D' is a subset repair of D under Γ'. The proof in [39] extends immediately to the setting of graph databases and GAV C2RPCs. That is, checking whether a graph database H is a $\subseteq$-repair of G under a set Γ of C2RPCs, is in polynomial time.

Hence, we have the following coNP procedure to check whether a tuple $\bar{a}$ belongs to $\subseteq$-Cons(G, q, Γ). We nondeterministically pick a subset H of G. We then check in polynomial time whether H is a $\subseteq$-repair of G with respect

to Γ. We can then check in polynomial time whether $\bar{a}$ belongs to $q(G)$ from Proposition 1. This finishes the proof of the first part of the theorem.

We prove now the second part of the theorem concerning the hardness. We provide a reduction from MONOTONE 1-IN-3 SAT to $\subseteq$-CQA(L_0, γ_0), where γ_0 is the constraint $tf \subseteq e$ and L_0 will be defined later. Recall that MONOTONE 1-IN-3 SAT is the problem of checking whether a formula in 3-CNF, without negative literals, admits a satisfying assignment that makes true exactly one propositional variable per clause. Let ϕ be a boolean formula in conjunctive normal form $C_1 \wedge \cdots \wedge C_m$, where each clause C_i is of the form:

$$x_{i1} \vee x_{i2} \vee x_{i3}.$$

We associate with ϕ a graph database G_ϕ and a node u_0 in such a way that:

$$\phi \in \text{MONOTONE 1-IN-3 SAT} \quad \text{iff} \quad (u_0, u_0) \notin \subseteq\text{-Cons}(G_\phi, L_0, \gamma_0). \tag{8}$$

The graph database G_ϕ is defined as follows. Its set V of nodes is:

$$\{n_{ij} : 1 \leq i \leq m, 1 \leq j \leq 3\}.$$

That is, we associate a node n_{ij} with each occurence of a variable x_{ij} in ϕ. Note that even if $x_{ij} = x_{kl}$ the nodes n_{ij} and n_{kl} associated with x_{ij} and x_{kl}, respectively, are distinct. We define the node u_0 as the node n_{11}. The schema is equipped with relations e, s, d, r, t and f, which are interpreted in the following way:

$$\begin{aligned}
e^{G_\phi} &= \emptyset, \\
s^{G_\phi} &= V \times V, \\
d^{G_\phi} &= \{(n_{ij}, n_{kl}) : x_{ij} = x_{kl}\}, \\
r^{G_\phi} &= \{(n_{i1}, n_{i2}), (n_{i2}, n_{i3}), (n_{i3}, n_{i1}) : 1 \leq i \leq m\}, \\
t^{G_\phi} &= f^{G_\phi} = \{(n, n) : n \in V\}.
\end{aligned}$$

Intuitively, the relation d specifies which nodes correspond to the same variable. The relation r specifies which variables occur in the same clause. The relations t and f are loops on each node. The intuition is as follows. Since e is empty in G_ϕ, the relation e is also empty in any $\subseteq$-repair H of G_ϕ. Hence, the constraint γ_0 specifies that a node in a repair H cannot have a loop with label t and a loop with label f. This allows us to define a boolean assignment

V_H associated with H in the following way: If n_{ij} has a loop with label t, we map n_{ij} to $\top$, otherwise, n_{ij} is mapped to $\bot$.

Note that V_H might not define a valuation as there might variables x_{ij} and x_{kl} such that $x_{ij} = x_{kl}$, the node n_{ij} has a loop with label t and n_{kl} has a loop with label f. By definition of d, this is equivalent to say that there might be a path with label tdf, between the nodes n_{ij} and n_{kl}. In fact, the map V_H is a valuation iff there is no path with label tdf in the repair H.

So, informally, each $\subseteq$-repair of G_ϕ with no such path corresponds to a valuation. Now we want to define the query L_0 in such a way that if L_0 is true in a repair H with an associated valuation V_H, then V_H is not a "witness" for the membership of ϕ in Monotone 1-in-3 Sat (we say that a valuation is a *witness* if exactly one variable in each clause of ϕ is true under the valuation). But a valuation V_H associated with a repair H is not a witness for the membership of ϕ in Monotone 1-in-3 Sat if and only if there exists an $1 \leq i \leq m$ such that:

(†) V_H takes value $\top$ in at least two elements of $\{x_{i1}, x_{i2}, x_{i3}\}$, or

(††) $V_H(x_{i1}) = V_H(x_{i2}) = V_H(x_{i3}) = \bot$.

By definition of r and V_H, condition (†) means that trt is the label of a path in H, while (††) means that $frfrf$ is the label of a path in H. We let W be the RPQ:

$$trt + frfrf.$$

So if H is a $\subseteq$-repair of G_ϕ, either it is not associated with a valuation (that is, there is a path with label tdf) or it is associated with a valuation and, in that case, it must admit a path with label W (in order to ensure that the valuation is not a witness). Intuitively, this suggests us to define L_0 as the RPQ:

$$stdfs + sWs.$$

We prove now that equivalence (8) holds. First we show the implication from left to right. Suppose that ϕ belongs to Monotone 1-in-3 Sat. Let V_0 be a valuation such that for all clauses of ϕ, exactly one variable is true under the valuation V_0. We need to find a repair H_0 such that (u_0, u_0) does not belong to the answer of L_0. We let H_0 be the following graph database. For all $u \in \{e, s, d, r\}$, we define u^{H_0} as u^{G_ϕ}, while t^{H_0} and f^{H_0} are given by:

$$\begin{aligned} t^{H_0} &= \{n_{ij} : V_0(x_{ij}) = \top\}, \\ f^{H_0} &= \{n_{ij} : V_0(x_{ij}) = \bot\}. \end{aligned}$$

It is easy to check that H_0 is a $\subseteq$-repair of G_ϕ with respect to γ_0. Now it remains to show that (u_0, u_0) does not belong to $L_0(H_0)$. By definition of L_0 and s^{H_0}, this is equivalent to proving that:

$$\text{There is no path with label } tdf \text{ or label } W. \tag{9}$$

First we show that there is no path with label tdf. Intuitively, this simply comes from the fact that nodes corresponding to the same variable admit the same loops. Suppose for contradiction that there is such a path. By definition of H_0, there must be nodes n_{ij} and n_{kl} such that n_{ij} has a loop with label t, the node n_{kl} has a loop with label f and $(n_{ij}, n_{kl}) \in d^{H_0}$. By definition of d, this means that $x_{ij} = x_{kl}$. Moreover, since n_{ij} has a loop with label t in H_0, this means that $V_0(x_{ij}) = \top$. Similarly, we can infer that $V_0(x_{kl}) = \bot$. This contradicts the fact that $x_{ij} = x_{kl}$.

In order to prove (9), it remains to show that there is no path in H_0 with label satisfying W. This comes from the fact that V_0 is a witness valuation. Suppose for contradiction that there is such a path. Its label is either trt or $frfrf$. We only treat the case where the label is $frfrf$, as the other case is analogous. By definition of r and t, if there is a path with label $frfrf$, this means that there are nodes n_{i1}, n_{i2} and n_{i3}, all of them with a loop with label f. By definition of f in H_0, this means that $V_0(x_{i1}) = \bot$, $V_0(x_{i2}) = \bot$ and $V_0(x_{i3}) = \bot$. This contradicts the fact that V_0 makes at least one variable true in each clause of ϕ.

Next we prove the implication from right to left of (8). Suppose that (u_0, u_0) does not belong to the answer of L_0 in a $\subseteq$-repair H of G_ϕ. Since there is no edge with label e in G_ϕ and H is a subset of G_ϕ, we also have that $e^H = \emptyset$. Since H is a repair, the constraint $tf \subseteq e$ holds in H. As there is no edge with label e in H, this means that no node has loops both with labels t and f. Moreover, using the maximality condition of $\subseteq$-repairs, we can show that each node has at least one loop with label t or f. This allows us to define a map F over the set of nodes of G_ϕ in the following way:

$$F(n_{ij}) = \begin{cases} \top & \text{if } n_{ij} \text{ has a loop with label } t, \\ \bot & \text{if } n_{ij} \text{ has a loop with label } f. \end{cases}$$

We would like now to define a valuation V_F such that $V_F(x_{ij}) = F(n_{ij})$. In order to do that we need to show that V_F is well-defined, i.e., that the

following does not hold:

$$\text{There are nodes } n_{ij} \text{ and } n_{kl} \text{ such that } F(n_{ij}) \neq F(n_{kl}) \text{ and } x_{ij} = x_{kl}. \tag{10}$$

Using the fact (u_0, u_0) does not belong to the answer of L_0 in H, we prove that (10) does not happen. Intuitively, this comes from the fact that H does not have a path with label tdf.

Suppose then, for the sake of contradiction, that there are nodes n_{ij} and n_{kl} such that $F(n_{ij}) \neq F(n_{kl})$ and $x_{ij} = x_{kl}$. Since $F(n_{ij}) \neq F(n_{kl})$, either (a) n_{ij} has a loop with label t and n_{kl} has a loop with label f, (b) or n_{ij} has a loop with label f and n_{kl} has a loop with label t. Suppose that (a) happens, the other case is identical. Since $x_{ij} = x_{kl}$, by definition of d there is an edge with label d from n_{ij} to n_{kl} in G_ϕ. By Lemma 2, $d^H = d^{G_\phi}$. Hence, there is an edge with label d from n_{ij} to n_{kl} in H. Together with (a), this implies that there is a path with label tdf.

Again, by Lemma 2, we have $s^H = s^G$, that is, $s^H = V \times V$. Together with the fact that there is a path with label tdf, this implies that there is a path with label $stdfs$ from u_0 to u_0, which contradicts the fact that (u_0, u_0) does not belong to $L_0(H)$. This finishes the proof that (10) does not happen. Hence, we can define a valuation V_F such that $V_F(x_{ij}) = F(n_{ij})$. That is,

$$V_F(x_{ij}) = \begin{cases} \top & \text{if } n_{ij} \text{ has a loop with label } t \\ \bot & \text{if } n_{ij} \text{ has a loop with label } f. \end{cases}$$

In order to finish the proof, we need to prove that V_F is a valuation such that exactly one variable is satisfied in each clause of ϕ. Suppose for contradiction that V_F is not such a valuation. That is, there is a clause $x_{i1} \vee x_{i2} \vee x_{i3}$ such that either two variables are true or all variables are false under V_F. We treat the second case, the other case is analogous.

Suppose then that $V_F(x_{i1}) = \bot$, $V_F(x_{i2}) = \bot$ and $V_F(x_{i3}) = \bot$. By definition of V_F, this means that in H the nodes n_{i1}, n_{i2} and n_{i3} have loops with label f. By Lemma 2, we have that $r^H = r^{G_\phi}$. Together with the definition of r^{G_ϕ}, we obtain that $(n_{i1}, n_{i2}) \in r^H$ and $(n_{i2}, n_{i3}) \in r^H$. Since n_{i1}, n_{i2} and n_{i3} have loops with label f, this means that there is a path with label $frfrf$ between the nodes n_{i1} and n_{i3}. Recall that by Lemma 2, $s^H = s^G$, that is, $s^H = V \times V$. Together with the fact that there is a path with label $frfrf$, this implies that there is a path with label $sfrfrfs$ from u_0 to u_0. This contradicts the fact that (u_0, u_0) does not belong to $L_0(H_0)$. $\square$

As before, interpreting GAV RPCs from the origin does not help CQA under $\subseteq$-repairs. In particular:

Proposition 5. 1. *For each C2RPQ q and finite set Γ of GAV 2RPCs over the same alphabet Σ, we have that $\{o, \subseteq\}$-CQA(q,Γ) is in* coNP.

2. *There exist a finite alphabet Σ, a non-recursive RPQ L, and a finite set Γ of GAV word constraints over Σ, such that checking whether (o, o) is an $\{o, \star\}$-consistent answer of L over G under Γ is* coNP*-complete.*

Proof. The upper bound uses exactly the same line of reasoning that the upper bound in the proof of Proposition 4. For the hardness, we apply the techniques developed in the proof of the lower bound of Proposition 3, and correspondingly modify the reduction used in the proof of the lower bound of Proposition 4. The main reason why such modification works in this case is because the only word constraint used in the proof of the lower bound of Proposition 4 is of the form $tf \subseteq e$, and the interpretation of e is empty in the underlying graph database used in such proof. This helps proving that Claim 3 continues to hold, which was the main ingredient used in the reduction given in the proof of Proposition 3. □

Note that in the setting of relational databases, the data complexity of consistent query answering for unions of conjunctive queries with respect to GAV constraints, was also shown to be complete for the class coNP.

Notice that the second part of the previous proposition shows that, in a sense, the tractability result for LAV C2RPCs in Theorem 2 is optimal: Allowing two-letter words on the left-hand side of RPCs leads to intractability, even if the right-hand side consists of a single letter.

Restricting graph databases. Our CQA problem under the subset repair semantics can be reformulated as a *monadic second-order logic* (MSO) evaluation problem over a relational representation of graph databases. This allows us to apply results establishing the tractability of MSO over structures of bounded treewidth [22]. From those results, we obtain tractability for the CQA problem over graph databases of bounded treewidth.

Formally, let $G = (V, E)$ be a graph database. A *tree decomposition* of G is a pair (T, λ), where T is a tree and $\lambda : T \to 2^V$ maps each node t in T to a nonempty set $\lambda(t)$ of nodes in V, that satisfies the following conditions:

- For each $v \in V$ the set $\{t \in T : v \in \lambda(t)\}$ is a connected subset of T.

- For each edge $(u, a, v) \in E$, it is the case that $\{u, v\} \subseteq \lambda(t)$, for some $t \in T$.

The *width* of the tree decomposition (T, λ) is $\max\{|\lambda(t)| - 1 : t \in T\}$. The *treewidth* of G is the minimum width of a tree decomposition of G. For instance, the treewidth of G is one if and only if the underlying undirected graph of G is a tree.

We then obtain the following:

Theorem 3. *Let q be a C2RPQ and Γ a set of C2RPCs over the same alphabet Σ. Then $\subseteq$-CQA(q,Γ) can be solved in linear time over graph databases of treewidth $\leq k$, for each $k \geq 1$.*

Proof. Let G be a graph database over Σ. We assume without loss of generality that each symbol in Σ is mentioned either in q or in Γ. Otherwise we can safely remove edges labeled with such symbol from G, as they affect neither the satisfaction of the constraints nor the satisfaction of the query. Thus, we can assume $|\Sigma|$ to be fixed.

Consider now the following two-sorted relational representation $\mathcal{T}(G) = (A, N, (E^i_a, E^o_a)_{a\in\Sigma})$ of G. The domain A of $\mathcal{T}(G)$ consists of elements representing each node and each edge in G. The interpretation of unary relation N consists of all elements in A that represent nodes. The interpretation of the binary relation E^i_a, for $a \in \Sigma$, is the set of all pairs (e, v) such that e corresponds to an edge of the form $(v', a, v) \in E$. Finally, the interpretation of the binary relation E^o_a, for $a \in \Sigma$, is the set of all pairs (v, e) such that e corresponds to an edge of the form $(v, a, v') \in E$.

Tree decompositions of structures of the form $\mathcal{T}(G)$ are defined exactly as tree decompositions of G, save for the fact that we now ask each pair (e, v) in some E^i_a or (e, v) in some E^o_a to be contained in some set of the form $\lambda(t)$, for $t \in T$. It can be proved that if the treewidth of G is at most k, for $k \geq 1$, then the treewidth of the relational structure $\mathcal{T}(G)$ is at most $(k+1)^2 \cdot |\Sigma| - 1$. In fact, consider a tree decomposition (T, λ) of G of width at most k. We can obtain a tree decomposition of $\mathcal{T}(G)$ by doing the following: For each edge $e = (v, a, v') \in E$ we arbitrarily pick an element $t \in T$ such that $\lambda(t)$ contains v and v'. Such t must exist by tree decomposition properties. Then we add e to this bag. Clearly, each node in the resulting tree decomposition contains at most $(k + 1)$ nodes and $(k + 1)^2 \cdot |\Sigma|$ edges from G. We conclude that the treewidth of $\mathcal{T}(G)$ is also fixed.

It is easy to prove that for every C2RPQ q and finite set Γ of C2RPCs, there is an MSO formula Φ over the vocabulary $\langle N, (E_a^i)_{a\in\Sigma}, (E_a^o)_{a\in\Sigma}\rangle$, such that for each graph database G and tuple $\bar{a}$ of nodes in G:

$$\bar{a} \in \subseteq\text{-Cons}(G, q, \Gamma) \quad \iff \quad \mathcal{T}(G) \not\models \Phi.$$

The formula Φ expresses that there is a subset of A that represents a graph database G' that is contained in G, for which the following holds:

(1) G' is a $\subseteq$-repair of G. That is, we need to express that G' satisfies Γ (which can be done in MSO over this representation using standard automata techniques), and that for every subset G'' of G that satisfies Γ we have that $G'' \not\models \Gamma$ or it is false that $G' \subsetneq G''$. It is clear that this can be expressed in MSO over $\mathcal{T}(G)$.

(2) It is not the case that $\bar{a} \in q(G')$. This can again be expressed in MSO over this representation using standard automata techniques.

Now the result follows from the fact that MSO formulas can be evaluated in linear time over structures of bounded treewidth [22]. □

4. CQA under Superset and Symmetric Difference Repairs

We prove in this section that our CQA problem is undecidable under the semantics of $\supseteq$- and $\oplus$-repairs. This holds even for queries defined as non-recursive and word constraints:

Theorem 4. *Assume $\star \in \{\supseteq, \oplus\}$. There exist a finite alphabet Σ, a non-recursive RPQ L, and a set Γ of word constraints over Σ, such that $\star$-CQA(L,Γ) is undecidable.*

In order to prove this theorem we establish a connection with the *implication problem* for RPCs [1, 28]. Recall that this is the problem of, given a finite set Γ of RPCs and an RPC $L_1 \subseteq L_2$, checking whether $\Gamma \models L_1 \subseteq L_2$, i.e., if $G \models \Gamma$ implies $G \models L_1 \subseteq L_2$, for every graph database G. Grahne and Thomo proved this problem to be undecidable, even for word constraints, using a reduction from the *word rewrite* problem [28]. We develop nontrivial adaptations of such reduction to prove Theorem 4. The reason why we have to develop such adaptations is that there exist differences in nature between CQA and the implication problem for constraints. First, in the CQA problem

we do not reason about all graph databases that satisfy the constraints (as in the case of the implication problem), but only about those that minimally differ from the original graph database. (Note, however, that in the special case of the superset semantics, this problem does not apply. Indeed, given a graph database G, the intersection of the answers of a *monotone* query L over all the $\supseteq$-repairs is equal to the the intersection of the answers of L over all the databases containing G and satisfying the constraints).

Second, we study the data complexity of the CQA problem, and, therefore, our goal is to prove undecidability of CQA for a *fixed* set of RPCs and a *fixed* RPQ. This is different to the case of the implication problem in which RPCs and RPQs define the input, and, therefore, cannot be fixed.

Proof of Theorem 4 in the case when $\star$ is $\supseteq$: We start by recalling the basic notions of rewrite systems. Let Δ be a finite alphabet. A *semi-Thue rewrite system* $\mathcal{R}$ over Δ is a finite subet of $\Delta^* \times \Delta^*$. A rewrite system $\mathcal{R}$ induces a single-step reduction relation $\rightarrow_{\mathcal{R}}$ over Δ^* defined as:

$$\rightarrow_{\mathcal{R}} = \{(v, w) : v = xty, w = xuy, \text{ for some } (t, u) \in \mathcal{R} \text{ and } x, y \in \Delta^*\}.$$

We let $\rightarrow^*_{\mathcal{R}}$ be the reflexive transitive closure of $\rightarrow_{\mathcal{R}}$. If $\mathcal{R}$ is clear from the context, we simply write $\rightarrow$ instead of $\rightarrow_{\mathcal{R}}$ and $\rightarrow^*$ instead of $\rightarrow^*_{\mathcal{R}}$. We define $Anc(v)$, the set of *ancestors of* v, as the set:

$$\{u \in \Delta^* : u \rightarrow^*_{\mathcal{R}} v\}.$$

The problem of testing whether a pair (u, v) belongs to $\rightarrow^*_{\mathcal{R}}$ is called the *rewrite problem* for $\mathcal{R}$. It is well known that there is a fixed semi-Thue rewrite system such that its rewrite problem is undecidable (see e.g., [9]).

The construction. Let $\mathcal{R}$ be a fixed semi-Thue rewrite system over alphabet Δ such that its rewrite problem is undecidable. We define a set Γ of RPCs and a non-recursive RPQ L_0 such that for all words w_1 and w_2 over Δ, there is a graph database G with a node n_0 satisfying:

$$w_1 \rightarrow^* w_2 \quad \text{iff} \quad (n_0, n_0) \in \supseteq\text{-Cons}(G, L_0, \Gamma).$$

In our construction, only G depends on (w_1, w_2), while L_0 and Γ are fixed.

Let w_1 and w_2 be two words in Δ^*. We assume that

$$\begin{aligned} w_1 &= w_{11}w_{12}\dots w_{1k}, \\ w_2 &= w_{21}w_{22}\dots w_{2l}, \end{aligned}$$

where $w_{1i}, w_{2j} \in \Delta$ for each $1 \leq i \leq k$ and $1 \leq j \leq l$. Therefore k is the length of w_1 and l is the length of w_2. We start by defining the graph database G. We let V, the set of nodes of G, to be defined as follows:

$$\{n_i : 0 \leq i \leq k\} \cup \{m_i : 0 < i < l\}.$$

We define m_0 as n_0 and m_l as n_k. Now, for each $a \in \Delta$, we have edges with label a and $\hat{a}$. We also have special edges with label \$. The graph database G is defined by the following binary relations:

$$\begin{array}{rcl} a^G & = & \{(n_{i-1}, n_i) : w_{1i} = a, 1 \leq i \leq k\} \\ \hat{a}^G & = & \{(m_i, m_{(i-1)}) : w_{2i} = a, 1 \leq i \leq l\}, \\ \$^G & = & \{(n_k, n_k)\}, \end{array}$$

where $a \in \Delta$. Thus, G consists of a path with label w_1 from n_0 to n_k and a path with label $\hat{w}_{2l}\hat{w}_{2(l-1)} \ldots \hat{w}_{21}$ from n_k to n_0.

We now define Γ as the following set of RPCs:

$$\begin{array}{rcl} u & \subseteq & v, \\ a\$\hat{a} & \subseteq & \$, \end{array}$$

where $(u, v) \in \mathcal{R}$ and $a \in \Delta$. (Recall that $\mathcal{R}$ is fixed, and thus Γ is fixed). We define L_0 as the symbol \$.

The intuition. Basically we start with the path with label w_1 in G. The idea is that if $w_1 \rightarrow^* w_2$, then applying the constraints of the form $u \subseteq v$, we will construct a path with label w_2. Now the query is \$ and we have to check whether n_0 has a loop with label \$.

The idea is that the presence of a path with label w_2 from n_0 to n_k is witnessed by a loop with label \$ at n_0, using the constraints of the form $a\$\hat{a} \subseteq \$$. Indeed, suppose that $s_0 w_{21} s_1 w_{22} \ldots s_l$ is a path with label w_2 from n_0 to n_k. Then, by induction on i, using the constraints $a\$\hat{a} \subseteq \$$ and the fact that n_k has a loop with label \$, we can prove that:

$$\text{There is an edge with label \$ from } s_{l-i} \text{ to } m_{l-i}.$$

In particular, there is an edge with label \$ from s_0 to m_0. Since $s_0 = m_0 = n_0$, this implies that n_0 has a loop with label \$.

In order to obtain a reduction from the rewrite problem, we show that:

$$w_1 \rightarrow^* w_2 \quad \text{iff} \quad (n_0, n_0) \in \supseteq\text{-Cons}(G, L_0, \Gamma). \tag{11}$$

This is what we do next.

Implication from left to right. Suppose that $w_1 \to^* w_2$. We prove that $(n_0, n_0) \in \supseteq\text{-Cons}(G, L_0, \Gamma)$. Let H be an arbitrary $\supseteq$-repair of G with respect to Γ. The proof that n_0 has a loop with label \$ in H is in two steps:

(A) We first show that there is a path with label w_2 from n_0 to n_k. This is proved by induction on the length of the derivation from w_1 to w_2 using the rules in $\mathcal{R}$.

(B) We prove that the existence of that path implies the existence of a loop at n_0 with label \$.

We start by proving (A). Since $w_1 \to^* w_2$, there is a sequence $u_1, \ldots, u_p$ such that $u_1 = w_1$, $u_p = w_2$ and $u_i \to u_{i+1}$, for all $i < p$. We can prove by induction on i that:

$$\text{For all } 1 \leq i \leq p, \text{ there is a path with label } u_i \text{ from } n_0 \text{ to } n_k \text{ in } H. \quad (12)$$

If $i = 1$, it follows by definition of G that there is a path with label w_1 from n_0 to n_k. Since $u_1 = w_1$ and $G \subseteq H$, this suffices to prove the base case. For the induction step, suppose there is a path with label u_i, for $1 \leq i < p$, from n_0 to n_k in H. Since $u_i \to u_{i+1}$, there is a rule $(x, y) \in \mathcal{R}$ such that for some v and v' we have $u_i = vxv'$ and $u_{i+1} = vyv'$. Since the constraint $x \subseteq y$ holds in H, the existence of a path with label vxv' from n_0 to n_k implies the existence of one with label vyv' from n_0 to n_k. This finishes the proof of (12). From (12) and the fact that $u_p = w_2$, it follows that there is a path with label w_2 from n_0 to n_k in H.

Next we prove (B). Using the fact that in G there is path with label w_2 from n_0 to n_k and using the constraints $a\$\hat{a} \subseteq \$$, we are going to prove that this implies the existence of a loop with label \$ at the node n_0.

Let $s_0 w_{21} s_1 w_{22} s_2 \ldots w_{2l} s_l$ be the path from n_0 to n_k with label w_2. We prove by induction on i that:

$$\text{For all } 0 \leq i \leq l, \text{ there is an edge with label \$ from } s_{l-i} \text{ to } m_{l-i}. \quad (13)$$

For the base case, recall that $s_l = n_k$ and $m_l = n_k$. As in G the node n_k has a loop with label \$ and $G \subseteq H$, this proves the base case.

For the induction step, suppose that there is an edge with label \$ from s_{l-i} to m_{l-i}. By definition of the nodes $s_0, \ldots, s_l$, there is an edge with label $w_{2(l-i)}$ from s_{l-i-1} to s_{l-i}. Moreover, by definition of G, there is an edge with label $\hat{w}_{2(l-i)}$ from m_{l-i} to m_{l-i-1}. Putting everything together, we

obtain that there is a path with label $w_{2(l-i)}\$\hat{w}_{2(l-i)}$ from s_{l-i-1} to m_{l-i-1}. Since the constraint $w_{2(l-i)}\$\hat{w}_{2(l-i)} \subseteq \$$ holds in H, there is an edge with label \$ from s_{l-i-1} to m_{l-i-1}. This finishes the proof of (13).

Now, it follows from (13) that there is an edge with label \$ from s_0 to m_0. Recall that $s_0 = n_0$ and $m_0 = n_0$. It follows that n_0 has a loop with label \$. That is, $(n_0, n_0) \in L_0(H)$. We conclude that $(n_0, n_0) \in \supseteq\text{-Cons}(G, L_0, \Gamma)$.

Implication from right to left. Suppose that for all $\supseteq$-repairs H of G, the node n_0 has a loop with label \$. We have to prove that $w_1 \rightarrow^* w_2$. The strategy is as follows.

(A) We construct a graph database H_0 such that:

  (i) $G \subseteq H_0$,

  (ii) $H_0 \models \Gamma$, and

  (iii) If there is a path with label w_2 from n_0 to n_k in H_0, then $w_1 \rightarrow^* w_2$.

(B) Since $H_0 \models \Gamma$, there is a $\supseteq$-repair H_0' of G such that $G \subseteq H_0' \subseteq H_0$. As the consistent answer of L_0 contains the pair (n_0, n_0), this implies that n_0 has a loop with label \$ in H_0'. In particular, n_0 has a loop with label \$ in H_0.

(C) We prove that if n_0 has a loop with label \$ in H_0, then there is a path with label w_2 from n_0 to n_k in H_0. Together with (A)(iii), this finishes the proof that $w_1 \rightarrow^* w_2$.

The construction of the graph database H_0 is similar to the graph constructed in the undecidability proof of [28]. In fact, H_0 is an extension of such graph database and property (A)(iii) will immediately follow from Lemma 2 in the proof of Theorem 2 in [28].

We start by defining H_0. Let k_0 be a natural number such that both the sizes of w_1 and w_2 are bounded by k_0. We define the set of nodes V_0 of H_0 as follows:

$$\{[u] : u \in \Delta^*, |u| \leq k_0\} \cup \{m_i : 0 < i < l\}.$$

We identify n_0 with $[\varepsilon]$. For all $1 \leq i \leq k$, we identify n_i with $[w_{11} \dots w_{1i}]$. In particular, n_k is $[w_1]$. Note that this implies that the domain of H_0 is a

superset of the domain of G. Relations in H_0 are defined as follows:

$$\begin{aligned} a^{H_0} &= \{([u],[v]) \in V_0 \times V_0 : v \in Anc(ua)\}, \\ \hat{a}^{H_0} &= \{(m_i, m_{(i-1)}) : w_{2i} = a, 1 \leq i \leq l\}, \\ \$^{H_0} &= \bigcup\{X_i : 0 \leq i \leq l\}, \end{aligned}$$

where X_i is given by:

$$\{([u], m_{(l-i)}) : \text{ there is a path with label } w_{2(l-i+1)} \dots w_{2l} \text{ from } [u] \text{ to } n_k\}.$$

Recall that $Anc(ua)$ is the set of ancestors of ua. If $i = 0$, we define $w_{2(l-i+1)} \dots w_{2l}$ as the empty word ε by convention. It is worth mentioning that if we consider the restriction of H_0 to the relations $\{a : a \in \Delta\}$, the graph H_0 is nothing but the graph $DB_{\mathcal{C}}$ of the proof of Theorem 2 in [28]. Notice that H_0 cannot be explicitly constructed since the predicate $Anc(ua)$ is a non-computable one. This does not affect the proof, however, as in this direction we only need to show the existence of such an H_0.

We prove that (A)(i), (A)(ii) and (A)(iii) hold. First, we prove that (A)(i) is true. We already observed that the domain of G is a subset of the domain of H_0. It is clear that $\hat{a}^G = \hat{a}^{H_0}$, for all $a \in \Delta$. For the relation \$, we have to prove that (n_k, n_k) belongs $\$^{H_0}$. It is enough to show that (n_k, n_k) belongs to X_0. The set X_0 is given by:

$$\{([u], m_l) : \text{ there is a path with label } \varepsilon \text{ from } [u] \text{ to } n_k\}.$$

Since $m_l = n_k$, there is an emtpy path from n_k to n_k, and $[w_1] = n_k$, the pair (n_k, n_k) belongs to X_0. Next we prove that $a^G \subseteq a^{H_0}$, for all $a \in \Delta$. These relations are interpreted in G in such a way that there is a path with label w_1 from n_0 to n_k, namely the path:

$$n_0 w_{11} n_1 w_{12} \dots w_{1k} n_k.$$

We have to show that this path also appears in H_0. That is, for all $1 \leq i \leq k$:

$$\text{There is an edge with label } w_{1i} \text{ from } n_{i-1} \text{ to } n_i \text{ in } H_0. \tag{14}$$

Recall that:

$$n_{i-1} = [u], \tag{15}$$

$$n_i = [uw_{1i}], \tag{16}$$

where $u = w_{11} \dots w_{1i-1}$. By definition of H_0, we have that:

$$([u], [uw_{1i}]) \in w_{1i}^{H_0},$$

since $uw_{1i} \to^* uw_{1i}$. Together with (15) and (16), we obtain that (n_{i-1}, n_i) belongs to $w_{1i}^{H_0}$, which finishes the proof of (14).

Next we prove that (A)(iii) follows from Lemma 2 in the proof of Theorem 2 in [28]. So suppose that there is a path from n_0 to n_k with label w_2. In Lemma 2 in [28], it is shown that in the graph H_0, for all words $u \in \Delta^*$:

$$\mathit{Reach}([\varepsilon], u) = \mathit{Anc}^{[]}([u]),$$

where $\mathit{Anc}^{[]}([u])$ is defined as the set:

$$\{[v] : v \in \mathit{Anc}(u), |v| \leq k_0\}.$$

and $\mathit{Reach}(n, u)$ is defined as the set

$$\{n' : \text{ there is a path in } H_0 \text{ with label } u \text{ from } n \text{ to } n'\}.$$

In particular:

$$\mathit{Reach}([\varepsilon], w_2) = \mathit{Anc}^{[]}([w_2]). \tag{17}$$

Since there is a path from n_0 to n_k with label w_2 and $n_0 = [\varepsilon]$, the node n_k belongs to $\mathit{Reach}([\varepsilon], w_2)$. By (17), it follows that n_k belongs to $\mathit{Anc}^{[]}([w_2])$. Recall that $n_k = [w_1]$. Hence, $[w_1]$ belongs to $\mathit{Anc}^{[]}([w_2])$. That is, $w_1 \to^* w_2$.

Now we prove (A)(ii). It follows immediately from the proof of Theorem (2) in [28] that each constraint of the form:

$$u \subseteq v,$$

where $(u, v) \in \mathcal{R}$, holds in H_0. Hence, it remains to show that each constraint of the form:

$$a\$\hat{a} \subseteq \$,$$

where $a \in \Delta$, holds in H_0. Let a be a letter in Δ. Suppose that there is a path with label $a\$\hat{a}$. We assume that the path is of the form

$$o_1 a o_2 \$ o_3 \hat{a} o_4.$$

We have to prove that there is an edge with label \$ from o_1 to o_4 in H_0. By definition of H_0, the only edges with labels $\hat{a}$ are edges between the m_i's.

Hence, for some i, the node o_4 is equal to m_{l-i-1} and o_3 is equal to m_{l-i}. Moreover, a must be equal to $w_{2(l-i)}$, as the only edge between m_{l-i} and m_{l-i-1} is the edge with label $\hat{w}_{2(l-i)}$.

By definition of \$ in H_0, if $(o_2, o_3) \in \$^{H_0}$ and o_3 is equal to m_{l-i}, then (o_2, o_3) belongs to X_i. Recall that X_i is given by:

$$\{([u], m_{(l-i)}) : \text{ there is a path with label } w_{2(l-i+1)} \dots w_{2l} \text{ from } [u] \text{ to } n_k\}.$$

Hence, o_2 is node of the form $[u]$ and there is a path with label $w_{2(l-i+1)} \dots w_{2l}$ from $[u]$ to n_k. Since $o_1 a o_2$ and $a = w_{2(l-i)}$, this implies that

$$\text{There is a path with label } w_{2(l-i)} \dots w_{2l} \text{ from } o_1 \text{ to } n_k.$$

By definition of X_{i-1}, this means that (o_1, m_{l-i-1}) belongs to X_{i-1}. It follows from the definition of \$ in H_0 that (o_1, m_{l-i-1}) belongs to $\$^{H_0}$. Since $o_4 = m_{l-i-1}$, this means that (o_1, o_4) belongs to $\$^{H_0}$, finishing the proof of (A)(ii).

Since (B) holds by definition, it only remains to prove (C). Assume that n_0 has a loop with label \$ in H_0. We have to prove that there is a path with label w_2 from n_0 to n_k. Recall that:

$$\$^{H_0} = \bigcup \{X_i : 0 \leq i \leq l\},$$

where X_i is given by:

$$\{([u], m_{(l-i)}) : \text{ there is a path with label } w_{2(l-i+1)} \dots w_{2l} \text{ from } [u] \text{ to } n_k\}.$$

Since (n_0, n_0) belongs to $\$^{H_0}$, the pair (n_0, n_0) must belong to X_{i_0} for some i_0. Since $n_0 = m_0$ (and all the m_i's are pairwise distinct), the pair (n_0, n_0) belongs to the set X_l. By definition of X_l, if $(n_0, n_0) \in X_l$, there is a path with label $w_{21} \dots w_{2l}$ from n_0 to n_k. That is, there is a path with label w_2 from n_0 to n_k. This finishes the proof of Theorem 4 when $\star = \supseteq$. □

Proof of Theorem 4 in the case when $\star$ is $\oplus$: The proof is similar to the proof in the case when $\star = \supseteq$. As in that proof, we show undecidability by reducing from the rewrite problem. Let $\mathcal{R}$ be a fixed rewrite system over Δ such that its rewrite problem is undecidable. We define a set Γ_1 of RPCs and a non-recursive RPQ L_1 such that for all words w_1 and w_2 over Δ, there is a graph database G_1 with a node n_0 satisfying:

$$w_1 \to^* w_2 \quad \text{iff} \quad (n_0, n_0) \in \oplus\text{-Cons}(G_1, L_1, \Gamma_1).$$

The construction. Let w_1 and w_2 be two words in Δ^*. As before:

$$\begin{aligned} w_1 &= w_{11}w_{12}\dots w_{1k}, \\ w_2 &= w_{21}w_{22}\dots w_{2l}, \end{aligned}$$

where $w_{1i}, w_{2j} \in \Delta$ for each $1 \leq i \leq k$ and $1 \leq j \leq l$. Thus, k is the length of w_1 and l is the length of w_2. We start by defining the graph database G_1. We let V, the set of nodes of G_1, to be defined as follows:

$$\{n_i : 0 \leq i \leq k\} \cup \{m_i : 0 < i < l\}.$$

We define m_0 as n_0 and m_l as n_k. Note that the domain of G_1 is equal to the domain of the graph G in the proof of Theorem 4 for the case when $\star = \supseteq$.

The edge relations over G_1 are defined as follows, where a ranges over Δ:

1. $a^{G_1} = \emptyset$.
2. $(a')^{G_1} = \{(n_{i-1}, n_i) : w_{1i} = a, 1 \leq i \leq k\}$.
3. $\hat{a}^{G_1} = \{(m_i, m_{(i-1)}) : w_{2i} = a, 1 \leq i \leq l\}$.
4. $(r')^{G_1} = \{(n_i, n_{i-1}) : 1 \leq i \leq k\}$.
5. $(\hat{r})^{G_1} = \{(m_{i-1}, m_i) : 1 \leq i \leq l\}$.
6. $\$^{G_1} = \{(n_k, n_k)\}$.
7. $(\$')^{G_1} = \{(n_k, n_k)\}$.
8. $e^{G_1} = \emptyset$.
9. $s^{G_1} = V \times V$.

Thus, G_1 contains a path with label $w'_{11}\dots w'_{1k}$ from n_0 to n_k and each edge on that path has an "inverse" edge with label r'. It also contains a path with label $\hat{w}_{2l}\dots\hat{w}_{21}$ from n_k to n_0 and each edge on that path has an "inverse" edge with label $\hat{r}$. The node n_k has loops with label \$ and \$′. The relation e is empty and s is the full relation.

The set of RPCs Γ_1 consists of the following constraints:

$$\begin{aligned} u \subseteq v, \qquad & \qquad a\$\hat{a} \subseteq \$, \\ a'r' \subseteq e, \qquad & \qquad \hat{a}\hat{r} \subseteq e, \\ \$\$' \subseteq e, \qquad & \qquad a' \subseteq a, \end{aligned}$$

for each $a \in \Delta$ and (u, v) in $\mathcal{R}$. The RPQ L_1 is given by:

$$s\hat{r}s + sr's + s\$'s + \$.$$

The intuition. The basic idea is as follows. Suppose that $w_1 \to^* w_2$. We start with the path with label $w'_{11} \dots w'_{1k}$ in G_1. Now we apply the constraints of the form $a' \subseteq a$. Since we can add or remove edges, either we remove (at least) one edge with label a' or we obtain a path with label w_1 (by only adding edges). In the first case, using the constraint $a'r' \subseteq e$ and the minimality property of the repairs, we can show that there must be an edge with label r'. This guarantees that (n_0, n_0) belongs to the answer of L_1. In the second case, we have a path with label w_1. Next we apply the constraints $u \subseteq v$ (where $(u, v) \in \mathcal{R}$) and we use the fact that $w_1 \to^* w_2$. By induction on the length of the derivation of w_2 from w_1, we can show that there is a path with label w_2 from n_0 to n_k.

Then there are three possibilities. Either: (I) we have deleted the loop with label \$ at the node n_k, or (II) we have deleted an edge with label $\hat{a}$, or (III) we did not delete neither the loop with label \$ nor any edge with label $\hat{a}$. In case (I), using the constraint $\$\$' \subseteq e$ and the minimality property of the repairs, we can show that there must be an edge with label \$'. This implies that (n_0, n_0) belongs to the answer of L_1. Case (II) is similar. Using the constraint $\hat{a}\hat{r} \subseteq e$ and the minimality property of the repairs, we can show that there must be an edge with label $\hat{r}$. This means that (n_0, n_0) belongs to the answer of L_1. Case (III) is similar to what happens in the proof of Theorem 4 for the case when $\star = \supseteq$. We show that the presence of a path with label w_2 implies the existence of a loop with label \$ at n_0. In particular, (n_0, n_0) belongs to the answer of L_1.

We now give a formal proof of the reduction. We show that:

$$w_1 \to^* w_2 \quad \text{iff} \quad (n_0, n_0) \in \oplus\text{-Cons}(G_1, L_1, \Gamma_1).$$

Implication from left to right. Assume that $w_1 \to^* w_2$. Let H be an $\oplus$-repair of G_1 with respect to Γ_1. We have to prove that (n_0, n_0) belongs to the answer of L_1 in H. We make the following case distinction. Either:

(i) It is the case that n_k has a loop with label \$, that $(a')^H = (a')^{G_1}$, and that $\hat{a}^H = \hat{a}^{G_1}$, for all $a \in \Delta$, or

(ii) for some $a_0 \in \Delta$ we have $(a'_0)^H \neq (a'_0)^{G_1}$, or

(iii) for some $a_0 \in \Delta$ we have $\hat{a}_0^H \neq \hat{a}_0^{G_1}$, or

(iv) n_k does not have a loop with label \$ in H.

We first look at case (i). It follows from the constraint $a' \subseteq a$ that for all $1 \leq i \leq k$, the pair (n_{i-1}, n_i) belongs to a^H if $w_{1i} = a$. That is, there is a path with label w_1 from n_0 to n_k. Then, using exactly the same proof as the proof of the implication from left to right of equivalence (11) in the proof of Theorem 4 for the case when $\star = \supseteq$, we can show that (n_0, n_0) belongs to the answer of \$ in H. Hence, (n_0, n_0) belongs to the answer of L_1 in H.

We only provide a sketch of this fact. As in the proof of Theorem 4 for the case when $\star = \supseteq$, we prove by induction on the length of the derivation that for all words w such that $w_1 \rightarrow^* w$, there is a path with label w from n_0 to n_k. The base case is true since there is a path with label w_1 from n_0 to n_k. For the induction step, we use the constraints $u \subseteq v$ where $(u, v) \in \mathcal{R}$. Since $w_1 \rightarrow^* w_2$, there is a path with label w_2 from n_0 to n_k. Let $s_0 w_{21} s_1 \dots w_{2l} s_k$ be such path. Using the fact that n_k has a loop with label \$ and the constraint $a\$\hat{a} \subseteq \$$ holds in H, we can show by induction that for all i, the pair (s_{l-i}, m_{l-i}) has an edge with label \$ in H. In particular, there is an edge with label \$ from s_0 to m_0. That is, from n_0 to n_0. This shows that (n_0, n_0) belongs to the answer of \$, and thus of L_1, in H.

Next we treat case (ii). The idea is that in this case, using the constraint $a'r' \subseteq e$ together with the minimality property of repairs, there must be an edge with label r'. This implies that (n_0, n_0) belongs to the answer of the query $sr's$; in particular, to the answer of L_1.

Formally, suppose that $(a'_0)^H \neq (a'_0)^{G_1}$ for some a_0. Since relations of the form a', for $a \in \Delta$, only appear in the left-hand side of the constraints, it follows from Lemma 2 that $(a')^H \subseteq (a')^{G_1}$ for every such a'. Together with the fact that $(a'_0)^H \neq (a'_0)^{G_1}$, it follows that for some i with $1 \leq i \leq k$ the pair (n_{i-1}, n_i) does not belong to any a' in H.

Now we use this fact together with the constraint $a'r' \subseteq e$. Let H' be the graph database obtained from H by adding an edge with label r' from n_i to n_{i-1}. Using the fact that (n_{i-1}, n_i) does not belongs to any a' in H, the constraint $a'r' \subseteq e$ remains true in H'. All other constraints continue to hold in H' since r' does not appear in any of them. Hence, $H' \models \Gamma_1$. Moreover, $G_1 \oplus H' \subseteq G_1 \oplus H$ since there is an edge with label r' from n_i to n_{i-1} in G. As H is an $\oplus$-repair, this implies that $H = H'$. In particular, H has an edge with label r'.

It follows from Lemma 2 that $s^H = s^{G_1}$ since s does not occur in any constraint. That is, $s^H = V \times V$. Together with the fact that H has an edge with label r', this implies that (n_0, n_0) belongs to the answer of $sr's$ in H and in particular to the answer of L_1 in H. Case (iii) is similar to case (ii).

Finally, we treat (iv). Suppose that n_k does not have a loop with label \$ in H. Let H'' be the database obtained by adding an edge with label \$′ from n_k to n_k and by removing all the edges with label \$. Since n_k does not have a loop with label \$ in H, we have $G_1 \oplus H'' \subseteq G_1 \oplus H$. Moreover, $H'' \models \Gamma_1$. In fact, the constraint $\$\$' \subseteq e$ holds in H'' as $\$^{H''}$ is empty. All other constraints also hold in H'' since neither \$′ nor \$ is mentioned in them. We then obtain $H = H''$. In particular, H has an edge with label \$′. This implies that (n_0, n_0) belongs to the answer of $s\$'s$ in H, and, in particular, to the answer of L_1 in H.

Implication from right to left. Suppose that (n_0, n_0) belongs to the answer of L_1 in all $\supseteq$-repairs of G_1. We prove that $w_1 \rightarrow^* w_2$. To do this, we construct a graph database H_1 satisfying the following:

(A) Γ_1 holds in H_1. Moreover, if there is a path with label w_2 from n_0 to n_k in H_1, then $w_1 \rightarrow^* w_2$.

(B) Hence there is a repair H_1' such that $G_1 \oplus H_1' \subseteq G_1 \oplus H_1$. As the consistent answer of L_1 contains (n_0, n_0), this implies that (n_0, n_0) belongs to the answer of L_1 in H_1'. We prove that this implies that n_0 has a loop with label \$ in H_1.

(C) We prove that if n_0 has a loop with label \$ in H_1, then there is a path with label w_2 from n_0 to n_k in H_1. Together with (a), this implies that $w_1 \rightarrow^* w_2$.

The graph database H_1 is a slight modification of the graph database H_0 defined in the proof of Theorem 4 for the case when $\star = \supseteq$ (in the proof of the implication from right to left of equivalence (11)). Let H_0 be the graph database defined in such proof. We define H_1 as the graph database obtained by extending H_0 in the following way:

We add an edge with label w'_{1i} from n_{i-1} to n_i for each $1 \leq i \leq k$.

So H_1 is obtained by adding a path with label $w'_{11} \dots w'_{1k}$ to the database H_0. Notice, in particular, that there is no edge with label r', $\hat{r}$ or \$′ in H_1.

We start by proving (A). Let us consider first the fact that if there is a path with label w_2 from n_0 to n_k in H_1, then $w_1 \rightarrow^* w_2$. This is proved exactly as in the case $\star = \supseteq$, when we proved that if there is a path with

label w_2 from n_0 to n_k in H_0, then $w_1 \rightarrow^* w_2$. The fact that H_1 extends H_0 with a path with label $w'_{11} \dots w'_{1k}$ does not change anything.

To finish the proof of (A), we have to show that Γ_1 holds in H_1. The fact that the RPCs:

$$u \subseteq v \quad \text{and} \quad a\$\hat{a} \subseteq \$,$$

hold in H_1, where $a \in \Delta$ and $(u, v) \in \mathcal{R}$, is shown in the same way that we prove that such constraints hold in the graph database H_0 (in the proof for the case $\star = \supseteq$). The constraints of the form:

$$a'r' \subseteq r, \ \hat{a}\hat{r} \subseteq r, \ \text{and} \ \$\$' \subseteq r,$$

hold because r', $\hat{r}$ and $\$'$ are empty in H_1. It remains to look at the constraints of the form:

$$a' \subseteq a.$$

By definition of H_1, if there is an edge with label a', there is $1 \leq i \leq n$ such that $a' = w'_{1i}$ and the edge is from node n_{i-1} to n_i. In order to check that the constraint is verified, we have to show that there is an edge with label w_{1i} from n_{i-1} to n_i. In the proof of Theorem 4, case $\star = \supseteq$, we showed that G (as defined in such proof) is contained in H_0. Since in G there is an edge with label w_{1i} from n_{i-1} to n_i, such edge also exists in H_0, and, therefore, in H_1. This finishes the proof that Γ_1 holds in H_1.

Now we prove (B). We know that $(n_0, n_0) \in L_1(H'_1)$. We have to show that n_0 has a loop with label \$. To do so, we show that:

$$\text{Relations } r', \ \hat{r} \text{ and } \$' \text{ are empty in } H'_1. \tag{18}$$

This implies that n_0 has a loop with label \$. Indeed, suppose that (18) holds. Recall that (n_0, n_0) belongs to $L_1(H'_1)$, where L_1 is given by $(s\hat{r}s + sr's + s\$'s + \$)$. By (18), we have that the answer of $L'_1 := (s\hat{r}s + sr's + s\$'s)$ over H'_1 is empty. Hence, (n_0, n_0) must belong to the evaluation of \$ over H'_1, i.e., n_0 has a loop labeled \$ in H'_1. Since $G \oplus H'_1 \subseteq G \oplus H_1$ and n_0 does not have a loop labeled \$ in G_1, we conclude that n_0 has a loop labeled \$ in H_1.

Hence, we are left with the proof of (18). We only prove that there is no edge with label r'. The other proofs are similar. Suppose for the sake of contradiction that there is an edge e with label r' in H'_1. Since $G_1 \oplus H'_1 \subseteq G_1 \oplus H_1$ and there is no edge with label r' in H_1, this edge must appear in G_1. By definition of G_1, the edge e must then go from n_i to n_{i-1} for some $1 \leq i \leq k$.

Recall now that there is an edge with label w'_{1i} from n_{i-1} to n_i both in H_1 and in G_1. Since $G_1 \oplus H'_1 \subseteq G_1 \oplus H_1$, there must also be an edge with label w'_{1i} from n_{i-1} to n_i in H'_1. We can then conclude that there is a path with label $w'_{1i}r'$ from n_{i-1} to n_{i-1}. Since the constraint $w'_{1i}r' \subseteq e$ holds in H'_1, there is a loop labeled e on n_{i-1}. In particular, e is not empty in H'_1. This contradicts the facts that e is empty both in G_1 and in H_1 and that $G_1 \oplus H'_1 \subseteq G_1 \oplus H_1$.

The proof that (C) holds mimics the proof of the fact that if n_0 has a loop with label \$ in H_0, then there is a path with label w_2 from n_0 to n_k in H_0 (from the case $\star = \supseteq$). □

4.1. Decidable restrictions

Since the CQA problem in this context is undecidable, it is crucial to look for decidable (and, ideally, tractable) restrictions of it. We provide three such restrictions in this section: The first one is based on the class of LAV C2RPCs, while the second one is based on the class of GAV C2RPCs. The third one is obtained by modifying C2RPC interpretation to be *from the origin*. It is worth noticing that the restriction to classes of graph databases of bounded treewidth, which leads to tractability under the semantics of subset repairs, is not useful in this context: The undecidability result in Theorem 4 holds even over graph databases of treewidth two.

Restriction to the class of LAV C2RPCs. As mentioned before, in the relational scenario the CQA problem for unions of CQs under LAV tgds is tractable, no matter which repair semantics is used. We already stated a similar result for CQA over graph databases under LAV C2RPCs and the subset repair semantics (Theorem 2). We can further extend those techniques to obtain tractability for our CQA problem under the semantics of $\oplus$-repairs.

Theorem 5. *For each C2RPQ q and finite set Γ of LAV C2RPCs over the same alphabet Σ, it is the case that* $\oplus$-CQA(q,Γ) *is in* NLogspace.

Proof. The proof is an immediate adaptation of techniques in [20]. We still provide it for the sake of completeness. Basically the result holds because LAV C2RPCs, i.e., those of the form $(x, a, y) \subseteq q'(x, y)$, are closed under union (that is, if two graph databases satisfy the constraints, then so does their union).

Let G be a graph database. It follows from the proof of Theorem 2 that there is a unique $\subseteq$-repair H_0 of G with respect to Γ, which can be computed

in NLogspace. We prove that for each C2RPQ q and tuple $\bar{a}$ of nodes in G it is the case that:

$$\bar{a} \in \oplus\text{-Cons}(G, q, \Gamma) \quad \text{iff} \quad \bar{a} \in q(H_0). \tag{19}$$

Since H_0 can be computed in NLogspace, and checking whether $\bar{a}$ belongs to the evaluation of q over H_0 is in NLogspace from Proposition 1, this suffices to prove that $\oplus$-CQA(L, Γ) is in NLogspace (as NLogspace computable functions are closed under composition).

Now we prove (19). Since H_0 is a $\subseteq$-repair, and these are also $\oplus$-repairs, the implication from left to right follows from the definition of consistent answer. For the implication from left to right, suppose that a tuple $\bar{a}$ belongs to $q(H_0)$. Let H be a $\oplus$-repair of G with respect to Γ. We show that $\bar{a} \in q(H)$.

Consider the graph database $H_0 \cup H$. Note that this a union, not a disjoint union. Since $H_0 \subseteq G$, we have that:

$$G \oplus (H_0 \cup H) \subseteq G \oplus H. \tag{20}$$

Moreover, since the C2PRCs in Γ hold both in H_0 and H and they are of the form $(x, a, y) \subseteq q'(x, y)$, for $a \in \Sigma^{\pm}$ and q' a C2RPQ, they continue to hold in $H_0 \cup H$, that is:

$$H_0 \cup H \models \Gamma.$$

Together with (20) and the fact that H is an $\oplus$-repair of G, we obtain that $H_0 \cup H = H$. That is, $H_0 \subseteq H$, and thus $q(H_0) \subseteq q(H)$ since C2RPQs are monotone. In particular, since $\bar{a}$ belongs to $q(H_0)$, it is also in $q(H)$. □

The case of the $\supseteq$-repair semantics is different: We do not know whether LAV C2RPCs (not even LAV RPCs) yield decidability in this context, but we prove next that at least they do not yield tractability in data complexity. This establishes a first difference in complexity between CQA under LAV tgds in the relational context and under LAV C2RPCs over graph databases.

Proposition 6. *There exist a finite alphabet Σ, a non-recursive RPQ L, and a finite set Γ of LAV RPCs (without Kleene-star) over Σ, such that $\supseteq$-CQA(L,Γ) is* coNP*-hard.*

Proof. We provide a reduction from Monotone 1-in-3 Sat to the problem $\supseteq$-CQA(L, γ), where γ is the constraint $a \subseteq t + f$ and L will be defined later.

Let ϕ be a boolean formula in conjunctive normal form $C_1 \wedge \cdots \wedge C_m$ where each clause C_i is of the form:

$$x_{i1} \vee x_{i2} \vee x_{i3}.$$

We associate with ϕ a graph database G_ϕ and a node u_0 in such a way that:

$$\phi \in \textsc{Monotone 1-in-3 Sat} \quad \text{iff} \quad (u_0, u_0) \notin \supseteq\text{-Cons}(G_\phi, L, \Gamma). \tag{21}$$

The graph G_ϕ is defined in the following way. Its set V of nodes is:

$$\{n_{ij} : 1 \leq i \leq m, 1 \leq j \leq 3\}.$$

That is, with each occurrence of a variable x_{ij} in ϕ we associate a node n_{ij}. Note that even if $x_{ij} = x_{kl}$, then the nodes associated with x_{ij} and x_{kl} are distinct. We define the node u_0 as the node n_{11}. The relations of G_ϕ are defined as follows (the schema will be self-evident from the definition):

1. $a^{G_\phi} = \{(n_{ij}, n_{ij}) : 1 \leq i \leq m, 1 \leq j \leq 3\}$
2. $d^{G_\phi} = \{(n_{ij}, n_{kl}) : x_{ij} = x_{kl}\}$.
3. $r^{G_\phi} = \{(n_{i1}, n_{i2}), (n_{i2}, n_{i3}), (n_{i3}, n_{i1}) : 1 \leq i \leq m\}$.
4. $s^{G_\phi} = V \times V$.
5. $t^{G_\phi} = f^{G_\phi} = \emptyset$.

Intuitively, the relation d specifies which nodes correspond to the same variable. The relation r specifies which variables occur in the same clause. The relation a is a loop on each node. The relations t and f are empty. The relation s is the full relation.

As mentioned before, γ is the RPC:

$$a \subseteq t + f.$$

The intuition is as follows. Since each node has a loop with label a, it will also have such a loop in any $\supseteq$-repair H with respect to γ. The constraint enforces that it also has either a loop with label f or with label t. This allows us to define a map V_H associated with H in the following way: If n_{ij} has a loop with label t, we map n_{ij} to $\top$, otherwise, n_{ij} is mapped to $\bot$.

Note that V_H might not define a valuation on the x_{ij}'s as there might variables x_{ij} and x_{kl} such that $x_{ij} = x_{kl}$, the node n_{ij} has a loop with label t, and n_{kl} has a loop with label f. By definition of d, this is equivalent to say

that there is a path with label tdf between the nodes n_{ij} and n_{kl}. In fact, the map V_H is a valuation if and only if there is no path with label tdf in the repair H. So, informally, each $\supseteq$-repair with no such path corresponds to a valuation.

Now, we want to define the RPQ L in such a way that if (u_0, u_0) belongs to $L(H)$, where H is a $\supseteq$-repair H of G_ϕ with respect to γ, and H has associated valuation V_H, then V_H is not a "witness" for the membership of ϕ in MONOTONE 1-IN-3 SAT (as before, we say that a valuation is a witness iff exactly one variable for each clause of ϕ is true under the valuation).

Notice that a valuation V_H associated with a repair H is not a witness for the membership of ϕ in MONOTONE 1-IN-3 SAT if for there is an $1 \leq i \leq m$ such that:

(†) V_H takes value $\top$ in at least two elements from $\{x_{i1}, x_{i2}, x_{i3}\}$, or

(††) $V_H(x_{i1}) = V_H(x_{i2}) = V_H(x_{i3}) = \bot$.

By definition of s and V_H, (†) implies that trt is the label of a path in H, while (††) implies that $frfrf$ is the label of a path in H. We let W be the regular expression:

$$trt + frfrf.$$

So if H is a repair, either it is not associated with a valuation (that is, there is a path with label tdf) or it is associated with a valuation and, in that case, it must admit a path with label W (in order to ensure that the valuation is not a witness). This suggests us to define L as the RPQ:

$$stdfs + sWs.$$

We prove now that equivalence (21) holds. The proof is similar to the proof of Proposition 4. First we show the implication from left to right of equivalence (21). Suppose that ϕ belongs to MONOTONE 1-IN-3 SAT. Let V_0 be a valuation such that for all clauses of ϕ, exactly one variable receives value $\top$ under V_0. We need to find a repair H_0 such that (u_0, u_0) does not belong to $L(H_0)$. We let H_0 be the graph database obtained from G_ϕ by adding loops with labels t and f in the following way:

- If $V_0(x_{ij}) = \top$, we add a loop with label t at the node n_{ij}.
- If $V_0(x_{ij}) = \bot$, we add a loop with label f at the node n_{ij}.

Since each node has either a loop with label t or f, the constraint γ is true. Moreover, as each node has exactly one loop with either label t or f, the graph H_0 is a $\supseteq$-repair of G with respect to γ (as the minimality condition of $\supseteq$-repairs is satisfied).

It remains to show that (u_0, u_0) does not belong to $L(H_0)$. The proof is almost identical to the proof that (u_0, u_0) does not belong to $L_0(H_0)$ in the proof of Proposition 4. So we only provide a sketch here. We have to show that there is neither a path with label tdf nor a path with label W. First, there is a path with label tdf in H if and only if there are two nodes that are linked with an edge with label d and such that the first node has a loop with label t and the other node has a loop with label f. The fact that two nodes are linked with an edge with label d means that they correspond to the same variable. Hence, by definition of H, they must have the same loops. So it will never be the case that one node have a loop with label t while the other one does not. We conclude that there is no path with label tdf in H.

The fact that there is no path with label W comes from the fact that V_0 is a witness valuation. Recall that W is the RPQ $trt + frfrf$. If there is a path with label trt, there are two nodes with loops labeled t that are linked by an edge with label r. By definition of r, this means that the two nodes correspond to variables appearing in the same clause. Since the two nodes have a loop with label t, the variables corresponding to the two nodes receive value $\top$ under the valuation V_0. This contradicts the fact that V_0 is a witness valuation. Similarly we can show that there is no path with label $frfrf$.

Next we prove the implication from right to left of (21). Suppose that (u_0, u_0) does not belong to the answer of L in a $\supseteq$-repair H of G_ϕ with respect to γ. Since each node has a loop with label a in G and H is a $\supseteq$-repair, each node also has a loop with label a in H. Moreover, the constraint $a \subseteq t + f$ holds in H. Since each node has a loop with label a in H, this implies that each node has at least one loop with label t or f. Moreover, using the minimality condition of repairs, we can show that each node has at most one loop with label t or f.

So each node in H has exactly one loop with label t or f. The proof that this implies that ϕ belongs to Monotone 1-in-3 Sat is almost identical to the proof of Proposition 4. So we only give a short sketch. We define a map F assigning to each node the value $\bot$ if it has a loop with label f, and the value $\top$ otherwise. Using the fact that there is no path with label tdf, we can prove that nodes corresponding to the same variables, are assigned the same value by F. Hence, we can define a valuation V_1 such that $V_1(x_{ij}) = F(n_{ij})$

for all i, j. That is,

$$V_1(x_{ij}) = \begin{cases} \top & \text{if } n_{ij} \text{ has a loop with label } t \\ \bot & \text{if } n_{ij} \text{ has a loop with label } f. \end{cases}$$

We can prove that V_1 is a valuation such that exactly one variable receives value $\top$ in each clause of ϕ. This comes from the fact that there is neither a path with label $frfrf$ nor a path with label trt (since by assumption there is no path in H with label W). □

Notice that, unlike all previous lower bounds, the one in Proposition 6 is not stated in terms of the class of word constraints. In fact, the techniques developed for studying CQA under tgds in the relational case [20] can be adapted to show that under a set Γ of LAV word constraints the problem $\supseteq$-CQA(L,Γ) is tractable.

Restriction to the class of GAV RPCs. In the case of the symmetric difference semantics, it is easy to adapt the proof of Proposition 4 in order to show that when restricting to GAV C2RPCs, our CQA problem is coNP-hard. Note that in the relational case, a similar result holds.

Proposition 7. 1. *For each C2RPQ q and finite set Γ of GAV C2RPCs over the same alphabet Σ, it is the case that $\oplus$-CQA(q,Γ) is in* coNP.
2. *There exist a finite alphabet Σ, a non-recursive RPQ L over Σ, and a single GAV RPC γ of the form $ab \subseteq c$, where $a, b, c \in \Sigma$, such that $\oplus$-CQA(L,γ) is* coNP*-complete.*

Proof. The proof is similar to the proof of Proposition 4. Membership in coNP is identical. For the hardness proof, we have to modify the original proof. The problem with the original proof under the symmetric difference semantics, is that here, it is not always the case that there is no edge with label e in a repair. So if L is the RPQ of the original proof, we modify it into an RPQ L' defined by:

$$L + ses.$$

Recall that s is the full relation. In that way, we ensure that if e is non-empty, then the answer of the RPQ contains all the pairs of nodes. Using a proof similar to the proof of Proposition 4, we can show that for any formula ϕ:

$$\phi \in \text{Monotone 1-in-3 Sat} \quad \text{iff} \quad (u_0, u_0) \notin \oplus\text{-Cons}(G_\phi, L', \gamma),$$

where γ, G_ϕ and u_0 are defined as in the proof of Proposition 4. □

In the case of the superset semantics, the restriction to GAV C2RPCs leads to tractability. Given a graph database G and a set of GAV C2RPCs Γ, using a classical chase argument, we can easily compute in LOGSPACE the *unique* superset repair of G with respect to Γ. This is identical to what happens in the setting of relational databases.

Proposition 8. *For each C2RPQ q and finite set Γ of GAV C2RPCs over the same alphabet Σ, it is the case that $\supseteq$-CQA(q,Γ) is in* NLOGSPACE.

Proof. Suppose that q is a C2RPQ and Γ is a set of GAV C2RPCs of the form $q'(x,y) \subseteq (x,a,y)$, for q' a C2RPQ and $a \in \Sigma^{\pm}$. Consider a graph database $G = (V,E)$. We show how to construct the unique superset repair in NLOGSPACE. We iteratively construct graph databases $H_0 \subseteq H_1 \subseteq \ldots$ such that $H_0 = G$, and for each $i \geq 0$ we have that H_{i+1} is constructed from H_i as follows. For each pair (u,v) of nodes and for each constraint $q'(x,y) \subseteq (x,a,y)$ such that $a \in \Sigma$ and the edge (u,a,v) is not in H_i, we check whether (u,v) belongs to $q'(H_i)$. If it does, then we add an edge with label a from u to v in H_{i+1}. Similarly, for each pair (u,v) of nodes and for each constraint $q'(x,y) \subseteq (x,a^-,y)$ such that $a \in \Sigma$ and the edge (v,a,u) is not in H_i, we check whether (u,v) belongs to $q'(G)$. If it does, then we add an edge with label a from v to u in H_{i+1}. Clearly, this process stops yielding new edges after finitely many steps. We assume that m is the smallest integer such that $H_m = H_{m+1}$. Notice that H_m can be constructed in NLOGSPACE from G. This is because each step can be constructed in NLOGSPACE and NLOGSPACE computable functions are closed under composition.

By construction, Γ holds in H_0. We prove next that if H is a $\supseteq$-repair of G under Γ, then $H_m \subseteq H$. Notice that these two facts show that H_m is the unique $\supseteq$-repair of G.

Let H be a $\supseteq$-repair of G under Γ. We prove by induction on $0 \leq i \leq m$ that $H_i \subseteq H$. By definition, $G \subseteq H$. Since $G = H_0$, the base case follows. Assume now by inductive hypothesis that $H_i \subseteq H$, for $0 \leq i < m$. Consider an arbitrary edge $(u,a,v) \in H_{i+1} \setminus H_i$. This edge has been added to H_{i+1} by the application of a constraint in Γ. Suppose that such a constraint is of the form $q'(x,y) \subseteq (x,a,y)$, for $a \in \Sigma$ (the case when the constraint is of the form $q'(x,y) \subseteq (x,a^-,y)$ is analogous). Then by construction $(u,v) \in q'(H_i)$. Since $H_i \subseteq H$ and C2RPQs are monotone, it is also the case that $(u,v) \in q'(H)$. But H is a $\supseteq$-repair of G, and thus it satisfies Γ. It follows that the edge (u,a,v) belongs to H. We conclude that $H_{i+1} \subseteq H$.

Thus, to check whether $\bar{a} \in \subseteq\text{-Cons}(G, q, \Gamma)$, it suffices to construct H_0 and check whether $\bar{a} \in q(H_0)$. Since the latter is in NLOGSPACE, it follows that the whole process can be done in NLOGSPACE. □

Modifying the interpretation of RPCs. By interpreting 2RPCs under the relation $\models_o$, we obtain decidability for the CQA problem for 2RPQs under the semantics of $\supseteq$-repairs. We do not know whether this can be extended to the semantics of $\oplus$-repairs. Notice the difference with the restriction to LAV 2RPCs we studied before: For the latter we could only obtain decidability under the $\oplus$-repairs semantics.

The difference here is that the implication problem for 2RPCs becomes decidable if 2RPCs are interpreted under the relation $\models_o$ [1]. We adapt the techniques used to prove this fact in order to obtain Theorem 6.

Recall that, G' is a $\{o, \supseteq\}$-repair of G under Γ, if (1) $G \subseteq G'$, (2) $G' \models_o \Gamma$, and (3) there is no graph database G'' such that $G \subseteq G'' \subsetneq G'$ and $G'' \models_o \Gamma$. Furthermore, if L is a 2RPQ and Γ is a finite set of 2RPCs, we define $\{o, \supseteq\}$-CQA(L,Γ) as the problem of, given a graph database $G = (V, E)$ and a pair (u, v) of nodes in V, checking whether (u, v) is an $\{o, \supseteq\}$-consistent answer of L over G under Γ.

Theorem 6.
1. *For each 2RPQ L and finite set Γ of 2RPCs over the same alphabet Σ, it is the case that $\{o, \supseteq\}$-CQA(L,Γ) is in* coNP.
2. *There is a 2RPQ L and a LAV RPC γ such that checking whether (o, o) is an $\{o, \supseteq\}$-consistent answer of L over G under Γ is* coNP*-hard.*

Proof. We start by proving (1). Consider a graph database $G = (V, E)$ over Σ such that the origin o is in V, and a pair (u, u') of nodes in V. We start by proving the following:

Claim 4. *If there is an $\{o, \supseteq\}$-repair $G' = (V', E')$ of G under Γ such that $(u, u') \notin L(G')$, then there is one such G' of size at most polynomial in G.*

In order to do this we apply standard "filtering" techniques on graph databases (such as the ones used by Abiteboul and Vianu to prove that implication of RPCs is decidable under the $\models_o$-interpretation [1]). Assume $\Gamma = \{L_i \subseteq L'_i \mid 1 \leq i \leq n\}$. Take the NFA $\mathcal{A}$ over $\Sigma^{\pm}$ that is equivalent to the product of all L_i's, all L'_i's and the 2RPQ L. For each state q in $\mathcal{A}$, we denote by $q_o(V')$ the set of nodes v in G' such that there is a path π from o to v in $(G')^{\pm}$ for which the following holds: There is a run of the NFA $\mathcal{A}$

over $\lambda(\pi)$ that leads from the initial state to q. Equivalently, we define the set $q_u(v)$, this time for paths that start in u.

Let V_1 be the set $V' \setminus V$. We "filter" the nodes of V_1 with respect to the states of $\mathcal{A}$. This is done as following: Define an equivalence relation $\sim$ over V_1 such that $v \sim v'$, for nodes $v, v' \in V_1$, if and only if for every state q of $\mathcal{A}$ it is the case that:

- $v \in q_o(V') \Leftrightarrow v' \in q_o(V')$, and
- $v \in q_u(V') \Leftrightarrow v' \in q_u(V')$.

We define a new graph database G'' that is obtained from G' by collapsing all nodes in V_1 that belong to the same equivalence class with respect to $\sim$. Using standard techniques (see, e.g., [1]) one can prove the following:

Lemma 3. *It is the case that* $G'' \models_o \Gamma$ *and* $(u, u') \notin L(G'')$.

Since Γ and L are fixed, we have that the size of G'' is polynomial (in fact, linear) in the size of G'. The problem is that we do not know whether G'' is an $\{o, \supseteq\}$-repair of G under Γ. Suppose that this is not the case. Then there is an $\{o, \supseteq\}$-repair G^* of G under Γ such that $G \subseteq G^* \subsetneq G''$. Since 2RPQs are monotone, we have that $(u, u') \notin L(G^*)$. Clearly, G^* is of size at most polynomial (in fact, linear) on G. This proves Claim 4.

Therefore, in order to solve the complement of the $\{o, \supseteq\}$-CQA(L,Γ) problem, for an input given by a graph database G and a pair (u, u) of nodes in G, it is possible to use the following NP algorithm: First guess a polynomial size graph database G' that extends G. Then check in polynomial time that $G' \models_o \Gamma$ and $(u, u') \notin L(G')$. In fact, if the algorithm returns true for some G', then by using the same reasoning than in the previous paragraph we can conclude that there is an $\{o, \supseteq\}$-repair G^* of G under Γ such that $(u, u') \notin L(G^*)$. On the other hand, if the algorithm does not find such G' then we can conclude from Claim 4 that there is no $\{o, \supseteq\}$-repair G^* of G under Γ such that $(u, u') \notin L(G^*)$.

We now prove (2). We reduce from the problem of 3-colorability. Assume we are given an undirected graph H. From H we construct a graph database $G = (V, E)$, such that (1) V corresponds to the set of nodes of H plus the origin o, and (2) E contains edges (o, a, v), for each node v in H, and (u, e, v) and (v, e, u), for each edge $\{u, v\}$ in H. Assume that γ corresponds to the RPC $a \subseteq c_1 \cup c_2 \cup c_3$. Intuitively, this tells us that a $\supseteq$-repair of G contains, for each node $v \neq o$, one, and only one, edge of the form (o, c_i, v), for $1 \leq i \leq 3$.

This edge represents the color assigned to node v by an assignment of three colors to the nodes of H.

It is not hard to prove that H is 3-colorable if and only if there exists a $\supseteq$-repair G' of G such that no two nodes linked by an edge labeled e in G' are assigned the same color. This is equivalent to checking that it is not the case that there are paths labeled $c_i e$ and c_i in G', for $1 \leq i \leq 3$, that start in the origin o and reach the same node v. This can be expressed as the 2RPQ $L := \bigcup_{1 \leq i \leq 3} c_i e c_i^-$. We then have that H is 3-colorable if and only if (o, o) is not an $\{o, \supseteq\}$-consistent answer of L over G under γ. □

Interestingly, we do not know whether the bound in Theorem 6 is also tight for RPQs in the presence of RPCs.

5. Comparison with CQA in the relational context

We compare our results with previous results on CQA obtained in the relational context. We assume familiarity with relational schemas and CQs. Tuple-generating dependencies, or tgds, define one of the most important classes of relational database constraints. They subsume several other classes of interest, such as inclusion dependencies. In addition, they have important applications in data integration, data exchange and ontological query answering [35, 25, 11]. Formally, a tgd over a relational schema σ is a formula of the form $\forall \bar{x}(\phi(\bar{x}) \rightarrow \psi(\bar{x}))$, where both $\phi(\bar{x})$ and $\psi(\bar{x})$ are CQs over σ and each variable in $\bar{x}$ is mentioned in $\phi(\bar{x})$. A relational database D over σ satisfies this tgd if $D \models \phi(\bar{a})$ implies $D \models \psi(\bar{a})$, for each tuple $\bar{a}$ of elements in D of the same length as $\bar{x}$.

As mentioned in the introduction, each word constraint can be naturally seen as a tgd over the standard relational representation of graph databases. However, lower bounds for CQA under tgds in the relational setting, such as the ones obtained by ten Cate et al. [20], cannot be used to obtain lower bounds for CQA under word constraints (or even RPCs) in the graph database context. This is because word constraints correspond to a restricted class of tgds defined by *chain* CQs (which we call *chain tgds*). However, none of the lower bounds developed for the data complexity of CQA under tgds applies to this class.

We formalize the class of chain tgds as follows. Let σ be a relational schema that contains only binary relation symbols. A chain CQ over σ is a

CQ of the form

$$\phi(x,y) \;:=\; \exists u_1 u_2 \ldots u_{m-1}\big(R_1(x,u_1) \wedge R_2(u_1,u_2) \wedge \cdots \wedge R_{m-1}(u_{m-1},y)\big),$$

where each R_i is a relation symbol in σ [23]. That is, the underlying *directed* graph of a chain CQ is a path. A chain tgd is one of the form $\forall x \forall y(\phi(x,y) \rightarrow \psi(x,y))$, where both $\phi(x,y)$ and $\psi(x,y)$ are chain CQs. It is easy to see that each word constraint can be represented as a chain tgd over the standard relational representation of graph databases (in which, for each $a \in \Sigma$, there is a binary relation symbol E_a that contains all pairs of nodes that are linked by an a-labeled edge in the graph database). Conversely, each chain tgd is the representation of a word constraint.

This allows us to use our proof techniques to obtain lower bounds for CQA under the restricted class of chain tgds in the relational context:

Proposition 9. 1. *Consider a semantics based on subset repairs of relational databases. There is a relational schema σ that contains only binary relation symbols, a finite set $\mathcal{T}$ of chain tgds and a union Q of CQs over σ, such that the problem of evaluating certain answers for Q under $\mathcal{T}$ is Π_2^P-hard.*

2. *Consider a semantics based on superset repairs of relational databases. There is a relational schema σ that contains only binary relation symbols, a finite set $\mathcal{T}$ of chain tgds and a union Q of CQs over σ, such that evaluating certain answers for Q under $\mathcal{T}$ is undecidable. The same holds for the semantics of symmetric difference repairs.*

6. Conclusions and Future Work

In this work we initiated the study of CQA over graph databases. The data complexity of the problem is in general undecidable or highly intractable, which motivated our search for decidable, and even tractable restrictions. In the case of subset repair semantics we obtain tractability by either restricting to the class of LAV C2RPCs or to the class of graph databases of bounded treewidth. The class of LAV C2RPCs also yields tractability for our problem under the semantics of $\oplus$-repairs. On the other hand, for the semantics of superset repairs we obtain decidability if we restrict to the class of GAV C2RPCs, or if only consider 2RPQs and 2RPCs which are interpreted from the origin. A picture of some of the complexity bounds obtained for the

	General C2RPCs	word RPCs	LAV (C2)RPCs	GAV (C2)RPCs
$\subseteq$-repairs	Π_2^{P}-complete (Theorem 1)	Π_2^{P}-complete (Theorem 1)	NLogspace (Theorem 2)	coNP-complete (Prop. 4)
$\supseteq$-repairs	Undecid. (Theorem 4)	Undecid. (Theorem 4)	coNP-hard (Prop. 6)	NLogspace (Prop. 8)
$\oplus$-repairs	Undecid. (Theorem 4)	Undecid. (Theorem 4)	NLogspace (Theorem 5)	coNP-complete (Prop. 7)
$\{o, \subseteq\}$-repairs	Π_2^{P}-c (Prop. 3)	Π_2^{P}-c (Prop. 3)	NLogspace (Theorem 2)	coNP-c (Prop. 5)
$\{o, \supseteq\}$-repairs	coNP-complete (Theorem 6)	coNP-complete (Theorem 6)	coNP-complete (Theorem 6)	coNP-complete (Theorem 6)
$\{o, \oplus\}$-repairs	?	?	?	?

Figure 1: Known complexity bounds for the CQA problem for C2RPQs over graph databases under different classes of constraints and repair semantics. Lower bounds hold even for RPQs, save for those for $\{o, \supseteq\}$-repairs that hold for 2RPQs.

problems studied in this paper (for different classes of C2RPCs and under different semantics) is shown in Figure 1.

Several questions regarding CQA under the semantics of $\supseteq$- and $\oplus$-repairs remain open. For instance, we do not know whether CQA under $\supseteq$-repairs is decidable when C2RPCs are in LAV form. Neither we know whether CQA under $\oplus$-repairs is decidable when 2RPCs are interpreted from the origin. We plan to study this in the future.

It would also be interesting to look for different kinds restrictions that yield decidability for our CQA problem. For instance, in the relational scenario it is possible to obtain tractability in data complexity for CQA under $\supseteq$- and $\oplus$-repair semantics if the set Γ of tgds is *weakly acyclic* [20]. The reason is that in this case there is a polynomial that bounds the size of each repair of a database D under Γ. We would like to develop a meaningful adaptation of this notion to the scenario of RPCs in search for similar positive results. However, this is more difficult than in the relational case, since the notion of acyclicity will have to consider how regular expressions interact with each other. Another way in which positive results for our CQA problem under $\supseteq$- and $\oplus$-repair semantics could be obtained, is by restricting to classes of RPCs for which the implication problem is decidable. This includes, for instance, classes of word constraints for which the associated rewrite problem is decidable [28].

Acknowledgements: We are grateful to Aidan Hogan and Leonid Libkin for their helpful comments in earlier version of the paper. Carsten Lutz and Meghyn Bienvenu also provided us with important insights on the nature of DL repairs. Barceló and Fontaine are funded by the Millenium Nucleus Center for Semantic Web Research under grant NC120004. Fontaine is also funded by Fondecyt postdoctoral grant 3130491.

Bibliography

[1] S. Abiteboul, V. Vianu. Regular path queries with constraints. *JCSS*, 58(3), pages 428-452, 1999.

[2] M. Arenas, L. E. Bertossi, Jan Chomicki. Consistent query answers in inconsistent databases. In *PODS* 1999, pages 68-79.

[3] M. Arenas, W. Fan, L. Libkin. On the complexity of verifying consistency of XML specifications. *SIAM J. Comput.* 38(3), pages 841-880, 2008.

[4] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider (Eds.). The description logic handbook: Theory, implementation, and applications. *Cambridge University Press*, 2003.

[5] P. Barceló. Querying graph databases. In *PODS* 2013, pages 175-188.

[6] P. Barceló, G. Fontaine. On the data complexity of consistent query answering over graph databases. In *ICDT* 2015, pages 380-397.

[7] M. Bienvenu. On the complexity of consistent query answering in the presence of simple ontologies. In *AAAI* 2012.

[8] M. Bienvenu, R. Rosati. Tractable approximations of consistent query answering for robust ontology-based data access. In *IJCAI* 2013.

[9] R. Book, F. Otto String. *Rewriting Systems.* Springer Verlag, 1993.

[10] P. Buneman, W. Fan, S. Weinstein. Path constraints in semistructured databases. *JCSS* 61(2), pages 146-193, 2000.

[11] A. Calì, G. Gottlob, M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *JAIR* 48, pages 115-174, 2013.

[12] Andrea Calì, D. Lembo, R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *PODS* 2003, pages 260-271.

[13] D. Calvanese, G. de Giacomo, M. Lenzerini. Structured objects: Modeling and reasoning. In *DOOD* 1995, pages 229-246.

[14] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *KR* 2000, pages 176-185.

[15] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. Rewriting of regular expressions and regular path queries. *JCSS*, 64(3), pages 443-465, 2002.

[16] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. Reasoning on regular path queries. *SIGMOD Record* 32(4), pages 83-92, 2003.

[17] D. Calvanese, G. de Giacomo, M. Lenzerini. Conjunctive query containment and answering under description logic constraints. *ACM TOCL* 9(3), 2008.

[18] D. Calvanese, M. Ortiz, M. Simkus. Containment of regular path queries under description logic constraints. In *IJCAI* 2011, pages 805-812.

[19] D. Calvanese, M. Ortiz, M. Simkus. Verification of evolving graph-structured data under expressive path constraints. In *ICDT* 2016, pages 1-19.

[20] B. ten Cate, G. Fontaine, Ph. G. Kolaitis. On the data complexity of consistent query answering. In *ICDT* 2012, pages 22-33.

[21] J. Chomicki, J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.* 197(1-2), pages 90-121, 2005.

[22] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Inf. Comput.* 85(1), pages 12-75, 1990.

[23] G. Dong. On datalog linearization of chain queries. *Theoretical Studies in Computer Science* 1992, pages 181-206.

[24] W. Fan, J. Siméon: Integrity constraints for XML. *JCSS*, 66(1), pages 254-291, 2003.

[25] R. Fagin, Ph. G. Kolaitis, R. J. Miller, L. Popa. Data exchange: semantics and query answering. *TCS* 336(1), pages 89-124, 2005.

[26] G. Fontaine. Why is it hard to obtain a dichotomy for consistent query answering? In *LICS* 2013, pages 550-559.

[27] A. Fuxman, R. J. Miller. First-order query rewriting for inconsistent databases. *JCSS* 73(4), pages 610-635, 2007.

[28] G. Grahne, A. Thomo. Query containment and rewriting using views for regular path queries under constraints. In *PODS* 2003, pages 111-122.

[29] A. Hogan, A. Harth, A. Passant, S. Decker, A. Polleres. Weaving the pedantic web. In *LDOW* 2010.

[30] Ph. G. Kolaitis, Enela Pema. A dichotomy in the complexity of consistent query answering for queries with two atoms. *Inf. Process. Lett.* 112(3), pages 77-85, 2012.

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

[32] G. Lausen, M. Meier, M. Schmidt. SPARQLing constraints for RDF. In *EDBT* 2008, pages 499-509.

[33] D. Lembo, M. Ruzzi. Consistent query answering over description logic ontologies. In *DL* 2007.

[34] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, D. Fabio Savo. Inconsistency-tolerant semantics for description logics. In *RR* 2010, pages 103-117.

[35] M. Lenzerini. Data integration: A theoretical perspective. In *PODS* 2002, pages 233-246.

[36] Th. Lukasiewicz, M. V. Martinez, G. I. Simari. Complexity of inconsistency-tolerant query answering in Datalog+/-. In *OTM Conferences* 2013, pages 488-500.

[37] R. Rosati. On the complexity of dealing with inconsistency in description logic ontologies. In *IJCAI* 2011, pages 1057-1062.

[38] K.-D. Schewe, B. Thalheim, J. W. Schmidt, I. Wetzel. Integrity enforcement in object-oriented databases. In *FMLDO* 1992, pages 174-195.

[39] S. Staworko. Declarative Inconsistency Handling in Relational and Semi-structured Databases. *PhD thesis*, 2007.

[40] J. Wijsen. Condensed representation of database repairs for consistent query answering. In *ICDT* 2003, pages 375-390.

[41] J. Wijsen. Certain conjunctive query answering in first-order logic. *ACM TODS* 37(2), 9, 2012.

[42] P. T. Wood. Query languages for graph databases. *SIGMOD Record* 41(1), pages 50-60, 2012.

[43] Y. Yuan, G. Wang, L. Chen, H. Wang. Efficient subgraph similarity search on large probabilistic graph databases. In *PVLDB* 5(9), pages 800-811, 2012.