Faculty of Computer Science, Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
Tel: +39 04710 16000, fax: +39 04710 16009

KRDB Research Centre Technical Report:

SHACL Constraint Validation over Ontology-enhanced KGs via Rewriting [Extended Version]

Ognjen Savković[1], Evgeny Kharlamov[2], Steffen Lamparter [3]

Affiliation	1: Free University of Bozen-Bolzano, Bolzano, Italy 2: University of Oslo, Oslo, Norway 3: Siemens CT, Siemens AG, Munich, Germany
Corresponding author	Ognjen Savković: ognjen.savkovic@unibz.it
Keywords	SHACL, Ontologies RDF Graph Validation, Graph Constraints, RDF, DL-Lite
Number	KRDB18-03
Date	December, 2018
URL	http://www.inf.unibz.it/krdb/

Acknowledgments

This work has been partially funded by the FUB projects QUEST and OBATS.

SHACL Constraint Validation over Ontology-enhanced KGs via Rewriting

Ognjen Savković[1], Evgeny Kharlamov[2], and Steffen Lamparter[3]

[1] Free University of Bozen-Bolzano, Bolzano, Italy
[2] University of Oslo, Oslo, Norway
[3] Siemens CT, Siemens AG, Munich, Germany

Abstract. Constraints have traditionally been used to ensure data quality. Recently, several constraint languages, among which the W3C recommendation SHACL, have been proposed for Knowledge Graphs (KGs), together with validation mechanisms. An important feature of KGs is that they are often enhanced with ontologies that define relevant background knowledge in a formal language such as OWL 2 QL. At the same time, existing systems for constraint validation either ignore these ontologies, or compile ontologies and constraints into rules that should be executed by some rule engine. In the latter case, one has to rely on different systems when validating constrains over KGs and over ontology-enhanced KGs. In this work, we address this problem by defining rewriting techniques that allow to compile an OWL 2 QL ontology and a set of SHACL constraints into another set of SHACL constraints. In particular, we show that in the general case of OWL 2 QL and SHACL the rewriting may not be possible, and establish restrictions to these languages for which rewriting always exists and also establish complexity bounds as well as identify tractable cases. Our rewriting techniques allow to validate constraints over KGs with and without ontologies using the same SHACL validation engines.

1 Introduction

Constraints has traditionally been used to ensure quality of data in relational [5] and semi-structured databases [4]. Recently constraints have attracted a considerable attention in the context of graph data [18, 17], and in particular for *Knowledge graphs* (KGs) (e.g, [27, 30, 25]), that is, large collections of interconnected entities that are annotated with data values and types [6]. KGs have become powerful assets for enhancing search and data integration and they are now widely used in both academia and industry [29, 1, 2, 21, 22]. Prominent examples of constraint languages for KGs include SHACL [23][4], ShEx[5]; examples of constraint validation systems include Stardog[6] and TopBraid[7].

KGs are often enhanced with OWL 2 *ontologies* [3] that capture the relevant domain background knowledge with axioms over the terms from the KGs vocabulary e.g., by assigning attributes to classes, defining relationships between classes, composing classes,

[4] `https://www.w3.org/TR/shacl/`
[5] `https://www.w3.org/2001/sw/wiki/ShEx`
[6] `https://www.stardog.com/`
[7] `https://www.topquadrant.com/technology/shacl/`

class hierarchies, etc. We will refer to ontology enhanced KGs as *Knowledge Bases*, or *KBs*. Ontologies significantly impact constraint validation over KGs. Indeed, constraints over KGs have Closed-World semantics, or Assumption (CWA) in the sense that their validation over a KG boils down to checking whether sub structures of the KG comply with the patterns encoded in the constraints [9, 16, 12]. On the other hand, KBs have open-world semantics (OWA) in the sense that ontologies allow to derive information from a KG that is not explicitly there.

As a result, constraint validation over KGs in the presence of ontologies requires to bridge the CWA of constraints and OWA of ontologies [25, 30, 20]. A promising semantics that offers such bridge was proposed in [25]: given a set of constraints $\mathcal{C}$, an ontology $\mathcal{O}$, and a KG $\mathcal{G}$, validating the KB $\langle \mathcal{O}, \mathcal{G} \rangle$ against $\mathcal{C}$ requires validating all first-order logic models of $\mathcal{O}$ and $\mathcal{G}$ that are set-inclusion minimal against $\mathcal{C}$. In practice this can be implemented via a rewriting mechanism: in order to validate $\langle \mathcal{O}, \mathcal{G} \rangle$ against $\mathcal{C}$, one can compile $\mathcal{O}$ and $\mathcal{C}$ into a (possibly disjunctive) logic program and then evaluate the program over $\mathcal{G}$ [25, 20]. A disadvantage of such approach is that constraint validation in presence of ontologies requires a different evaluation engine than in their absence: it requires an engine for disjunctive logic programs, rather than an engine for validating graph constraints. It is preferable to have a mechanism that allows to evaluate constraints over KBs using the same engine as over KGs.

In this work we address this issue. We first formally formulate the problem of constraints rewriting over ontologies: we require that the result of rewriting is again a set of constraints $\mathcal{C}'$ in the same formalism as the original $\mathcal{C}$. We then study the existence of such a rewriting function for the constraint language SHACL and OWL 2 QL, commonly used profile of OWL 2. Our results show that rewriting may not exist in the general case unless CO-NP $=$ NP, since constraint validation in presence of ontologies is CO-NP-complete, while in absence it is NP-complete.

We next turn our attention to restrictions, on the one hand, of the SHACL language and, on the other hand, of OWL QL. We do this by observing that a source of non-existence of the rewiring function comes from the combination of two factors: *(1)* the capacity of OWL QL to express the existence of entities that are not explicitly mentioned in a knowledge graph, and *(2)* the capacity of SHACL to express negation. We then study how elimination of the first factor for OWL 2 QL or the second one for SHACL solves the rewriting problem and propose a corresponding rewriting algorithms.[8] Moreover, we show that if both factors are eliminated for OWL 2 QL, then constraint validation in presence of constraints can be performed in polynomial time, and show both a PTIME rewriting algorithm as well as a PTIME constraint validation algorithm.

The paper is organised as follows: in Section 2 we give preliminaries. In Section 3 we introduce our notion of constrain rewriting over ontologies, and show negative results for SHACL and OWL 2 QL. In Section 4 we present our rewriting algorithms for restricted SHACL constrains and restricted OWL 2 QL ontologies. In Section 5 we put all complexity results together. In Section 6 we discuss related work and extensions our results to other ontology languages. In Section 7 we conclude.

[8] We note that elimination of the first factor brings OWL 2 QL close to the W3C recommended ontology language RDFS [6] and thus the resulting language is still useful in practice.

An extended version of the paper with detailed proofs can be found on-line at: `https://www.inf.unibz.it/krdb/KRDB%20files/tech-reports/KRDB18-03.pdf`.

2 Preliminaries and Running Example

In this section we recall required definitions. We assume a signature Σ that consists of three infinite countable sets of *constants*, that correspond to entities, *classes* of unary predicates, that correspond to types, and *properties* or binary predicates that correspond to object properties or a special predicate "a" that essentially labels entities with classes. Note that we consider neither datatypes nor data properties in this work and leave them for the future study. We also consider an infinite countable *domain* Δ of entities.

2.1 Knowledge Graph

A Knowledge Graph (KG) $\mathcal{G}$ in our work is a possibly infinite directed label graph that consists of triples of the form (s, p, o) over Σ, where s is a constant, p is a property, and o is either a constant or a class and only in the latter case p is the special predicate "a".

Example 1. Consider the following fragment of the Siemens KG $\mathcal{G}_{SIEM}$ from [22], which describes Siemens industrial assets including two turbines with the identifiers `:t177` and `:t852` and one power plant (`PPlant`) with the identifier `:p063`, as well as information about equipment (turbine) categories (`hasTuCat`, `hasCat`), their deployment sites (`deplAt`), and enumeration of turbines at plants (`hasTurb`):

$\{$(`:p063, a, :PPlant`), (`:p063, :hasTurb, :t852`),
(`:t852, a, :Turbine`), (`:t852, :deplAt, :p063`), (`:t852, :hasCat, :SGT-800`),
(`:t177, :deplAt, :p063`), (`:t177, :hasTuCat, :SGT-800`), $\}$. □

2.2 SHACL Syntax

We next briefly recall relevant notions of SHACL using a compact syntax of [12] which is equivalent to SHACL's "Core Constraint Components" [12]. SHACL stands for *Shapes Constraint Language*. Each SHACL constraint in a set of constraints $\mathcal{C}$, usually referred to as *shape*, is defined as a triple: $\langle s, \tau_s, \phi_s \rangle$, where

- s is the *name*,
- τ_s is the *target definition*, a SPARQL query with one output variable whose purpose is to retrieve *target entities* of s from $\mathcal{G}$, i.e., entities (nodes) occurring in $\mathcal{G}$ for which the following constraint of the shape should be verified,
- and ϕ_s is the *constraint*, an expression defined according to the following grammar:

$$\phi \ ::= \ \top \mid s' \mid c \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \ \geq_n r.\phi \mid \mathrm{EQ}(r_1, r_2), \tag{1}$$

where $\top$ stands for the Boolean truth values, s' is a shape name occurring in $\mathcal{C}$, c is a constant, r is a SPARQL property path, and $n \in \mathbb{N}^+$; moreover, $\wedge$ denotes the conjunction, $\neg$ – negation, "$\geq_n r.\phi$"–"must have at least n-successors in $\mathcal{G}$ verifying ϕ", and "EQ(r_1, r_2)"–"r_1 and r_2-successors of a node must coincide".[9]

[9] One may also use $\vee$ and $\leq_n r.\phi$ as syntactic sugar, with their expected meaning.

With a slight abuse of notation we identify the shape with its name. We note that the syntax for constraints allows for shapes to reference each other. We call a set of constraints *recursive* if it contains a shape that reference itself, either directly or via a reference cycle.

Example 2. Consider $\mathcal{C}_{SIEM} = \{\langle s_i, \tau_{s_i}, \phi_{s_i}\rangle \mid i = 1, 4\}$, where:

$$\begin{aligned}
\tau_{s_1} &= \exists y(\texttt{:deplAt}(x, y)), & \phi_{s_1} &= (\geq_1 \texttt{:hasCat}.\top),\\
\tau_{s_2} &= \exists y(\texttt{:hasTuCat}(x, y)), & \phi_{s_2} &= (\geq_1 \texttt{a.:Turbine}),\\
\tau_{s_3} &= \texttt{:PPlant}(?x), & \phi_{s_3} &= (\geq_1 \texttt{:hasTurb}.s_4),\\
\tau_{s_4} &= \texttt{:Turbine}(?x), & \phi_{s_4} &= (\geq_1 \texttt{:deplAt}.s_3).
\end{aligned}$$

Here s_1 essentially says that any deployed artifact should have a category, and s_2 says that only turbines can have a turbine category. The last two shapes s_3 and s_4 are mutually recursive, and they respectively say that each power plant should have at least one turbine and each turbine should be deployed in at least one location. □

2.3 SHACL Semantics

Given a shape s, a KG $\mathcal{G}$, and an entity e occurring $\mathcal{G}$, we say that e *verifies* s in $\mathcal{G}$ if the constraint ϕ_s applied to e is valid in $\mathcal{G}$. Finally, $\mathcal{G}$ is *valid* against $\mathcal{C}$ if for each $s \in \mathcal{C}$, each target entity retrieved by τ_s from $\mathcal{G}$ verifies s in $\mathcal{G}$. Since a constraint ϕ_s may refer to a shape s', the definition of validity for KGs is non-trivial. Indeed, the SHACL specification leaves the difficult case of recursion up to the concrete implementation[10] and a formal semantics via so-called *shape assignments* has only recently been proposed [12]. Intuitively, $\mathcal{G}$ is valid against $\mathcal{C}$ if one can label its entities with shape names, while respecting targets and constraints. A shape assignment σ is a function mapping each entity of $\mathcal{G}$ to a set of shape names in $\mathcal{C}$. We call an assignment *target-compliant* if it assigns (at least) each shape to each of its targets, *constraint-compliant* if it complies with the constraints, and *valid* if it complies with both targets and constraints. Then, $\mathcal{G}$ is valid against $\mathcal{C}$ if there exists a valid assignment for $\mathcal{G}$ and $\mathcal{C}$.

Example 3. Observe that $\mathcal{G}_{SIEM}$ is not valid against $\mathcal{C}_{SIEM}$. Shape s_1 has targets `:t852` and `:t177`, since both are deployed. `:t852` satisfies the constraint for s_1, since it has a category, but `:t177` violates it. Shape s_2 has target `:t177` only, which violates it, since it is not declared to be a turbine. Shape s_3 has no target in $\mathcal{G}_{SIEM}$. The case of shape s_4 is more involved. It has only `:t852` as target, and one may assign s_4 to `:t852` and s_3 to `:p063`, in order to satisfy the recursive constraint. But since `:t177` violates s_1 and s_2, there is no "global" valid shape assignment for $\mathcal{G}$ and $\mathcal{S}$, i.e. which would satisfy all targets and constraints simultaneously. □

2.4 OWL 2 QL

We now recall the syntax and semantics of OWL 2 QL relying on the the Description Logics *DL-Lite*$_R$ [11] that is behind this profile. (Complex) classes and properties in OWL 2 QL are recursively defined as follows:

$$B ::= A \mid \exists R,\ C ::= B \mid \neg B,\ R ::= P \mid P^-, \text{ and } E ::= R \mid \neg R,$$

[10] `https://www.w3.org/TR/shacl/`

where A is a class from Σ, P a property from Σ, and P^- the inverse of P. A *DL-Lite*$_R$ ontology is a finite set of axioms of the form $B \sqsubseteq C$ or $R \sqsubseteq E$. A *Knowledge Base* (KB) is a pair $\langle \mathcal{O}, \mathcal{G} \rangle$ of an ontology and a KG. The formal semantics of *DL-Lite*$_R$ is given in terms of standard first-order logic interpretations $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ over Δ in the standard way.

Example 4. Consider the following OWL 2 QL ontology $\mathcal{O}_{SIEM}$:

$$\{\texttt{:hasTuCat} \sqsubseteq \texttt{:hasCat},\ \exists \texttt{:hasTuCat}.\top \sqsubseteq \texttt{:Turbine}\},$$

that says tat if x has y as a turbine category, then x has y as a category, and also x can be inferred to be a turbine. □

A useful property of *DL-Lite*$_R$ exploited in Section 4, is the existence, for any satisfiable KB $\langle \mathcal{O}, \mathcal{G} \rangle$, of a so-called *canonical model*, which can be homomorphically mapped to any model of $\langle \mathcal{O}, \mathcal{G} \rangle$.

2.5 Constraint Validation over Ontology-Enhanced KGs

Consider the semantics of [25], that naturally extends constraint validation from KGs to ontology-enhanced KGs and has been adopted in, e.g., [20]. Given a KG $\mathcal{G}$, ontology $\mathcal{O}$, and a set of constraints $\mathcal{C}$, the idea of this semantics is to validate $\mathcal{C}$ over all set inclusion minimal models of $\mathcal{G}$ and $\mathcal{O}$. Formally, $\mathcal{G}$ enhanced with $\mathcal{O}$ is *valid* against $\mathcal{C}$ if for each minimal model $\mathcal{M}$ of $\mathcal{G}$ with $\mathcal{O}$, the KG *skol*$(\mathcal{M})$ is valid against $\mathcal{C}$, where *skol*$(\mathcal{M})$ is the Skolemization of models.

Example 5. Observe that $\langle \mathcal{O}, \mathcal{G}, \rangle$ is valid agains $\mathcal{C}_{SIEM}$. Indeed, shape s_1 is still satisfied by `:t852`, since no new information can be entailed about `:t852` from $\langle \mathcal{O}_{SIEM}, \mathcal{G}_{SIEM} \rangle$. But it is not violated anymore by `:t177`, since $\langle \mathcal{O}_{SIEM}, \mathcal{G}_{SIEM} \rangle$ entails that `:t177` has a category. Similarly, shape s_2 is not violated anymore by `:t177`, which can now be inferred to be a turbine. As for shape s_4, it now has an additional target (`:t177`), and it is verified by both its targets, thanks to the following assignment:

$$\{s_1 \mapsto \{\texttt{:t852}, \texttt{:t177}\}, s_2 \mapsto \{\texttt{:t177}\}, s_3 \mapsto \{\texttt{:p063}\}, s_4 \mapsto \{\texttt{:t852}, \texttt{:t177}\}\}.$$

3 The Problem of Constraint Rewriting

We now formalise and discuss the problem of constraint rewriting over ontologies.

3.1 SHACL-Rewriting

Our notion adapts the notion of rewriting (or reformulation) of queries over ontologies from [11, 19].

Definition 1. *Let $\mathcal{C}$ be a set of constraints and $\mathcal{O}$ an ontology. A set of constraints $\mathcal{C}'$ is a* constraint-rewriting *of $\mathcal{C}$ over $\mathcal{O}$ if for any KG $\mathcal{G}$ it holds that:*

$$\langle \mathcal{O}, \mathcal{G} \rangle \textit{ is valid against } \mathcal{C} \textit{ iff } \mathcal{G} \textit{ is valid against } \mathcal{C}'.$$

We now illustrate this notion on the following example.

Example 6. Consider a set of SHACL constraints and an OWL 2 QL ontology:

$$\mathcal{C} = \{\langle s, \tau_s, \phi_s\rangle\}, \text{ where } \tau_s = \texttt{:MechDevice}(x) \text{ and } \phi_s = (\geq_1 \texttt{:hasModel}.\top),$$
$$O = \{\texttt{:Turbine} \sqsubseteq \texttt{:MechDevice}, \exists\texttt{:hasTuCat} \sqsubseteq \exists\texttt{:hasCat}\}.$$

One can show that a rewiring of $\mathcal{C}$ over $\mathcal{O}$ is $\mathcal{S}' = \{\langle s, \tau'_s, \phi'_s\rangle\}$, where

$\tau'_s = \texttt{:MechDevice}(x) \vee \texttt{:Turbine}(x)$ and $\phi'_s = (\geq_1 \texttt{:hasCat}.\top) \vee (\geq_1 \texttt{:hasTuCat}.\top)$□

Observe that in the example both the target definition τ_s and the constraint definition ϕ_s were rewritten over $\mathcal{O}$ in order to guarantee that the ontology $\mathcal{O}$ can be safely ignored. In particular, the rewriting of τ_s guarantees that in any graph $\mathcal{G}$, each instance of :Turbine should also be verified against s, whereas the rewriting of ϕ_s guarantees that any entity in $\mathcal{G}$ with a :hasTuCat-successor validates s, even if it has no :hasCat-successor.

Thus, despite the similarity of query and constraint rewriting overt ontologies there are significant differences. [11] The first difference as illustrated above is that a shape contains a target definition and a constraint that in the general case should be rewritten independently. But more importantly, as opposed to queries, SHACL constraints can be recursive which makes the rewriting significantly more involved (see Section 4 for details).

In what follows we study rewritability for SHACL, i.e. *SHACL-rewritability*, for different fragments of SHACL. Before proceeding we show that in the general case rewriting does not exist.

3.2 Non-Existence of SHACL-Rewritings

We start with the hardness of SHACL validation.

Theorem 1. *There exists an DL-Lite$_R$ ontology, a set of SHACL constraints $\mathcal{C}$, and a KG $\mathcal{G}$ such that deciding whether $\langle\mathcal{O}, \mathcal{G}\rangle$ is valid against $\mathcal{C}$ is* co-NP*-hard in the size of $\mathcal{G}$.*

Proof (Sketch). The proof is based on encoding the 3-coloring co-problem. For a given undirected graph $\mathcal{F} = \langle V, E\rangle$ (with vertices V and edges E), we construct the following KG $G_{\mathcal{F}}$:

$$\{(v_i,\ \mathsf{a},\ V) \mid v_i \in V\} \cup \{(v_i,\ E,\ v_j) \mid (v_i,\ v_j) \in E\}$$
$$\cup \{(v',\ U,\ v_i) \mid v_i \in V\} \cup \{(v',\ \mathsf{a},\ T)\},$$

where v', U and T are needed for technical reasons as will be explained below.

Then, we define $\mathcal{O} = \{V \sqsubseteq \exists R.C, C_{red} \sqsubseteq C, C_{blue} \sqsubseteq C, C_{red} \sqsubseteq \neg C_{blue}\}$, where the axiom $V \sqsubseteq \exists R.C$ enforces that in each minimal model $\mathcal{M}$ of $\langle\mathcal{O}, \mathcal{G}_{\mathcal{F}}\rangle$, each vertex v_i has an R-successor a_i, which intuitively stands for the color of vertex v_i in $\mathcal{F}$ [12]. The

[11] Recall that for query rewriting the input is a query q and ontology $\mathcal{O}$ and the output is another query q' such that for any database D so-called certain answers of q over $\langle\mathcal{O}, \mathcal{D}\rangle$ coincide with the answers of q' over $\mathcal{D}$ alone [11].

[12] The axiom of kind $V \sqsubseteq \exists R.C$ in syntactically not in *DL-Lite$_R$* but can be expressed by using fresh role R_1 and three axioms: $V \sqsubseteq \exists R_1$, $R_1 \sqsubseteq R$ and $\exists R_1^- \sqsubseteq C$.

two other axioms intuitively enforce that either $(a_i, \mathsf{a}, C_{red}) \in \mathcal{M}$ or $(a_i, \mathsf{a}, C_{blue}) \in \mathcal{M}$, or none of the two. Intuitively, v_i is either red, or blue or none of the two (i.e. green).

Now we introduce a singleton set of constraints $\mathcal{C} = \{\langle s, \tau_s, \phi_s \rangle\}$ that requires that at least one pair of adjacent vertices has the same color:

$$\tau_s = T(x), \text{ and } \phi_s = (\geq_1 U.(\phi_1 \vee \phi_2 \vee \phi_3)), \text{ where}$$

$$\phi_1 = (\geq_1 R. \geq_1 \mathsf{a}.C_{red}) \wedge (\geq_1 E. \geq_1 R. \geq_1 \mathsf{a}.C_{red})$$
$$\phi_2 = (\geq_1 R. \geq_1 \mathsf{a}.C_{blue}) \wedge (\geq_1 E. \geq_1 R. \geq_1 \mathsf{a}.C_{blue})$$
$$\phi_3 = (\geq_1 R. \geq_1 \mathsf{a}.(\neg C_{red} \wedge \neg C_{blue})) \wedge (\geq_1 E. \geq_1 R. \geq_1 \mathsf{a}.(\neg C_{red} \wedge \neg C_{blue})).$$

Intuitively, formula ϕ_1 evaluates to true at node v_i node if v_i is colored red and has a red neighbor. Formulas ϕ_2 and ϕ_3 evaluate similarly, but for blue and green. Finally, shape s has node v' as a unique target, and v has every other node in $G_{\mathcal{F}}$ as a U-successor, ensuring that $G_{\mathcal{F}}$ is valid against $\mathcal{C}$ iff there is no 3-colouring for $\mathcal{F}$. □

In [13] it has been shown that validation of SHACL constraints over KG without ontologies is NP-complete in the size of the graph. Thus, we can immediately conclude the following negative result that holds under the assumption that CO-NP $\not\subseteq$ NP.

Corollary 1. *There exists an DL-Lite$_R$ ontology and a set of SHACL constraints for which no SHACL-rewriting over this ontology exists.*

In order to overcome the non-existence problem for OWL 2 QL we found possible restrictions on both the ontology language and SHACL (we leave the study of restrictions for OWL 2 EL for future work). In particular, a combination of *DL-Lite$_R$* axioms of the form $A \sqsubseteq \exists R$ on the one hand, and SHACL constraints with negation on the other hand is sufficient to show that a SHACL-rewriting may not exist. In the next section we investigate rewritability for fragments where such combinations are ruled out.

4 Shape Rewriting

In this section we introduce our rewriting algorithms. As discussed above, a rewriting may not exist for an arbitrary set of SHACL shapes and a *DL-Lite$_R$* ontology. Thus to gain rewritability, one needs to restrict the expressivity of either SHACL or the ontology language.

In Section 4.1, we define *DL-Lite$_R$*$^-$, the fragment of *DL-Lite$_R$* where extensional quantification (i.e. concepts of the form $\exists R$) is not allowed on the right-hand side of a general concept inclusion (GCI). *DL-Lite$_R$*$^-$ is interesting from a practical point of view, since it corresponds to RDFS. We also define *positive SHACL* as the fragment of SHACL without negation (but where disjunction is allowed). For *DL-Lite$_R$*$^-$ and whole SHACL we developed a rewriting algorithm Algorithm 1.

In Section 4.2 we restrict SHACL to positive SHACL shapes while we do not restrict *DL-Lite$_R$*. For this setting we develop Algorithm 2 that allows to compute constraint rewritings. Note that this algorithm is more involved that Algorithm 1 since *DL-Lite$_R$* KBs may have canonical models of arbitrary size (even infinite).

We observe that the PERFREF algorithm defined in [11] to rewrite (unions of) conjunctive queries in the presence of a *DL-Lite$_R$* ontology cannot be applied in general

to SHACL shapes, since they may be recursive or contain negations. We nonetheless use PERFREF below when applicable (in particular, for rewriting target definitions). If $\mathcal{O}$ is a *DL-Lite*$_R$ ontology and ψ a conjunctive query (resp. if C is a *DL-Lite*$_R$ concept) then PERFREF$(\psi, \mathcal{O})$ (resp. PERFREF$(C, \mathcal{O})$) designates the rewriting of ψ (resp. the conjunctive query corresponding to C) w.r.t. to $\mathcal{O}$.

In order to make notation more concise, we use $\mathcal{G}, \mathcal{C} \models \phi(v)$ below to indicate that node v satisfies constraint ϕ in graph $\mathcal{G}$ given a set $\mathcal{C}$ of shapes. Similarly, we use $\langle \mathcal{O}, \mathcal{G} \rangle, \mathcal{C} \models \phi(v)$ to indicate that node v satisfies constraint ϕ in graph that corresponds to the canonical model $\langle \mathcal{O}, \mathcal{G} \rangle$ given a set $\mathcal{C}$ of shapes.

We also assume w.l.o.g. that shape constraints in $\mathcal{C}$ are normalised, i.e. contain at most one operator, which can always be obtained by introducing nested shape names.

4.1 Rewriting for *DL-Lite*$_R^-$

In this section, we introduce a rewriting algorithm for *DL-Lite*$_R^-$ ontologies and full SHACL. We observe that a *DL-Lite*$_R^-$ KB $\langle \mathcal{O}, \mathcal{G} \rangle$, because it has no existential quantifier on the right-hand side of a GCI, must have a canonical model $can(\mathcal{O}, \mathcal{G})$ of finite size. This makes rewriting significantly simpler than for arbitrary *DL-Lite*$_R$ ontologies (investigated in Section 4.2).

Given an ontology $\mathcal{O}$ and set $\mathcal{C}$ of shapes, the rewriting $\mathcal{C}'$ of $\langle \mathcal{O}, \mathcal{C} \rangle$ is the union of two sets of shapes: a set $\mathcal{C}_\mathcal{O}$ built out of $\mathcal{O}$, which is used to mimic ontological reasoning, and a set $\mathcal{C}''$ built out of $\mathcal{C}$ and $\mathcal{O}$, obtained by *(i)* rewriting target definitions w.r.t. $\mathcal{O}$, and *(ii)* rewriting constraints w.r.t. $\mathcal{C}$ and $\mathcal{O}$.

Constructing $\mathcal{C}_\mathcal{O}$ For every concept of the form A (resp. $\exists R$) in $\mathcal{O}$, we introduce a shape s_A (resp. $s_{\exists R}$), with no target (i.e. $\tau_{s_A} = \tau_{s_{\exists R}} = \bot(x)$) and with constraint:

$$\phi_{s_A} = (\geq_1 \mathsf{a}.A) \vee \bigvee_{C \sqsubseteq A \in \mathcal{O}} s_C, \qquad \phi_{s_{\exists R}} = (\geq_1 R.\top) \vee \bigvee_{C \sqsubseteq \exists R \in \mathcal{O}} s_C \vee \bigvee_{R' \sqsubseteq R \in \mathcal{O}} s_{\exists R'},$$

where R, R' may be inverse roles.

Further, we introduce shapes that encode negative assertions. For each GCI of the form $(C \sqsubseteq \neg D)$ in $\mathcal{O}$, we introduce one shape $s_{C \sqsubseteq \neg D}$, whose targets are all instances of C and D in $\mathcal{G}$, and whose constraint is always violated. To this end, we exploits results based on PERFREF (see [11]). Namely $\tau_{s_{C \sqsubseteq \neg D}} =$ PERFREF$(C(x) \wedge D(x), \mathcal{O})$, and $\phi_{s_{C \sqsubseteq \neg D}} = \bot$. Similarly, for negative role inclusions, we use $s_{R_1 \sqsubseteq \neg R_2}$, with $\tau_{s_{R_1 \sqsubseteq \neg R_2}} =$ PERFREF$(\exists y R1(x, y) \wedge R_2(x, y), \mathcal{O})$, and $\phi_{s_{R_1 \sqsubseteq \neg R_2}} = \bot$.

We denote the set of shapes produced above with $\mathcal{C}_\mathcal{O}$, and the corresponding translation function as SHAPET, i.e., SHAPET$(\mathcal{O}) = \mathcal{C}_\mathcal{O}$. Then the following holds:

Lemma 1 (Finite Canonical model). *Let $\mathcal{O}$ be a DL-Lite$_R$ ontology, C a DL-Lite$_R$ concept, $\mathcal{G}$ a graph, v a node in $\mathcal{G}$, $can(\mathcal{O}, \mathcal{G})$ the canonical model of $\mathcal{O}$ and $\mathcal{G}$. Then $\mathcal{G}, \mathcal{C}_\mathcal{O} \cup \mathcal{C}'' \models \phi_{s_C}(v)$ iff $(v, \mathsf{a}, C) \in can(\mathcal{O}, \mathcal{G})$.*

Next, we rewrite the shapes in $\mathcal{C}$. If s is an initial shape in $\mathcal{C}$, its rewriting will be designated with s'. As already illustrated, both target definitions and constraints need to be rewritten. We start with target definitions.

Algorithm 1 CONSTRAINT REWRITING FOR *DL-Lite*$_R^-$

Input: *DL-Lite*$_R^-$ ontology $\mathcal{O}$, set $\mathcal{C}$ of shapes
1: $\mathcal{C}_\mathcal{O} \leftarrow$ SHAPET($\mathcal{O}$)
2: $\mathcal{C}'' \leftarrow \{\langle s', \text{PERFREF}(\tau_s, \mathcal{O}), \text{REWRITESIM}(\phi_s, \mathcal{O})\rangle \mid s \in \mathcal{C}\}$
3: **return** $\mathcal{C}_\mathcal{O} \cup \mathcal{C}''$

Rewriting of Constraints In the absence of ontology, the targets of s are retrieved by evaluating the target definition τ_s over graph $\mathcal{G}$, written $[\![\tau_s]\!]^\mathcal{G}$. In SHACL, a target definition is a monadic query with a single atom that corresponds to a basic concept in an ontology.

In the presence of an ontology, we follow the semantics described in Section 2.5, and retrieve targets over all minimal models, or equivalently over the canonical model, written $[\![\tau]\!]^{\langle\mathcal{O},\mathcal{G}\rangle}$. To achieve this, since τ_s is a unary conjunctive query, one can apply PERFREF.

Lemma 2. *For any shape s, ontology $\mathcal{O}$ and graph $\mathcal{G}$:*
$[\![\tau_s]\!]^{\langle\mathcal{O},\mathcal{G}\rangle} = [\![\text{PERFREF}(\tau_s, \mathcal{O})]\!]^\mathcal{G}$

Proof (Sketch). Follows from the properties of the canonical model and query answering over *DL-Lite*$_R$ ontologies (see [11])

In other words, the targets of s according to the KB $\langle\mathcal{O}, \mathcal{G}\rangle$ can be retrieved by evaluating the query $\text{PERFREF}(\tau_s, \mathcal{O})$ over $\mathcal{G}$ alone.

Rewriting of Constraints Finally, we rewrite the constraints in $\mathcal{C}$. We replace each shape s by shape s' such that:

$$\phi_{s'} = s'_1 \wedge s'_2 \text{ if } \phi_s = s_1 \wedge s_2, \qquad \phi_{s'} = s'_1 \vee s'_2 \text{ if } \phi_s = s_1 \vee s_2,$$
$$\phi_{s'} = \neg s'_1 \text{ if } \phi_s = \neg s_1, \qquad \phi_{s'} = I \text{ if } \phi_s = I,$$
$$\phi_{s'} = (\geq_k R.s'_1) \vee \bigvee_{R' \sqsubseteq R \in \mathcal{O}} (\geq_k R'.s'_1) \text{ if } \phi_s = (\geq_k R.s_1),$$
$$s_{R_1,R_2} = \bigwedge_{\mathcal{O} \models R'_1 \sqsubseteq R_1, R'_2 \sqsubseteq R_2} \text{EQ}(R'_1, R'_2) \text{ if } \phi_s = \text{EQ}(R_1, R_2).$$

Theorem 2. *Given a DL-Lite*$_R^-$ *ontology $\mathcal{O}$, graph $\mathcal{G}$, node v in $\mathcal{G}$, set $\mathcal{C}$ of shapes and shape s in $\mathcal{C}$:*

$$\langle\mathcal{O}, \mathcal{G}\rangle, \mathcal{C} \models \phi_s(v) \text{ iff } \mathcal{G}, \mathcal{C}_\mathcal{O} \cup \mathcal{C}'' \models \phi_{s'}(v)$$

We denote with REWRITESIM this constraint rewriting function, i.e. $\text{REWRITESIM}(\phi_s, \mathcal{O}) = \phi_{s'}$ for each s in $\mathcal{C}$.

Algorithm 1 summarise the whole (ontology and shape) rewriting procedure. As an illustration, the SHACL-rewriting described in Example 6 is the one produced by this procedure.

4.2 Rewriting for *DL-Lite*$_R$

We now consider the case of an arbitrary *DL-Lite*$_R$ $\mathcal{O}$, together with a set $\mathcal{C}$ of positive shapes. For any graph $\mathcal{G}$, $can(\mathcal{O}, \mathcal{G})$ may now be arbitrary large (even infinite) and it may introduce fresh nodes that may be needed to check constraint. This makes rewriting significantly more involved.

Example 7. Consider the ontology $\mathcal{O} = \{A \sqsubseteq \exists U, \exists U^- \sqsubseteq \exists P\}$ and graph $\mathcal{G} = \{(v, \mathsf{a}, A)\}$. Then $can(\mathcal{O}, \mathcal{G}) = \{(v, \mathsf{a}, A), (v, U, a_1), (a_1, P, a_2)\}$ where a_1 and a_2 are fresh nodes. Now consider shapes $\langle s_1, A(x), (\geq_1 U.s_2)\rangle$ and $\langle s_2, \bot(x), (\geq_1 P.\top)\rangle$. It is not hard to see that $\mathcal{C}$ is valid over $(\mathcal{O}, \mathcal{G})$.

Since SHACL constraints cannot express fresh values, we need to introduce additional shapes that mimic the construction of the canonical model.

We start by introducing additional technical notions. Let $cl(\mathcal{O}, \mathcal{G})$ be the maximal subset of $can(\mathcal{O}, \mathcal{G})$ which contains no fresh node. We observe that facts in $cl(\mathcal{O}, \mathcal{G})$ can be validated using $\mathcal{C} \cup \text{SHAPET}(\mathcal{O})$. Now we need to introduce additional shapes to validates facts in $\mathcal{G}' = can(\mathcal{O}, \mathcal{G}) \setminus cl(\mathcal{O}, \mathcal{G})$. Graph $\mathcal{G}'$ is in fact a forest (follows from the construction of $can(\mathcal{O}, \mathcal{G})$) where each tree has a root in some assertion in $cl(\mathcal{O}, \mathcal{G})$. We call this root the *witness* of the tree. In example above, (v, a, A) is the only witness.

Then for each concept C appearing in a GCI in $\mathcal{O}$, we introduce a shape $s_C^{virtual}$, such that, for a node v in $\mathcal{G}$, verifies $s_C^{virtual}(v)$ iff there is a node v' in $\mathcal{G}'$ with v as witness such that $\mathcal{G}' \models C(v')$. For instance, in Example 7, we introduce shape $s_{\exists R}^{virtual}$ which is verified by witness v. Note that from this definition, v' is not necessarily an immediate successor of v in $\mathcal{G}'$.

More formally, for concept C, the virtual shape $\langle s_C^{virtual}, \bot(x), s_C\rangle$ is created. Then a function similar to REWRITESIM is applied to each $\phi_{s_C^{virtual}}$, in order to ensure the above property. In our running example, this yields $\phi_{s_A^{virtual}} = s_A$, i.e. $\phi_{s_A^{virtual}}$ remains unchanged, but $\phi_{s_{\exists U}^{virtual}} = s_{\exists U} \vee s_A^{virtual} \vee s_{\exists U^-}^{virtual}$. Here, sub-formula $s_A^{virtual}$ is added because of the GCI $A \sqsubseteq U$, and $s_{\exists U^-}$ is added because if $\exists U$ holds at some node a_1 in the tree of $\mathcal{G}'$ rooted in v, then $\exists U^-$ must hold at some U-successor a_2 of a_1. Let SHAPEVIRTUAL be the function which produces (and rewrites) these "virtual" shapes.

A second kind of shape is needed in order to check if two roles are concatenated in the same tree in $\mathcal{G}'$. For each pair of roles R_1 and R_2 in $\mathcal{O}$, we introduce shape s_{R_1,R_2}^{succ} such that a node $v \in \mathcal{G}$ verifies $\phi_{s_{R_1,R_2}^{succ}}$ iff (a_1, R_1, a_2) and (a_2, R_2, a_3) are on the subtree with witness v, for some a_1, a_2, a_3 in $\mathcal{G}'$. In our running example, v verifies $\phi_{s_{R,P}^{succ}}$, but not $\phi_{s_{P,R}^{succ}}$. Formally, for every two roles R_1 and R_2 in $\mathcal{O}$, $\tau_{s_{R_1,R_2}^{succ}} = \bot(x)$, and if $\mathcal{O} \models \exists R_1^- \sqsubseteq R_2$, then $\phi_{s_{R_1,R_2}^{succ}} = s_{\exists R_1}$, otherwise $\phi_{s_{\exists R_1,\exists R_2}^{succ}} = \bot$. The special case $R_2 = R_1^-$, is also covered by the definition $\phi_{s_{R_1,R_1^-}^{succ}} = s_{\exists R_1}$. Let SUCCESSORT denote the function creating these fresh shapes.

Finally, we need to rewrite the shapes in $\mathcal{C}$. To this end, we extend the procedure REWRITESIM in the following way. For each shape s in $\mathcal{C}$, we set $s' =$ REWRITECOMPL$(s) \vee s^{virtual}$ where REWRITECOMPL is identical to REWRITESIM for operators $\wedge$, $\vee$ and constant I but it changes for $\phi_s = (\geq_k R.s_1)$ as follows:

$$\phi'_s = (\geq_k R.s'_1) \vee s^{virtual} \quad \text{where} \quad \phi_{s^{virtual}} = s_{\exists R}^{virtual} \wedge s_1^{virtual} \wedge s_{\exists R, s_1}$$

Algorithm 2 CONSTRAINT REWRITING 2

Input: ontology $\mathcal{O}$ possible with existential rules, set of positive shapes $\mathcal{C}$

1: $\mathcal{C}_\mathcal{O} \leftarrow$ SHAPET($\mathcal{O}$)
2: $\mathcal{C}^v_\mathcal{O} \leftarrow$ SHAPEVIRTUAL($\mathcal{O}$)
3: $\mathcal{C}^s_{\mathcal{O},\mathcal{C}} \leftarrow$ SUCCESSORT($\mathcal{O}, S$)
4: $\mathcal{C}'' \leftarrow \{\langle$PERFREF$(\tau_s, \mathcal{O})$, REWRITECOMPL$(\phi_s, \mathcal{O})\rangle \mid s \in \mathcal{C}\}$
5: **return** $\mathcal{C}_\mathcal{O} \cup \mathcal{C}^v_\mathcal{O} \cup \mathcal{C}^s_{\mathcal{O},\mathcal{C}} \cup \mathcal{C}''$

In other words, witness v verifies $s^{virtual}$ if it verifies both $s^{virtual}_{\exists R}$ and $s^{virtual}_1$ (that is, both are verified by some anonymous node with v as witness), and the range of R can be validated against s_1, expressed with the new shape $s_{\exists R,s_1}$. Then $\phi_{s_{\exists R,s_1}} = s_{\exists R,s_2} \wedge s_{\exists R,s_3}$ if $\phi_{s_1} = s_2 \wedge\ s_3$ (and similarly for $\vee$). If $\phi_{s_1} = (\geq_k P.s_2)$ then $\phi_{\exists R,s_1} = s^{succ}_{R,P}$, that is P has to be successor of R in $\mathcal{G}'$. Let REWRITECOMPLT denote the corresponding rewriting of $\mathcal{C}$.

We summarise the rewriting procedure in Algorithm 2.

Lemma 3. *Let $\mathcal{O}$ be a DL-Lite ontology, C a concept in $\mathcal{O}$, R and P properties in $\mathcal{O}$, $\mathcal{G}$ a graph, v a node in $\mathcal{G}$, $can(\mathcal{O},\mathcal{G})$ the canonical model of $\mathcal{O}$ and $\mathcal{G}$, $\mathcal{C}$ a set of positive shapes and $\mathcal{C}'$ the shapes returned by Algorithm 2. Then the following holds:*

- $\mathcal{G}, \mathcal{C}' \models \phi_{s^{virtual}_C}(v)$ *iff there is a node a_1 in $can(\mathcal{O},\mathcal{G})$ with witness v s.t. $can(\mathcal{O},\mathcal{G}) \models C(a_1)$*
- $\mathcal{G}, \mathcal{C}' \models \phi_{s^{\text{succ}}_{R,P}(v)}$ *iff there are nodes a_1, a_2, a_3 in $can(\mathcal{O},\mathcal{G})$ with witness v s.t. $can(\mathcal{O},\mathcal{G}) \models R(a_1, a_2)$ and $can(\mathcal{O},\mathcal{G}) \models R(a_2, a_3)$*

Example 8. We illustrate the rewriting of the running example. Shapes that are not relevant for reasoning are omitted. The presented shapes are ordered in the way one would reason with them, starting bottom-up (which is possible if $\mathcal{C}$ is not recursive). To illustrate the reasoning, we underline in each formula the disjuncts for which one can construct a satisfying shape assignment.

$$\phi_{s^{virtual}_A} = \underline{s_A}, \quad \phi_{s^{virtual}_{\exists U}} = s_{\exists U} \vee \underline{s^{virtual}_A} \vee s^{virtual}_{\exists U^-},$$
$$\phi_{s_{\exists U}} = (\geq_1 U.\top) \vee \underline{s^{virtual}_{\exists U}}, \quad \phi_{s^{succ}_{U,P}} = \underline{s_{\exists U}}, \quad \phi_{s_{\exists U,s_2}} = \underline{s^{succ}_{U,P}},$$
$$\phi_{s^{virtual}_1} = \underline{s^{virtual}_{\exists U}} \wedge \underline{s^{virtual}_2} \wedge \underline{s_{\exists U,s_2}}, \quad \phi_{s'_1} = \geq_1 U.s'_2 \vee \underline{s^{virtual}_1},$$
$$\phi_{s^{virtual}_2} = \underline{s^{virtual}_{\exists P}}, \quad \phi_{s^{virtual}_{\exists P}} = s_{\exists P} \vee \underline{s^{virtual}_{\exists U^-}} \vee s^{virtual}_{\exists P^-}, \quad \phi_{s^{virtual}_{\exists U^-}} = \underline{s^{virtual}_{\exists U}}.$$

The only target of s is v, and v verifies ϕ_s w.r.t the rewritten set of shapes.

Theorem 3. *Let $\mathcal{O}$ be a DL-Lite ontology, $\mathcal{C}$ a set of positive shapes, s a shape in $\mathcal{C}$, $\mathcal{C}'$ the shapes returned by Algorithm 2, and s' the rewriting of s in $\mathcal{C}'$.*
For any graph $\mathcal{G}$ and node v in $\mathcal{G}$

$$\langle \mathcal{O}, \mathcal{G}\rangle, \mathcal{C} \models \phi_s(v) \text{ iff } \mathcal{G}, \mathcal{C}' \models \phi'_s(v)$$

We also note that the size of the returned rewriting is polynomial in size of $\mathcal{O}$ and $\mathcal{C}$.

	Full SHACL	*Positive SHACL*
DL-Lite$_R$	DP-hard	PTIME-complete
DL-Lite$_R^-$	NP-complete	PTIME-complete

Table 1. Combined complexity of graph validation against a (compact) SHACL-rewriting, for different fragments of SHACL and *DL-Lite*

5 Complexity of Validation

In this section we discuss the complexity of validating a KB against SHACL constraints as summarised in Table 1. These results follow from Sections 3 and 4 and previous results about SHACL constraint validation.

We start by observing that despite the fact that for full SHACL and *DL-Lite*$_R$, rewriting in general does not exist, we can still provide a lower-complexity bound on validation of KBs over constraints. We show that the problem is DP-hard [26]. We remind the reader that DP $=$ NP $\wedge$ CO-NP $= \{L_1 \cap L_2 \mid L_1 \in$ NP and $L_2 \in$ CO-NP$\}$ Then, from Theorem 1, validation is at least CO-NP-hard in data complexity. Since it is also at least NP-hard (shown by SAT encoding in [12]) for full SHACL, it is not hard to combine two encodings into single one and obtain the problem that is DP-hard.

It was shown in [12] that in the case without ontologies validation for the full SHACL language is NP-complete in both data (i.e. the size of the graph) and combined (i.e. graph and constraints) complexity, e.g., due to the combination of recursion and negation. At the same time tractability can be gained by restricting their usage as in positive SHACL, for which the validation is PTIME-complete in both data and combined complexity. Another way to gain the tractability is as [14] where the notion of *strict stratification* for SHACL constraints was introduced and it strengthen the classical notion of stratification in Datalog. It was shown that validation for such constraints is also in PTIME. In Section 4.1 our rewriting over ontologies of disjointness constraints for *DL-Lite*$_R$ and positive SHACL is strictly stratified, and this is the only possible source of negation. Since our rewriting is of polynomial size, we can conclude that validating a *DL-Lite*$_R$ KB against positive SHACL constraints is in PTIME.

6 Discussion And Related work

In this section we discuss what we thinks are interesting properties of SHACL rewritings as well as related work.

6.1 On Compactness of Rewriting

The rewriting technique presented in this paper also have a desirable property of *compactness*. Compactness of rewritability has been studied query rewriting over OWL 2 QL ontologies [8], and it has been shown (under some separation assumptions) that a polynomial rewriting cannot exist in general for several non-recursive languages (including non-recursive Datalog). In comparison, our techniques take the advantage of recursive shape references to produce a worst-case polynomial SHACL-rewriting.

6.2 General Rewriting Algorithm

In general, rewriting of OWL 2 QL can be seen a special case of general *backward-chaining algorithm* [24] algorithm over so-called $\forall\exists$-rules (where the body and the head are conjunctions of atoms, and variables that occur only in the head are existentially quantified). The existence of a rewriting is undecidable in general for arbitrary rules, and even for some decidable fragments, this algorithm may not terminate. Nevertheless, since it is a general one it is a good starting point to investigate terminations and optimizations when applied to more restrictive ontologies.

6.3 Beyond OWL 2 QL

So far we focused on restricting SHACL constraints to gain rewritability. A natural question to ask what if we go beyond *DL-Lite*$_R$ even if for restricted SHACL constraints or consider languages like OWL 2 EL. First, observe that for OWL 2 EL one can prove the non existence of rewriting of constraints over ontologies using the same argument is in Theorem 1. Still, compact rewriting in case of positive SHACL may be possible and we leave it as an open question. Then, consider $\mathcal{ALC}$ [7], a DL that is contained in OWL 2, has been long studied and that properly contains both QL and EL profiles of OWL 2. One can show that $\mathcal{ALC}$ is too expressive so that can be rewritten into SHACL even if we are given a single shape of the formula of the form "$\top$". For example, one can reduce the problem of "concept satisfiability" in $\mathcal{ALC}$ into checking validity of SHACL constraints of over the same $\mathcal{ALC}$ KB. Since the former is known to be PSPACE-complete [7], we have a problem that cannot be encoded as SHACL validation problem (which is worst case NP-complete). The Datalog$^{\pm}$ family [10] is another popular family of ontological languages that extend the idea of first-order query rewritability to $\forall\exists$-rules but under certain syntactic restrictions. For Datalog$^{\pm}$ despite the fact that instance checking is very efficient in data complexity (AC^0) meaning that the query answering can be reduced to evaluation of first-order queries, the price of the combined complexity of the problem is rather high (EXPTIME-hard) and thus one cannot obtain polynomial rewritings in SHACL.

6.4 Translating SHACL to known query language

Alternative idea of rewriting SHACL is to relate SHACL in some know ontological (query) language and then apply rewriting algorithm. Since SHACL may contain recursion such query language cannot be of conjunctive queries. Naturally, one may think of relating SHACL to Datalog programs [15]. However, Datalog programs can at most have one unique minimal model, and SHACL constraints are checking for all possible assignments [12] (including also non-minimal one). If we consider more expressive version of Datalog like Datalog with negation under the stable model semantics (SMS) [15], then relating it to SHACL is more promising, while the actual relation is not obvious as SMS is also based on minimal models. A possible way to relate the two semantics, at least for SHACL with single maximal assignments, is to reason about shape "complements" under SMS. Nevertheless, our preliminary results show that this is not straightforward.

7 Conclusion

In this work, we study the problem of rewriting constraints over ontologies. We focused on a prominent language for graph constraints, namely SHACL, and on ontologies from the widely used OWL 2 QL ontology language. We defined semantics for constraint rewriting, showed the non-existence of such rewritings in the general case, and identified restrictions of OWL 2 QL and SHACL for which they always exist. For these restricted cases, we showed how to rewrite ontologies and SHACL into unique set of SHACL constraints. Moreover, validation over OWL 2 QL is tractable for the positive (but still recursive) fragment of SHACL. And validation with full SHACL expressivity becomes NP-complete. For the case where the existence of rewritings cannot be guaranteed, we established lower complexity bound.

We see this work as an important step towards practical constraint rewriting algorithms and systems. In particular, instead of combining OWL reasoning and SHACL validation as an additional layer may create an unnecessary overhead, our rewritings allow the validation that could be performed against a single set shapes. Next, we plan to analyze optimization techniques in order to obtain more efficient rewritings. For instance, we plan to consider datatypes. They can be used to optimize eliminate unnecessary rewritings, but this need to be done in a controlled way to ensure tractability (e.g., [28]). Finally, in the future we plan to extend this work to account for OWL 2 EL. Moreover, we plan to implement our approach and evaluate it.

Bibliography

[1] Freebase: an open, shared database of the world's knowledge. `www.freebase.com/`.

[2] Google KG. `google.co.uk/insidesearch/features/search/knowledge.html`.

[3] W3C: OWL 2 Web Ontology Language. `http://www.w3.org/TR/owl2-overview/`.

[4] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[5] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[6] M. Arenas, C. Gutiérrez, and J. Pérez. Foundations of RDF Databases. In *Reasoning Web. Semantic Technologies for Information Systems*, pages 158–204, 2009.

[7] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The description logic handbook: theory, implementation and applications*. Cambridge university press, 2003.

[8] M. Bienvenu, S. Kikot, R. Kontchakov, V. V. Podolskii, and M. Zakharyaschev. Ontology-mediated queries: Combined complexity and succinctness of rewritings via circuit complexity. *Journal of the ACM (JACM)*, 65(5):28, 2018.

[9] I. Boneva, J. E. Labra-Gayo, and E. G. Prud'hommeaux. Semantics and Validation of Shapes Schemas for RDF. In *ISWC*, 2017.

[10] A. Calì, G. Gottlob, and A. Pieris. Query answering under non-guarded rules in datalog+/-. In *RR*.

[11] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *JAR*, 2007.

[12] J. Corman, J. L. Reutter, and O. Savkovic. Semantics and validation of recursive shacl. *ISWC*, 2018.
[13] J. Corman, J. L. Reutter, and O. Savkovic. Semantics and validation of recursive shacl (extended version). *Technical Report KRDB18-1, KRDB Research Center, Free Univ. Bozen-Bolzano*, 2018.
[14] J. Corman, J. L. Reutter, and O. Savkovic. A tractable notion of stratification for SHACL. In *(ISWC*, 2018.
[15] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.
[16] F. J. Ekaputra and X. Lin. SHACL4p: SHACL constraints validation within Protégé ontology editor. In *ICoDSE*, 2016.
[17] W. Fan, Z. Fan, C. Tian, and X. L. Dong. Keys for graphs. *PVLDB*, 8(12):1590–1601, 2015.
[18] W. Fan, Y. Wu, and J. Xu. Functional dependencies for graphs. In *SIGMOD*, pages 1843–1857, 2016.
[19] P. Hansen, C. Lutz, I. Seylan, and F. Wolter. Efficient query rewriting in the description logic EL and beyond. In *IJCAI*, pages 3034–3040, 2015.
[20] E. Kharlamov, B. C. Grau, E. Jiménez-Ruiz, S. Lamparter, G. Mehdi, M. Ringsquandl, Y. Nenov, S. Grimm, M. Roshchin, and I. Horrocks. Capturing industrial information models with ontologies and constraints. In *ISWC*, pages 325–343, 2016.
[21] E. Kharlamov, D. Hovland, M. G. Skjæveland, D. Bilidas, E. Jiménez-Ruiz, G. Xiao, A. Soylu, D. Lanti, M. Rezk, D. Zheleznyakov, M. Giese, H. Lie, Y. E. Ioannidis, Y. Kotidis, M. Koubarakis, and A. Waaler. Ontology Based Data Access in Statoil. *J. Web Sem.*, 44:3–36, 2017.
[22] E. Kharlamov, T. Mailis, G. Mehdi, C. Neuenstadt, Ö. L. Özçep, M. Roshchin, N. Solomakhina, A. Soylu, C. Svingos, S. Brandt, M. Giese, Y. E. Ioannidis, S. Lamparter, R. Möller, Y. Kotidis, and A. Waaler. Semantic Access to Streaming and Static Data at Siemens. *J. Web Sem.*, 44:54–74, 2017.
[23] H. Knublauch and A. Ryman. Shapes constraint language (SHACL). *W3C Recommendation*, 11:8, 2017.
[24] M. König, M. Leclère, M. Mugnier, and M. Thomazo. Sound, complete and minimal ucq-rewriting for existential rules. *Semantic Web*, 6(5):451–475, 2015.
[25] B. Motik, I. Horrocks, and U. Sattler. Bridging the gap between OWL and relational databases. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(2):74–89, 2009.
[26] C. M. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.
[27] P. F. Patel-Schneider. Using Description Logics for RDF Constraint Checking and Closed-World Recognition. In *AAAI*, 2015.
[28] O. Savkovic and D. Calvanese. Introducing datatypes in dl-lite. In *ECAI*, pages 720–725, 2012.
[29] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proc. of WWW*, pages 697–706, 2007.
[30] J. Tao, E. Sirin, J. Bao, and D. L. McGuinness. Integrity Constraints in OWL. In *AAAI*, 2010.

SHACL Constraint Validation over Ontology-enhanced KGs via Rewriting

Ognjen Savković[1], Evgeny Kharlamov[2], and Steffen Lamparter[3]

[1] Free University of Bozen-Bolzano, Bolzano, Italy
[2] University of Oslo, Oslo, Norway
[3] Siemens CT, Siemens AG, Munich, Germany

Table of Contents

1 Appendix Structure

The structure of the appendix we organize as follows. In Section 2 we introduce a axillary notions and properties that we use to prove main lemmas and theorems from the paper. Then in Section 3 we show the proofs for Lemma 1, Theorem 2 that are proving SHACL rewritings for with *DL-Lite*$_R^-$ and then for Lemma 2, Theorem 3 that are proving rewritings for *DL-Lite*$_R$. When it is clear from the context we just named them as *DL-Lite* ontologies or just ontologies.

We remind the reader that expression $\langle \mathcal{O}, \mathcal{G}, \mathcal{C} \rangle \models s(v)$ is defined as follows. For each minimal model of $\mathcal{G}'$ of $\langle \mathcal{O}, \mathcal{G} \rangle$ there exists a satisfying shape assignment σ over $\mathcal{G}'$ to shape names in $\mathcal{C}$. Here, σ is satisfying shape assignment means if for each node v in $\mathcal{G}$ and each assigned shape s to v, written, $s \in \sigma(v)$, the shape formula ϕ_s evaluates to true over given graph $\mathcal{G}$, written $[\![\phi_s]\!]^{\mathcal{G},v,\sigma} = \textit{true}$, or shortly $\sigma(v, s) = \textit{true}$ when it is clear from the context.

2 Auxiliary Properties

First we show that to check validity over all minimal models it is sufficient to check validity over the canonical model. In particular, this holds for our two cases for which we establish SHACL rewritings: *(i)* DL-Lite without existentials and whole SHACL and *(ii)* DL-Lite and positive SHACL.

First we introduce an axillary property that shows that it is sufficient to consider only satisfiable KGs.

Lemma 4. *Let $\mathcal{O}$ be a DL-Lite ontology and $\mathcal{G}$ a graph. If $\langle \mathcal{O}, \mathcal{G} \rangle$ is unsatisfiable then for any shape s and any node in v in $\mathcal{G}$ there is exists no satisfying shape assignment over $\mathcal{G}$ and the set of shapes $\mathcal{C}_\mathcal{O}$ that validates $s(v)$.*

Proof. Assume that $\langle \mathcal{O}, \mathcal{G} \rangle$ is unsatisfiable. Wlog we assume that cause of being unsatisfiable is the following: it holds $\langle \mathcal{O}, \mathcal{G} \rangle \models C(a) \wedge D(a)$ for some node a in $\mathcal{G}$ and some basic concepts C and D, and at same time $\mathcal{O} \models C \sqsubseteq \neg D$ (similarly it can be shown for role disjointness).

Since $\langle \mathcal{O}, \mathcal{G} \rangle \models C(a) \wedge D(a)$ we have (from the properties on PERFREF in [1]) that $\mathcal{G} \models \text{PERFREF}(C(a) \wedge D(a), \mathcal{O})$. Then, since we have $\tau_{s_{C \sqsubseteq \neg D}} = \text{PERFREF}(C(x) \wedge D(x), \mathcal{O})$ it follows that a is the target of shape $s_{C \sqsubseteq \neg D}$. On the other hand $\phi_{s_{C \sqsubseteq \neg D}} = \bot$, thus for every shape assignment σ it must be $\sigma(s_{C \sqsubseteq \neg D}) = \bot$, i.e., there exist no satisfying assignment for $s_{C \sqsubseteq \neg D}$. Hence, there exist no satisfying assignment over $\mathcal{G}$ and $\mathcal{C}_\mathcal{O}$ that also validates $s(v)$. □

Lemma 5 (canonical model characterization I). *Let $\mathcal{O}$ be a DL-Lite ontology without existential rules, C a basic concept, $\mathcal{G}$ a graph , v a node in $\mathcal{G}$, $\mathcal{C}$ set of shapes and s shape in $\mathcal{C}$. Then if $\langle \mathcal{O}, \mathcal{G} \rangle$ is satisfiable we have that the following holds: $\langle \mathcal{O}, \mathcal{G}, \mathcal{C} \rangle \models \phi_s(v)$ iff $can(\mathcal{O}, \mathcal{G}), \mathcal{C} \models \phi_s(v)$.*

Proof. Since $\mathcal{O}$ is without existential rules, $\langle \mathcal{O}, \mathcal{G} \rangle$ has the unique minimal model and that model is the same as canonical model. Thus the property follows from there directly. □

If a DL-Lite ontology has existentials it can have many different minimal models. The canonical model is one them, but the canonical model is also the most "conservative" one,

in the sense that formulas that hold over canonical model would hold in all models but not vice-versa. To show this property we rely on the notion of homomorphism between models.

Definition 1 (FO-homomorphism). *Given two interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ over the same set $\mathcal{P}$ of predicate symbols, a* homomorphism *μ from $\mathcal{I}$ to $\mathcal{J}$ is a mapping $\mu : \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ such that, for each predicate $P \in \mathcal{P}$ of arity n and each tuple $(o_1, \ldots, o_n) \in (\Delta^{\mathcal{I}})^n$, if $(o_1, \ldots, o_n) \in P^{\mathcal{I}}$, then $(\mu(o_1), \ldots, \mu(o_n)) \in P^{\mathcal{J}}$.*

In [1], the authors showed that a canonical model of a KB can be homomorphically mapped to any other model of that KB. Now we extend this notion to shapes.

Definition 2 (SHACL-homomorphism). *Given two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with set of constants Δ_1 and Δ_2 respectively, and set of shapes $\mathcal{C}$, a* SHACL-homomorphism *μ from $\mathcal{G}$ and $\mathcal{G}'$ is a mapping $\mu : \Delta_1 \to \Delta_2$ such that, for each shape $s \in \mathcal{C}$ and each constant $v \in \Delta_1$, if $\mathcal{G}_1, \mathcal{C} \models \phi_s(v)$ then $\mathcal{G}_2, \mathcal{C} \models \phi_s(\mu(v))$.*

Lemma 6 (canonical homomorphism for positive shapes). *Let $\mathcal{O}$ be an ontology, $\mathcal{G}$ a graph, and let $\mathcal{M}$ be a minimal model of $\langle \mathcal{O}, \mathcal{G} \rangle$. Let $\mathcal{C}$ be a set of positive shapes. Then, there is a SHACL-homomorphism from $can(\mathcal{O}, \mathcal{G})$ to $\mathcal{M}$ given $\mathcal{C}$. In particular, there exists a SHACL-homomorphism that maps every node from $\mathcal{G}$ to itself.*

Proof. From [1], we have that there exists a homomorphism μ from $can(\mathcal{O}, \mathcal{G})$ to $\mathcal{M}$ such that for a basic concept C and node v it holds if $C(v) \in can(\mathcal{O}, \mathcal{G})$ (resp. $R(v_1, v_2) \in can(\mathcal{O}, \mathcal{G})$) then $C(\mu(v)) \in \mathcal{M}$ (resp. $R(\mu(v_1), \mu(v_2)) \in \mathcal{M}$. In particular, it is possible to select μ such that $\mu(v) = v$ for $v \in \mathcal{G}$. We also notice that μ has to be surjective; otherwise the μ-image of $can(\mathcal{O}, \mathcal{G})$ would be "smaller" minimal model than $\mathcal{M}$ which is contradiction.

Assume now that $can(\mathcal{O}, \mathcal{G}), \mathcal{C} \models \phi_s(v)$ for some shape s from $\mathcal{C}$ and node v in $can(\mathcal{O}, \mathcal{G})$. Let σ be a satisfying assignment for $can(\mathcal{O}, \mathcal{G}), \mathcal{C}$ such that $[\![\phi_s]\!]^{can(\mathcal{O},\mathcal{G}),\sigma,v} = \mathit{true}$. We define an assignment σ' over $\mathcal{M}, \mathcal{C}$ in the following way: for a shape s_1 and node v_1 in $\mathcal{M}$ we set $s_1 \in (v_1, \sigma')$ iff exists a node v_2 in $can(\mathcal{O}, \mathcal{G})$ such that $\mu(v_2) = v_1$ and $s_1 \in (v_2, \sigma)$. Now analyzing different cases for s: $\phi_s = \top$, $\phi_s = I$, $\phi_s = \geq_k R.s_1$, $\phi_s = s_1 \wedge s_2$, $\phi_s = s_1 \vee s_2$ and $\phi_s = \text{EQ}(r_1, r_2)$, it is not hard to show that $[\![\phi_s]\!]^{can(\mathcal{O},\mathcal{G}),\sigma,v} = \mathit{true}$ iff $[\![\phi_s]\!]^{\mathcal{M},\sigma',\mu(v)} = \mathit{true}$. □

Using the lemmas above we are able to show the following property of the canonical model.

Lemma 7 (canonical model characterization II). *For a DL-Lite TBox $\mathcal{O}$, a basic concept C, a graph G, node v in $\mathcal{G}$, set S of positive shapes and shape s defined in $\mathcal{C}$ we have that: $\langle \mathcal{O}, \mathcal{G}, \mathcal{C} \rangle \models \phi_s(v)$ iff $can(\mathcal{O}, \mathcal{G}), \mathcal{C} \models \phi_s(v)$.*

Proof. ($\Leftarrow$) Statement $\langle \mathcal{O}, \mathcal{G}, \mathcal{C} \rangle \models \phi_{s(v)}$ is true if $\mathcal{M}, \mathcal{C} \models \phi_{s(v)}$ is true each minimal models $\mathcal{M}$. Since $can(\mathcal{O}, \mathcal{G})$ is one minimal model the claim follows directly.

($\Rightarrow$) Let $\mathcal{M}$ be a minimal model of $\langle \mathcal{O}, \mathcal{G} \rangle$. From Lemma 6 we have that there is a homomorphism μ from $can(\mathcal{O}, \mathcal{G})$ to $\mathcal{M}$ such that $\mathcal{M}, \mathcal{C} \models \phi_s(\mu(v))$ where $\mu(v) = v$. □

The following lemma characterizes "virtual" shapes.

Lemma 8. *Assume we are given a DL-Lite ontology $\mathcal{O}$, a graph $\mathcal{G}$, node v in $\mathcal{G}$, a set $\mathcal{C}$ of positive shapes and shape s defined in $\mathcal{C}$. Then let $\mathcal{C}''$, $\mathcal{C}_\mathcal{O}$, $\mathcal{C}^v_\mathcal{O}$ and $\mathcal{C}_\mathcal{O}$ be sets of shapes constructed from $\mathcal{C}$ and $\mathcal{O}$ as defined in Algorithm 2. It holds:*

$\mathcal{G}, \mathcal{C}'' \cup \mathcal{C}_\mathcal{O} \cup \mathcal{C}^v_\mathcal{O} \cup \mathcal{C}^s_\mathcal{O} \models \phi_{s^{virtual}}(v)$ *iff*
there is a node a_1 in $can(\mathcal{O}, \mathcal{G})$ with witness v s.t. $can(\mathcal{O}, \mathcal{G}), \mathcal{C} \models \phi_s(a_1)$.

Proof. ($\Rightarrow$) Let σ' be a satisfying assignment for $\mathcal{G}, \mathcal{C}'' \cup \mathcal{C}_\mathcal{O} \cup \mathcal{C}^v_\mathcal{O} \cup \mathcal{C}^s_\mathcal{O}$.

We set σ to be an assignment for $\mathcal{C}$ over $can(\mathcal{O}, \mathcal{G})$ to be based on σ' by replacing each $s^{virtual}$ with s and eliminating assignments for shapes in $\mathcal{C}_\mathcal{O} \cup \mathcal{C}^v_\mathcal{O} \cup \mathcal{C}^s_\mathcal{O}$.

From Lemma 3 we have that each shape $s^{\text{succ}}_{R,P}$ in $\mathcal{C}^s_\mathcal{O}$ is true in v iff there are nodes a_1, a_2, a_3 in $can(\mathcal{O}, \mathcal{G})$ with witness v s.t. $can(\mathcal{O}, \mathcal{G}) \models R(a_1, a_2)$ and $can(\mathcal{O}, \mathcal{G}) \models R(a_2, a_3)$, and *(ii)* each shape $s^{virtual}_C$ in $\mathcal{C}^v_\mathcal{O}$ is true in v iff there is a node a_1 in $can(\mathcal{O}, \mathcal{G})$ with witness v s.t. $can(\mathcal{O}, \mathcal{G}) \models C(a_1)$. Thus, for cases $\phi_s = \top$, $\phi_s = I$, $\phi_s = \geq_1 \mathsf{a}.C_1$ for some basic concept C_1 it is clear that $\sigma(v, s)$ is true iff $\sigma'(v, s')$ is true.

Cases $\phi_s = \phi_1 \wedge \phi_2$, $\phi_s = \phi_1 \vee \phi_2$ and $\phi_s = \text{EQ}(r_1, r_2)$ can be shown in the same way as in the proof of Theorem 1.

For $\phi_s = (\geq_k R.s_1)$ we have the following rewriting: $\phi'_s = (\geq_k R'.s'_1) \vee s^{virtual}$. Then $\phi_{s^{virtual}}$ evaluates to true if either $(\geq_k R'.s'_1)$ is true or $\phi_{s^{virtual}}$ is true.

Assume first that $\geq_k R'.s'_1$ evaluates to true. Since $R' = R_1 | \ldots | R_i | \ldots | R_n$ where $\mathcal{O} \models R_i \sqsubseteq R$ there must be at least k facts of the form (v, R_i, v_i) in $\mathcal{G}$. So there are at least k R-successors in $can(\mathcal{O}, \mathcal{T})$ (actually, there are already in $cl(\mathcal{O}, \mathcal{G})$). The we set $a_1 = v$ and $\geq_k R.s_1(v)$ evaluates to true since $s_1 \in \sigma'(v)$ iff $s_1 \in \sigma(v)$.

Assume now that $s^{virtual}$ evaluates to true. First we analyze the shapes of the form $s_{\exists R, s_1}$. We observe that such shapes are forming a propositional Datalog program where $\phi_{s_{\exists R,s_1}} = s_{\exists R,s_2} \wedge s_{\exists R,s_3}$ if $\phi_{s_1} = s_2 \wedge\ s_3$ (and similarly for $\vee$) and $\phi_{\exists R,s_1} = s^{succ}_{R,P}$ if $\phi_{s_1} = (\geq_k P.s_2)$. The program that only depends on $\mathcal{O}$ and not on shapes in $\mathcal{C}$ since literals are defined as: $s^{succ}_{R,P} = s^{virtual}_{\exists R}$ if $\mathcal{O} \models \exists R^- \sqsubseteq \exists P$ and $s^{succ}_{R,P} = \bot$ if $\mathcal{O} \models \exists R^- \sqsubseteq \exists P$. Special cases are $\phi_{s_1} = \geq_1 \mathsf{a}.A$ and $\phi_{s_1} = \geq_1 R.\top$. then $\phi_{\exists R,s_1} = s^{virtual}_A$ and $\phi_{\exists R,s_1} = s^{virtual}_{\exists R}$ respectively.

Overall, formula of a shape $\phi_{\exists R,s_1}$ evaluates to true (in any node in $\mathcal{G}$) if *(i)* all successors necessary for s_1 and other shapes references by s_1 exist in canonical model, *(ii)* "terminating" shape with formulas $\phi_{s_1} = \geq_1 \mathsf{a}.A$ and $\phi_{s_1} = \geq_1 R.\top$ are also true in v.

Second, shape $s^{virtual}_{\exists R}$ evaluates to true in v and thus from Lemma 3 we have that canonical model contains a tree with root in v that branch is of the form (v, R, v') for some fresh v' introduced when generating canonical model. From Lemma 3 we have that there exists a node a_1 in $can(\mathcal{O}, \mathcal{G})$ where $s_{\exists R}$ evaluates to true.

Thus all "necessary successors" of s are in $can(\mathcal{O}, \mathcal{G})$ and we set σ accordingly so $\sigma(a_1, s) = \textit{true}$.

($\Leftarrow$) Let σ be a satisfying assignment for $can(\mathcal{O}, \mathcal{G}), \mathcal{C} \models \phi_s(v)$. Let now σ' be the same as σ where each shape name s is replaced with s'. In addition, we extend σ' to shape names in $\mathcal{C}_\mathcal{O} \cup \mathcal{C}^v_\mathcal{O} \cup \mathcal{C}^s_\mathcal{O}$. In particular, a shape $s^{succ}_{R,P}$ in $\mathcal{C}^s_\mathcal{O}$ are assigned to every node in $\mathcal{G}$ such that $\sigma'(s^{succ}_{R,P}, v) = \textit{true}$ iff $\mathcal{O} \models \exists R^- \sqsubseteq \exists P$. This determines the assignment for shapes of kind $s_{\exists R,s_1}$ by setting $\sigma'(s_{\exists R,s_1}, v) = \textit{true}$ iff propositional $s_{\exists R,s_1}$ is *true* in the Datalog program that contains shapes $\mathcal{C}^s_\mathcal{O}$ and shapes of the kind "$s_{\exists R,s_2}$". For the

shapes on form "$\phi_{s_C^{virtual}}$" we set to evaluate to *true* in a node v iff there is a node a_1 in $can(\mathcal{O},\mathcal{G})$ with witness v s.t. $can(\mathcal{O},\mathcal{G}) \models C(a_1)$. Then, we extend σ' to shape names in $\mathcal{C}_\mathcal{O}$ and "virtual" $\mathcal{C}_\mathcal{O}$. For each shape name s_A and node $v \in \mathcal{G}$ we add $s_A \in \sigma'(v)$ iff $(v, \mathsf{a}, A) \in cl(\mathcal{O},\mathcal{G})$. Similarly, we add $s_{\exists R} \in \sigma'(v)$ iff $(v, R, v') \in cl(\mathcal{O},\mathcal{G})$ for $v, v' \in \mathcal{G}$. Similarly, we extend σ' for shape of the form $s_A^{virtual}$ and $s_{\exists R}^{virtual}$.

Finally, one can show for each shape type in $\mathcal{C}'' \cup \mathcal{C}_\mathcal{O} \cup \mathcal{C}_\mathcal{O}^v \cup \mathcal{C}_\mathcal{O}^s$ that σ' is a satisfying assignment given graph $\mathcal{G}$ and that $\sigma'\phi_{s^{virtual}}(v)$. This can be shown similarly to the opposite direction by analyzing each shape type and operator. □

3 Proofs of the Properties from the Paper

3.1 Proof of Lemma 1

Proof. ($\Rightarrow$) By assumption there exists a satisfying assignment σ over shape names in $\mathcal{C}_\mathcal{O} \cup \mathcal{C}''$ such that $[\![\phi_{s_C}]\!]^{\mathcal{G},v,\sigma_\mathcal{M}} = \textit{true}$. Wlog assume C is an atomic concept A By definition $\phi_{s_A} = (\geq_1 \mathsf{a}.A) \vee \bigvee_{\mathcal{O} \models C' \sqsubseteq A} s_{C'}$ and thus either $(\geq_1 \mathsf{a}.A)$ evaluates to true or for some C', $(\geq_1 \mathsf{a}.C')$ evaluates to true if $C' = B$ or $(\geq_1 R.\top)$ if $C' = \exists R$. In either case we have $\mathcal{G} \models C'(v)$. Then since $\mathcal{O} \models C' \sqsubseteq A$ by construction of the canonical model it must be $(v, \mathsf{a}, A) \in can(\mathcal{O},\mathcal{G})$.

($\Leftarrow$) We construct an assignment for $\mathcal{C}_\mathcal{O}$ in the following way: $s_C \in \sigma(v)$ iff $(v, \mathsf{a}, C) \in can(\mathcal{O},\mathcal{G})$. Now we have to show that it is a satisfying assignment. Wlog $C = A$ where A is a basic concept. Then by definition target query of $\tau_{s_A} = \textsc{PerfRef}(A)$. Assume that v' is returned by $\textsc{PerfRef}(A)$ over $\mathcal{G}'$. Then must exits a fact $(v', \mathsf{a}, A') \in \mathcal{G}$ for a atomic concept A' or $(v, R, v'') \in \mathcal{G}$ for some role R such that $\mathcal{O} \models A' \sqsubseteq A$ or $\mathcal{O} \models \exists R \sqsubseteq A$ respectively. If the former then $\sigma(v, s_{A'}) = \textit{true}$, or if the latter $\sigma(v, s_{\exists R}) = \textit{true}$. In either case, it follows that $\sigma(v, s_A) = \textit{true}$. Similarly, we can show the case $C = \exists R$. □

3.2 Proof of Theorem 2

Proof. First we consider the case when $\langle \mathcal{O}, \mathcal{G} \rangle$ is unsatisfiable. Then there exists no minimal model thus the left-hand side is false. On the other hand from Lemma 4 there is no satisfying assignment for $\mathcal{G}, \mathcal{C}_\mathcal{O}$ and hence neither for $\mathcal{G}, \mathcal{C}_\mathcal{O} \cup \mathcal{C}''$. Thus, the right-hand side is also false.

Assume now that $\langle \mathcal{O}, \mathcal{G} \rangle$ is satisfiable. Then, from Lemma 5 we have that: $\langle \mathcal{O}, \mathcal{G}, \mathcal{C} \rangle \models \phi_s(v)$ iff $can(\mathcal{O},\mathcal{G}), \mathcal{C} \models \phi_s(v)$. Thus it is sufficient to show:

$$can(\mathcal{O},\mathcal{G}), \mathcal{C} \models \phi_s(v) \text{ iff } \mathcal{G}, \mathcal{C}_\mathcal{O} \cup \mathcal{C}'' \models \phi_{s'}(v).$$

($\Rightarrow$) Let σ be a satisfying assignment for $can(\mathcal{O},\mathcal{G}), \mathcal{C} \models \phi_s(v)$. We define σ' as a shape assignment for $\mathcal{C}' \cup \mathcal{C}_\mathcal{O}$ over $\mathcal{G}$. In particular, let σ' be the same as σ where each shape name s is replaced with s'. In addition, we extend σ' to shape names in $\mathcal{C}_\mathcal{O}$. For each shape name s_A and node $v \in \mathcal{G}$ we add $s_A \in \sigma'(v)$ iff $(v, \mathsf{a}, A) \in can(\mathcal{O},\mathcal{G})$. Similarly, we add $s_{\exists R} \in \sigma'(v)$ iff $(v, R, v') \in can(\mathcal{O},\mathcal{G})$ for $v, v' \in \mathcal{G}$. We observe that σ' is an assignment from G to shape names in $\mathcal{C}_\mathcal{O} \cup \mathcal{C}''$.

First we show that σ' is satisfying assignment for shape names in $\mathcal{C}_\mathcal{O}$. From Lemma 1 we have that $\mathcal{G}, \mathcal{C}_\mathcal{O} \models \phi_{s_C}(v)$ iff $(v, \mathsf{a}, C) \in can(\mathcal{O},\mathcal{G})$. Wlog assume $C = A$ then for

each node v such that $(v, \mathsf{a}, A) \in can(\mathcal{O}, \mathcal{G})$ we have that $s_A \in \sigma'(v)$ (and similarly for the shape names of kind "$s_{\exists R}$"). Thus σ' is an satisfying assignment for shapes in $\mathcal{C}_\mathcal{O}$.

Now we are going to show that σ' is also a satisfying assignment such that

$$\sigma(v, s) = \textit{true} \text{ iff } \sigma'(v, s') = \textit{true}$$

We note that $\sigma(v, s) = [\![\phi_s]\!]^{can(\mathcal{O},\mathcal{G}),v,\sigma}$ and $\sigma'(v, s') = [\![\phi_{s'}]\!]^{\mathcal{G},v,\sigma'}$.

We analyze s case by case. For the cases: $\phi_s = \top$, $\phi_s = I$, and $\phi_s = \neg s_1$ it is clear that $\sigma(v, s)$ is true iff $\sigma'(v, s')$ is true.

If $\phi_s =\geq_k R.s_1$ then $\sigma(v, s) = \textit{true}$ if there are at least k R-successors in the canonical model. By construction of canonical model there must be then at least k facts of the form (v, R_i, v_i) in $can(\mathcal{O}, \mathcal{C})$ such that $\mathcal{O} \models R_i \sqsubseteq R$. Since, $\phi_{s'} = \geq_k R'.s'_1$ where $R' = R_1|\ldots|R_i|\ldots|R_n$ then v must have at least k R'-successors. Moreover, $\sigma(v_i, s_1) = \textit{true}$ iff $\sigma'(v_i, s'_1) = \textit{true}$. Then it holds $\sigma(v, s') = \textit{true}$.

Two syntactically special cases are $\phi_s =\geq_1 \mathsf{a}.A$ and $\phi_s =\geq_k R.\top$, however they are not covered by the case $\geq_k R.s_1$. The reasons is that concept inclusions in $\mathcal{O}$ that create new facts in the canonical model Assume $s_\phi =\geq_1 \mathsf{a}.A$. If $\sigma(v, s) = \textit{true}$ then according to the construction of $can(\mathcal{O}, \mathcal{G})$ there must be a finite chain of concept and role inclusion that starting from a fact of form (v, a, A_0) or (v, R_0, v_1) in $\mathcal{G}$ creates the fact (v, a, A) in $can(\mathcal{O}, \mathcal{G})$. If (v, a, A_0) is the case we have that $\mathcal{O} \models A_0 \sqsubseteq A$, otherwise we have that $\mathcal{O} \models \exists R \sqsubseteq A$. In either case, we have that $[\![\phi_{s'}]\!]^{\mathcal{G},v,\sigma'} = \textit{true}$. Similarly, we can reason in the case $\phi_s =\geq_k R.\top$.

If $\phi_s = \text{EQ}(r_1, r_2)$ then we set $\phi_{s'} = \text{EQ}(r'_1, r'_2)$ where r'_1 and r'_2 are obtained by replacing each occurrence of property R with $R_1 \mid \ldots \mid R_n$ where $\mathcal{O} \models R_i \sqsubseteq R$. Then a fact (v_1, R, v_2) is in $can(\mathcal{O}, \mathcal{G})$ iff for some i a fact (v_1, R_i, v_2) is in $\mathcal{G}$, that is R' returns v_1 and v_2 over $\mathcal{G}$. Thus, paths in r_1 corresponds to one in r'_1 and the same for r_2, and so $\sigma(v, s)$ is true iff $\sigma'(v, s')$ is true.

If $\phi_s = s_1 \wedge s_2$. then $\sigma(v, s) = \textit{true}$ iff $\sigma(v, s_1) = \sigma(v, s_2) = \textit{true}$ which by definition of σ is the case iff $\sigma'(v, s'_1) = \sigma'(v, s'_2) = \textit{true}$ iff $\sigma'(v, s') = \textit{true}$. Similarly, one can show the same property for the case $\phi_s = \phi_1 \vee \phi_2$.

($\Leftarrow$) Let σ' be an satisfying assignment for $\mathcal{G}, \mathcal{C}_\mathcal{O} \cup \mathcal{C}'' \models \phi_{s'}(v)$. We construct an satisfying assignment σ for $can(\mathcal{O}, \mathcal{G}), \mathcal{C} \models \phi_s(v)$. We set that σ is obtained from σ' by replacing each s' with s and eliminating shape names from $\mathcal{C}_\mathcal{O}$. Then analogously to the direction ($\Rightarrow$) for each node v in $\mathcal{G}$ and shape name we can show that it holds: $\sigma(v, s) = \textit{true}$ iff $\sigma'(v, s') = \textit{true}$.

We analyze s case by case. For the cases: $\phi_s = \top$, $\phi_s = I$, and $\phi_s = \neg s_1$ it is clear that $\sigma(v, s)$ is true iff $\sigma'(v, s')$ is true.

Again we have two special cases when $s_\phi =\geq_1 \mathsf{a}.A$ and $s_\phi =\geq_k R.\top$. Let us consider that $s_\phi =\geq_1 \mathsf{a}.A$ and that $s' \in \sigma(v)$. Then from Lemma 1 we have that $s' \in \sigma(v)$ iff $(v, \mathsf{a}, A) \in can(\mathcal{O}, \mathcal{G})$ and thus $[\![s_\phi]\!]^{can(\mathcal{O},\mathcal{G}),\sigma,v} = \textit{true}$. So, $\sigma(v, s) = \textit{true}$. Similarly, we can show the same property for the case $s_\phi =\geq_k R.\top$.

Finally for the cases $\phi_s = \text{EQ}(r_1, r_2)$, $\phi_s = \phi_1 \wedge \phi_2$, $\phi_s = \phi_1 \vee \phi_2$, $\phi_s =\geq_k R.s_1$ we can show similarly to the direction ($\Rightarrow$). □

3.3 Proof of Lemma 3

Proof. We remind the reader that the virtual shapes are defined as follows:

$$\phi_{s_A^{virtual}} = s_A \vee \bigvee_{\mathcal{O} \models C \sqsubseteq A} s_C^{virtual}, \quad \phi_{s_{\exists R}^{virtual}} = s_{\exists R} \vee \bigvee_{\mathcal{O} \models C \sqsubseteq \exists R} s_C^{virtual} \vee \bigvee_{\mathcal{O} \models R' \sqsubseteq R} s_{\exists R'}^{virtual}.$$

Then it is sufficient to show that there exists a satisfying assignment over $\mathcal{C}_\mathcal{O} \cup \mathcal{C}_\mathcal{O}^v \cup \mathcal{C}_\mathcal{O}^s$ that verifies $s_C^{virtual}(v)$ iff there is a node v' in $\mathcal{G}'$ with v as witness such that $\mathcal{G}' \models C(v')$. This can be shown in a similar fashion to the proof of Lemma 1. The second claim can also shown in a similar way. □

3.4 Proof of Theorem 3

Proof. First we consider the case when $\langle \mathcal{O}, \mathcal{G} \rangle$ is unsatisfiable. Then there exists no minimal model thus the left-hand side is false. On the other hand from Lemma 4 there is no satisfying assignment for $\mathcal{G}, \mathcal{C}_\mathcal{O}$ and hence neither for $\mathcal{G}, \mathcal{C}'$. Thus, the right-hand side is also false.

Assume now that $\langle \mathcal{O}, \mathcal{G} \rangle$ is satisfiable. Then, from Lemma 7 we have that: $\langle \mathcal{O}, \mathcal{G}, \mathcal{C}' \rangle \models \phi_s(v)$ iff $can(\mathcal{O}, \mathcal{G}), \mathcal{C} \models \phi_s(v)$. Thus it is sufficient to show:

$$can(\mathcal{O}, \mathcal{G}), \mathcal{C} \models \phi_s(v) \text{ iff } \mathcal{G}, \mathcal{C}' \models \phi_{s'}(v).$$

($\Rightarrow$) Let σ be a satisfying assignment for $\mathcal{C}$ over $can(\mathcal{O}, \mathcal{G})$ such that $[\![\phi_s]\!]^{can(\mathcal{O},\mathcal{G}),v,\sigma} = \textit{true}$. We construct a satisfying assignment σ' for $\mathcal{C}'$ over $\mathcal{G}$ such that $[\![\phi_s]\!]^{\mathcal{G},v,\sigma'} = \textit{true}$.

By definition $\mathcal{C}' = \mathcal{C}_\mathcal{O} \cup \mathcal{C}_\mathcal{O}^v \cup \mathcal{C}_{\mathcal{O},\mathcal{C}}^s \cup \mathcal{C}''$ where shape names in $\mathcal{C}''$ are referencing shape in $\mathcal{C}_\mathcal{O} \cup \mathcal{C}_\mathcal{O}^v \cup \mathcal{C}_{\mathcal{O},\mathcal{C}}^s$ but not vice-versa. So we first define shape assignment for $\mathcal{C}_\mathcal{O} \cup \mathcal{C}_\mathcal{O}^v \cup \mathcal{C}_{\mathcal{O},\mathcal{C}}^s$ and then we extend it to $\mathcal{C}''$.

From Lemma 3 we have that *(i)* $\mathcal{G}, \mathcal{C}_\mathcal{O} \cup \mathcal{C}_\mathcal{O}^v \cup \mathcal{C}_\mathcal{O}^s \models \phi_{s_{R,P}^{succ}(v)}$ iff there are nodes a_1, a_2, a_3 in $can(\mathcal{O}, \mathcal{G})$ with witness v s.t. $can(\mathcal{O}, \mathcal{G}) \models R(a_1, a_2)$ and $can(\mathcal{O}, \mathcal{G}) \models R(a_2, a_3)$, and *(ii)* $\mathcal{G}, \mathcal{C}_\mathcal{O} \cup \mathcal{C}_\mathcal{O}^v \cup \mathcal{C}_\mathcal{O}^s \models \phi_{s_C^{virtual}}(v)$ iff there is a node a_1 in $can(\mathcal{O}, \mathcal{G})$ with witness v s.t. $can(\mathcal{O}, \mathcal{G}) \models C(a_1)$. From the proof of Lemma 3 we have that there exists a satisfying assignment σ_1 that is satisfying assignments for the both claims above.

Let σ_2 be the satisfying assignment for $\mathcal{C}''$ that is the same as σ where each s is replaced with s'. Finally, we set $\sigma' = \sigma_1 \cup \sigma_2$. Obviously, σ' is satisfying assignment for shape names in $\mathcal{C}_\mathcal{O} \cup \mathcal{C}_\mathcal{O}^v \cup \mathcal{C}_\mathcal{O}^s$. It remains to show that it is satisfying assignment for $\mathcal{C}''$.

We show this for each case. For the cases: $\phi_s = \top$ and $\phi_s = I$ it is clear that $\sigma(v, s)$ is true iff $\sigma'(v, s')$ is true.

Cases $\phi_s = \phi_1 \wedge \phi_2$, $\phi_s = \phi_1 \vee \phi_2$ and $\phi_s = \text{EQ}(r_1, r_2)$ can be shown in the same way as in the proof of Theorem 1.

If $\phi_s = \geq_k R.s_1$ the translation is $\phi_s' = (\geq_k R'.s_1') \vee s^{virtual}$ where $R' = R_1 | \ldots | R_i | \ldots | R_n$. By definition $\sigma(v, s) = \textit{true}$ if there are at least k R-successors in the canonical model. By construction of canonical model there must be then at least k facts of the form (v, R_i, v_i) in $can(\mathcal{O}, \mathcal{C})$ such that $\mathcal{O} \models R_i \sqsubseteq R$. We distinguish between two cases. Either all facts (v, R_i, v_i) are in the closure $cl(\mathcal{O}, \mathcal{G})$ and thus (as shown in Theorem 2) formula $\geq_k R'.s_1'$ evaluates to *true*. Otherwise, there are R-successors necessary for s to evaluate to *true* that are in $can(\mathcal{O}, \mathcal{G}) \setminus cl(\mathcal{O}, \mathcal{G})$. The

minimal canonical model introduces at most one node a_1 for each node a in $can(\mathcal{O}, \mathcal{C})$ such that $R(v, a) \in can(\mathcal{O}, \mathcal{C})$. Moreover, all nodes necessary to evaluate $\sigma(v, s)$ to *true* are located on the tree t in $can(\mathcal{O}, \mathcal{G}) \setminus cl(\mathcal{O}, \mathcal{G})$ that is rooted in v. That is, for each shape s_1 referenced by s that evaluates to *true* on t with σ there exists a node a_1 on t such that $can(\mathcal{O}, \mathcal{G}), \mathcal{C} \models \phi_{s_1}(a_1)$. Now we apply Lemma 8, and get that it holds $G, \mathcal{C}'' \cup \mathcal{C}_{\mathcal{O}} \cup \mathcal{C}^v_{\mathcal{O}} \cup \mathcal{C}^s_{\mathcal{O}} \models \phi_{s_1^{virtual}}(v)$. If we set $s_1 = s$ and the node $a_1 = v$ and apply Lemma 8 again, we have that $\sigma'(s^{virtual}, v) = \textit{true}$.

($\Leftarrow$) Let σ' be a satisfying assignment for $\mathcal{G}, \mathcal{C}'' \cup \mathcal{C}_{\mathcal{O}} \cup \mathcal{C}^v_{\mathcal{O}} \cup \mathcal{C}^s_{\mathcal{O}}$ such that $[\![\phi_s]\!]^{\mathcal{G}, v, \sigma'} = \textit{true}$. We set σ to be an assignment for $\mathcal{C}$ over $can(\mathcal{O}, \mathcal{G})$ to be based on σ' by replacing each $s^{virtual}$ with s and eliminating assignments for shapes in $\mathcal{C}_{\mathcal{O}} \cup \mathcal{C}^v_{\mathcal{O}} \cup \mathcal{C}^s_{\mathcal{O}}$.

Then this claim can be shown in a similar fashion to the direction "$\Rightarrow$" of Lemma 8 by applying Lemmas 3,4 and 8 for different types of shape definitions. □

Bibliography

[1] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *JAR*, 2007.