Consistent Query Answering over SHACL Constraints

Shqiponja Ahmetaj , Timo Camillo Merkl , Reinhard Pichler
TU Wien
{shqiponja.ahmetaj, timo.merkl, reinhard.pichler}@tuwien.ac.at

Abstract

The Shapes Constraint Language (SHACL) was standardized by the World Wide Web as a constraint language to describe and validate RDF data graphs. SHACL uses the notion of shapes graph to describe a set of *shape* constraints paired with *targets*, that specify which nodes of the RDF graph should satisfy which shapes. An important question in practice is how to handle data graphs that do not validate the shapes graph. A solution is to *tolerate* the non-validation and find ways to obtain meaningful and correct answers to queries despite the non-validation. This is known as *consistent query answering* (CQA) and there is extensive literature on CQA in both the database and the KR setting. We study CQA in the context of SHACL for a fundamental fragment of the Semantic Web query language SPARQL. The goal of our work is a detailed complexity analysis of CQA for various semantics and possible restrictions on the acceptable repairs. It turns out that all considered variants of the problem are intractable, with complexities ranging between the first and third level of the polynomial hierarchy.

1 Introduction

The SHACL Shapes Constraint Language is the World Wide Web recommendation language for expressing and validating constraints on RDF data graphs (Knublauch and Kontokostas 2017). The normative standard emerged as a need for a *prescriptive* language to provide guarantees on the structure and content of RDF data (W3C 2013). SHACL uses the notion of *shapes graph* to describe a set of *shape* constraints paired with *targets*, that specify which nodes of the RDF graph should satisfy which shapes. The main computational problem in SHACL is to check whether an RDF graph *validates* a shapes graph. However, the W3C specification does not clearly define, and sometimes leaves undefined, certain aspects of validation, such as the semantics of validation for *recursive* constraints, which involve cyclic dependencies. A formal logic-based syntax and semantics for (recursive) SHACL was proposed in (Corman, Reutter, and Savkovic 2018), and has served as bases for most subsequent works (Andresel et al. 2020; Ahmetaj et al. 2023; Bogaerts, Jakubowski, and den Bussche 2022).

SHACL is particularly close to expressive Description Logics (DLs), the logics underlying OWL, as already observed in several works (Bogaerts, Jakubowski, and den Bussche 2022; Leinberger et al. 2020; Ortiz 2023; Ahmetaj et al. 2021). Specifically, since SHACL adopts the closed world assumption, it is closely related to an extension of the DL $\mathcal{ALCOIQ}$ with regular role expressions and equalities, where all roles and some concept names are viewed as *closed predicates* (see e.g. (Lutz, Seylan, and Wolter 2013)).

It is commonly recognized that real-world data, and in particular, graph-structured data and large RDF triple stores, which are subject to frequent change, may be incomplete or contain faulty facts. It thus seems inevitable to expect a data graph to not validate a SHACL shapes graph, e.g., because it is missing some facts to validate a target or it has conflicting or contradictory facts. The question of how to handle such data graphs is very relevant in practice. A possible solution is to fix the data graph before reasoning. In the style of database *repairs*, (Ahmetaj et al. 2021; Ahmetaj et al. 2022) propose to fix the data graph through (minimal) additions or removals of facts such that the resulting data graph validates the shapes graph. However, this may not always be desirable in practice as there may be a large number of possible repairs and selecting one may keep wrong facts, or remove true facts.

Another alternative is to *tolerate* the non-validation and find ways to leverage the consistent part of the data and obtain meaningful and correct answers to queries despite the non-validation. This view is known as *consistent query answering* (CQA), and it has gained a lot of attention since the seminal paper (Arenas, Bertossi, and Chomicki 1999). The idea is to accept as answers to a query those that are true over all (minimal) repairs of the input database. This is also known as the *AR* semantics (Arenas, Bertossi, and Chomicki 1999; Bertossi 2011; Lembo et al. 2010; Arming, Pichler, and Sallinger 2016). Several other inconsistency-tolerant semantics have also been studied such as *brave* (Bienvenu and Rosati 2013) and *IAR* (Lembo et al. 2010) semantics. The former accepts answers that are true in *some* repair, and the latter accepts the most reliable answers, that is those that are true in the *intersection of all* repairs. CQA has been extensively studied in various database and knowledge representation settings; we refer to (Bertossi 2011; Wijsen 2019; Bienvenu and Bourgaux 2016) for nice surveys.

In this work, we focus on CQA in the presence of (recursive) SHACL shapes, which to our knowledge, has not yet been explored. As a query language we consider a fundamental fragment of SPARQL, which is the standardized

language to query RDF data. Specifically, we focus on *basic graph patterns* (BGPs), which are essentially conjunctive queries (CQs), and the well-behaved extension with the OPTIONAL operator – the so-called *well-designed* fragment of SPARQL) (Pérez, Arenas, and Gutierrez 2009), referred to as *well-designed queries* (WDQs, for short) in this paper.

Our main goal is a detailed complexity analysis of the CQA problem. We thus build on (Ahmetaj et al. 2021), which analyzes the complexity of the main reasoning problems for repairs w.r.t. SHACL constraints, such as checking the existence of a repair and deciding if a particular fact is added or deleted in at least one or in all repairs. These problems were further refined by restricting to repairs that are minimal w.r.t. cardinality or set inclusion. Of course, these restrictions are also highly relevant for the CQA problem. In total, we will thus study numerous variants of the CQA problem by considering 4 query languages (BGPs and WDQs, with or without projection), under 3 semantics (brave, AR, IAR), with or without (cardinality or subset inclusion) minimality-restrictions. Moreover, we distinguish data complexity (where the SHACL constraints and the query are considered as fixed and only the data is allowed to vary) and combined complexity. We refer to Table 2 for an overview of our main results. Formal definitions of all terms appearing in this table are given in Sections 2 and 3.

We note that the settings considered for CQA in the literature crucially differ from ours in several respects. The typical constraint languages studied for databases are (fragments of) tuple generating dependencies (tgds, i.e., rules with conjunctive queries (CQs) in both the body and in the head) and equality generating dependencies (egds, i.e., rules with a CQ in the body and an equality in the head), see e.g., (Afrati and Kolaitis 2009; Arming, Pichler, and Sallinger 2016; ten Cate, Fontaine, and Kolaitis 2015). SHACL also has implications as the crucial building blocks. However, in contrast with these dependencies, SHACL allows (among other features not present in tgds and egds) *(unrestricted) negation* in the rule body, which has significant semantical implications. There is also a large body of works on CQA in the context of Description Logic knowledge bases, see e.g., (Bienvenu 2012; Bienvenu, Bourgaux, and Goasdoué 2019; Lembo et al. 2015; Du, Qi, and Shen 2013). However, to our knowledge, there are no works studying CQA for (expressive) DLs with closed predicates. Finally, the query languages considered in the literature focus on CQs with a few works considering extensions such as tree/path queries (Koutris, Ouyang, and Wijsen 2021; Koutris, Ouyang, and Wijsen 2024), datalog (Koutris and Wijsen 2021), and counting (Khalfioui and Wijsen 2023). Apart from CQs (i.e., BGPs), we also study *non-monotonic queries* in the form of wdQs. To the best of our knowledge, wdQs have not been considered in the context of CQA.

In this paper, we proceed as follows: We initially investigate the above mentioned problems, considering the scenario that a repair always exists. Indeed, it seems plausible to assume that a SHACL shapes graph is carefully designed so that no conflicting constraints are introduced. If this is not guaranteed, the existence of a repair can be tested with NP-power as shown in (Ahmetaj et al. 2021). In case of a negative outcome of this test, it may still be possible to provide a repair that validates a subset of the targets. To address this, in the spirit of inconsistency tolerance, (Ahmetaj et al. 2022) proposes a relaxed notion of repairs, which aims at validating a maximal subset of the targets. We also study the complexity of CQA over *maximal repairs*.

Our main contributions are summarized as follows:

- We first study the complexity of CQA in settings where we can validate all targets. It turns out that brave and AR semantics behave very similarly in terms of algorithms (to establish membership results) and in terms of methods for proving hardness results. We therefore study these cases simultaneously in Section 4.
- In Section 5, we study the complexity for IAR semantics. It turns out that the influence of choosing different query languages and/or minimality conditions on the repairs gives a yet more colorful picture than with brave and AR semantics.
- Finally, in Section 6, we extend our complexity analysis to the settings where the data cannot be fully repaired. Thus, we resort back to validating as many targets as possible. It turns out that this increases the complexity beyond the first level of the polynomial hierarchy. However, in all cases in Table 2 with complexity of $\Theta_2\mathsf{P}$ or above, the complexity classification remains the same.

In all cases, we provide a complete complexity classification in the form of matching upper and lower bounds. We note that all results hold for both non-recursive and recursive constraints. Due to lack of space, proof details have to be omitted. Full proofs of all results presented here are given in the arXiv version (Ahmetaj, Merkl, and Pichler 2024).

2 Preliminaries

In this section, we introduce RDF graphs, SHACL, *validation* against RDF graphs, and (well-designed) SPARQL queries. We follow the abstract syntax and semantics for the fragment of SHACL core studied in (Ahmetaj et al. 2021); for more details on the W3C specification of SHACL core we refer to (Knublauch and Kontokostas 2017).

RDF Graphs. We let N_N, N_C, N_P denote countably infinite, mutually disjoint sets of *nodes* (constants), *class names*, and *property names*, respectively. An RDF (data) graph G is a finite set of (ground) *atoms* of the form $B(c)$ and $p(c, d)$, where $B \in N_C$, $p \in N_P$, and $c, d \in N_N$. The set of nodes appearing in G is denoted with $V(G)$.

SHACL Validation. We assume a countably infinite set N_S of *shape names*, disjoint from $N_N \cup N_C \cup N_P$. A *shape atom* is an expression of the form $\mathsf{s}(a)$, where $\mathsf{s} \in N_S$ and $a \in N_N$. A *path expression* E is a regular expression built using the usual operators $*$, $\cdot$, $\cup$, property names $p \in N_P$ and *inverse properties* p^-, where $p \in N_P$. A *(complex) shape* is an expression φ obeying the syntax:

$$\varphi, \varphi' ::= \top \mid \mathsf{s} \mid B \mid c \mid \varphi \wedge \varphi' \mid \neg\varphi \mid \geq_n E.\varphi \mid E = E',$$

where $\mathsf{s} \in N_S$, $p \in N_P$, $B \in N_C$, $c \in N_N$, n is a positive integer, and E, E' are path expressions. In what follows, we

$[\![\top]\!]^I = V(I) \quad [\![c]\!]^I = \{c\} \quad [\![B]\!]^I = \{c \mid B(c) \in I\}$
$[\![\mathsf{s}]\!]^I = \{c \mid \mathsf{s}(c) \in I\} \quad [\![p]\!]^I = \{(a,b) \mid p(a,b) \in I\}$
$[\![p^-]\!]^I = \{(a,b) \mid p(b,a) \in I\}$
$[\![E \cup E']\!]^I = [\![E]\!]^I \cup [\![E']\!]^I \quad [\![E \cdot E']\!]^I = [\![E]\!]^I \circ [\![E']\!]^I$
$[\![E^*]\!]^I = \{(a,a) \mid a \in V(I)\} \cup [\![E]\!]^I \cup [\![E \cdot E]\!]^I \cup \cdots$
$[\![\neg\varphi]\!]^I = V(I) \setminus [\![\varphi]\!]^I \quad [\![\varphi_1 \wedge \varphi_2]\!]^I = [\![\varphi_1]\!]^I \cap [\![\varphi_2]\!]^I$
$[\![\geq_n E.\varphi]\!]^I = \{c \mid |\{(c,d) \in [\![E]\!]^I \text{ and } d \in [\![\varphi]\!]^I\}| \geq n\}$
$[\![E = E']\!]^I = \{c \mid \forall d : (c,d) \in [\![E]\!]^I \text{ iff } (c,d) \in [\![E']\!]^I\}$

Table 1: Evaluation of complex shapes

write $\varphi \vee \varphi'$ instead of $\neg(\neg\varphi \wedge \neg\varphi')$; $\geq_n E$ instead of $\geq_n E.\top$; $\exists E.\varphi$ instead of $\geq_1 E.\varphi$; $\forall E.\varphi$ instead of $\neg\exists E.\neg\varphi$; $=_n E.\varphi$ instead of $\leq_n E.\varphi \wedge \geq_n E.\varphi$.

A *(shape) constraint* is an expression $\mathsf{s} \leftrightarrow \varphi$ where $\mathsf{s} \in N_S$ and φ is a complex shape. W.l.o.g., we view *targets* as shape atoms of the form $\mathsf{s}(a)$, where $\mathsf{s} \in N_S$ and $a \in N_N$, which asks to check whether the shape name s is validated at node a of the input data graph. The SHACL specification allows for a richer specification of targets but these do not affect the results in this paper. A *shapes graph* is a pair $(\mathcal{C}, \mathcal{T})$, where $\mathcal{C}$ is a set of constraints and $\mathcal{T}$ is a set of targets. We assume that each shape name appearing in $\mathcal{C}$ occurs exactly once on the left-hand side of a constraint. A set of constraints $\mathcal{C}$ is *recursive*, if there is a shape name in $\mathcal{C}$ that directly or indirectly refers to itself.

The evaluation of shape expressions is given by assigning nodes of the data graph to (possibly multiple) shape names. More formally, a *(shape) assignment* for a data graph G is a set $I = G \cup L$, where L is a set of shape atoms such that $a \in V(G)$ for each $\mathsf{s}(a) \in L$. The evaluation of a complex shape w.r.t. an assignment I is given in terms of a function $[\![\cdot]\!]^I$ that maps a shape expression φ to a set of nodes, and a path expression E to a set of pairs of nodes (see Table 1).

There are several validation semantics for SHACL with recursion (Corman, Reutter, and Savkovic 2018; Andresel et al. 2020; Chmurovic and Simkus 2022), which coincide on non-recursive SHACL. Here, we follow (Ahmetaj et al. 2021) and consider the supported model semantics from (Corman, Reutter, and Savkovic 2018). Assume a SHACL shapes graph $(\mathcal{C}, \mathcal{T})$ and a data graph G such that each node that appears in $\mathcal{C}$ or $\mathcal{T}$ also appears in G. Then, an assignment I for G is a *(supported) model* of $\mathcal{C}$ if $[\![\varphi]\!]^I = \mathsf{s}^I$ for all $\mathsf{s} \leftrightarrow \varphi \in \mathcal{C}$. The data graph G *validates* $(\mathcal{C}, \mathcal{T})$ if there exists an assignment $I = G \cup L$ for G such that (i) I is a model of $\mathcal{C}$, and (ii) $\mathcal{T} \subseteq L$.

Example 1. *Consider G and the shapes graph $(\mathcal{C}, \mathcal{T})$:*

$$G = \{Prof(\texttt{Ann}), worksWith(\texttt{Lea}, \texttt{Ann}), Student(\texttt{Ben}), id(\texttt{Ben}, \texttt{ID1}), id(\texttt{Ben}, \texttt{ID2}), enrolledIn(\texttt{Ben}, c), id(\texttt{John}, \texttt{ID3}), Student(\texttt{John})\}$$

$$\mathcal{C} = \{\mathsf{Profshape} \leftrightarrow Prof \vee \exists worksWith.\mathsf{Profshape}, \mathsf{Studshape} \leftrightarrow Student \wedge =_1 id \wedge \exists enrolledIn\}$$

$$\mathcal{T} = \{\mathsf{Studshape}(\texttt{Ben}), \mathsf{Studshape}(\texttt{John})\}$$

The first constraint is recursive and intuitively, it states that nodes validating the shape name Profshape *must either belong to the class $Prof$ or have a $worksWith$ connection with some node validating* Profshape*. The second constraint states that nodes validating* Studshape *must belong to the class $Students$, have exactly one id, and belong to some $enrolledIn$ fact. The targets ask to check whether* Ben *and* John *satisfy the constraint for* Studshape*. The data graph G does not validate the shapes graph. Intuitively, the reason is that G contains more than one id for Ben and it is missing an $enrolledIn$ fact for $John$. In other words, data graph G is inconsistent w.r.t. shapes graph $(\mathcal{C}, \mathcal{T})$. Conversely, G validates $(\mathcal{C}, \mathcal{T}')$ with $\mathcal{T}' = \{\mathsf{Profshape}(\texttt{Lea})\}$ as witnessed by the assignment $I = G \cup \{\mathsf{Profshape}(\texttt{Lea}), \mathsf{Profshape}(\texttt{Ann})\}$.*

Well-Designed SPARQL. Let N_V be an infinite set of variables, disjoint from $N_N \cup N_C \cup N_P \cup N_S$. A *basic graph pattern (*BGP*)* is a conjunction of atoms $\psi_1 \wedge \cdots \wedge \psi_n$, where $n \geq 0$ and each ψ_i is of the form $B(t)$ or $p(t_1, t_2)$ with $B \in N_C$, $p \in N_P$, and $t, t_1, t_2 \in N_V \cup N_N$. We denote the empty conjunction with $\top$. Intuitively, a BGP is a conjunctive query built from atoms over class and property names over variables and constants, where all the variables are output variables. We focus on SPARQL queries built from BGPs and the *OPTIONAL* (or OPT) operator. We thus may assume the so-called "OPT-normal form" (which disallows OPT-operators in the scope of a $\wedge$-operator) introduced by (Pérez, Arenas, and Gutierrez 2009).

A *(SPARQL) mapping* is any partial function μ from N_V to N_N. Given a unary or binary atom $\psi(\vec{t})$ and a mapping μ, we use $\mu(\psi(\vec{t}))$ to denote the ground atom obtained from $\psi(\vec{t})$ by replacing every variable x in $\vec{t}$ by $\mu(x)$. We write $\mathsf{dom}(\mu)$ to denote the domain of μ and $\mathsf{vars}(Q)$ for the set of variables in Q. Mappings μ_1 and μ_2 are *compatible* (written $\mu_1 \sim \mu_2$) if $\mu_1(x) = \mu_2(x)$ for all $x \in \mathsf{dom}(\mu_1) \cap \mathsf{dom}(\mu_2)$.

The evaluation of a SPARQL query Q over an RDF graph G is defined as follows:

1. $[\![Q]\!]_G = \{\mu \mid \mathsf{dom}(\mu) = \mathsf{vars}(Q)$, and $\mu(\psi_i) \in G$ for $i = 1, \ldots, n\}$, where Q is a BGP $\psi_1 \wedge \cdots \wedge \psi_n$.
2. $[\![Q_1 \text{ OPT } Q_2]\!]_G = \{\mu_1 \cup \mu_2 \mid \mu_1 \in [\![Q_1]\!]_G, \mu_2 \in [\![Q_2]\!]_G,$ and $\mu_1 \sim \mu_2\} \cup \{\mu_1 \in [\![Q_1]\!]_G \mid \forall \mu_2 \in [\![Q_2]\!]_G : \mu_1 \nsim \mu_2\}$

As in (Pérez, Arenas, and Gutierrez 2009), we assume set semantics. A SPARQL query Q is *well-designed* (WDQs for short), if there is no subquery $Q' = (P_1 \text{ OPT } P_2)$ of Q and a variable x, such that x occurs in P_2, outside of Q', but not in P_1. It was shown in (Pérez, Arenas, and Gutierrez 2009) that the complexity of query evaluation with unrestricted OPT is PSPACE-complete, while for WDQs is coNP-complete.

Projection in SPARQL is realized via the SELECT result modifier on top of queries. For a mapping μ and a set X of variables, we let $\mu|_X$ denote the mapping μ' that restricts μ to the variables in X, that is $\mathsf{dom}(\mu') = X \cap \mathsf{dom}(\mu)$ and $\mu'(x) = \mu(x)$ for all $x \in \mathsf{dom}(\mu')$. The result of evaluating a query Q with projection to the variables in X over a graph G is defined as $[\![\pi_X Q]\!]_G = \{\mu|_X \mid \mu \in [\![Q]\!]_G\}$. We refer to queries $\pi_X Q$ as a "projected WDQ" (or π-WDQ), and as a "projected BGP" (or π-BGP) if Q is just a BGP. Note that there is no gain in expressiveness when we allow projections to also appear inside Q. It was shown in (Letelier et

al. 2013) that checking if some mapping μ is an answer to a π-WDQ $\pi_X Q$ over a graph G is Σ_2P-complete. By inspecting the Σ_2P-membership proof, it turns out that, w.l.o.g., we may assume for an arbitrary π-WDQ $\pi_X Q$, that Q is of the form $((\dots((P \text{ OPT } P_1) \text{ OPT } P_2)\dots) \text{ OPT } P_k)$, such that $\mathsf{vars}(P) \cap X = \mathsf{dom}(\mu)$ and, for every i, $\mathsf{vars}(P_i) \subseteq X$ and P_i contains at least one variable from $X \setminus \mathsf{vars}(P)$. Then μ is an answer to $\pi_X Q$, iff *there exists* an extension ν of μ to the variables in $Y = \mathsf{vars}(P) \setminus X$ s.t. *there does not exist* an extension of ν to an answer of one of the queries P_i.

3 Querying Non-valid Data Graphs

In this section, we recall the notion of repairs for a data graph in the presence of a SHACL shapes graph proposed in (Ahmetaj et al. 2021) and then introduce the three inconsistency-tolerant semantics in this setting.

Repairing Non-Validation. We can explain non-validation of a SHACL shapes graph in the style of database repairs. Hence, a repair is provided as a set A of facts to be added and a set D of facts to be deleted, so that the resulting data graph validates the shapes graph. We recall the definition below, but instead of "*explanations*" we speak of "*repairs*".

Definition 1 ((Ahmetaj et al. 2021))**.** *Let G be a data graph, let $(\mathcal{C}, \mathcal{T})$ be a SHACL shapes graph, and let the set of* hypotheses *H be a data graph disjoint from G. A* repair *for $(G, \mathcal{C}, \mathcal{T}, H)$ is a pair (A, D), such that $D \subseteq G$, $A \subseteq H$, and $(G \setminus D) \cup A$ validates $(\mathcal{C}, \mathcal{T})$; we call $(G \setminus D) \cup A$ the* repaired *(data) graph G_R of G w.r.t. $R = (A, D)$.*

As usual in databases, instead of considering all possible repairs, we consider *preference relations* given by a pre-order $\preceq$ (a reflexive and transitive relation) on the set of repairs. Following (Ahmetaj et al. 2021), we study *subset-minimal* ($\subseteq$), and *cardinality-minimal* ($\leq$) repairs. For two repairs $(A, D), (A', D')$, we write $(A, D) \subseteq (A', D')$ if $A \subseteq A'$ and $D \subseteq D'$, and $(A, D) \leq (A', D')$ if $|A| + |D| \leq |A'| + |D'|$. A preferred repair under the pre-order $\preceq$, called $\preceq$-repair, is a repair R such that there is no repair R' for Ψ with $R' \preceq R$ and $R \not\preceq R'$. In that case, we also call G_R a $\preceq$-repaired graph. Clearly, every $\leq$-repair is also a $\subseteq$-repair, but not vice versa. We denote with $=$ when there is no preference order, and we use $\preceq$ as a placeholder for $\subseteq$, $\leq$, and $=$. We illustrate the notion of repairs by revisiting Example 1.

Example 2. *Consider G and $(\mathcal{C}, \mathcal{T})$ from Example 1 and H defined as follows:*

$$H = \{\mathit{enrolledIn}(\texttt{John}, \texttt{c1}), \mathit{enrolledIn}(\texttt{Ben}, \texttt{c2})\}$$

Recall from Example 1 that G does not validate $(\mathcal{C}, \mathcal{T})$. Validation can be obtained by repairing G with the subset- and cardinality-minimal repairs $R_1 = (A, D_1)$ and $R_2 = (A, D_2)$, where $A = \{\mathit{enrolledIn}(\texttt{John}, \texttt{c1})\}$, and each D_j includes the fact $\mathit{id}(\texttt{Ben}, \texttt{IDj})$. There are more repairs, e.g., $R_3 = (A', D_1)$ and $R_4 = (A', D_2)$ with $A' = H$, but they are neither $\subseteq$-minimal nor $\leq$-minimal.

Constructs such as existential restrictions in constraints may sometimes enforce repairs to add atoms over fresh nodes. This is supported by the set of hypothesis H. Observe that H can only introduce a limited number of fresh nodes. Indeed, it was shown in (Ahmetaj et al. 2021) that if H is left unrestricted, most problems related to repairs, such as checking the existence of a repair, become undecidable.

Query Answering Semantics under Repairs. We now define the three inconsistency-tolerant semantics *brave*, *AR*, and *IAR semantics*, which we will refer to by the symbols $\exists$, $\forall$, and $\cap$, respectively. Consider a query Q, a mapping μ, a data graph G, a shapes graph $(\mathcal{C}, \mathcal{T})$, and hypotheses H. Then, μ is an answer of Q over $\Psi = (G, \mathcal{C}, \mathcal{T}, H)$ and preference order $\preceq$:

- under *brave semantics*, if there *exists* a $\preceq$-repair R for Ψ, such that $\mu \in [\![Q]\!]_{G_R}$,
- under *AR semantics*, if $\mu \in [\![Q]\!]_{G_R}$ *for all* $\preceq$-repairs R for Ψ,
- under *IAR semantics*, if $\mu \in [\![Q]\!]_{G_\cap}$, where $G_\cap = \bigcap\{G_R \mid R \text{ is a } \preceq\text{-repair for } \Psi\}$, i.e., the *intersection of all* $\preceq$-repaired graphs G_R.

We illustrate the semantics by continuing Example 2.

Example 3. *Consider again $\Psi = (G, \mathcal{C}, \mathcal{T}, H)$ from Example 2 together with the* BGP *$Q = \mathit{Student}(x) \wedge \mathit{id}(x, y)$. Clearly, the mapping $\mu_1 = \{x \to \texttt{John}, y \to \texttt{ID3}\}$ is an answer of Q over Ψ under brave, AR, and IAR semantics. The mappings $\mu_2 = \{x \to \texttt{Ben}, y \to \texttt{ID1}\}$ and $\mu_3 = \{x \to \texttt{Ben}, y \to \texttt{ID2}\}$ are answers of Q over G_{R_1} and G_{R_2}, respectively, and hence under brave semantics, but not under AR and IAR semantics. Now consider the* WDQ *$Q2 = \mathit{Student}(x)$ OPT $\mathit{id}(x, y)$. In this case, μ_1 and $\mu_4 = \{x \to \texttt{Ben}\}$ are solutions under IAR semantics. Clearly, μ_1, μ_2, and μ_3 are still answers to $Q2$ under brave semantics and μ_1 under AR semantics. Note that the above statements hold for each preference order $\preceq$.*

Observe that for BGPs (with and without projection), if μ is an answer under IAR semantics, then μ is an answer under AR and brave semantics. This is due to the monotonicity property of BGPs. For well-designed queries, this may not be the case. For instance, μ_4 in Example 3 is an answer under IAR semantics, but not under AR and brave semantics. However, μ_4 *can be extended* to a solution over every repaired graph G_{R_i} of G. In fact, it is known that well-designed queries have some weak form of monotonicity, in the sense that a solution is not lost, but it may be extended if new facts are added to the data.

Decision Problems. For a given SPARQL query language $\mathcal{L} \in \{\text{BGP}, \pi\text{-BGP}, \text{WDQ}, \pi\text{-WDQ}\}$, preference order $\preceq \in \{=, \leq, \subseteq\}$, and inconsistency-tolerant semantics $\mathcal{S} \in \{\exists, \forall, \cap\}$, we define the CQA problem as follows:

Problem: CQA($\mathcal{L}, \preceq, \mathcal{S}$)
Input: A query $Q \in \mathcal{L}$, $\Psi = (G, \mathcal{C}, \mathcal{T}, H)$, and a mapping μ.
Question: Is μ an answer of Q over Ψ and preference order $\preceq$ under $\mathcal{S}$-semantics?

For all settings, we analyze both the *data* and *combined* complexity. We refer to Table 2 for the complete picture

$\mathcal{L} \setminus \preceq, \mathcal{S}$	$=, \exists$	$\leq, \exists$	$\subseteq, \exists$	$=, \forall$	$\leq, \forall$	$\subseteq, \forall$	$=, \cap$	$\leq, \cap$	$\subseteq, \cap$
BGP (DC)	NP ▲	Θ_2P ▲	Σ_2P ▲	coNP ▲	Θ_2P ▲	Π_2P ▲	coNP ▲	Θ_2P ▲	Π_2P ▲
π-BGP (DC)	NP	Θ_2P	Σ_2P	coNP	Θ_2P	Π_2P	coNP ▼	Θ_2P	Π_2P
WDQ (DC)	NP	Θ_2P	Σ_2P	coNP	Θ_2P	Π_2P	DP ▲▼	Θ_2P	DP_2 ▲
π-WDQ (DC)	NP ▼	Θ_2P ▼	Σ_2P	coNP ▼	Θ_2P ▼	Π_2P ▼	Θ_2P ▲▼	Θ_2P ▼	Θ_3P ▲
BGP (CC)	NP	Θ_2P	Σ_2P	coNP	Θ_2P	Π_2P	coNP ▼	Θ_2P	Π_2P
π-BGP (CC)	NP ▼	Θ_2P ▼	Σ_2P	Π_2P ▲▼	Π_2P ▲▼	Π_2P ▼	Θ_2P ▲▼	Θ_2P ▼	Π_2P ▼
WDQ (CC)	Σ_2P ▲	Σ_2P ▲	Σ_2P	coNP ▼	Θ_2P ▼	Π_2P ▼	Θ_2P ▲▼	Θ_2P ▼	DP_2 ▼
π-WDQ (CC)	Σ_2P ▼	Σ_2P ▼	Σ_2P ▼	Π_3P ▲▼	Π_3P ▲▼	Π_3P ▲▼	Σ_2P ▲▼	Σ_2P ▲▼	Θ_3P ▼

Table 2: Complexity Results of the CQA($\mathcal{L}, \preceq, \mathcal{S}$) problem for query language $\mathcal{L} \in \{\text{BGP}, \pi\text{-BGP}, \text{WDQ}, \pi\text{-WDQ}\}$, preference order $\preceq \in \{=, \leq, \subseteq\}$, and semantics $\mathcal{S} \in \{\exists, \forall, \cap\}$ as abbreviations for brave, AR and IAR semantics, respectively. Moreover *DC* and *CC* stand for data and combined complexity. The up- and down-triangles (▲▼) indicate hardness and membership proofs, respectively, in this work. All complexity classifications are completeness results but not all have to be proved separately, since hardness results carry over from more special to more general cases and membership results carry over in the opposite direction. Colors indicate the theorem where a (hardness or membership) result is proved, namely Theorem 1 (▲▼), Theorem 2 (▲▼), Theorem 3 (▲▼), Theorem 4 (▲▼), and Theorem 5 (▲▼).

of our main results, which will be discussed in detail in Sections 4 and 5. In Section 6, we will then study the "max-variants" of these problems, i.e., settings where the existence of a repair is not guaranteed and we may have to settle for validating a maximal subset of the targets. There may be several reasons for the non-existence of a repair – including conflicting constraints with target shape atoms, unsatisfiable constraints, or insufficient hypothesis set. For instance, consider constraints $\mathsf{s1} \leftrightarrow B$ and $\mathsf{s2} \leftrightarrow \neg B$ and targets $\mathsf{s1}(a)$ and $\mathsf{s2}(a)$; in this case adding $B(a)$ violates the second constraint and not adding it violates the first constraint. Hence, there exists no repair for any input data graph. Note that we never use recursive constraints in our lower bounds, and the combined complexity lower bounds hold even for fixed constraints and hypotheses. The results in Sections 4 and 5 hold even for fixed targets.

4 Brave and AR Semantics

We start our complexity analysis of CQA with two of the most basic cases, which yield the lowest complexity classifications in Table 2, namely the data complexity of the CQA(BGP, $=, \exists$) and CQA(BGP, $=, \forall$) problems. The NP-membership of the former and the coNP-membership of the latter are immediate: Given a graph G, shapes graph $(\mathcal{C}, \mathcal{T})$, hypotheses H, BGP Q, and mapping μ, do the following: (1) guess a repaired graph G_R together with a supported model I and (2) check that $\mu \in [\![Q]\!]_{G_R}$ holds (for CQA(BGP, $=, \exists$)) or $\mu \notin [\![Q]\!]_{G_R}$ holds (for CQA(BGP, $=, \forall$)), respectively; moreover, check that I is indeed a supported model. That is, yes-instances in the case of brave semantics and no-instances in the case of AR semantics are identified by essentially the same procedure. We, therefore, study the two semantics simultaneously in this section. Note that, switching to π-WDQ as the most expressive query language considered here, does not increase the *data complexity*, since the check $\mu \in [\![Q]\!]_{G_R}$ or $\mu \notin [\![Q]\!]_{G_R}$ is still feasible in P for π-WDQs (this is even true for arbitrary SPARQL queries).

Now consider the CQA($\mathcal{L}, \leq, \exists$) and CQA($\mathcal{L}, \leq, \forall$) problems for $\mathcal{L} \in \{\text{BGP}, \pi\text{-BGP}, \text{WDQ}, \pi\text{-WDQ}\}$. To arrive at a cardinality minimal repair R, one first has to compute the minimal cardinality k of a repair. This can be done by asking NP-questions of the form: "Does there exist a repair of size $\leq c$?". With binary search, only a logarithmic number of NP-oracle calls are required for this task. After that, we can check with another oracle call if there exists a repair $R = (A, D)$ *of size* $|A| + |D| = k$ with $\mu \in [\![Q]\!]_{G_R}$ or $\mu \notin [\![Q]\!]_{G_R}$, respectively. In total, we thus end up in Θ_2P.

Finally, for the CQA($\mathcal{L}, \subseteq, \exists$) and CQA($\mathcal{L}, \subseteq, \forall$) problems, we would still start by (1) guessing a repaired graph G_R plus supported model I. But then, in step (2), we have to additionally check that R is $\subseteq$-minimal, which requires coNP-power (i.e., there *does not exist a* $\subseteq$*-smaller repair*). In total, this gives us a Σ_2P-procedure for CQA($\mathcal{L}, \subseteq, \exists$) and a Π_2P-procedure for CQA($\mathcal{L}, \subseteq, \forall$).

As can be seen in Table 2, the *combined complexity* gives a much more varied picture. Compared with data complexity, we note that the complexity may possibly increase by up to 2 levels in the polynomial hierarchy. Moreover, the increase of complexity now also differs between brave and AR semantics. However, in some cases, the combined complexity remains the same as the data complexity. In particular, this applies to all cases of CQA($\mathcal{L}, \preceq, \exists$) with $\mathcal{L} \in \{\text{BGP}, \pi\text{-BGP}\}$ and $\preceq \in \{=, \leq, \subseteq\}$. The reason for this is that, for BGPs, no additional complexity arises anyway (this also holds for CQA(BGP, $\preceq, \forall$)). For π-BGPs, checking $\mu \in [\![Q]\!]_{G_R}$ requires an additional non-deterministic guess for the extension of μ to the bound variables. However, this additional guess does not push the complexity out of the complexity classes NP, Θ_2P, Σ_2P, and, likewise, out of Π_2P in the case of CQA(π-BGP, $\subseteq, \forall$). On the other hand, checking $\mu \in [\![Q]\!]_{G_R}$ for a WDQ Q requires an additional coNP-check (namely, that μ cannot be extended to an answer for one of the OPT parts). Hence, only in cases where a coNP-check is already needed for data complexity is there no increase of complexity for combined complexity. Most notably, this applies to CQA(π-WDQ, $\subseteq, \exists$) and CQA(WDQ, $\preceq, \forall$).

Theorem 1 below states that all membership results sketched here are indeed tight.

Theorem 1. *The following statements are true for data complexity:*

- CQA($\mathcal{L}, =, \exists$) *is* NP*-c for* $\mathcal{L} \in \{\text{BGP}, \pi\text{-BGP}, \text{WDQ},$

π-WDQ$\}$.

- CQA$(\mathcal{L}, \leq, \exists)$ *and* CQA$(\mathcal{L}, \leq, \forall)$ *are* Θ_2P*-c for* $\mathcal{L} \in \{$BGP, π-BGP, WDQ, π-WDQ$\}$.
- CQA$(\mathcal{L}, \subseteq, \exists)$ *is* Σ_2P*-c for* $\mathcal{L} \in \{$BGP, π-BGP, WDQ, π-WDQ$\}$.
- CQA$(\mathcal{L}, =, \forall)$ *is* coNP*-c for* $\mathcal{L} \in \{$BGP, π-BGP, WDQ, π-WDQ$\}$.
- CQA$(\mathcal{L}, \subseteq, \forall)$ *is* Π_2P*-c for* $\mathcal{L} \in \{$BGP, π-BGP, WDQ, π-WDQ$\}$.

The following statements are true for combined complexity:

- CQA$(\mathcal{L}, =, \exists)$ *is* NP*-c for* $\mathcal{L} \in \{$BGP, π-BGP$\}$.
- CQA$(\mathcal{L}, \leq, \exists)$ *is* Θ_2P*-c for* $\mathcal{L} \in \{$BGP, π-BGP$\}$.
- CQA$(\mathcal{L}, \subseteq, \exists)$ *is* Σ_2P*-c for* $\mathcal{L} \in \{$BGP, π-BGP, WDQ, π-WDQ$\}$.
- CQA$(\mathcal{L}, =, \forall)$ *is* coNP*-c for* $\mathcal{L} \in \{$BGP, WDQ$\}$.
- CQA$(\mathcal{L}, \leq, \forall)$ *is* Θ_2P*-c for* $\mathcal{L} \in \{$BGP, WDQ$\}$.
- CQA$(\mathcal{L}, \subseteq, \forall)$ *is* Π_2P*-c for* $\mathcal{L} \in \{$BGP, π-BGP, WDQ$\}$.

Proof sketch. We only discuss the hardness part. Our reductions are from prototypical complete problems on propositional formulas for the complexity classes NP, coNP, Θ_2P, Σ_2P, Π_2P. We thus encode the semantics of propositional logic into the constraints $\mathcal{C}$ and the concrete propositional formula φ (in 3-CNF) into the data graph G. This is done in a way such that repairs correspond to truth assignments under which φ is evaluated. As an example, consider the following constraints $\mathcal{C}$ used for the case CQA(BGP, $=, \exists$):

$$\mathsf{lit} \leftrightarrow \mathit{Lit} \wedge ((T \wedge \neg F \wedge \exists \mathit{dual}.F) \vee (F \wedge \neg T \wedge \exists \mathit{dual}.T))$$
$$\mathsf{cl} \leftrightarrow \mathit{Cl} \wedge =_3 \mathit{or}^- \wedge =_1 \mathit{and} \wedge ($$
$$(F \wedge \neg T \wedge \forall \mathit{or}^-.F) \vee (T \wedge \neg F \wedge \exists \mathit{or}^-.T))$$
$$\mathsf{phi} \leftrightarrow \mathit{Phi} \wedge ((\forall \mathit{and}^-.T \wedge T \wedge \neg F) \vee (\exists \mathit{and}^-.F \wedge F \wedge \neg T))$$
$$\mathsf{val} \leftrightarrow \forall \mathit{next}^*.(e \vee \mathsf{lit} \vee \mathsf{cl} \vee \mathsf{phi}) \wedge \exists \mathit{next}^*.e$$

The target is $\mathcal{T} = \{\mathsf{val}(s)\}$, $H = \emptyset$, $\mu = \{x \mapsto s\}$, and the query is $Q = T(x)$. The nodes of G are the literals and the clauses of φ and two auxiliary nodes s, e, where s represents φ itself. The first three constraints of $\mathcal{C}$ are respectively meant for the literals (put into the class Lit), clauses (put into the class Cl), and φ itself (put into the class Phi), ensuring that all of these are either true or false in a repaired graph G_R. I.e., they are all both in the classes T and F in G but can only remain in one of them in a repaired graph G_R. These constraints ensure that truth values are correctly propagated through Boolean expressions. Concretely, nodes of G are connected via properties $\mathit{dual}, \mathit{and}, \mathit{or}$ such that dual literals have dual truth values, clauses are true iff one of its literals is true, and φ is true iff all of its clauses are true. Lastly, the property next encodes the immediate successors and predecessors of an arbitrary linear order $\preceq_{\mathit{next}}$ that starts in node s, goes through all other nodes of G, and ends in e. This property is used in the last constraint to enforce the first three constraints on literals, clauses, and φ itself (s represents φ) without having to explicitly name them in the shapes graph. Concretely, in G_R the shape name val has to be validated at s and, thus, the last constraint ensures that every node n of G that appears after s in the order $\preceq_{\mathit{next}}$, i.e., every node of G has to either be e or validate lit, cl, or phi. The remaining parts of the constraints ensure, together with the target, that repairs do not alter the property names $\mathit{dual}, \mathit{or}, \mathit{and}, \mathit{next}$, and the class names $\mathit{Lit}, \mathit{Cl}, \mathit{Phi}$ (remove atoms over these names). The satisfiability of φ is then checked by asking whether there is a repaired graph G_R s.t. φ is in class T in G_R. Thus, we prove NP-hardness.

When we are only interested in $\leq$-repairs, we can have repairs be "penalized" for setting variables to true. Therefore, with a BGP query, we can check whether a concrete variable x is true in some *minimal* model of φ, establishing a reduction from the Θ_2P-complete problem CARDMINSAT (Creignou, Pichler, and Woltran 2018).

For $\subseteq$-repairs, we proceed one step higher up the polynomial hierarchy. We do this by splitting the variables of φ into X variables and Y variables. Now, the question is if there is a truth assignment of the X variables that can *not* be extended to a model of φ, i.e., whether $\exists X \forall Y \neg \varphi$. We thus construct $(G, \mathcal{C}, \mathcal{T}, H)$ s.t., intuitively, repairs have the choice between instantiating only the X variables or both the X and the Y variables. Crucially, when the instantiation is the same on the X variables, a repair of the second kind is then a subset of the repair of the first kind. But a repair of the second kind has to represent a model of φ. Thus, we can answer $\exists X \forall Y \neg \varphi$ by asking whether there is a $\subseteq$-repair that instantiates only the X variables. □

Next, we consider the remaining cases for brave and AR semantics. We have already seen above that when considering combined complexity, two additional sources of complexity may arise: (1) another non-deterministic guess if the query involves projection (namely, to see if an extension of μ to the bound variables exists) and (2) another coNP-check in case of WDQs (namely, to check that μ cannot be extended to one of the OPT parts). Hence, for CQA$(\mathcal{L}, =, \exists)$ and CQA$(\mathcal{L}, \leq, \exists)$ with $\mathcal{L} \in \{$WDQ, π-WDQ$\}$, the additional coNP-check increases the combined complexity to Σ_2P. For CQA$(\pi$-BGP$, \preceq, \forall)$ with $\preceq \in \{=, \leq\}$, the additional nondeterministic guess increases the complexity to Π_2P. The most dramatic increase of complexity (namely from coNP to Π_3P) happens for CQA$(\pi$-WDQ$, =, \forall)$ where the additional guess and coNP-check introduce orthogonal new sources of complexity. Of course, also for CQA$(\pi$-WDQ$, \preceq, \forall)$ with $\preceq \in \{\leq, \subseteq\}$, the complexity rises to Π_3P.

In Theorem 2 below, we again state that the above-sketched membership results are actually tight.

Theorem 2. *The following statements are true for combined complexity:*

- CQA$(\mathcal{L}, =, \exists)$ *and* CQA$(\mathcal{L}, \leq, \exists)$ *are* Σ_2P*-c for* $\mathcal{L} \in \{$WDQ, π-WDQ$\}$.
- CQA$(\pi$-BGP$, =, \forall)$ *and* CQA$(\pi$-BGP$, \leq, \forall)$ *are* Π_2P*-c.*
- CQA$(\pi$-WDQ$, \preceq, \forall)$ *is* Π_3P*-c for* $\preceq \in \{=, \leq, \subseteq\}$.

Proof sketch. For the hardness proofs, we use a Σ_kP-complete k-round coloring game played on a graph G (Ajtai, Fagin, and Stockmeyer 2000). In the game, Player 1 starts Round 1 by coloring the degree 1 vertices. Then Player 2 proceeds to color the degree 2 vertices. Then Player 1 again colors the degree 3 vertices, and so on .... In the last Round

k, the Turn Player colors all remaining vertices and wins if the final coloring is a valid 3-coloring. The question is whether Player 1 has a winning strategy.

We show the reduction for CQA(WDQ, $\leq, \exists$) and CQA(WDQ, $=, \exists$) from 2-ROUNDS-3-COLORABILITY. Let G_{col} be an arbitrary instance of the problem with n nodes. We construct a data graph G whose nodes are the vertices of G_{col} and 5 extra nodes r, g, b, s, e. For every $v \in V(G_{col})$, we add $col(v, r)$, $col(v, g)$, $col(v, b)$ to indicate that every vertex can be colored by every color. We add $neq(c, c')$ for $c, c' \in \{r, g, b\}$, $c \neq c'$ to distinguish colors; $L(u)$ to G for every leaf (degree 1 node) u of G_{col}; $I(v)$ for every non-leaf node v of G_{col}. We again encode an arbitrary linear order $\preceq_{next}$ on all the nodes of G that starts in s and ends in e. We construct $(\mathcal{C}, \mathcal{T})$, where $\mathcal{C}$ is:

$$\mathsf{leaf} \leftrightarrow L \wedge =_1 col \quad \mathsf{inner} \leftrightarrow I \wedge =_3 col \quad \mathsf{valC} \leftrightarrow =_2 neq$$
$$\mathsf{valV} \leftrightarrow \forall next^*.(s \vee e \vee \mathsf{leaf} \vee \mathsf{inner}) \wedge \exists next^*.e$$

and $\mathcal{T} = \{\mathsf{valV}(s), \mathsf{valC}(r), \mathsf{valC}(g), \mathsf{valC}(b)\}$. Moreover, $H = \emptyset$, $\mu = \{\}$, and the query is $Q = \top$ OPT P, where $P = \bigwedge_{i=1}^{n} col(v_i, x_i) \wedge \bigwedge_{(v_i, v_j) \in G_{col}} neq(x_i, x_j)$. Then, there exists a $\preceq$-repair R for $(G, \mathcal{C}, \mathcal{T}, H)$ such that $\mu \in [\![Q]\!]_{G_R}$ and $\preceq \in \{=, \leq\}$ iff there is a coloring of the leaf nodes that cannot be extended to a coloring of the whole G_{col}, i.e., Player 1 has a winning strategy. Intuitively, the constraints with the targets ensure that every repaired graph G_R provides a possible coloring of the leaves (constraint for leaf), but leaves the three colors for the inner nodes (constraint for inner). The query Q together with μ then asks whether there is no valid coloring of the rest of the nodes.

For AR semantics and π-BGPs, we can use the query $\pi_\emptyset P$. Then, μ is an answer to $\pi_\emptyset P$ over every repaired graph G_R iff there exists a coloring of the whole graph given any coloring of the leaves, i.e., Player 1 has no winning strategy. The idea is similar for AR semantics and π-WDQs. □

5 IAR Semantics

We now turn our attention to CQA under IAR semantics. For BGP, the AR- and IAR-semantics coincide as an atom α appears in every $\preceq$-repaired graph iff it appears in the intersection of all $\preceq$-repaired graphs. Thus, we start by looking at the case of π-BGP in data complexity. The natural idea seems to consist in modifying the basic guess-and-check algorithm from Section 4 by guessing the intersection $G_\cap$ of all repaired graphs in step (1) and extending step (2) by a check that $G_\cap$ is indeed the desired intersection. However, this approach introduces an additional source of complexity since we apparently need coNP-power to check that the atoms in $G_\cap$ are indeed contained in every repaired graph.

In this section, we show that we can, in fact, do significantly better. The key idea is to *guess a superset* $G'_\cap$ of the intersection $G_\cap$ of all repaired graphs. Given graph G and hypotheses H, we know that every repair (and, hence, also $G_\cap$) must be a subset of $G \cup H$. Now the crux is to guess *witnesses* for atoms that are definitely *not* in the intersection $G_\cap$. That is, for each atom $\alpha \in (G \cup H) \setminus G'_\cap$ (there are at most linearly many) guess a repair R_α with $\alpha \notin G_{R_\alpha}$. To sum up, for CQA($\pi$-BGP, $=, \cap$), we identify no-instances as follows: (1) guess a subset $G'_\cap \subseteq (G \cup H)$ together with repaired graphs G_{R_α} (and their supported models I_α) for each $\alpha \in (G \cup H) \setminus G'_\cap$ and (2) check that $\mu \notin [\![Q]\!]_{G'_\cap}$ holds; moreover, for every α, check that $\alpha \notin G_{R_\alpha}$ (and that I_α is indeed a supported model for repair R_α). By the monotonicity of π-BGPs, it does not matter if $G'_\cap$ is a strict superset of $G_\cap$. This algorithm, therefore, establishes the coNP-membership of CQA(π-BGP, $=, \cap$).

For $\subseteq$-repairs, the check in step (2) has to be extended by checking that all of the guessed repairs R_α are $\subseteq$-minimal. This only requires coNP-checks and we remain in Π_2P for CQA(π-BGP, $\subseteq, \cap$) for data and combined complexity.

Let us now consider the case of $\leq$-repairs. Recall from Section 4 that, with Θ_2P-power, we can compute the minimal cardinality k of any repair. We can then, again with Θ_2P-power, compute the exact cardinality K of the intersection $G_\cap$ of all $\leq$-repaired graphs. This can be achieved by logarithmically many oracle calls of the form: "is the size of the intersection of all $\leq$-repaired graphs less than c?" (or, equivalently, are there at least $|G \cup H|$ - c atoms in $G \cup H$ not contained in some $\leq$-repaired graph). We can then modify the guess in step (1) to guessing $G'_\cap \subseteq (G \cup H)$ with $|G'_\cap| = K$. This means that we get $G_\cap = G'_\cap$. Hence, our guess-and-check algorithm now also works for WDQs since we no longer rely on monotonicity of the query language. Moreover, for data complexity, projection does no harm and we get Θ_2P-membership of CQA($\mathcal{L}, \leq, \cap$) for all query languages considered here. For the combined complexity of CQA(π-BGP, $\leq, \cap$) or CQA(WDQ, $\leq, \cap$), we also end up in Θ_2P, since checking $\mu \in [\![Q]\!]_{G'_\cap}$ just requires yet another oracle call. By analogous considerations, we establish Θ_2P-membership also for CQA(π-BGP, $=, \cap$).

Again, we can show that all membership results sketched here are indeed tight:

Theorem 3. *The following statements are true for data complexity:*

- CQA($\mathcal{L}, =, \cap$) *is* coNP*-c for* $\mathcal{L} \in \{$BGP, π-BGP$\}$.
- CQA($\mathcal{L}, \leq, \cap$) *is* Θ_2P*-c for* $\mathcal{L} \in \{$BGP, π-BGP, WDQ, π-WDQ$\}$.
- CQA($\mathcal{L}, \subseteq, \cap$) *is* Π_2P*-c for* $\mathcal{L} \in \{$BGP, π-BGP$\}$.

The following statements are true for combined complexity:

- CQA(BGP, $=, \cap$) *is* coNP*-c.*
- CQA(π-BGP, $=, \cap$) *is* Θ_2P*-c.*
- CQA($\mathcal{L}, \leq, \cap$) *is* Θ_2P*-c for* $\mathcal{L} \in \{$BGP, π-BGP, WDQ$\}$.
- CQA($\mathcal{L}, \subseteq, \cap$) *is* Π_2P*-c for* $\mathcal{L} \in \{$BGP, π-BGP$\}$.

Proof sketch. The only remaining case is the Θ_2P-hardness of CQA(π-BGP, $=, \cap$). To that end, we reduce from the problem CARDMIN-PRECOLORING that asks whether for a graph G' and *pre-coloring* (i.e., an assignment of admissible colors for each vertex) $c : V(G') \to 2^{\{r,g,b\}}$, a given vertex $v_1 \in V(G') = \{v_1, \ldots, v_n\}$ is colored g in some 3-coloring of G' that minimizes the use of the color g.

The bulk of the reduction, i.e., the definition of the data graph and the shapes graph is very technical and therefore omitted here. But the intuition behind it is quite simple. The idea is to split the "computation" into two steps. The first

step is to compute the minimum number of vertices that need to be colored g and this is handled by the (intersection of the) repaired graphs. The second step is to try to compute such a minimal coloring while assigning color g to vertex v_1. This is handled by the (boolean) query $\pi_\emptyset Q$ where Q is

$$\bigwedge_{(v_i,v_j)\in G'} neqCol(y_i,y_j) \wedge Pre_{\{g\}}(y_1) \wedge \bigwedge_i Pre_{c(v_i)}(y_i)$$
$$\wedge \bigwedge_i allowed(x_i,y_i) \wedge \bigwedge_{i\neq j} neqCnt(x_i,x_j).$$

In this query, the variables y_i represent colors of the vertices of the graph G' (concretely, y_i corresponds to v_i) with the classes Pre_C ensuring that the coloring adheres to the given pre-coloring and, additionally, v_1 is colored g. The property $neqCol$ in the query then ensures that adjacent vertices are assigned different colors. Thus, in total, the first line of the query ensures that we are dealing with a valid 3-coloring of G' and v_1 is colored g. This part is more or less unimpacted by repairs. However, the repairs impact the $allowed$ property in an important way, thus restricting the number of vertices that are "allowed" to be colored g (the colors r, b are always "allowed"). Concretely, there are n possible values for the variables x_i and all have to be different due to $neqCnt$. These values are the counters $i_1, \dots, i_n$ and $allowed(i_j, g)$ is in the intersection of all repaired graphs iff g has to be used $\geq j$-times. Let k be such that at least k vertices have to be colored g. Then, to satisfy the last line of the query, the vertices colored g by the variables y_i have to be matched to $i_1, \dots, i_k$, ensuring that there are at most k vertices colored g. Thus, looking at the whole query, a valid instantiation of the X and Y variables constitutes a minimal 3-coloring that assigns color g to vertex v_1. □

Next, we take a look at the remaining cases for WDQs. Recall from Section 2 that we may assume that a WDQ Q is of the form $((\dots((P \text{ OPT } P_1) \text{ OPT } P_2)\dots) \text{ OPT } P_k)$ with $\mathsf{vars}(P) = \mathsf{dom}(\mu)$ (we are considering WDQs without projection here) and each of the subqueries P_i contains at least one free variable outside $\mathsf{vars}(P)$.

Then μ is an answer to Q if and only if μ is an answer to P but, for all i, μ *cannot* be extended to an answer of P_i. Checking if μ is an answer to P corresponds to identifying yes-instances of CQA(BGP, $=, \cap$); checking that μ cannot be extended to an answer of any P_i corresponds to identifying no-instances of CQA(π-BGP, $=, \cap$) (note that the variables in $\mathsf{vars}(P_i) \setminus \mathsf{vars}(P)$ behave like bound variables in this case, since we are not interested in a particular extension of μ to these variables but in *any* extension). The problem CQA(WDQ, $\preceq, \cap$) can, therefore, be seen as the intersection of CQA(BGP, $\preceq, \cap$) and (multiple) CQA(π-BGP, $\preceq, \cap$). This proves the DP- and Θ_2P-membership in case of $=$, respectively for data and combined complexity, and the DP_2-membership in case of $\subseteq$ for both settings. We thus get the following complexity classification for WDQs.

Theorem 4. *The following statements are true for data complexity:*

- CQA(WDQ, $=, \cap$) *is* DP*-c.*
- CQA(WDQ, $\subseteq, \cap$) *is* DP_2*-c.*

The following statements are true for combined complexity:

- CQA(WDQ, $=, \cap$) *is* Θ_2P*-c.*
- CQA(WDQ, $\subseteq, \cap$) *is* DP_2*-c.*

Proof sketch. The hardness part follows a similar idea as the membership: We reduce from instance pairs $(\mathcal{I}, \mathcal{J})$, where $\mathcal{I}$ comes from CQA(BGP, $\preceq, \cap$) and $\mathcal{J}$ comes from CQA(π-BGP, $\preceq, \cap$). For the reduction, we simply merge (take the union of) the data graphs, the hypotheses, the constraints, and the targets. Then, the intersection $G_\cap$ of all $\preceq$-repaired graphs (of the new instance) is exactly the union of the intersections $G_{\mathcal{I},\cap}$ and $G_{\mathcal{J},\cap}$ (of the old instances $\mathcal{I}$ and $\mathcal{J}$). Furthermore, the two mappings $\mu_\mathcal{I}, \mu_\mathcal{J}$ are merged to μ, while the queries $Q_\mathcal{I} = P_1$ and $Q_\mathcal{J} = \pi_Y P_2$ are combined into $Q = (P_1 \wedge \top(Y))\,\text{OPT}\,P_2$. Here, $\top(Y)$ is a shorthand for a conjunction that allows instantiating the variables in Y to any node of $\mathcal{J}$. Then, for μ to be an answer to Q over $G_\cap$, the part $\mu_\mathcal{I}$ has to be an answer of P_1 over the part $G_{\mathcal{I},\cap}$ while $\mu_\mathcal{J}$ cannot be extended to an answer of P_2 over the part $G_{\mathcal{J},\cap}$. □

We now consider the last remaining cases for π-WDQ. As we have already discussed above, Θ_2P-power suffices to compute the size K of the intersection of all ($\leq$-)repaired graphs. For $\subseteq$-minimality, Θ_3P-power is needed to compute K by analogous arguments and the fact that checking $\subseteq$-minimality only requires access to an NP-oracle. For data complexity, given K, we can then simply "guess" the intersection $G_\cap$ and answer the queries in polynomial time. Hence, we end up in Θ_2P for CQA(π-WDQ, $=, \cap$) and in Θ_3P for CQA(π-WDQ, $\subseteq, \cap$). For combined complexity, we have to take the Σ_2P-completeness of evaluating π-WDQs into account. This fits into the Θ_3P-bound in the case of CQA(π-WDQ, $\subseteq, \cap$) but leads to Σ_2P-completeness of CQA(π-WDQ, $=, \cap$).

Theorem 5. *The following statements are true for data complexity:*

- CQA(π-WDQ, $=, \cap$) *is* Θ_2P*-c.*
- CQA(π-WDQ, $\subseteq, \cap$) *is* Θ_3P*-c.*

The following statements are true for combined complexity:

- CQA(π-WDQ, $=, \cap$), CQA(π-WDQ, $\leq, \cap$) *are* Σ_2P*-c.*
- CQA(π-WDQ, $\subseteq, \cap$) *is* Θ_3P*-c.*

Proof sketch. The cases CQA(π-WDQ, $=, \cap$) and CQA(π-WDQ, $\leq, \cap$) immediately follow from the hardness of answering π-WDQ in combined complexity.

For CQA(π-WDQ, $=, \cap$), as in the proof of Theorem 1, we encode the semantics of propositional logic into the constraints $\mathcal{C}$ and the concrete propositional formulas φ into the data graph G. However, this time, we do not simply encode a single propositional formula into our instance but a list $L = (\varphi_1, \psi_1), \dots, (\varphi_n, \psi_n)$ of pairs of propositional formulas. L is a yes-instance if there is a pair (φ_i, ψ_i) such that φ_i is unsatisfiable while ψ_i is satisfiable. This constitutes a natural Θ_2P-complete problem.

Intuitively, the constructed instance is such that the repairs have the (independent) choice of what truth values to assign to each variable appearing in any formula. The truth values

of the formulas themselves are then functionally determined by this choice. Thus, $F(\varphi_i)$ (resp. $F(\psi_i)$) is in the repaired graph if and only if φ_i (resp. ψ_i) evaluates to false under the given choice of the truth values for the variables in φ_i (resp. ψ_i). Since each truth value assignment constitutes a repair, $F(\varphi_i)$ (resp. $F(\psi_i)$) appears in the intersection of all repaired graphs if and only if φ_i (resp. ψ_i) is unsatisfiable. Furthermore, formulas φ_i (resp. ψ_i) are connected to their indices via $is_\varphi(\mathtt{i}_i, \varphi_i)$ (resp. $is_\psi(\mathtt{i}_i, \psi_i)$). Thus, the empty mapping $\mu = \{\,\}$ is an answer to the query $\pi_z Q$ with

$$Q = is_\varphi(x, y) \wedge F(y) \text{ OPT } is_\psi(x, z) \wedge F(z)$$

if and only if there is an $\mathtt{i}_i$ (variable x) such that φ_i (variable y) is unsatisfiable but ψ_i (variable z) is not unsatisfiable, i.e., ψ_i is satisfiable.

For CQA(π-WDQ, $\subseteq, \cap$), we have to combine the ideas of the previous case CQA(π-WDQ, $=, \cap$) with the ideas of the case CQA(BGP, $\subseteq, \exists$). That is, we split the variables of each formula φ into X variables and Y variables and then interpret the list as consisting of QBFs $\forall X \exists Y \varphi$. Then, similar to CQA(BGP, $\subseteq, \exists$), $\subseteq$-repairs "know" whether a concrete truth assignment of X can be extended to a model of φ and, the intersection "knows" whether this is the case for all truth assignments of X. Thus, a similar query as before is enough to identify if there is a (φ_i, ψ_i) such that $\forall X_i \exists Y_i \varphi_i$ and $\neg \forall X_i' \exists Y_i' \psi_i$ hold. □

6 Max-Variants of CQA

We now consider CQA for settings where the existence of a repair is not guaranteed. Following (Ahmetaj et al. 2022), we relax the notion of repairs and aim at satisfying the maximum number of targets. Formally, given a graph G, a shapes graph $(\mathcal{C}, \mathcal{T})$ and a set of hypotheses H, we define *max-repairs* for $(G, \mathcal{C}, \mathcal{T}, H)$ to be pairs (A, D) satisfying (1) $A \subseteq H$, $D \subseteq G$, (2) there exists $\mathcal{T}' \subseteq \mathcal{T}$ such that (A, D) is a repair for $(G, \mathcal{C}, \mathcal{T}', H)$, and (3) for every $\mathcal{T}'' \subseteq \mathcal{T}$ with $|\mathcal{T}''| > |\mathcal{T}'|$, there exists no repair for $(G, \mathcal{C}, \mathcal{T}'', H)$. We thus consider the max-variant MCQA($\mathcal{L}, \preceq, \mathcal{S}$) of the CQA problem, where max-repairs play the role of repairs for given query language $\mathcal{L}$, pre-order $\preceq$ on the repairs, and semantics $\mathcal{S} \in \{\exists, \forall, \cap\}$.

Note that, in our hardness proofs for the CQA problem, we can always choose the target problem instance consisting of a graph G, shapes graph $(\mathcal{C}, \mathcal{T})$, and set of hypotheses H in such a way that a repair exists. In this case, the CQA and MCQA problems coincide. Hence, the hardness results of each variant of the CQA problem carry over to the MCQA problem. Therefore, whenever we manage to show that the membership result for a variant of CQA also holds for the corresponding variant of MCQA, we may immediately conclude the completeness for this variant of MCQA.

The algorithms sketched in Section 4 and 5 to illustrate our membership results for all variants of the CQA problem now have to be extended by ensuring maximum cardinality of the targets covered by the repaired graphs that are guessed in step (1) of our algorithms. In case of data complexity, we thus have to combine constantly many NP-problems deciding the question if there exists a repair that covers a subset $\mathcal{T}' \subseteq \mathcal{T}$ of cardinality c. This requires the power of some level in the Boolean Hierarchy BH. In case of combined complexity, the maximally attainable cardinality of subsets $\mathcal{T}' \subseteq \mathcal{T}$ allowing for a repair has to be determined by binary search with logarithmically many NP-oracle calls of the form "does there exist $\mathcal{T}' \subseteq \mathcal{T}$ with $|\mathcal{T}'| \geq c$ such that $\mathcal{T}'$ allows for a repair?", which requires Θ_2-power.

Hence, for all variants of CQA whose data complexity is at least hard for every level of BH or whose combined complexity is at least Θ_2P-hard, respectively, the membership and, therefore, also the completeness carries over to MCQA. Moreover, the data (resp. combined) complexity of the remaining cases is never lifted beyond BH (resp. Θ_2P).

In Theorem 6 below, we state that the BH-membership results for the data complexity are actually tight by showing BH_k-hardness for every level $k > 0$ of BH.

Theorem 6. MCQA(BGP, $=, \mathcal{S}$) *is* BH_k*-hard data complexity for every* $k > 0$ *and* $\mathcal{S} \in \{\exists, \forall, \cap\}$. *Furthermore,* MCQA($\pi$-WDQ, $=, \exists$)*,* MCQA(π-WDQ, $=, \forall$)*, and* MCQA(WDQ, $=, \cap$) *are in* BH *data complexity.*

Proof sketch. We proceed similarly to the case CQA(π-WDQ, $=, \cap$). That is, we encode the semantics of a list of propositional formulas $(\varphi_1, \psi_1), \ldots, (\varphi_k, \psi_k)$ into the constraints $\mathcal{C}$ and the formulas themselves into the data graph G. However, this time, the length of the list is fixed to k and thus, it becomes BH_{2k}-complete to decide if there is a pair (φ_i, ψ_i) such that φ_i is unsatisfiable while ψ_i is satisfiable.

We construct an instance of MCQA with targets $\mathcal{T}_1$ that intuitively require repaired graphs to simulate the semantics of propositional formulas (similar to the proof of Theorem 1), and targets $\mathcal{T}_2$ that require the formulas to be satisfied. The targets $\mathcal{T}_1$ are chosen in such a way that (1) either all are satisfied or none and (2) there always exist repairs that validate $\mathcal{T}_1$. Thus, by choosing $|\mathcal{T}_1| > |\mathcal{T}_2|$, staying true to the semantics of propositional formulas is a "hard" constraints. On the other hand, $\mathcal{T}_2$ is such that, for each of the formulas φ_i and ψ_i, $\mathcal{T}_2$ contains a target that is validated when the corresponding formula evaluates to true. These constitute "soft" constraints, and max-repairs try to satisfy as many formulas as possible. Therefore, a max-repair can "decide" whether there exists a pair (φ_i, ψ_i) as desired. In particular, a formula φ_i is definitely unsatisfiable, if the corresponding target in $\mathcal{T}_2$ is not validated by a max-repair. □

In Theorem 7 below, we state that, for combined complexity, the Θ_2P-membership results are actually tight.

Theorem 7. MCQA(BGP, $=, \mathcal{S}$) *is* Θ_2P*-hard combined complexity for* $\mathcal{S} \in \{\exists, \forall, \cap\}$. *Furthermore,* MCQA($\pi$-BGP, $=, \exists$)*,* MCQA(WDQ, $=, \forall$)*, and* MCQA(BGP, $=, \cap$) *are in* Θ_2P *combined complexity.*

Proof sketch. We again reduce from the Θ_2P-complete problem CARDMINSAT. That is, given a propositional formula φ in 3-CNF, the question is whether the variable x_1 is true in some model of φ that minimizes the number of variables set to true.

We construct an instance of MCQA with multiple types of targets $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4$. The targets $\mathcal{T}_1$ are essentially the same as before, ensuring that max-repaired graphs simulate

the semantics of propositional logic. $\mathcal{T}_2$ asks for φ to be satisfied. $\mathcal{T}_3$ is such that, for every variable x_i, $\mathcal{T}_3$ contains two targets $t(x_i), t'(x_i)$ which are validated when x_i is set to false. Lastly, $\mathcal{T}_4$ consists of a single target that asks for x_1 to be set to true. By setting $|\mathcal{T}_2| > |\mathcal{T}_3|+|\mathcal{T}_4|$ we ensure that, if φ is satisfiable, then all max-repairs will represent models of φ. Thus, max-repairs set both φ and x_1 to true if and only if there is a minimal model of φ in which x_1 is true. □

As we have already discussed at the beginning of this section, the complexity of all other cases remains the same.

Theorem 8. *For all cases* $\text{MCQA}(\mathcal{L}, \preceq, \mathcal{S})$ *not mentioned in Theorem 6 or 7,* $\text{MCQA}(\mathcal{L}, \preceq, \mathcal{S})$ *is* C-*c if and only if* $\text{CQA}(\mathcal{L}, \preceq, \mathcal{S})$ *is* C-*c. This holds for data and combined complexity.*

7 Conclusion and Future Work

In this work, we have carried out a thorough complexity analysis of the CQA problem for data graphs with SHACL constraints and we have pinpointed the complexity of this problem in a multitude of settings – considering various query languages, inconsistency-tolerant semantics of CQA, and preference relations on repairs. Several new proof techniques had to be developed to obtain these results. For instance, this has allowed us – in contrast to (Ahmetaj et al. 2021) – to prove all our hardness results without making use of recursion in shapes constraints. Hence, by carrying over our techniques to the problems studied in (Ahmetaj et al. 2021), we could extend the hardness to the non-recursive cases left open there.

The targets we considered here are shape atoms of the form $\mathsf{s}(c)$. The SHACL standard also allows for richer targets over class and property names. Specifically, it allows to state that a data graph must validate a shape name at each node of a certain class name, or domain (or range) of a property name. All the membership results in this paper can be immediately updated to support these richer targets. Some features of SHACL (such as disjointness and closed constraints) are not considered here, but we strongly believe they do not change the complexity results.

We considered the supported model semantics. Other validation semantics, which are inspired from the stable model semantics (Andresel et al. 2020) or the well-founded semantics (Chmurovic and Simkus 2022) for logic programs with negation are left for future work. Again, we do not expect changes of the complexity: all our hardness proofs should hold since the three semantics coincide for non-recursive SHACL, and the membership results can be immediately carried over since they rely on the problem of validation, which is not harder than for the supported model semantics.

An immediate direction for future work is to investigate the notion of optimal repairs of prioritized data graphs over SHACL constraints and specifically, the notions of global, Pareto and completion optimality of repairs (Staworko, Chomicki, and Marcinkowski 2012). In particular, it would be of interest to study the computational properties of the main reasoning tasks such as repair checking and repair existence, and to analyze CQA for each of the three notions of optimal repairs and the various settings studied in this paper. Devising practical algorithms for CQA over SHACL constraints and in particular, identifying meaningful and relevant fragments of SHACL that admit better complexity results is also an important next step.

Acknowledgements

This work was supported by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT2201, 10.47379/VRG18013]. In addition, Ahmetaj was supported by the FWF and netidee SCIENCE project T1349-N.

References

Afrati, F. N., and Kolaitis, P. G. 2009. Repair checking in inconsistent databases: algorithms and complexity. In Fagin, R., ed., *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, volume 361 of *ACM International Conference Proceeding Series*, 31–41. ACM.

Ahmetaj, S.; David, R.; Ortiz, M.; Polleres, A.; Shehu, B.; and Simkus, M. 2021. Reasoning about explanations for non-validation in SHACL. In Bienvenu, M.; Lakemeyer, G.; and Erdem, E., eds., *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, 12–21.

Ahmetaj, S.; David, R.; Polleres, A.; and Simkus, M. 2022. Repairing SHACL constraint violations using answer set programming. In Sattler, U.; Hogan, A.; Keet, C. M.; Presutti, V.; Almeida, J. P. A.; Takeda, H.; Monnin, P.; Pirrò, G.; and d'Amato, C., eds., *The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Virtual Event, October 23-27, 2022, Proceedings*, volume 13489 of *Lecture Notes in Computer Science*, 375–391. Springer.

Ahmetaj, S.; Ortiz, M.; Oudshoorn, A.; and Simkus, M. 2023. Reconciling SHACL and ontologies: Semantics and validation via rewriting. In *ECAI 2023 - 26th European Conference on Artificial Intelligence*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, 27–35. IOS Press.

Ahmetaj, S.; Merkl, T. C.; and Pichler, R. 2024. Consistent query answering over SHACL constraints. *CoRR* abs/2406.16653.

Ajtai, M.; Fagin, R.; and Stockmeyer, L. J. 2000. The closure of monadic NP. *J. Comput. Syst. Sci.* 60(3):660–716.

Andresel, M.; Corman, J.; Ortiz, M.; Reutter, J. L.; Savkovic, O.; and Simkus, M. 2020. Stable model semantics for recursive SHACL. In Huang, Y.; King, I.; Liu, T.; and van Steen, M., eds., *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, 1570–1580. ACM / IW3C2.

Arenas, M.; Bertossi, L. E.; and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In Vianu, V., and Papadimitriou, C. H., eds., *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, 68–79. ACM Press.

Arming, S.; Pichler, R.; and Sallinger, E. 2016. Complexity of repair checking and consistent query answering.

In Martens, W., and Zeume, T., eds., *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016*, volume 48 of *LIPIcs*, 21:1–21:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

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

Bienvenu, M., and Bourgaux, C. 2016. Inconsistency-tolerant querying of description logic knowledge bases. In Pan, J. Z.; Calvanese, D.; Eiter, T.; Horrocks, I.; Kifer, M.; Lin, F.; and Zhao, Y., eds., *Reasoning Web, Tutorial Lectures*, volume 9885 of *Lecture Notes in Computer Science*, 156–202. Springer.

Bienvenu, M., and Rosati, R. 2013. Tractable approximations of consistent query answering for robust ontology-based data access. In Rossi, F., ed., *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 775–781. IJCAI/AAAI.

Bienvenu, M.; Bourgaux, C.; and Goasdoué, F. 2019. Computing and explaining query answers over inconsistent dl-lite knowledge bases. *J. Artif. Intell. Res.* 64:563–644.

Bienvenu, M. 2012. On the complexity of consistent query answering in the presence of simple ontologies. In Hoffmann, J., and Selman, B., eds., *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*, 705–711. AAAI Press.

Bogaerts, B.; Jakubowski, M.; and den Bussche, J. V. 2022. SHACL: A description logic in disguise. In Gottlob, G.; Inclezan, D.; and Maratea, M., eds., *Logic Programming and Nonmonotonic Reasoning - 16th International Conference, LPNMR 2022, Genova, Italy, September 5-9, 2022, Proceedings*, volume 13416 of *Lecture Notes in Computer Science*, 75–88. Springer.

Chmurovic, A., and Simkus, M. 2022. Well-founded semantics for recursive SHACL. In Alviano, M., and Pieris, A., eds., *Proceedings of (Datalog-2.0 2022)*, volume 3203 of *CEUR Workshop Proceedings*, 2–13. CEUR-WS.org.

Corman, J.; Reutter, J. L.; and Savkovic, O. 2018. Semantics and validation of recursive SHACL. In Vrandecic, D.; Bontcheva, K.; Suárez-Figueroa, M. C.; Presutti, V.; Celino, I.; Sabou, M.; Kaffee, L.; and Simperl, E., eds., *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, volume 11136 of *Lecture Notes in Computer Science*, 318–336. Springer.

Creignou, N.; Pichler, R.; and Woltran, S. 2018. Do hard sat-related reasoning tasks become easier in the Krom fragment? *Log. Methods Comput. Sci.* 14(4).

Du, J.; Qi, G.; and Shen, Y. 2013. Weight-based consistent query answering over inconsistent $${\mathcal {SHIQ}}$$ knowledge bases. *Knowl. Inf. Syst.* 34(2):335–371.

Khalfioui, A. A. E., and Wijsen, J. 2023. Consistent query answering for primary keys and conjunctive queries with counting. In Geerts, F., and Vandevoort, B., eds., *26th International Conference on Database Theory, ICDT 2023, March 28-31, 2023, Ioannina, Greece*, volume 255 of *LIPIcs*, 23:1–23:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Knublauch, H., and Kontokostas, D. 2017. Shapes constraint language (SHACL). W3C Recommendation, W3C. https://www.w3.org/TR/shacl/.

Koutris, P., and Wijsen, J. 2021. Consistent query answering for primary keys in datalog. *Theory Comput. Syst.* 65(1):122–178.

Koutris, P.; Ouyang, X.; and Wijsen, J. 2021. Consistent query answering for primary keys on path queries. In Libkin, L.; Pichler, R.; and Guagliardo, P., eds., *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, 215–232. ACM.

Koutris, P.; Ouyang, X.; and Wijsen, J. 2024. Consistent query answering for primary keys on rooted tree queries. *Proc. ACM Manag. Data* 2(2):76.

Leinberger, M.; Seifer, P.; Rienstra, T.; Lämmel, R.; and Staab, S. 2020. Deciding SHACL shape containment through description logics reasoning. In Pan, J. Z.; Tamma, V. A. M.; d'Amato, C.; Janowicz, K.; Fu, B.; Polleres, A.; Seneviratne, O.; and Kagal, L., eds., *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I*, volume 12506 of *Lecture Notes in Computer Science*, 366–383. Springer.

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2010. Inconsistency-tolerant semantics for description logics. In Hitzler, P., and Lukasiewicz, T., eds., *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings*, volume 6333 of *Lecture Notes in Computer Science*, 103–117. Springer.

Lembo, D.; Lenzerini, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2015. Inconsistency-tolerant query answering in ontology-based data access. *J. Web Semant.* 33:3–29.

Letelier, A.; Pérez, J.; Pichler, R.; and Skritek, S. 2013. Static analysis and optimization of semantic web queries. *ACM Trans. Database Syst.* 38(4):25.

Lutz, C.; Seylan, I.; and Wolter, F. 2013. Ontology-based data access with closed predicates is inherently intractable(sometimes). In Rossi, F., ed., *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 1024–1030. IJCAI/AAAI.

Ortiz, M. 2023. A short introduction to SHACL for logicians. In Hansen, H. H.; Scedrov, A.; and de Queiroz, R. J. G. B., eds., *Logic, Language, Information, and Computation - 29th International Workshop, WoLLIC 2023, Halifax, NS, Canada, July 11-14, 2023, Proceedings*, volume 13923 of *Lecture Notes in Computer Science*, 19–32. Springer.

Pérez, J.; Arenas, M.; and Gutierrez, C. 2009. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.* 34(3):16:1–16:45.

Staworko, S.; Chomicki, J.; and Marcinkowski, J. 2012.

Prioritized repairing and consistent query answering in relational databases. *Ann. Math. Artif. Intell.* 64(2-3):209–246.

ten Cate, B.; Fontaine, G.; and Kolaitis, P. G. 2015. On the data complexity of consistent query answering. *Theory Comput. Syst.* 57(4):843–891.

W3C. 2013. RDF Validation Workshop Report: Practical Assurances for Quality RDF Data. `http://www.w3.org/2012/12/rdf-val/report`.

Wijsen, J. 2019. Foundations of query answering on inconsistent databases. *SIGMOD Rec.* 48(3):6–16.