ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230250

Reconciling SHACL and Ontologies: Semantics and Validation via Rewriting

Shqiponja Ahmetaj[a], **Magdalena Ortiz**[b], **Anouk Oudshoorn**[a;*] **and Mantas Šimkus**[b]

[a]TU Wien, Austria
[b]Umeå University, Sweden

Abstract. OWL and SHACL are two prominent W3C standards for managing RDF graphs, the data model of the Web. They are used for different purposes and make different assumptions about the completeness of data: SHACL is used for expressing integrity constraints on complete data, while OWL allows inferring implicit facts from incomplete data; SHACL reasoners perform validation, while OWL reasoners do logical inference. Integrating these two tasks into one uniform approach is a relevant but challenging problem. The SHACL standard envisions graph validation in combination with OWL entailment, but it does not provide technical guidance on how to realize this. To address this problem, we propose a new intuitive semantics for validating SHACL constraints with OWL 2 QL ontologies based on a suitable notion of the *chase*. We propose an algorithm that rewrites a set of recursive SHACL constraints (with stratified negation) and an OWL 2 QL ontology into a stand-alone set of SHACL constraints that preserves validation for every input graph, which can in turn be evaluated using an off-the-shelf SHACL validator. We show that validation in this setting is EXPTIME-complete in combined complexity, but only PTIME-complete in data complexity, i.e., if the constraints and the ontology are fixed.

1 Introduction

SHACL and OWL are two prominent W3C standards for managing RDF data, the graph-based data model of the Web. They were specifically designed to target two different issues. OWL was standardized in parallel with RDF to address information incompleteness of RDF data by means of ontological axioms that complete the data with missing information. OWL and its profiles are based on *Description Logics (DLs)* [4] and make the open-world assumption (OWA), which intuitively means that the data only presents a partial description of the domain of interest and missing facts may also be true. RDF and OWL were soon adopted by increasingly many applications, including enterprise systems and critical areas such as healthcare, and making decisions based on correct data became particularly crucial.

To check the correctness of RDF data, W3C proposed the so-called *Shapes Constraint Language (or SHACL)* [17], a machine-readable constraint language for describing and validating RDF graphs. Unlike OWL, SHACL operates under the closed-world assumption (CWA) and assumes completeness of data. SHACL specifies the notion of a *shapes graph*, which consists of a set of shape constraints paired with the so-called *targets*, which is a selection of nodes of the data graph that must be validated against the constraints. The precise semantics of SHACL in the presence of recursion was not described in the W3C standard, which led to recent works that propose semantics based on first-order logic and logic programming [7, 15, 2].

Combining SHACL and OWL into a setting that allows to perform RDF data validation while taking into account the implicit facts inferred using an OWL ontology is a relevant but challenging problem. Indeed, the W3C SHACL specification envisions graph validation in the presence of OWL entailment but does not provide guidance on how to realize this.

To our knowledge, this has only been addressed in [16], which considers *positive* SHACL constraints only.

To see the benefits of taking into account ontologies when performing validation, consider an example of a simplistic database of pet owners containing the facts $hasWingedPet(linda, blu)$, $Bird(blu)$, $PetOwner(john)$, $hasPet(john, ace)$ and consider the simple constraint

$$petOwnerShape \leftarrow PetOwner \lor \exists hasPet$$

with the target $petOwnerShape(linda)$, which asks to verify whether $linda$ is a pet owner. Clearly, one would expect the input data to validate $linda$ as a pet owner given that she has a winged pet. However, this is not the case since the setting is missing the background knowledge that owning a winged pet implies owning a pet. The latter can be expressed through the ontological axiom $hasWingedPet \sqsubseteq hasPet$, which would allow us to obtain the desired validation result.

Identification of a proper semantics in this setting requires integrating the OWA of OWL and the CWA of SHACL. There are several proposals by the DL and database communities to relax the OWA and combine it with CWA [11, 14, 13, 10]. Another challenge when defining a validation semantics is dealing with the non-monotonic behavior of SHACL constraints due to the presence of negation. Roughly speaking, adding facts to the input data graph may cause a previously valid setting to become invalid. Such non-monotonic behavior is known when combining ontologies and negation in the so-called conjunctive queries or database constraints (see e.g., [5, 14, 8]).

The contributions of this paper can be summarized as follows:

○ We present a novel notion of SHACL validation in the presence of a DL-Lite$_\mathcal{R}$ ontology, the logic underlying OWL 2 QL [12]. Specifically, we consider *stratified* SHACL constraints, which support a limited form of recursion (specifically, limiting the interaction between recursion and negation). Our notion of stratification is derived

[*] Corresponding Author. Email: anouk.oudshoorn@tuwien.ac.at.

from the well-known class of stratified logic programs [3]. We note that the current SHACL standard defines the semantics only for non-recursive constraints, leaving the recursive case open.

∘ Since SHACL constraints involve negation, defining a semantics of validation in the presence of ontologies is challenging. In our approach, knowledge stemming from the ontology is included by completing the input data graph with additional facts to satisfy the ontological axioms. This completion is reminiscent to the *chase procedure* known in databases. We adopt a completion that is *austere* in the sense that only a minimal amount of new facts is added at each step of the procedure. Validation of SHACL constraints over a data graph in the presence of an ontology is defined as validation of the constraints in the possibly infinite *austere canonical model* that we introduce.

∘ Since validation in this paper is defined over the (potentially infinite) austere canonical model, its computational complexity is not obvious. We prove that this problem is decidable and is PTIME-complete in data complexity. This coincides with the complexity of stratified constraints over plain data graphs [2], and shows that adding a DL-Lite$_\mathcal{R}$ ontology actually does not incur additional costs in data complexity. This is different for combined complexity, which turns out to be EXPTIME-complete. The high combined complexity is somewhat surprising, since individually standard reasoning in DL-Lite$_\mathcal{R}$ ontologies and validation of stratified SHACL constraints over plain data graphs are tractable in combined complexity.

∘ Our upper bounds on complexity follow from a *constraint rewriting technique* that we introduce in this paper. We design an inference procedure that takes as input an ontology $\mathcal{T}$ together with a set $\mathcal{C}$ of stratified constraints, and produces as output a new set $\mathcal{C}_\mathcal{T}$ of stratified constraints such that $\mathcal{C}_\mathcal{T}$ alone is equivalent to the pair of $(\mathcal{T}, \mathcal{C})$, i.e. for validation, $\mathcal{C}_\mathcal{T}$ and $(\mathcal{T}, \mathcal{C})$ behave the same on any input data graph.[1] Thus an infinite austere canonical model does not need to be built explicitly in order to perform validation. The rewriting method is interesting in its own right as it opens the way to reuse standard SHACL validators to perform validation in the presence of ontologies, and it thus joins the ranks of other rewriting-based methods for reasoning with infinite structures (see, e.g., [9]).

2 Preliminaries

Data Graphs and Interpretations. Let $\mathrm{N}_C, \mathrm{N}_R, \mathrm{N}_I$ denote countably infinite, mutually disjoint sets of *concept names* (also known as *class names*), *role names* (or, *property names*), and *individuals* (or, *constants*), respectively. An *assertion* is an expression of the form $A(c)$ or of the form $p(c,d)$, where $A \in \mathrm{N}_C$, $p \in \mathrm{N}_R$ and $c, d \in \mathrm{N}_I$. An *ABox* (or a *data graph*) $\mathcal{A}$ is a finite set of assertions. An *interpretation* is a pair $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty set (called *domain*) and $\cdot^\mathcal{I}$ is a function that maps every $A \in \mathrm{N}_C$ to a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, every $p \in \mathrm{N}_R$ to a binary relation $p^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and every individual $c \in \mathrm{N}_I$ to an element $c^\mathcal{I} \in \Delta^\mathcal{I}$. The *canonical interpretation* $\mathcal{I}_\mathcal{A}$ for an ABox $\mathcal{A}$ is defined by setting $\Delta^{\mathcal{I}_\mathcal{A}} = \mathrm{N}_I$, $A^{\mathcal{I}_\mathcal{A}} = \{c \mid A(c) \in \mathcal{A}\}$ for all $A \in \mathrm{N}_C$, $p^{\mathcal{I}_\mathcal{A}} = \{(c,d) \mid p(c,d) \in \mathcal{A}\}$ for all $p \in \mathrm{N}_R$, and $c^{\mathcal{I}_\mathcal{A}} = c$ for every individual $c \in \mathrm{N}_I$.

Syntax and Standard Semantics of OWL 2 QL. As usual in the literature, we consider the underlying logic DL-Lite$_\mathcal{R}$. We let $\mathrm{N}_R^+ = \{r, r^- \mid r \in \mathrm{N}_R\}$ denote the set of *roles*. For $r \in \mathrm{N}_R$, we let $(r^-)^- = r$. A *basic concept* is a concept name $A \in \mathrm{N}_C$ or an expression of the form $\exists r$ such that $r \in \mathrm{N}_R^+$. A *concept inclusion* is an expression of the form $C \sqsubseteq D$ or $C \sqsubseteq \neg D$, where C, D are basic concepts. A *role inclusion* is an expression of the form $r \sqsubseteq s$ or $r \sqsubseteq \neg s$, where r, s are roles. A *TBox* $\mathcal{T}$ is a set of concept and role inclusions. We use $\mathcal{K}$ to denote a pair $(\mathcal{T}, \mathcal{A})$ of a TBox and an ABox. The semantics is defined in terms of interpretations $\mathcal{I}$ in the usual way. The notions of satisfaction and of model of an inclusion, a TBox, and an ABox are standard, as is the definition of logical entailment. In particular, we write $\mathcal{T} \models \gamma$ if every model of the TBox $\mathcal{T}$ is also a model of γ (where the latter may be an inclusion, an TBox, or an ABox). We say that $\mathcal{A}$ is *consistent* with $\mathcal{T}$ (or $\mathcal{K}$ is consistent) if there is a model of $\mathcal{A}$ and $\mathcal{T}$.

$$
\begin{aligned}
c^{\mathcal{I},S} &= \{c^\mathcal{I}\}\\
s^{\mathcal{I},S} &= \{c \mid s(c) \in S\}\\
(\neg s)^{\mathcal{I},S} &= \{c \in \Delta^\mathcal{I} \mid s(c) \notin S\}\\
A^{\mathcal{I},S} &= A^\mathcal{I}\\
(\neg A)^{\mathcal{I},S} &= \Delta^\mathcal{I} \setminus A^\mathcal{I}\\
(\varphi_1 \land \varphi_2)^{\mathcal{I},S} &= (\varphi_1)^{\mathcal{I},S} \cap (\varphi_2)^{\mathcal{I},S}\\
(\exists r.\varphi)^{\mathcal{I},S} &= \{e \in \Delta^\mathcal{I} \mid \exists e' : (e,e') \in r^\mathcal{I} \land e' \in \varphi^{\mathcal{I},S}\}
\end{aligned}
$$

Figure 1: Evaluating shape expressions

Basic SHACL notions. We let N_S denote a countably infinite set of *shape names*, disjoint from N_C, N_R, and N_I. A *shape expression* φ is an expression built according to the following grammar:

$$\varphi ::= c \mid A \mid \neg A \mid s \mid \neg s \mid \varphi \land \varphi \mid \exists r.\varphi,$$

where $c \in \mathrm{N}_I$, $A \in \mathrm{N}_C$, $s \in \mathrm{N}_S$, and $r \in \mathrm{N}_R^+$. In the following, we use $\exists r$ as a shorthand for $\exists r.\top$. A *shape constraint* is an expression of the form $s \leftarrow \varphi$, where $s \in \mathrm{N}_S$ and φ is a shape expression. A *shapes graph* is a pair $(\mathcal{C}, \mathcal{G})$, where $\mathcal{C}$ is a set of shape constraints and $\mathcal{G}$ is a set of shape atoms (*targets* or *goals*) of the form $s(c)$, where $s \in \mathrm{N}_S$ and $c \in \mathrm{N}_I$.

A *shape assignment* for an interpretation $\mathcal{I}$ is a set S of atoms of the form $s(c)$, where $s \in \mathrm{N}_S$ and $c \in \Delta^\mathcal{I}$. Given a shape expression φ, a shape assignment S and an interpretation $\mathcal{I}$ with $\mathrm{N}_I \subseteq \Delta^\mathcal{I}$, we define the set $\varphi^{\mathcal{I},S} \subseteq \Delta^\mathcal{I}$ inductively as shown in Figure 1.

Given an interpretation $\mathcal{I}$ and a shapes graph $(\mathcal{C}, \mathcal{G})$, we are interested in *validating* the targets in $\mathcal{G}$ given the information in $\mathcal{I}$ and the constraints in $\mathcal{C}$. Specifically, the goal of validation is to find a shape assignment S with $\mathcal{G} \subseteq S$ and such that S *satisfy* $\mathcal{C}$ and $\mathcal{I}$. For recursive SHACL, there are several validation semantics, i.e., different meanings of "satisfy". For illustration, we recall the *supported model semantics* from [7]. We say S is a *supported model* of $\mathcal{C}$ and $\mathcal{I}$, if the following holds: $s(a) \in S$ iff there exists $s \leftarrow \varphi \in \mathcal{C}$ with $a \in \varphi^{\mathcal{I},S}$. Validation under the supported model semantics consists of finding a supported model S for $\mathcal{I}$ and $\mathcal{C}$ such that $\mathcal{G} \subseteq S$. The supported model semantics suffers from *unfounded justifications*. Consider the singleton constraint set $\mathcal{C} = \{s \leftarrow \exists r.s\}$, the target set $\mathcal{G} = \{s(a)\}$, an interpretation $\mathcal{I}$ with $(a,a) \in r^\mathcal{I}$, and a shape assignment $S = s(a)$. Then S is a supported model of $\mathcal{I}$ and $\mathcal{C}$ and it validates $s(a)$. However, the validation of $s(a)$ is clearly not well-founded. Another semantics is the *stable model semantics* introduced in [2], which allows to eliminate such unfounded inferences.

[1] The impossibility of such a rewriting for SHACL with negation given in Theorem 1 of [16] does not apply to our semantics, nor to the minimal-model semantics adopted in [16], as acknowledged by the authors in personal communication.

3 Stratified SHACL

In this paper, we are interested in shape assignments where each shape atom has a proper justification. We do not resort to the full stable model semantics, as we are interested in cases with polynomial time data complexity. We focus on *stratified* sets of constraints which support negation, but also admit a partition of constraints (stratification) such that a justified shape assignment can be constructed by processing each partition individually. This approach is based on the syntax and semantics of stratified logic programs, and thus we mainly only adapt the notions from [3].

Definition 3.1. We say a shape name s *occurs negatively* in a shape constraint $s' \leftarrow \varphi$ if $\neg s$ occurs in φ. We say a shape name s is *defined* in a set $\mathcal{C}$ of constraints if $s \leftarrow \varphi \in \mathcal{C}$ for some φ.

A set $\mathcal{C}$ of constraints is *stratified* if it can be partitioned into sets $\mathcal{C}_0, \dots, \mathcal{C}_k$ such that, for all $0 \leq i \leq k$, the following hold:

1. If $i < k$ and s' occurs in φ for some $s \leftarrow \varphi \in \mathcal{C}_i$, then s' is not defined in $\mathcal{C}_{i+1} \cup \ldots \cup \mathcal{C}_k$.
2. If s' occurs negatively in φ for some $s \leftarrow \varphi \in \mathcal{C}_i$, then s' is not defined in $\mathcal{C}_i \cup \ldots \cup \mathcal{C}_k$.

A set of constraints is *stratified* if it admits a stratification.

We now define the notion of a consequence operator $T_{\mathcal{I},\mathcal{C}}$ that, given a shape assignment S, adds new shape atoms to satisfy the constraints that are fired by the constraints in $\mathcal{C}$ based on S and $\mathcal{I}$.

Definition 3.2. For a set $\mathcal{C}$ of constraints, an interpretation $\mathcal{I}$ with $\mathrm{N}_I \subseteq \Delta^I$, we define an *immediate consequence operator* $T_{\mathcal{I},\mathcal{C}}$ that maps shape assignments to shape assignments as follows:

$$T_{\mathcal{I},\mathcal{C}}(S) = S \cup \{s(a) \mid s \leftarrow \varphi \in \mathcal{C} \text{ and } a \in (\varphi)^{\mathcal{I},S}\}.$$

We further let:

- $T_{\mathcal{I},\mathcal{C}} \uparrow^0 (S) = S$,
- $T_{\mathcal{I},\mathcal{C}} \uparrow^{n+1} (S) = T_{\mathcal{I},\mathcal{C}}(T_{\mathcal{I},\mathcal{C}} \uparrow^n (S))$,
- $T_{\mathcal{I},\mathcal{C}} \uparrow^\omega (S) = \bigcup_{n=0}^{\infty} T_{\mathcal{I},\mathcal{C}}(S)$.

The following two propositions are a direct consequence of the characterizations from [3] in the context of stratified logic programs.

Proposition 3.3. *If $\mathcal{C}$ is a constraint set that does not define any shape names that occur negatively in $\mathcal{C}$, then the following hold:*

1. *$T_{\mathcal{I},\mathcal{C}}$ is monotonic, i.e. if $S \subseteq S'$, then $T_{\mathcal{I},\mathcal{C}}(S) \subseteq T_{\mathcal{I},\mathcal{C}}(S')$;*
2. *$T_{\mathcal{I},\mathcal{C}}$ is finitary, i.e. $T_{\mathcal{I},\mathcal{C}}(\bigcup_{n=0}^{\infty} S_n) \subseteq \bigcup_{n=0}^{\infty} T_{\mathcal{I},\mathcal{C}}(S_n)$ for all infinite sequences $S_0 \subseteq S_1 \subseteq \cdots$;*
3. *$T_{\mathcal{I},\mathcal{C}}$ is growing, i.e. $T_{\mathcal{I},\mathcal{C}}(S_2) \subseteq T_{\mathcal{I},\mathcal{C}}(S_3)$ for all S_1, S_2, S_3 such that $S_1 \subseteq S_2 \subseteq S_3 \subseteq T_{\mathcal{I},\mathcal{C}} \uparrow^\omega (S_1)$.*

Proposition 3.4. *If $\mathcal{I}$ is an interpretation and $\mathcal{C}_0, \dots, \mathcal{C}_k$ is a stratification of $\mathcal{C}$, then all $T_{\mathcal{I},\mathcal{C}_0}, \dots, T_{\mathcal{I},\mathcal{C}_k}$ are monotone, finitary, and growing. Thus, for any shape assignment S and each $0 \leq j \leq k$, $T_{\mathcal{I},\mathcal{C}_j} \uparrow^\omega (S)$ is a fixpoint of $T_{\mathcal{I},\mathcal{C}_j}$.*

Based on the above, we can now define the computation of the desired shape assignment along a stratification $\mathcal{C}_0, \dots, \mathcal{C}_k$ of $\mathcal{C}$.

Definition 3.5. Assume $\mathcal{I}$ is an interpretation, $\mathcal{C}$ is a stratified set of constraints, and let $\mathcal{C}_0, \dots, \mathcal{C}_k$ be a stratification of $\mathcal{C}$. Then let

$$M_0 = T_{\mathcal{I},\mathcal{C}_0} \uparrow^\omega (\emptyset)$$
$$M_i = T_{\mathcal{I},\mathcal{C}_i} \uparrow^\omega (M_{i-1}) \quad \text{for each } 1 \leq i < k.$$

We let M_k be the *perfect assignment* for $\mathcal{C}$ and $\mathcal{I}$, and let $PA(\mathcal{C},\mathcal{I}){=}M_k$. An interpretation $\mathcal{I}$ (resp., ABox $\mathcal{A}$) *validates* a shapes graph $(\mathcal{C},\mathcal{G})$ if $\mathcal{G} \subseteq PA(\mathcal{C},\mathcal{I})$ (resp., $\mathcal{G} \subseteq PA(\mathcal{C},\mathcal{I}_{\mathcal{A}})$).

Assume an interpretation $\mathcal{I}$ and a shapes graph $(\mathcal{C},\mathcal{G})$, where $\mathcal{C}$ is a stratified set of constraints. We first note that $PA(\mathcal{C},\mathcal{I})$ does not depend on a concrete stratification, i.e. any stratification yields the same $PA(\mathcal{C},\mathcal{I})$. Moreover, $PA(\mathcal{C},\mathcal{I})$ is a supported model of $\mathcal{C}$ and $\mathcal{I}$, which follows from the corresponding result in [3] for stratified logic programs. Since the stable model semantics for SHACL in [2] was defined for finite structures only, we cannot precisely formalize here a connection between our semantics and the semantics in [2]. However, for a data graph $\mathcal{A}$ that contains all nodes mentioned in $(\mathcal{C},\mathcal{G})$, we have that $PA(\mathcal{C},\mathcal{I}_{\mathcal{A}})$ restricted to the nodes of $\mathcal{A}$ is the only stable model of $\mathcal{C}$. Thus $\mathcal{A}$ validates $(\mathcal{C},\mathcal{G})$ under the stable model semantics iff $\mathcal{G} \subseteq PA(\mathcal{C},\mathcal{I}_{\mathcal{A}})$.

To check whether a set of constraints admits a stratification we can construct the *dependency graph* [3]. The dependency graph of a set of constraints $\mathcal{C}$ is a marked directed graph, where the nodes are the shape names occurring in $\mathcal{C}$, and there is an edge (s_1, s_2) from node s_1 to node s_2 if there is a constraint $s_1 \leftarrow \phi$ in $\mathcal{C}$ such that s_2 occurs in ϕ, and the edge is *marked* if s_2 occurs negatively in ϕ. A set of constraints $\mathcal{C}$ is stratified iff the dependency graph of $\mathcal{C}$ has no cycles involving a marked edge. Computing a stratification is tractable: for a stratified set of constraints $\mathcal{C}$, any topological ordering of its dependency graph provides a stratification [3].

4 Validating SHACL with Ontologies

In this section we propose an intuitive validation semantics for SHACL in the presence of a DL-Lite$_{\mathcal{R}}$ ontology. More precisely, for a given a TBox $\mathcal{T}$, an ABox $\mathcal{A}$, and a shapes graph $(\mathcal{C},\mathcal{G})$, we need to define when $(\mathcal{T},\mathcal{A})$ validates $(\mathcal{C},\mathcal{G})$. A natural first idea would be to follow the usual open-world semantics of DL-Lite and check for validation over all models of $\mathcal{A}$ and $\mathcal{T}$. While this works for positive constraints, it does not yield a natural result in the presence of negation, as illustrated in the following simple example.

Example 4.1. Consider an ABox $\mathcal{A}$, consisting of the facts $\mathit{hasPet}(\mathit{linda}, \mathit{blu})$, $\mathit{Bird}(\mathit{blu})$, $\mathit{PetOwner}(\mathit{linda})$, and an empty TBox $\mathcal{T}$. Let $(\mathcal{C},\mathcal{G})$ be a shapes graph, where $\mathcal{C}$ only contains the constraint

$$s \leftarrow \exists \mathit{hasPet} \wedge \neg \exists \mathit{hasPet}.\mathit{Dog}$$

and the target to be checked for validation is $\mathcal{G} = \{s(\mathit{linda})\}$.

The perfect assignment $PA(\mathcal{C},\mathcal{I}_{\mathcal{A}})$ naturally includes the target $s(\mathit{linda})$ since linda has a pet, and does not have a pet that is a dog. Note that since we have an empty TBox, we are in the usual setting of validation here. Thus, $\mathcal{A}$ validates $(\mathcal{C},\mathcal{G})$ and clearly, one would expect $(\mathcal{T},\mathcal{A})$ to validate $(\mathcal{C},\mathcal{G})$. However, if we consider all possible models of $(\mathcal{T},\mathcal{A})$, we have non-validation since there may be a model $\mathcal{I}$ of $\mathcal{A}$ and $\mathcal{T}$ that includes a hasPet-fact for linda to some pet b that is a Dog. Indeed, the interpretation $\mathcal{I}$ does not validate the shapes graph since the target $s(\mathit{linda})$ is not included in $PA(\mathcal{C},\mathcal{I})$.

The problem in the above example is the *non-monotonicity* of the constraints. Roughly speaking, adding facts to the data may cause a previously valid setting to become invalid. We want an intuitive semantics that coincides with the usual validation in case the TBox is empty. As done in related settings (see e.g., [5, 14, 8]) we rely on the *chase* procedure [1] known from Knowledge Representation and Database Theory. Roughly speaking, a chase procedure takes as input an ABox and a TBox and iteratively applies the axioms of the TBox to the data by adding atoms over possibly fresh individuals until all the axioms in the TBox are satisfied. The result of the chase is a so-called *canonical* or *universal* model, and since it can be homomorphically embedded into every other model of the ABox and

TBox, the canonical model can be used as a representative of all the models. For DL-Lite$_{\mathcal{R}}$ ontologies, such chase procedures may not terminate and result in infinite models. There are several chase variants producing different canonical models [8]. While for positive constraints these differences do not matter, constraints with negation can distinguish between them, resulting in different validation answers as we illustrate in Example 4.6.

4.1 Semantics over the Austere Canonical Model

The semantics we propose is based on a special chase procedure that constructs an *austere* canonical model, in the sense that each chase step introduces as few fresh successors as possible, without merging any successors unless the TBox is forcing us to do so.

We first define an auxiliary notion of a *good successor configuration*, which, given a node e and the set of basic concepts that are assumed to hold at e, determines the minimal set of fresh successors of e and the roles connecting e to each of them.

Definition 4.2. For a set of roles R, let $cl_{\mathcal{T}}(R)$ be the set of all roles r' such that $\mathcal{T} \models r \sqsubseteq r'$ and $r \in R$. Given a *DL-Lite*$_{\mathcal{R}}$ TBox $\mathcal{T}$ and a set of basic concepts U, a *good successor configuration* $succ_{\mathcal{T}}(U)$ for U is a possibly empty set of roles such that:

1. There do not exist $r, r' \in succ_{\mathcal{T}}(U)$ such that $r \in cl_{\mathcal{T}}(\{r'\})$;
2. If $C \in U$, $\mathcal{T} \models C \sqsubseteq \exists r$ and $\exists r \notin U$, then there exists $r' \in succ_{\mathcal{T}}(U)$ such that $r \in cl_{\mathcal{T}}(\{r'\})$;
3. If $r \in succ_{\mathcal{T}}(U)$, there exists $C \in U$ such that $\mathcal{T} \models C \sqsubseteq \exists r$;
4. $succ_{\mathcal{T}}(U) \cap cl_{\mathcal{T}}(\{r \mid \exists r \in U\}) = \{\}$.

Lemma 4.3. *For each set of basic concepts U, there exists a unique good successor configuration $succ_{\mathcal{T}}(U)$ for U.*

The following example illustrates the notion of a good successor configuration.

Example 4.4. Consider the TBox $\mathcal{T}$ containing three axioms:

$$PetOwner \sqsubseteq \exists hasPet \qquad hasWingedPet \sqsubseteq hasPet$$
$$PetOwner \sqsubseteq \exists hasWingedPet.$$

Then, $succ_{\mathcal{T}}(\{PetOwner\})$ is the set $\{hasWingedPet\}$, which trivially satisfies items (1) to (4). Indeed $succ_{\mathcal{T}}(\{PetOwner\})$ cannot contain $hasPet$ since by item (1), $hasPet \in cl_{\mathcal{T}}(\{hasWingedPet\})$.

All other sets of basic concepts will have empty sets as successor configurations. In particular, the set $U = \{PetOwner, \exists hasWingedPet\}$ cannot have any of the roles of the TBox as a successor role due to item (4), since $cl_{\mathcal{T}}(\{hasWingedPet\}) = \{hasWingedPet, hasPet\}$.

The good successor configuration prescribes how to satisfy the TBox axioms locally. We use it to build a canonical model layer by layer that we call *austere*.

Definition 4.5. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a pair of a *DL-Lite*$_{\mathcal{R}}$ TBox $\mathcal{T}$ and ABox $\mathcal{A}$. Let $N_I(\mathcal{A})$ be the set of individuals occurring in $\mathcal{A}$. Let $N_{\mathcal{K}}$ be the set of finite words of the form $ar_1r_2 \dots r_n$, with $n \geq 1$, $a \in N_I(\mathcal{A})$ and $r_1, \dots, r_n$ are roles, such that the following hold:

1. Let $U_a = \{A \mid A(a) \in \mathcal{A}\} \cup \{\exists r \mid r(a, a') \in \mathcal{A}\}$, then $r_1 \in succ_{\mathcal{T}}(U_a)$;
2. For every $1 \leq i < n$, $r_{i+1} \in succ_{\mathcal{T}}(U_{r_i})$, where $U_{r_i} = \{\exists r_i^-\}$.

We denote by $tail(w)$ the last role appearing in a word $w \in N_{\mathcal{K}}$. The *austere canonical model* $can(\mathcal{K})$ of $\mathcal{K}$ is the interpretation $\mathcal{I}$ with domain $\Delta^{\mathcal{I}} := N_I(\mathcal{A}) \cup N_{\mathcal{K}}$ such that for all $a \in N_I(\mathcal{A})$, concept names A and roles r, the following hold:

1. $a^{\mathcal{I}} := a$;
2. $A^{\mathcal{I}} := \{a \mid \mathcal{K} \models A(a)\} \cup \{w \in N_{\mathcal{K}} \mid \mathcal{T} \models \exists tail(w)^- \sqsubseteq A\}$;
3. $r^{\mathcal{I}} := \{(a, a') \mid \mathcal{K} \models r(a, a')\}\} \cup \{(w_1, w_2) \mid w_2 \in N_{\mathcal{K}}, w_2 = w_1r', r \in cl_{\mathcal{T}}(\{r'\})$.

For a consistent $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, the austere canonical model $can(\mathcal{K})$ exists and it is unique. It is standard to show that $can(\mathcal{K})$ is a *canonical model* of $\mathcal{K}$, which is a model of $\mathcal{K}$ that can be homormorphically embedded into every other model of $\mathcal{K}$. The number of successor nodes in $can(\mathcal{K})$ is determined by the good successor configuration.

Our notion of austere canonical model is closely related to the *core* chase [8]. It will typically create fewer fresh successors than the *oblivious* chase, which, roughly speaking, applies the axioms of the TBox without first checking whether the axiom is already satisfied. It may also create fewer successors than the *restricted chase*, which may be sensitive to the order of rule applications.

Example 4.6. Consider $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ where the ABox

$$\mathcal{A} = \{PetOwner(linda), hasWingedPet(linda, blu), Bird(blu)\}$$

and the TBox $\mathcal{T}$ is as in Example 4.4. The good successor configuration $succ_{\mathcal{T}}(U_{linda})$ is the empty set. More precisely, $U_{linda} = \{PetOwner, \exists hasWingedPet\}$ and as shown in Example 4.4, it cannot have any of the roles of the TBox as a successor role. Thus, $N_{\mathcal{K}}$ in the canonical model is empty, and no fresh objects are introduced. The austere canonical model $can(\mathcal{K})$ (right in Figure 2) will only add a $hasPet$-role from $linda$ to $blue$. In contrast, the canonical model obtained from the oblivious chase (left in Figure 2) will introduce two fresh objects to satisfy the two existential axioms.

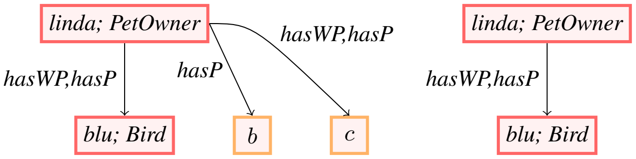

Figure 2: Result of oblivious chase (left) and austere canonical model (right). We use hasWP and hasP instead of hasWingedPet and hasPet.

Consider now the shapes graph $(\mathcal{C}, \mathcal{G})$ with $\mathcal{C} = \{s \leftarrow \exists hasPet.\neg Bird\}$ and $\mathcal{G} = \{s(linda)\}$. The shapes graph asks to validate whether *linda* has a pet that is not a bird. Clearly, the austere canonical model provides the expected answer, as it does not validate $(\mathcal{C}, \mathcal{G})$. In contrast, the canonical model on the left-hand-side of the figure—and thus the semantics of [14] adopted for SHACL in [16]—results in the unintended validation of $(\mathcal{C}, \mathcal{G})$.

The semantics of validation with *DL-Lite*$_{\mathcal{R}}$ ontologies is given in terms of validation over the austere canonical model.

Definition 4.7 (Validation with *DL-Lite*$_{\mathcal{R}}$)**.** Assume $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and $(\mathcal{C}, \mathcal{G})$ is a shapes graph, where $\mathcal{C}$ is a stratified set of constraints. We say $\mathcal{K}$ validates $(\mathcal{C}, \mathcal{G})$ if $can(\mathcal{K})$ validates $(\mathcal{C}, \mathcal{G})$.

We illustrate the notion of validation with an example.

Example 4.8. Consider again Example 4.6 and the shapes graph $(\mathcal{C}, \mathcal{G})$ with $\mathcal{C} = \{s \leftarrow \exists hasPet.\neg Bird\}$ and $\mathcal{G} = \{s(linda)\}$.

Intuitively, the shapes graph asks to validate whether *linda* has a pet that is not a bird. Clearly, $can(\mathcal{K})$ provides the expected answer, as it does not validate $(\mathcal{C}, \mathcal{G})$. In contrast, the canonical model on the left-hand-side of Figure 2 provides the unintended validation of $(\mathcal{C}, \mathcal{G})$.

5 Rewriting

In the rest of the paper, we will only consider ABoxes that are consistent with the given TBox. Furthermore, note that disjointness axioms will not play any role in the rewriting.

Given a TBox $\mathcal{T}$ and a set of stratified constraints $\mathcal{C}$, we want to compile $\mathcal{T}$ and $\mathcal{C}$ into a new set $\mathcal{C}_\mathcal{T}$ of stratified constraints so that for every ABox $\mathcal{A}$ consistent with $\mathcal{T}$, and every target $\mathcal{G}$, we have

$$(\mathcal{T}, \mathcal{A}) \text{ validates } (\mathcal{C}, \mathcal{G}) \text{ iff } \mathcal{A} \text{ validates } (\mathcal{C}_\mathcal{T}, \mathcal{G}).$$

This will be achieved by means of an inference procedure that uses a collection of inference rules to capture the possible “propagation” of shape names in the annonymous part of the cannonical model. To ease presentation, we assume $\mathcal{C}$ contains only *normalized* constraints of the following forms:

$$\begin{array}{ll} \text{(NC1) } s \leftarrow D & \text{(NC2) } s \leftarrow s_1 \wedge s_2 \\ \text{(NC3) } s \leftarrow \exists r.s' & \text{(NC4) } s \leftarrow \neg s' \end{array}$$

where D is an individual or a (DL-Lite$_\mathcal{R}$) basic concept. Every set of constraints can be easily normalized without affecting validation (see e.g., [2]). This can be done in polynomial time by introducing fresh shape names. Note that we may say *negative constraints* to refer to constraints of the form (NC4) and *positive constraints* to refer to the constraints of the form (NC1) to (NC3).

In the first stage, we define the rewriting procedure for constraints without negation. We later show how that procedure can be upgraded to handle stratified negation.

5.1 Rewriting Positive Constraints

Our rewriting algorithm uses pairs of the form (W, H), where W is a set of *basic shape expressions* and H is a set of shape names.

Definition 5.1. If A is a concept name, r is a role, s is a shape name, and c an individual, then c, A, $\exists r.\top$, $\exists r.s$, $\neg c$, $\neg A$, $\neg\exists r.\top$, and $\neg\exists r.s$ are called *basic shape expressions*; the former four are *positive*, the latter four are *negative*. For a set W of basic shape expressions, we let

$$cr(W) = \{\exists r \mid \exists r.\top \in W \text{ or } \exists r.s \in W\} \;\cup \{A \mid A \in W\}.$$

The core of our technique is an inference procedure that derives a set I of pairs (W, H). Intuitively, $(W, H) \in I$ tells us that an object that satisfies all expressions in W validates all shape names in H.

Example 5.2. Let $\mathcal{T} = \{\exists q \sqsubseteq \exists p, \exists p^- \sqsubseteq A\}$ and

$$\mathcal{C} = \{s \leftarrow \exists p.s_A, \qquad s_A \leftarrow A, \qquad s' \leftarrow \exists q\}.$$

We want our rewritten $\mathcal{C}_\mathcal{T}$ to expand the constraints $\mathcal{C}$ above with

$$s \leftarrow \exists q \qquad s_A \leftarrow \exists p^-$$

allowing us to validate the target $s(a)$ over the ABox $\mathcal{A} = \{q(a, b)\}$, or the target $s_A(b)$ over the ABox $\mathcal{A} = \{p(a, b)\}$, without having to build $can(\mathcal{T}, \mathcal{A})$. This will be achieved by deriving a set of pairs that includes $(\{\exists q\}, \{s, s'\})$, which keeps the constraint $s' \leftarrow \exists q$ while also adding the new constraint $s \leftarrow \exists q$, as well as the pairs $(\{\exists p^-\}, \{s_A\})$ and $(\{A\}, \{s_A\})$, witnessing the constraints for s_A.

In the following definition, we assume a given set I of pairs (W, H) as above. The role of this set will become clear later when we lift the algorithm to constraints with stratified negation.

Definition 5.3. Let $\mathcal{T}$ be a TBox and $\mathcal{C}$ a set of normalized constraints. Let $\Sigma_{\mathcal{C},\mathcal{T}}$ be the set of all basic shape expressions that can be formed from the concept, role, and shape names occurring in $\mathcal{T}$ and $\mathcal{C}$. We let $sat_{\mathcal{C},\mathcal{T}}(I)$ be the smallest set of pairs, containing I, closed under the following rules:

1. If $S \in \Sigma_{\mathcal{C},\mathcal{T}}$ is a positive basic shape expression, then $(\{S\}, \{\})$ belongs to $sat_{\mathcal{C},\mathcal{T}}(I)$.
2. If $s \leftarrow S \in \mathcal{C}$ for some basic shape expression S, then $(\{S'\}, \{s\})$ belongs to $sat_{\mathcal{C},\mathcal{T}}(I)$ for each $S' \in \Sigma_{\mathcal{C},\mathcal{T}}$ such that:
    - $S' = S = c$ for some individual c; or
    - $\mathcal{T} \models S' \sqsubseteq S$ and S, S' are basic DL-Lite$_\mathcal{R}$ concepts; or
    - $\mathcal{T} \models r' \sqsubseteq r$ and S, S' are of the forms $\exists r.s'$ and $\exists r'.s'$ respectively.
3. If $\{(W, H), (W', H')\} \subseteq sat_{\mathcal{C},\mathcal{T}}(I)$ such that there is no basic concept or individual D with $\{D, \neg D\} \subseteq W \cup W' \cup cr(W) \cup cr(W')$, and no two distinct individuals $\{c, c'\} \subseteq N_I$ with $\{c, c'\} \subseteq W \cup W'$, then $(W \cup W', H \cup H')$ belongs to $sat_{\mathcal{C},\mathcal{T}}(I)$.
4. If $s \leftarrow s_1 \wedge s_2 \in \mathcal{C}$, $(W, H) \in sat_{\mathcal{C},\mathcal{T}}(I)$ and $\{s_1, s_2\} \subseteq H$, then $(W, H \cup \{s\})$ belongs to $sat_{\mathcal{C},\mathcal{T}}(I)$.
5. If $s \leftarrow \exists r.s' \in \mathcal{C}$, $\{(W, H), (W', H')\} \subseteq sat_{\mathcal{C},\mathcal{T}}(I)$, $s' \in H'$ and there are $r' \in \mathsf{N}_R^+$ and $R \subseteq \mathsf{N}_R^+$ such that:
    - $r' \in succ_\mathcal{T}(cr(W))$, $R = cl_\mathcal{T}(\{r'\})$, and $r \in R$,
    - $cr(W') = \{\exists r \mid r^- \in R\}$,
    - $\{s'' \mid \exists r''.s'' \in W'\} \cup \{\neg s'' \mid \neg\exists r''.s'' \in W', r''^- \in R\} \subseteq H$,

    then $(W \cup \{\neg\exists p \mid r \in cl_\mathcal{T}(\{p\})\}, H \cup \{s\})$ belongs to $sat_{\mathcal{C},\mathcal{T}}(I)$.

We let $\mathcal{C}_{\mathcal{T},I}$ be the set that contains, for each $(W, H) \in I$ and each $s \in H$, the constraint

$$s \leftarrow \bigwedge_{S \in W} S.$$

We let $\mathcal{C}_\mathcal{T} = \mathcal{C}_{\mathcal{T},I}$ with $I = sat_{\mathcal{C},\mathcal{T}}(\emptyset)$.

The following example illustrate the application of each of the rules in Definition 5.3.

Example 5.4. Consider $\mathcal{T} = \{A \sqsubseteq \exists p, \exists p^- \sqsubseteq \exists q, \exists q^- \sqsubseteq B, p \sqsubseteq r, q \sqsubseteq r\}$ and

$$\begin{aligned} \mathcal{C} = \{s \leftarrow \exists r.s, \qquad & s \leftarrow s_B \wedge s', \qquad & s_B \leftarrow B, \\ & s' \leftarrow \exists r^-.s', \qquad & s' \leftarrow A\}. \end{aligned}$$

Given the ABox $\mathcal{A} = \{A(a)\}$, the austere canonical model $can(\mathcal{T}, \mathcal{A})$ can be found in Figure 3.

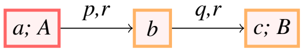

Figure 3: Austere canonical model $can(\mathcal{T}, \mathcal{A})$ from Example 5.4

Let us consider the target $\mathcal{G} = \{s(a)\}$. It is now an easy check that $can(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$; just compute the perfect assignment.

To see that we can reach the same result with our rewriting, first apply Rule 2 two times to find $\{(\{\exists q^-\}, \{s_B\}), (\{\exists r^-.s'\}, \{s'\})\} \subseteq$

I, where we use I as a shorthand for $sat_{\mathcal{C},\mathcal{T}}(\emptyset)$. By applying first Rule 3, then Rule 4 on these two pairs, we find that $(\{\exists q^-, \exists r^-.s'\}, \{s_B, s', s\}) \in I$ too. Call this pair (W', H'). Note that this pair is simulating the environment of c, which has as incoming q edge, and an incoming r edge from b, for which we assume that $s'(b)$ is validated. In such a context, we can correctly infer that $s_B(c)$, $s'(c)$, and $s(c)$ are validated. Let's now look at the environment W of b, which contains $\exists p^-$ and $\exists r^-$ and, according to the good successor configuration, such a node has a q-successor.

Let $(W, H) = (\{\exists p^-, \exists r^-.s'\}, \{s'\})$ be constructed from applying Rule 2 on $s' \leftarrow \exists r^-.s'$ (intuitively, this reflects that $s'(a)$ and $s'(b)$ are in the perfect assignment), followed by Rule 1 and 3 to add $\exists p^-$ to W. We can now apply Rule 5 where we consider the constraint $s \leftarrow \exists r.s$: let $r' = q$, $R = \{q, r\}$. Since (W', H') is simulating the environment of c, and $\exists r^-.s' \in W'$, we need to check that $s' \in H$, or in words, that $s'(b)$ is validated. It is now easy to see that all conditions of Rule 5 are met: $(\{\exists p^-, \exists r^-.s', \neg\exists q\}, \{s', s\}) \in I$. Note that the function of $\neg\exists q$ is to make sure that we can never add $\exists q$ to the left-hand side of the pair; adding this would break our assumption that a q-successor is introduced by the good successor configuration.

Next, we apply Rule 5 on the pair $(W, H) = (\{A\}, \{s'\})$ and the pair $(W', H') = (\{\exists p^-, \exists r^-.s', \neg\exists q\}, \{s', s\})$. We find that $(\{A, \neg\exists p\}, \{s', s\}) \in I$. This means that the following constraint is in $\mathcal{C}_\mathcal{T}$:

$$s \leftarrow A \wedge \neg\exists p,$$

which shows that indeed $\mathcal{A}$ validates $(\mathcal{C}_\mathcal{T}, \mathcal{G})$.

We generalize this in the following theorem: the rewriting gives a sound and complete set of constraints.

Theorem 5.5. *Let $\mathcal{T}$ be a TBox and $\mathcal{C}$ a set of positive constraints. Then for every target $\mathcal{G}$ and every ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$, we have that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}$ validates $(\mathcal{C}_\mathcal{T}, \mathcal{G})$.*

Proof. (Sketch) To show soundness, assume an arbitrary but fixed sequence of rule applications leading to $sat_{\mathcal{C},\mathcal{T}}(\emptyset)$, let I_i be the set of pairs after i steps, and let $C_{\mathcal{T},I_i}$ be the constraints obtained from I_i. It suffices to show by induction on i, that for every arbitrary target $s(a)$, if $can(\mathcal{T}, \mathcal{A})$ validates $(C_{\mathcal{T},I_i}, \{s(a)\})$, then $can(\mathcal{T}, \mathcal{A})$ validates $(C, \{s(a)\})$. For $i = 1$, only Rule 1 or 2 can be applied. In the case of Rule 1 the claim is trivial as pairs $(W, \emptyset)$ do not generate constraints. In the case it is Rule 2, it is also straightforward since the added (W, H) will result in a constraint of the form $s \leftarrow S'$ and if S' is a basic concept then there is some basic concept S such that $s \leftarrow S \in \mathcal{C}$, and from $\mathcal{T} \models S' \sqsubseteq S$ it is the case that $a \in S'^{can(\mathcal{T},\mathcal{A})}$ implies $a \in S^{can(\mathcal{T},\mathcal{A})}$. Hence, if $s(a) \in PA(C_{\mathcal{T},I_i}, can(\mathcal{T}, \mathcal{A}))$, then $s(a) \in PA(C, can(\mathcal{T}, \mathcal{A}))$. The argument for S, S' of the forms $\exists r.s'$ and $\exists r'.s'$ is similar. For the induction step, by inspecting the Rules 2 to 5 one can easily show that each pair (W, H) is correctly obtained at rule application $i + 1$ and it will generate sound constraints.

For completeness, suppose $can(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \{s(a)\})$. Then there exists a *justification tree* indicating which domain elements of $can(\mathcal{T}, \mathcal{A})$ validate which shape names that are used in the fixed point computations. In this tree, nodes are of the form (x, S), where $x \in N_I(\mathcal{A}) \cup N_\mathcal{K}$ and $S \in N_S \cup \{D \mid D \text{ a basic concept}\}$. The root is (a, s). Each node has none, one or two children and there are no infinite branches. Moreover, the following conditions hold.

- (x, S) has no children iff S is a basic concept. If (x, S) appears as a leaf and $S = A$ for some concept A, then $x \in A^{can(\mathcal{T},\mathcal{A})}$; and if $S = \exists r$ for some role r, there exists $y \in \Delta^{can(\mathcal{T},\mathcal{A})}$ such that $(x, y) \in r^{can(\mathcal{T},\mathcal{A})}$.
- If (x, s) has only one child, if it is of the form (x, D) or (x', s'). If the form is (x, D), then $s \leftarrow D \in \mathcal{C}$. If the form is (x', s'), then $x' = xr'$ such that $r \in cl_\mathcal{T}(\{r'\})$ or $x = x'r'$ such that $r^- \in cl_\mathcal{T}(\{r'\})$ and $s \leftarrow \exists r.s' \in \mathcal{C}$.
- If (x, s) has two children, they are of the form (x, s') and (x, s'') and $s \leftarrow s' \wedge s'' \in \mathcal{C}$.

Define the *weight* of a tree as the number of nodes (w, S) with $w \in N_\mathcal{K}$. We call a leaf *strictly justified* if there exists a pair (W, H) such that $cr(W) = \{\exists r \mid r \in cl_\mathcal{T}(\{tail(x)^-\})\}$ and $\{\exists r.s \mid r \in N_R^+, s \in N_S\} \cap W = \{\}$, moreover, we need to add some more constraints to make sure that negative shape expressions in W are not clashing with shape names satisfied by the parent of w in $can(\mathcal{T}, \mathcal{A})$ and the role connecting the two, $tail(w)$.

We claim we can decrease the weight of this tree while ensuring that: (i) each edge is either an edge occurring in the original tree, or a marked edge, and (ii) each leaf is strictly justified, or of the form (a', S) for $a' \in N_I(\mathcal{A})$, or of the form (x, S) for $S \in N_D$. Intuitively, a marked edge indicates that we can rewrite the pair (W, H) that 'justifies', a weaker notion than strict justification, also allowing $\exists r.s$ in W, the child node, as soon as we have it, into a justification for the parent. If on a branch of the tree there are no nodes such that (x, S) is the parent of (x', S') and $|x| > |x'|$, we call this branch a *loose end*. The weight can be lowered easily by cutting off leaves on loose ends.

However, a *U-turn* can also happen in the anonymous part of our justification: from (x, S) there exists a path to (x, S') such that all nodes appearing on this path are of the form (x', S''), where $x' = xr$ for some r. If we are cutting off leaves on loose ends, we might find a node with two children, one of them a leaf. In this case, we cut off the leaf and mark the edge to the other child. The other option is that we find a connected set of nodes M such that if $(x', S) \in M$, then $x' = xr$ and if some $(x'', S') \notin M$ is the child or parent of some $(x', S) \in M$, then $x'' = x$. We can take out complete sets M at once, in such a way that the parent of this set will become the new parent of M's children, but with a marked edge. Note that rule 5 deals explicitly with this case.

As the weight of the final tree is 0, we can use Rules 1, 2 and 3 to create pairs (W, H) for each leave node left. Normal edges correspond to already existing shapes that we can use here too. Marked edges tell us that there exists a rewriting from the child to the parent: we know we can rewrite the incoming pair (W, H) in the needed form. This tells us how $\mathcal{A}$ validates $(\mathcal{C}_\mathcal{T}, \{s(a)\})$. □

5.2 Constraints with Stratified Negation

We now extend the above rewriting to additionally handle constraint sets $\mathcal{C}$ that involve stratified negation. Intuitively, this is done by running the saturation procedure at each stratum of $\mathcal{C}$, starting with the lowermost. For this, we need to make sure that the outcome of the saturation at a non-topmost stratum is completed with negative information and is passed to the next stratum. To this end, we w.l.o.g. assume that all constraints from $\mathcal{C}$ with the same shape name on the left-hand-side occur together in the same stratum.

We will operate now on pairs (W, H), which are similar to the ones in the previous section, except that H might additionally contain expressions of the form $\neg s$ for a shape name s. For a set I of such pairs, we say (W, H) is *maximal* in I, if $(W, H) \in I$ and there is no $H' \supset H$ with $(W', H') \in I$ such that W' contains exactly the same basic shape expressions as W. Then the notion of *completion* is defined as follows:

Definition 5.6. The *completion* $comp_{\mathcal{C},\mathcal{T}}(I)$ of a set I of implication pairs w.r.t. a TBox $\mathcal{T}$ and a constraint set $\mathcal{C}$ is a set defined as follows:

$$comp_{\mathcal{C},\mathcal{T}}(I) = \{(W \cup \bar{W}, H \cup \bar{H}) \mid (W, H) \text{ is maximal in } I\} \text{ with}$$

$$\bar{H} := \{\neg s \mid s \text{ occurs in } \mathcal{C}, s \notin H\}$$

$$\bar{W} := \{\neg D \mid \text{ there is no } D' \in cr(W) \text{ s.t. } \mathcal{T} \models D' \sqsubseteq D\} \cup$$

$$\{\neg\exists r.s' \mid s \leftarrow \exists r.s' \in \mathcal{C}, s \notin H\} \cup \{\neg c \mid s \leftarrow c \in \mathcal{C}, s \notin H\}.$$

We need to augment the inference rules of the previous section with an additional rule to handle constraints of the form $s \leftarrow \neg s'$.

Definition 5.7. Let $\mathcal{T}$ be a TBox, $\mathcal{C}$ a set of normalized constraints. We let $sat_{\mathcal{C},\mathcal{T}}(I)$ be the smallest set of pairs containing I closed under Rules 1–5 in Def. 5.3, and additionally under the following Rule 6:

6. If $s \leftarrow \neg s' \in \mathcal{C}$ and $(W, H) \in sat_{\mathcal{C},\mathcal{T}}(I)$ such that $\neg s' \in H$, then $(W, H \cup \{s\})$ belongs to $sat_{\mathcal{C},\mathcal{T}}(I)$.

Now can we define the inference procedure that processes strata from the lowest to the topmost, performing saturation using the updated set of rules at every stratum, interleaved with a computation of the completion in between.

Definition 5.8. For a TBox $\mathcal{T}$ and a constraint set $\mathcal{C}$ with stratification $\mathcal{C}_0, \dots \mathcal{C}_n$, we let $I_0 = sat_{\mathcal{C}_0,\mathcal{T}}(\emptyset)$ and for $0 < i \leq n$

$$I_i = sat_{\mathcal{C}_i,\mathcal{T}}(comp_{\mathcal{C}_{i-1},\mathcal{T}}(I_{i-1})).$$

We let $\mathcal{C}_\mathcal{T} = \mathcal{C}_{\mathcal{T},I_n}$, where $\mathcal{C}_{\mathcal{T},I_n}$ is defined as in Definition 5.3.

Example 5.9. Consider $\mathcal{T} = \{A \sqsubseteq \exists p\}$ and the set $\mathcal{C}$ of constraints:

$$s \leftarrow \exists p.s' \qquad s' \leftarrow \neg s_B \qquad s_B \leftarrow B.$$

Let $\mathcal{C}_0, \mathcal{C}_1$ be a stratification of $\mathcal{C}$ such that $\mathcal{C}_0 = \{s_B \leftarrow B\}$ and $\mathcal{C}_1 = \{s \leftarrow \exists p.s', s' \leftarrow \neg s_B\}$. The rewriting, among others, will produce the pair $(\{\exists p^-\}, \{\})$ with Rule 1, which is maximal in I_0, and hence, the pair $(\{\exists p^-, \neg A, \neg B, \neg\exists p\}, \{\neg s_B\})$ is included in $comp_{\mathcal{C}_0,\mathcal{T}}(I_0)$. By Rule 6, the pair $(W', H') = (\{\exists p^-, \neg A, \neg B, \neg\exists p\}, \{\neg s_B, s'\})$ is added in I_1. By applying Rule 5 to the pairs (W', H') and $(W, H) = (\{A\}, \{\})$, the pair $(\{A, \neg\exists p\}, \{s\})$ is added to I_1. This is translated into the constraint $s \leftarrow A \wedge \neg\exists p \in \mathcal{C}_\mathcal{T}$. Consider now the ABox $\mathcal{A}_1 = \{A(a)\}$. Clearly, $(\mathcal{A}_1, \mathcal{T})$ validates $(\mathcal{C}, \{s(a)\})$ and $\mathcal{A}_1$ $(\mathcal{C}_\mathcal{T}, \{s(a)\})$.

The rewriting gives other pairs (W, H) in I_1 as well, including $(\{B\}, \{s_B\})$, $(\{\exists p.s'\}, \{s\})$ and constraints in which a subset of $\{A, B, \exists p, \exists p^-, \exists p.s'\}$ is added to W. In addition, by taking the completion of $sat_{\mathcal{C}_0,\mathcal{T}}(\emptyset)$, we find pairs with $\neg s_B \in H$ if $B \notin W$. Now consider the ABox $\mathcal{A}_2 = \{A(a), p(a, b), B(b)\}$. We cannot use any pair (W, H) introduced by Rule 5, as W then contains $\neg\exists p$, so we have that $\mathcal{A}_2$ does not validate $(\mathcal{C}_\mathcal{T}, \{s(a)\})$, as desired.

Now consider $\mathcal{A}_3 = \{A(a), p(a, b)\}$. $\mathcal{A}_3$ validates $(\mathcal{C}_\mathcal{T}, \{s'(b)\})$ witnessed by $(\{\exists p^-, \neg A, \neg B, \neg\exists p\}, \{\neg s_B, s'\}) \in I_1$, and it validates $(\mathcal{C}_\mathcal{T}, \{s(a)\})$, witnessed by $(\{\exists p.s'\}, \{s\}) \in I_1$.

Theorem 5.10. *Let $\mathcal{T}$ be a TBox and $\mathcal{C}$ a set of stratified constraints. Then, for every target $\mathcal{G}$ and every ABox $\mathcal{A}$ that is consistent with $\mathcal{T}$, we have that $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}$ validates $(\mathcal{C}_\mathcal{T}, \mathcal{G})$.*

Proof. (Sketch) Let $\mathcal{C}_0, \dots, \mathcal{C}_n$ be a stratification of $\mathcal{C}$. We want to show that, for each i, $(\mathcal{T}, \mathcal{A})$ validates $(\mathcal{C}_0 \cup \dots \cup \mathcal{C}_i, \{s(a)\})$ iff $\mathcal{A}$ validates $(\mathcal{C}_{\mathcal{T},I_i}, \{s(a)\})$, for an arbitrary target atom $s(a)$. We prove this together with another claim: that for each $i \leq k$, each $s \in \mathrm{N}_S$, and each $x \in \Delta^{can(\mathcal{T},\mathcal{A})}$,

$$s(x) \in PA(\mathcal{C}_{\mathcal{T},I_i}, can(\mathcal{T}, \mathcal{A})) \text{ iff } s(x) \in PA(\bigcup_{j \leq i} \mathcal{C}_j, can(\mathcal{T}, \mathcal{A})).$$

The proof is by induction on i. For the base case, note that the constraints in $\mathcal{C}_0$ are all positive, thus the first claim directly follows from Theorem 5.5. The second claim follows as $\mathcal{C}$ is contained in $\mathcal{C}_\mathcal{T}$ and the added constraints in $\mathcal{C}_\mathcal{T}$ are sound.

Now suppose we are considering the i-th stratum. We add another type of leaves to the definition of a justification tree: leaves of the form $(x, \neg s)$. Also the definition of justified leaves is adapted accordingly: for $(x, \neg s)$ to be correct, there must exist $(W, H) \in sat_{\mathcal{C}_i,\mathcal{T}}$ such that $\neg s \in H$ and the positive basic shape expressions in W are exactly the following: $\{A \mid x \in A^{can(\mathcal{T},\mathcal{A})}\} \cup \{\exists r \mid \text{ exists } y.(x, y) \in r^{can(\mathcal{T},\mathcal{A})}\} \cup \{\exists r.s' \mid \text{ exists } y.x = py, r \in cl(p), can(\mathcal{T}, \mathcal{A}) \text{ validates } (C_{\mathcal{T},I_{i-1}}, s'(y)\}$.

To make the step from one stratum to the next, suppose $s(x) \notin PA(\mathcal{C}_{\mathcal{T},I_{i-1}}, can(\mathcal{T}, \mathcal{A}))$. By IH, this happens iff there is no $(W, H) \in sat_{\mathcal{C}_{i-1},\mathcal{T}}$, maximal in I_{i-1}, such that $s \in H$ and the set of positive basic shape expressions in W is exactly as needed for the correctness of a leaf of the form $(x, \neg s)$. Since by construction $comp_{\mathcal{C}_{i-1},\mathcal{T}}(\{(W, H)\}) \subseteq sat_{\mathcal{C}_i,\mathcal{T}}(comp_{\mathcal{C}_{i-1},\mathcal{T}}(I_{i-1}))$, we use this as the basis for the rest of the induction step.

To prove the first claim, note that we can generalize both the soundness and the completeness of Theorem 5.5 with the updated justification tree, combined with what we just derived. For the second claim, the fact that we can build the justification tree is enough to show completeness. Soundness works similarly as in 5.5. □

6 Complexity

We now discuss the computational complexity of SHACL validation in the presence of *DL-Lite*$_\mathcal{R}$ TBoxes. Specifically, we discuss the *combined complexity* and the *data complexity* of the problem. The former is measured by in terms of the combined size of all input components, while the latter is measured assuming all components except the ABox are of fixed size.

Theorem 6.1. *In the presence of DL-Lite$_\mathcal{R}$ TBoxes, SHACL validation is* ExpTime*-complete in combined complexity and* PTime*-complete in data complexity. This holds also for positive constraints.*

Proof. For data complexity, the PTime lower bound was shown in [2], which already applies in the absence of ontologies. The matching PTime upper bound follows from Theorem 5.10 and the fact that validation under stratified constraints without ontologies in feasible in polynomial time in data complexity [2]. We note that checking if the input graph is consistent with a TBox is in logarithmic space.

For the upper bound in combined complexity, we discuss the rewriting algorithm of the previous section. Assume $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a set $\mathcal{C}$ of constraints. Observe that the number of different pairs (W, H) over the signature of $\mathcal{K}$ and $\mathcal{C}$ that can be added to I during the rewriting is bounded by an exponential in the size of $\mathcal{K}$ and $\mathcal{C}$. An application of any of the 6 rules takes polynomial time in the size of $\mathcal{K}, \mathcal{C}$ and I, which is bounded by an exponential in the size of $\mathcal{K}$ and $\mathcal{C}$. Computing $\mathcal{C}_{\mathcal{T},I}$ is polynomial in the size of I. Thus overall we get a procedure that runs in exponential time in the size of $\mathcal{K}$ and $\mathcal{C}$.

To prove ExpTime-hardness in combined complexity we reduce the word problem of polynomially space-bounded *Alternating Turing Machines (ATMs)* to validating shapes under a *DL-Lite*$_\mathcal{R}$ ontology.

An *ATM* is defined as a tuple of the form

$$\mathcal{M} = (\Sigma, Q_\exists, Q_\forall, q_0, q_{acc}, q_{rej}, \delta)$$

where Σ is an *alphabet*, $Q_\exists$ is a set of *existential states*, $Q_\forall$ is a set of *universal states*, disjoint from $Q_\exists$, $q_0 \in Q_\exists$ is an *initial state*, $q_{acc} \in Q_\exists \cup Q_\forall$ is an *accepting state* and $q_{rej} \in Q_\exists \cup Q_\forall$ is a *rejecting state*, and δ is a *transition relation* of the form

$$\delta \subseteq Q \times (\Sigma \cup \{B\}) \times Q \times (\Sigma \cup \{B\}) \times \{-1, 0, +1\}$$

with $Q = Q_\exists \cup Q_\forall$. Here B is the the *blank symbol*. We let $\delta(q, a) = \{(q', b, D) \mid (q, a, q', b, D) \in \delta\}$. W.l.o.g. we assume that in $\mathcal{M}$, universal and existential states are strictly alternating: if $(q, a, q', b, m) \in \delta$ and $q \in Q_\exists$ (resp., $q \in Q_\forall$), then $q' \in Q_\forall$ (resp., $q' \in Q_\exists$). We further assume that $|\delta(q, a)| = 2$ for all combinations of states $q \in Q$ and symbols $a \in \Sigma$. If $\delta(q, a) = \{(q_1, a_1, D_1), (q_2, a_2, D_2)\}$, we let $\delta_\ell(q, a) = (q_1, a_1, D_1)$ and $\delta_r(q, a) = (q_2, a_2, D_2)$.

A run of an ATM $\mathcal{M}$ on an input word w is defined as usual. We assume a word $w = d_1 \cdots d_n \in \Sigma^*$ with $n > 0$ together with an ATM $\mathcal{M}$ that only uses the tape cells where the input word was written, i.e., it only uses the first n cells. Checking if such $\mathcal{M}$ accepts w is an ExpTime-hard problem.

We show how to construct $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and $(\mathcal{C}, \{s(a)\})$ such that $\mathcal{M}$ accepts w iff $\mathcal{K}$ validates $(\mathcal{C}, \{s(a)\})$. The reduction takes polynomial time in the size of $\mathcal{M}$ and w. It uses the following symbols:

- a concept name *Init* and a shape name s_{acc};
- role names $succ$, $succ_\ell$, $succ_r$;
- shape names s_q for all states $q \in Q_\exists \cup q \in Q_\forall$;
- shape names h_i for all $1 \leq i \leq n$;
- shape names $c_b^{(i)}$ for all $b \in \Sigma \cup \{B\}$ and $1 \leq i \leq n$.

We let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be such that $\mathcal{A} = \{Init(a)\}$ and the TBox $\mathcal{T}$ consists of the following inclusions:

$$\begin{array}{lll} Init \sqsubseteq \exists succ_\ell & succ_\ell \sqsubseteq succ & \exists succ^- \sqsubseteq \exists succ_\ell \\ Init \sqsubseteq \exists succ_r & succ_r \sqsubseteq succ & \exists succ^- \sqsubseteq \exists succ_r. \end{array}$$

The interpretation *can*($\mathcal{K}$) will provide us an infinite binary tree. In there, the root is representing the starting configuration of $\mathcal{M}$ and each child of a node represents a next step in the run of the ATM $\mathcal{M}$.

To mimic the start configuration, we define the following shapes:

$$h_1 \leftarrow Init \qquad s_{q_0} \leftarrow Init \qquad c_{d_i}^{(i)} \leftarrow Init \text{ for all } 1 \leq i \leq n$$

Intuitively, this is setting the starting state to q_0 (s_{q_0}), putting the head in the starting position (h_1), and stating the starting symbol written on each tape cell ($c_{d_i}^{(i)}$).

The next step is to encode the transition relation of the $\mathcal{M}$. For each $1 \leq i \leq n$, each $(q, a) \in Q \times (\Sigma \cup \{B\})$, and $\gamma \in \{\ell, r\}$ we add the following shapes, where $(q', b, D) = \delta_\gamma(q, a)$:

$$\begin{aligned} s_{q'} &\leftarrow \exists succ_\gamma^-.(s_q \land h_i \land c_a^{(i)}) \\ c_b^{(i)} &\leftarrow \exists succ_\gamma^-.(s_q \land h_i \land c_a^{(i)}) \\ h_{i+D} &\leftarrow \exists succ_\gamma^-.(s_q \land h_i \land c_a^{(i)}). \end{aligned}$$

Furthermore, the tape cells that are not under the read-write head have their content preserved. Thus, for each $1 \leq i < j \leq n$, add

$$c_a^{(i)} \leftarrow \exists succ^-.(c_a^{(i)} \land h_j).$$

We now identify subtrees that represent accepting computations. For all $q \in Q_\exists$ and all $q' \in Q_\forall$ we add the following:

$$\begin{aligned} s_{acc} &\leftarrow s_{q_{acc}} \\ s_{acc} &\leftarrow s_q \land \exists succ.s_{acc} \\ s_{acc} &\leftarrow s_{q'} \land \exists succ_\ell.s_{acc} \land \exists succ_r.s_{acc}. \end{aligned}$$

This concludes the reduction. It is not too difficult to check that $\mathcal{M}$ accepts w iff $\mathcal{K}$ validates $(\mathcal{C}, \{s(a)\})$. □

7 Conclusions and Outlook

We have considered the validation of SHACL constraints in the presence of OWL 2 QL ontologies. We have defined the semantics over a carefully constructed notion of canonical model (aka chase) that minimizes the number of fresh successors introduced to satisfy the ontology axioms at each chase step, and we have argued that this semantics is natural and intuitive. We proposed a rewriting algorithm for recursive SHACL constraints with *stratified* negation. It takes as an input a SHACL shapes graph and an ontology, and constructs a new SHACL shapes graph (also with stratified negation) that can be used for sound and complete validation over the data graph alone, without needing to reason about the ontology at validation time. We showed that, under our semantics, validation in the presence of OWL 2 QL ontologies is complete for ExpTime, but it remains PTime complete in data complexity, and hence it is not harder than validation of stratified SHACL alone, without the ontology.

There are several directions for future work. The first is to extend our approach to support more syntactic features of SHACL, like *complex path expressions* and *counting* (number restrictions on paths). We believe the mentioned features can be incorporated and supported by our rewriting approach in principle, but it requires a substantial extension. Another direction is to support ontology languages that go beyond OWL 2 QL. We believe our approach can be elegantly generalized to ontologies expressed in *Horn*-$\mathcal{SHIQ}$, but it is more challenging to support non-Horn ontology languages. An implementation of our approach also remains for future work. The rewriting algorithm in this paper was meant to demonstrate the principle feasibility of the approach. As presented, our rewriting is best-case exponential since rule 3 in Definition 5.3 forces us to add exponentially many pairs (W, H). A way to avoid this problem will be needed in order to achieve an efficient implementation of the rewriting. Extending the SHACL fragment to consider *unstratified* negation is also an interesting direction for future work. For this, the *well-founded semantics* for SHACL seems like a promising starting point [2, 6].

Acknowledgements

The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101034440.

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. In addition, Šimkus was partially supported by the Austrian Science Fund (FWF) project P30873 and Ahmetaj was supported by the FWF and netidee SCIENCE project T1349-N.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu, *Foundations of Databases*, Addison-Wesley, 1995.

[2] Medina Andresel, Julien Corman, Magdalena Ortiz, Juan L. Reutter, Ognjen Savkovic, and Mantas Šimkus, 'Stable model semantics for recursive SHACL', in *Proc. of The Web Conference 2020*, p. 1570–1580. ACM, (2020).

[3] K. R. Apt, H. A. Blair, and A. Walker, *Towards a Theory of Declarative Knowledge*, 89–148, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[4] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler, *An Introduction to Description Logic*, Cambridge University Press, 2017.

[5] Stefan Borgwardt and Walter Forkel, 'Closed-world semantics for conjunctive queries with negation over ELH-bottom ontologies', in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, ed., Sarit Kraus, pp. 6131–6135. ijcai.org, (2019).

[6] Adrian Chmurovic and Mantas Simkus, 'Well-founded semantics for recursive SHACL', in *Proceedings of the 4th International Workshop on the Resurgence of Datalog in Academia and Industry (Datalog-2.0 2022)*, eds., Mario Alviano and Andreas Pieris, volume 3203 of *CEUR Workshop Proceedings*, pp. 2–13. CEUR-WS.org, (2022).

[7] Julien Corman, Juan L. Reutter, and Ognjen Savkovic, 'Semantics and validation of recursive SHACL', in *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I*, eds., Denny Vrandecic, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, volume 11136 of *Lecture Notes in Computer Science*, pp. 318–336. Springer, (2018).

[8] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel, 'The chase revisited', in *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, eds., Maurizio Lenzerini and Domenico Lembo, pp. 149–158. ACM, (2008).

[9] Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao, 'Query rewriting for Horn-SHIQ plus rules', in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*, eds., Jörg Hoffmann and Bart Selman. AAAI Press, (2012).

[10] Wenfei Fan and Floris Geerts, 'Relative information completeness', *ACM Trans. Database Syst.*, (November 2010).

[11] Matthias Knorr, José Júlio Alferes, and Pascal Hitzler, 'Local closed world reasoning with description logics under the well-founded semantics', *Artif. Intell.*, (2011).

[12] Markus Krötzsch, 'OWL 2 profiles: An introduction to lightweight ontology languages', in *Reasoning Web. Semantic Technologies for Advanced Query Answering - 8th International Summer School 2012, Vienna, Austria*, eds., Thomas Eiter and Thomas Krennwallner, volume 7487 of *Lecture Notes in Computer Science*, pp. 112–183. Springer, (2012).

[13] Boris Motik, Ian Horrocks, and Ulrike Sattler, 'Adding integrity constraints to OWL', in *Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions, Innsbruck, Austria, June 6-7, 2007*, eds., Christine Golbreich, Aditya Kalyanpur, and Bijan Parsia, volume 258 of *CEUR Workshop Proceedings*. CEUR-WS.org, (2007).

[14] Boris Motik, Ian Horrocks, and Ulrike Sattler, 'Bridging the gap between OWL and relational databases', in *Proceedings of WWW*. ACM, (2007).

[15] Paolo Pareti, George Konstantinidis, and Fabio Mogavero, 'Satisfiability and containment of recursive SHACL', *Journal of Web Semantics*, **74**, 100721, (2022).

[16] Ognjen Savkovic, Evgeny Kharlamov, and Steffen Lamparter, 'Validation of SHACL constraints over KGs with OWL 2 QL ontologies via rewriting', in *The Semantic Web - 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2-6, 2019, Proceedings*, eds., Pascal Hitzler, Miriam Fernández, Krzysztof Janowicz, Amrapali Zaveri, Alasdair J. G. Gray, Vanessa López, Armin Haller, and Karl Hammar, volume 11503 of *Lecture Notes in Computer Science*, pp. 314–329. Springer, (2019).

[17] W3C, 'Shape constraint language (SHACL)', *Technical Report.*, (2017). https://www.w3.org/TR/shacl.