

Deciding SHACL Shape Containment through Description Logics Reasoning (Extended Version)

Martin Leinberger¹, Philipp Seifer², Tjitze Rienstra¹, Ralf Lämmel², and
Steffen Staab^{3,4}

¹ Inst. for Web Science and Technologies, University of Koblenz-Landau, Germany

² The Software Languages Team, University of Koblenz-Landau, Germany

³ Institute for Parallel and Distributed Systems, University of Stuttgart, Germany

⁴ Web and Internet Science Research Group, University of Southampton, England

Abstract. The Shapes Constraint Language (SHACL) allows for formalizing constraints over RDF data graphs. A shape groups a set of constraints that may be fulfilled by nodes in the RDF graph. We investigate the problem of containment between SHACL shapes. One shape is contained in a second shape if every graph node meeting the constraints of the first shape also meets the constraints of the second. To decide shape containment, we map SHACL shape graphs into description logic axioms such that shape containment can be answered by description logic reasoning. We identify several, increasingly tight syntactic restrictions of SHACL for which this approach becomes sound and complete.

1 Introduction

RDF has been designed as a flexible, semi-structured data format. To ensure data quality and to allow for restricting its large flexibility in specific domains, the W3C has standardized the *Shapes Constraint Language (SHACL)*⁵. A set of SHACL shapes are represented in a *shape graph*. A shape graph represents constraints that only a subset of all possible RDF data graphs *conform* to. A SHACL processor may validate whether a given RDF data graph conforms to a given SHACL shape graph.

A shape graph and a data graph that act as a running example are presented in Fig. 1. The shape graph introduces a **PaintingShape** (line 1–4) which constrains all instances of the class **Painting**. It requires the presence of at least one `–exhibitedAt→` property (line 3) as well as that each node reachable via the `–creator→` property from a **Painting** conforms to the **PainterShape** (line 4). The **PainterShape** (lines 5–8) requires all incoming `–creator→` properties to conform to **PaintingShape** (line 6) as well as the presence of exactly one `–birthdate→` property. Lastly, the shapes define a **CubistShape** (lines 9–11) which must have an incoming `–creator→` property from a node that has an outgoing `–style→` property

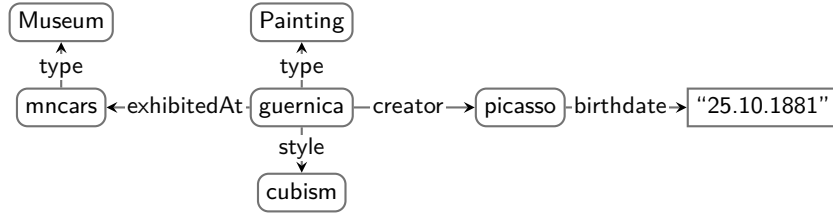
⁵ <https://www.w3.org/TR/shacl/>

```

1  :PaintingShape a sh:NodeShape;
2    sh:targetClass :Painting;
3    sh:property [ sh:path :exhibitedAt; sh:minCount 1; ];
4    sh:property [ sh:path :creator; sh:node :PainterShape; ].
5
6  :PainterShape a sh:NodeShape;
7    sh:property [ sh:inversePath :creator; sh:node :PaintingShape; ];
8    sh:property [ sh:path :birthdate; sh:minCount 1; sh:maxCount 1; ];
9
10 :CubistShape a sh:NodeShape
11   sh:property [ sh:path ( [sh:inversePath :creator] :style );
12     sh:minCount 1; sh:value :cubism; ].

```

(a) Example for a SHACL shape graph.



(b) Example for a data graph that conforms to the shape graph.

Fig. 1: Example of a shape graph (a) and a data graph (b).

to the node `cubism`. The graph shown in Fig. 1 conforms to this set of shapes as it satisfies the constraints imposed by the shape graph.

In this paper, we investigate the problem of *containment* between shapes: Given a shape graph S including the two shapes s and s' , intuitively s is contained in s' if and only if every data graph node that conforms to s is also a node that conforms to s' . An example of a containment problem is the question whether `CubistShape` is contained in `PainterShape` for all possible RDF graphs. While containment is not directly used in the validation of RDF graphs with SHACL, it offers means to tackle a broad range of other problems such as SHACL constraint debugging, query optimization [6,1,2] or program verification [17]. As an example of query optimization, assume that `CubistShape` is contained in `PainterShape` and that the graph being queried conforms to the shapes. A query querying for $?X$ and $?Y$ such that $?X$ -style→`cubism`, $?X$ -creator→ $?Y$ and $?X$ -exhibitedAt→ $?Z$ can be optimized. Since nodes that are results for $?Y$ must conform to `CubistShape` and `CubistShape` is contained in `PainterShape`, nodes that are results for $?X$ must conform to `PaintingShape`. Subsequently, the pattern $?X$ -exhibitedAt→ $?Z$ can be removed without consequence.

Given a set of shapes S , checking whether a shape s is contained in another shape s' involves checking whether there is no counterexample. That means, searching for a graph that conforms to S , but in which a node exists that con-

forms to s' but not to s . A similar problem is concept subsumption in description logics (DL). For DL, efficient tableau-based approaches [5] are known that either disprove concept subsumption by constructing a counterexample or prove that no counterexample can exist. Despite the fundamental differences between the Datalog-inspired semantics of SHACL [11] and the Tarski-style semantics used by description logics, we leverage concept subsumption in description logic by translating SHACL shapes into description logic knowledge bases such that the shape containment problem can be answered by performing a subsumption check.

Contributions We propose a translation of the containment problem for SHACL shapes into a DL concept subsumption problem such that the formal semantics of SHACL shapes as defined in [11] is preserved. Our contributions are as follows:

1. We define a syntactic translation of a set of SHACL shapes into a description logic knowledge base and show that models of this knowledge base and the idea of faithful assignments for RDF graphs in SHACL can also be mapped into each other.
2. We show that by using the translation, the containment of SHACL shapes can be decided using DL concept subsumption.
3. Based on the translation and the resulting description logic, we identify syntactic restrictions of SHACL for which the approach is sound and complete.

Organization The paper first recalls the basic syntax and semantics of SHACL and description logics in Section 2. We describe how sets of SHACL shapes are translated into a DL knowledge base in Section 3. Section 4 investigates how to use standard DL entailment for deciding shape containment. Finally, we discuss related work in Section 5 and summarize our results.

2 Preliminaries

2.1 Shape Constraint Language

The Shapes Constraint Language (SHACL) is a W3C standard for validating RDF graphs. For this, SHACL distinguishes between the *shape graph* that contains the schematic definitions (e.g., Fig. 1a) and the *data graph* that is being validated (e.g., Fig. 1b). A shape graph consists of *shapes* that group *constraints* and provide so called *target nodes*. Target nodes specify which nodes of the data graph have to be valid with respect to the constraints in order for the graph to be valid. In the following, we rely on the definitions presented by [11].

Data graphs We assume familiarity with RDF. We abstract away from concrete RDF syntax though, representing an RDF Graph G as a labeled oriented graph $G = (V_G, E_G)$ where V_G is the set of nodes of G and E_G is a set of triples of the form (v_1, p, v_2) meaning that there is an edge in G from v_1 to v_2 labeled with the property p . We use \mathcal{V} to denote the set of all possible graph nodes and \mathcal{E}

to denote the set of all possible triples. A subset $\mathcal{V}_C \subseteq \mathcal{V}$ represents the set of possible RDF classes. Furthermore, we use \mathcal{G} to denote the set of all possible RDF graphs.

Constraints While shape graphs and constraints are typically given as RDF graphs, we use a logical abstraction in the following. We use \mathcal{N}_S to refer to the set of all possible shape names. A constraint ϕ from the set of all constraints Φ is then constructed as follows:

$$\phi ::= \top \mid s \mid v \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \geq_n \rho.\phi \quad (1)$$

$$\rho ::= p \mid \hat{\rho} \mid \rho_1/\rho_2 \quad (2)$$

where \top represents a constraint that is always true, $s \in \mathcal{N}_S$ references a shape name, $v \in \mathcal{V}$ is a graph node, $\neg\phi$ represents a negated constraint and $\geq_n \rho.\phi$ indicates that there must be at least n successors via the path expression ρ that satisfy the constraint ϕ . For simplicity, we restrict ourselves to path expressions ρ comprising of either standard properties p , inverse of path $\hat{\rho}$, and concatenations of two paths ρ_1/ρ_2 . We therefore leave out operators for transitive closure and alternative paths. We use \mathcal{P} to indicate the set of all possible path expressions. A number of additional syntactic constructs can be derived from these basic constructors, including $\phi_1 \vee \phi_2$ for $\neg(\neg\phi_1 \wedge \neg\phi_2)$, $\leq_n \rho.\phi$ for $\neg(\geq_{n+1} \rho.\phi)$, $=_n \rho.\phi$ for $(\leq_n \rho.\phi) \wedge (\geq_n \rho.\phi)$, and $\forall \rho.\phi$ for $\leq_0 \rho.\neg\phi$. As an example, the constraint of **CubistShape** (see Fig. 1) can be expressed as $\geq_1(\hat{\text{creator/style}}).\text{cubism}$.

Evaluation of constraints is rather straightforward with the exception of reference cycles. To highlight this issue, consider a shape name **Local** with its constraint $\forall \text{knows}.\text{Local}$. In order to fulfill the constraint, any graph node reachable through $-\text{knows} \rightarrow$ must conform to **Local**. Consider a graph with a single vertex b_1 whose $-\text{knows} \rightarrow$ property points to itself (see Fig. 2).

$$\phi_{\text{Local}} = \forall \text{knows}.\text{Local}$$

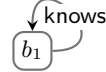


Fig. 2: Illustration of a problematic, recursive case.

Intuitively, there are two possible solutions. If b_1 is assumed to conform to **Local**, then the constraint is satisfied and it is correct to say that b_1 conforms to **Local**. If b_1 is assumed to not conform to **Local**, then the constraint is violated and it is correct to say that b_1 does not conform to **Local**. We follow the proposal of [11] and ground evaluation of constraints using *assignments*. An assignment σ maps graph nodes v to shape names s . Evaluation of constraints takes an assignment as a parameter and evaluates the constraints with respect to the given assignment. The case above is therefore represented through two different assignments—one in which $\text{Local} \in \sigma(b_1)$ and a different one where $\text{Local} \notin \sigma(b_1)$. We require assignments to be total, meaning that they map all graph nodes to the set of all shapes that the node supposedly conforms to. This

disallows certain combinations of reference cycles and negation in constraints, in essence requiring them to be stratified. In contrast, [11] also defines partial assignments, lifting this restriction. Due to the lack of space, we refer to [11] for an in depth discussion on the differences of total and partial assignments.

Definition 1 (Assignment). Let $G = (V_G, E_G)$ be an RDF graph and S a set of shapes with its set of shape names $\text{Names}(S)$. An assignment σ is a total function $\sigma : V_G \rightarrow 2^{\text{Names}(S)}$ mapping graph nodes $v \in V_G$ to subsets of shape names. If a shape name $s \in \sigma(v)$, then v is assigned to the shape name s . For all $s \notin \sigma(v)$, the node v is not assigned to the shape s .

Evaluating whether a node v in G satisfies a constraint ϕ , written $\llbracket \phi \rrbracket^{v,G,\sigma}$, is defined as shown in Fig. 3.

$$\begin{aligned}
\llbracket \top \rrbracket^{v,G,\sigma} &= \text{true} \\
\llbracket \phi_1 \wedge \phi_2 \rrbracket^{v,G,\sigma} &= \begin{cases} \text{true if } \llbracket \phi_1 \rrbracket^{v,G,\sigma} = \text{true} \wedge \\ \llbracket \phi_2 \rrbracket^{v,G,\sigma} = \text{true} \\ \text{false otherwise} \end{cases} & \llbracket \neg \phi \rrbracket^{v,G,\sigma} &= \begin{cases} \text{true if} \\ \llbracket \phi \rrbracket^{v,G,\sigma} = \text{false} \\ \text{false otherwise} \end{cases} \\
\llbracket \geq_n \rho. \phi \rrbracket^{v,G,\sigma} &= \begin{cases} \text{true if } |\{v_2 \mid (v_1, v_2) \in \llbracket r \rrbracket^G \wedge \\ \llbracket \phi \rrbracket^{v_2,G,\sigma} = \text{true}\}| \geq n \\ \text{false otherwise} \end{cases} & \llbracket v' \rrbracket^{v,G,\sigma} &= \begin{cases} \text{true if } v = v' \\ \text{false otherwise} \end{cases} \\
& & \llbracket s \rrbracket^{v,G,\sigma} &= \begin{cases} \text{true if } s \in \sigma(v) \\ \text{false otherwise} \end{cases} \\
\llbracket p \rrbracket^G &= \{(v_1, v_2) \mid \exists p : (v_1, p, v_2) \in E_G\} \\
\llbracket \wedge \rho \rrbracket^G &= \{(v_2, v_1) \mid (v_1, v_2) \in \llbracket \rho \rrbracket^G\} \\
\llbracket \rho_1 / \rho_2 \rrbracket^G &= \{(v_1, v_2) \mid \exists v : (v_1, v) \in \llbracket \rho_1 \rrbracket^G \wedge (v, v_2) \in \llbracket \rho_2 \rrbracket^G\}
\end{aligned}$$

Fig. 3: Evaluation rules for constraints and path expressions.

Shapes and Validation A shape is modelled by a triple (s, ϕ, q) . It consists of a shape name s , a constraint ϕ and a query for target nodes q . Target nodes denote those nodes which are expected to fulfill the constraint associated with the shape. Queries for target nodes are built according to the following grammar:

$$q ::= \perp \mid \{v_1, \dots, v_n\} \mid \text{class } v \mid \text{subjectsOf } p \mid \text{objectsOf } p \quad (3)$$

where \perp represents a query that targets no nodes, $\{v_1 \dots v_n\}$ targets all explicitly listed nodes with $v_1, \dots, v_n \in \mathcal{V}$, $\text{class } v$ targets all instances of the class represented by v where $v \in \mathcal{V}_C$, $\text{subjectsOf } p$ targets all subjects of the property p and $\text{objectsOf } p$ targets all objects of p . We use \mathcal{Q} to refer to the set of all possible queries and $\llbracket q \rrbracket_G$ to denote the set of nodes in the RDF graph G targeted by the query q (c.f. Fig. 4).

A shape graph is then represented by a set of shapes S whereas \mathcal{S} represents the set of all possible sets of shapes. We assume for each $(s, \phi, q) \in S$ that, if a shape name s' appears in ϕ , then there also exists a $(s', \phi', q') \in S$. Similar to

$$\begin{aligned}
\llbracket \perp \rrbracket_G &= \emptyset & \llbracket \text{class } v_2 \rrbracket_G &= \{v_1 \mid (v_1, \text{type}, v_2) \in E_G\} \\
\llbracket \{v_1, \dots, v_n\} \rrbracket_G &= \{v_1, \dots, v_n\} & \llbracket \text{subjectsOf } p \rrbracket_G &= \{v_1 \mid \exists v_2 : (v_1, p, v_2) \in E_G\} \\
& & \llbracket \text{objectsOf } p \rrbracket_G &= \{v_2 \mid \exists v_1 : (v_1, p, v_2) \in E_G\}
\end{aligned}$$

Fig. 4: Evaluation of target node queries.

$$\begin{aligned}
S_1 = \{ & \\
& (\text{PaintingShape}, \geq_1 \text{exhibitedAt}.\top \wedge \forall \text{creator.PainterShape, class Painting}), \\
& (\text{PainterShape}, =_1 \text{birthdate}.\top \wedge \forall \text{creator.PaintingShape}, \quad \perp), \\
& (\text{CubistShape}, \geq_1 \text{creator/style.cubism}, \quad \perp) \\
& \}
\end{aligned}$$

Fig. 5: Representation of the shape graph shown in Fig. 1a as a set of shapes.

[11], we refer to the language represented by the definitions above as \mathcal{L} . As an example, Fig. 5 shows the shape graph defined in Fig. 1a as a set of shapes.

Validating an RDF graph means finding a *faithful assignment*. That is, finding an assignment for which two conditions hold: First, if a node is a target node of a shape, then the assignment must assign that shape to the node. Second, if an assignment assigns a shape to a graph node, the constraint of the shape must evaluate to true. Third, when a constraint evaluates to true (false) on a node, that node must (not) be assigned to the corresponding shape.

Definition 2 (Faithful assignment). *An assignment σ for a graph $G = (V_G, E_G)$ and a set of shapes S is faithful, iff for each $(s, \phi, q) \in S$ and for each graph node $v \in V_G$, it holds that:*

- $s \in \sigma(v) \Leftrightarrow \llbracket \phi \rrbracket^{v, G, \sigma}.$
- $v \in \llbracket q \rrbracket_G \Rightarrow s \in \sigma(v).$

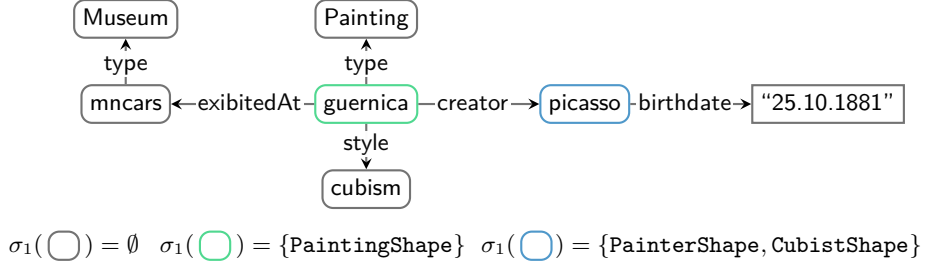
A graph that is valid with respect to a set of shapes is said to *conform* to the set of shapes.

Definition 3 (Conformance). *An RDF graph G conforms to a set of shapes S iff there is at least one faithful assignment σ for G and S . We write $\text{Faith}(G, S)$ to denote the set of all faithful assignments for G and S .*

For the data graph shown in Fig. 1b, there is a faithful assignment σ_1 that maps `PaintingShape` to `guernica` and both `PainterShape` and `CubistShape` to `picasso` (see Fig. 6). The assignment is faithful because all instances of `Painting` are assigned to `PaintingShape` and all nodes that are assigned to a shape satisfy the constraints of the shape.

2.2 Description Logics

In this paper, we focus on the highly-expressive DL $\mathcal{ALCOIQ}(o)$ as well as decidable subsets of this logic.

Fig. 6: Faithful assignment σ_1 for S_1 and the data graph shown in Fig. 1b.

Constructor Name	Syntax	Semantics
atomic property	p	$p^I \subseteq \Delta^I \times \Delta^I$
inverse role	r^-	$\{(o_2, o_1) \mid (o_1, o_2) \in r^I\}$
role composition	$r_1 \circ r_2$	$\{(o_1, o_2) \mid (o_1, o) \in r_1^I \wedge (o, o_2) \in r_2^I\}$
atomic concept	A	$A^I \subseteq \Delta^I$
nominal concept	$\{o_1, \dots, o_n\}$	$\{o_1^I, \dots, o_n^I\}$
top	\top	Δ^I
negation	$\neg C$	$\Delta^I \setminus C^I$
conjunction	$C \sqcap D$	$C^I \cap D^I$
qualified number restriction	$\geq_n r.C$	$\{o_1 \mid \{o_2 \mid (o_1, o_2) \in r^I \wedge o_2 \in C^I\} \geq n\}$

Fig. 7: Syntax and semantics of roles r (above the line) and concept expressions C, D (below the line).

Concept Expressions A description logic knowledge base K is comprised of a set of axioms. Axioms are either terminological statements, belonging to the so called TBox, or assertional statements belonging to the ABox. Statements of a knowledge base K are constructed using a signature $\text{Sig}(K)$. The signature defines atomic elements of the knowledge base, which can then be combined into expressions using a range of available constructors. The signature of a knowledge base K is a triple $\text{Sig}(K) = (N_A, N_P, N_O)$ consisting of a set of atomic concept names N_A (e.g., **Painting**) that is a subset of the set of all concept names \mathcal{N}_A , a set of relation or property names N_P (e.g., **creator**) which is a subset of \mathcal{N}_P and a set of atomic object names N_O (e.g., **guernica**) which is a subset of \mathcal{N}_O . Formal semantics of DL is based on first-order logic [4]. An interpretation I is a pair consisting of a non-empty universe Δ^I and an interpretation function \cdot^I . The interpretation function \cdot^I then maps each object $o \in N_O$ to an element of the universe $o^I \in \Delta^I$. Furthermore, the interpretation I assigns each atomic concept name $A \in N_A$ to a set $A^I \subseteq \Delta^I$ and each atomic property $p \in N_P$ to a binary relation $p^I \subseteq \Delta^I \times \Delta^I$.

As shown in Fig. 7, more complex expression can be built from the atomic elements defined in the signature. Similar to path expressions, role expressions, represented by the metavariable r , are either atomic properties p , the inverse of a role expressions r^- , or the concatenation of role expressions $r_1 \circ r_2$. Con-

Name	Syntax	Semantics
concept inclusion	$C \sqsubseteq D$	$C^I \subseteq D^I$
concept assertion	$o : C$	$o^I \in C^I$
role assertion	$(o_1, o_2) : r$	$(o_1^I, o_2^I) \in r^I$

Fig. 8: Syntax and semantics of axioms.

cept expressions, represented by the metavariables C and D , are either atomic concepts (represented by the metavariable A), nominal concepts denoted by $\{o\}$, or \top to represent the set of all objects. Concept expressions can also be composed through negation $\neg C$, conjunction $C \sqcap D$ or qualified number restrictions $\geq_n r.C$ expressing that there must be at least n successors via the role expression r that belongs to the concept C . Again, a number of additional constructors can be derived from the basic ones, such as \perp for $\neg\top$, and $\exists r.C$ for $\geq_1 r.C$. We use \mathcal{C} to denote the set of all possible concept expressions and \mathcal{R} for the set of all possible role expressions. As an example, a concept expression representing the set of paintings that have a creator relation can be expressed as **Painting** $\sqcap \exists \text{creator}.\top$.

Axioms Using the previously defined concept expressions, terminological and assertional axioms can be defined (see Figure 8). Two concept expressions can be in a subsumptive relationship $C \sqsubseteq D$. For example, the axiom $\exists \text{exhibitedAt}.\top \sqsubseteq \text{Painting}$ says that everything that has an **exhibitedAt** property is also a **Painting**. Furthermore, we use $C \equiv D$ as a shorthand for the two axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. In terms of the assertional axioms, objects can be an instance of a concept expression $o : C$ (e.g. **guernica** : **Painting**), or be connected to another object via a role expression $(o_1, o_2) : r$ (e.g. (**guernica**, **picasso**) : **creator**).

Entailment Logical entailment is defined through interpretations. In a given interpretation I , a axiom is either true or false. For an axiom ψ built according to the syntax in Fig. 8, we use the satisfaction relationship \models if its semantics constraint according to Fig. 8 is true in a interpretation I (written $I \models \psi$). An interpretation I satisfies a set of axioms Ψ if $\forall \psi \in \Psi : I \models \psi$. An interpretation I that satisfies all axioms of a knowledge base K , written $I \models K$, is called a model of K . We use $\text{Mod}(K)$ to refer to the set of all models of K . A axiom is entailed by a knowledge base K if it is true in all models of the knowledge base K .

Definition 4 (Entailment). *Let K be a knowledge base, let ψ refer to a axiom and let \mathcal{I} be the set of all possible interpretations. The axiom ψ is entailed by K , written $K \models \psi$, if $\forall I \in \mathcal{I} : (I \models K) \Rightarrow (I \models \psi)$.*

3 From SHACL Shape Containment to Description Logic Concept Subsumption

Given two shapes s and s' that are elements of the same set of shapes S , we say that s is contained in s' if any node that conforms to s will also conform to s' for any given RDF data graph G as well as any given faithful assignment for S and G .

Definition 5 (Shape Containment). *Let S be a set of shapes with $s, s' \in \text{Names}(S)$. The shape s is contained in shape s' if:*

$$\forall G \in \mathcal{G} : \forall \sigma \in \text{Faith}(G, S) : \forall v \in V_G : s \in \sigma(v) \Rightarrow s' \in \sigma(v) \text{ with } s, s' \in \text{Names}(S)$$

We use $s <_S s'$ to indicate that shape s is contained in s' with respect to S .

Both SHACL and description logics use syntactic formulas inspired by first-order logic. However, their semantics are fundamentally different. For SHACL, we follow the Datalog-inspired semantics introduced by [11]. Description logics on the other hand adopt Tarskian-style semantics. To decide shape containment, we map sets of shapes syntactically into description logic knowledge bases such that the difference in semantics can be overcome.

The function τ_{shapes} maps a set of shapes S to a description logic knowledge base $K^{\langle S \rangle}$ using four auxiliary functions (see Fig. 9): First, τ_{name} maps shape names, RDF classes as well as properties and graph nodes onto atomic concept names, atomic property names and object names. Second, τ_{role} maps SHACL path expressions to DL role expressions. Third, τ_{constr} maps constraints to concept expressions. Fourth, τ_{target} maps queries for target nodes to concept expressions. The function τ_{shapes} maps a set of shapes S to a set of axioms such that $s <_S s'$ is true if $K^{\langle S \rangle} \models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$.

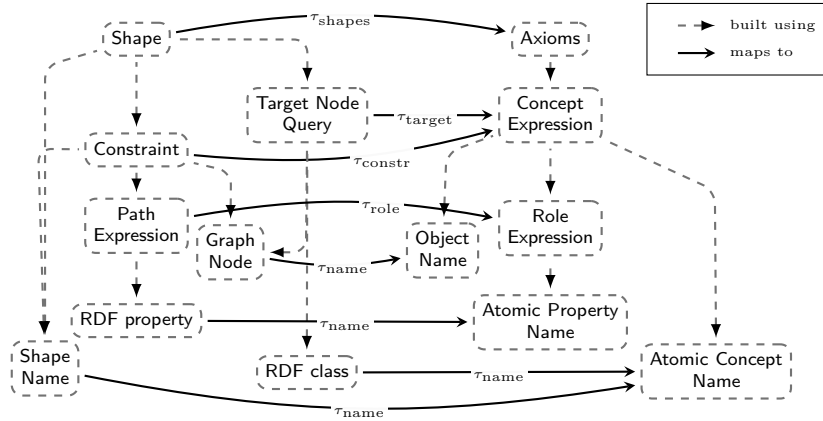


Fig. 9: Syntactic translation of SHACL to description logics.

To prove this property, we show that every finite model of $K^{\langle S \rangle}$ can be used to construct an RDF graph G and an assignment that is faithful with respect

to G and S . Likewise, a model of $K^{\langle S \rangle}$ can be constructed from an assignment that is faithful with respect to S and any given RDF graph G .

3.1 Syntactic Mapping

We map the set of shapes S into a knowledge base $K^{\langle S \rangle}$ by constraints and target node queries of each shape using the functions τ_{role} , τ_{constr} , τ_{target} , and τ_{shapes} . All those functions rely on τ_{name} which maps atomic elements used in SHACL to atomic elements of a DL knowledge base:

Definition 6 (Mapping atomic elements). *The function $\tau_{\text{name}} : \mathcal{N}_S \cup \mathcal{V}_C \cup \mathcal{V} \cup \mathcal{E} \rightarrow \mathcal{N}_A \cup \mathcal{N}_P \cup \mathcal{N}_O$ is an injective function mapping shape names and RDF classes onto atomic concept names, graph nodes onto object names as well as properties onto atomic property names.*

Definition 7 (Mapping path expressions to DL roles). *The path mapping function $\tau_{\text{role}} : \mathcal{P} \rightarrow \mathcal{R}$, is defined as follows:*

$$\begin{aligned}\tau_{\text{role}}(p) &= \tau_{\text{name}}(p) \\ \tau_{\text{role}}(\hat{\rho}) &= \tau_{\text{role}}(\rho)^- \\ \tau_{\text{role}}(\rho_1/\rho_2) &= \tau_{\text{role}}(\rho_1) \circ \tau_{\text{role}}(\rho_2)\end{aligned}$$

Definition 8 (Mapping constraints to DL concept expressions). *The constraint mapping $\tau_{\text{constr}} : \Phi \rightarrow \mathcal{C}$ is defined as follows:*

$$\begin{aligned}\tau_{\text{constr}}(\top) &= \top \\ \tau_{\text{constr}}(s) &= \tau_{\text{name}}(s) \\ \tau_{\text{constr}}(v) &= \{\tau_{\text{name}}(v)\} \\ \tau_{\text{constr}}(\phi_1 \wedge \phi_2) &= \tau_{\text{constr}}(\phi_1) \sqcap \tau_{\text{constr}}(\phi_2) \\ \tau_{\text{constr}}(\neg\phi) &= \neg\tau_{\text{constr}}(\phi) \\ \tau_{\text{constr}}(\geq_n \rho.\phi) &= \geq_n \tau_{\text{role}}(\rho).\tau_{\text{constr}}(\phi)\end{aligned}$$

Definition 9 (Mapping target node queries to DL concept expressions). *The target node mapping $\tau_{\text{target}} : \mathcal{Q} \rightarrow \mathcal{C}$ is defined as follows:*

$$\begin{aligned}\tau_{\text{target}}(\perp) &= \perp \\ \tau_{\text{target}}(\{v_1, \dots, v_n\}) &= \{\tau_{\text{name}}(v_1), \dots, \tau_{\text{name}}(v_n)\} \\ \tau_{\text{target}}(\text{class } v) &= \tau_{\text{name}}(v) \\ \tau_{\text{target}}(\text{subjectsOf } p) &= \exists \tau_{\text{name}}(p).\top \\ \tau_{\text{target}}(\text{objectsOf } p) &= \exists \tau_{\text{name}}(p)^-.\top\end{aligned}$$

The mapping $\tau_{\text{target}}(q)$ of a target query q is defined such that querying for the instances of q returns exactly the same nodes from the data graph. Likewise, the mapping $\tau_{\text{constr}}(\phi)$ is defined such that it contains those nodes for which ϕ evaluates to true and τ_{role} that the interpretation of the role expression contains those nodes that are also in the evaluation of the path expression. τ_{shapes} generalizes the construction to sets of shapes:

Definition 10 (Mapping sets of shapes to DL axioms). *The shape mapping function $\tau_{\text{shapes}} : \mathcal{S} \rightarrow \mathcal{K}$ is defined as follows:*

$$\tau_{\text{shapes}}(S) = \bigcup_{(s, \phi, q) \in S} \{\tau_{\text{target}}(q) \sqsubseteq \tau_{\text{name}}(s), \tau_{\text{constr}}(\phi) \equiv \tau_{\text{name}}(s)\}$$

To illustrate the function τ_{shapes} , the translation of the set of shapes $\tau_{\text{shapes}}(S_1) = K^{\langle S_1 \rangle}$ is shown in Fig. 10.

$$\begin{aligned}
K^{\langle S_1 \rangle} = \{ & \text{Painting} \sqsubseteq \text{PaintingShape}, \\
& \geq_1 \text{exhibitedAt}.\top \sqcap \forall \text{creator}.\text{PainterShape} \equiv \text{PaintingShape}, \\
& \perp \sqsubseteq \text{PainterShape}, \\
& \geq_1 \text{birthdate}.\top \sqcap \forall \text{creator}^-. \text{PaintingShape} \equiv \text{PainterShape}, \\
& \perp \sqsubseteq \text{Cubist}, \\
& \geq_1 \text{creator}^- \circ \text{style}.\{\text{cubism}\} \equiv \text{CubistShape} & \}
\end{aligned}$$

Fig. 10: Translation $\tau_{\text{shapes}}(S_1) = K^{\langle S_1 \rangle}$ of the set of shapes S_1 .

3.2 Construction of Faithful Assignments and Models

Given our translation, we now show that the notion of faithful assignments of SHACL and *finite models* in description logics coincide.

Definition 11 (Finite model). *Let K be a knowledge base and $I \in \text{Mod}(K)$ a model of K . The model I is finite, if its universe Δ^I is finite [8]. We use $\text{Mod}^{\text{fin}}(K)$ to refer to the set of all finite models of K .*

Given an RDF data graph G , a set of shapes S and an assignment σ that is faithful with respect to S and G , we construct an interpretation $I^{\langle G, \sigma \rangle}$ that is a finite model for the knowledge base $K^{\langle S \rangle}$.

Definition 12 (Construction of the finite model $I^{\langle G, \sigma \rangle}$). *Let S be a set of shapes, $G = (V_G, E_G)$ an RDF data graph and σ an assignment that is faithful with respect to S and G . Furthermore, let τ_{node} be the inverse of the function τ_{name} . The finite model $I^{\langle G, \sigma \rangle}$ for the knowledge base $\tau_{\text{shapes}}(S) = K^{\langle S \rangle}$ is constructed as follows:*

1. *All objects are interpreted as themselves: $\forall o \in N_O : o^I = o$.*
2. *A pair of objects is contained in the interpretation of a relation if the two objects are connected in the RDF data graph:*
 $\forall p \in N_P : \forall o_1, o_2 \in N_O : (o_1^{I^{\langle G, \sigma \rangle}}, o_2^{I^{\langle G, \sigma \rangle}}) \in p^{I^{\langle G, \sigma \rangle}}$ *if* $(\tau_{\text{node}}(o_1), p, \tau_{\text{node}}(o_2)) \in (E_G \setminus \{(v_1, \text{type}, v_2) \in E_G\})$.
3. *Objects are in the interpretation of a concept if this concept is a class used in the RDF data graph and the object is an instance of this class according to the graph:*
 $\forall A_v \in N_A : \forall o \in N_O : o^{I^{\langle G, \sigma \rangle}} \in A_v^{I^{\langle G, \sigma \rangle}}$ *if* $(\tau_{\text{node}}(o), \text{type}, \tau_{\text{node}}(A_v)) \in E_G$.
4. *Objects are in the interpretation of a concept if the concept is a shape name and the assignment σ assigns the shape to the object:*
 $\forall A_s \in N_A : \forall o \in N_O : o^{I^{\langle G, \sigma \rangle}} \in A_s^{I^{\langle G, \sigma \rangle}}$ *if* $\tau_{\text{node}}(A_s) \in \sigma(\tau_{\text{node}}(o))$.

The interpretation $I^{(G,\sigma)}$ is a model of the knowledge base $K^{(S)}$. Before we show this, it is important to notice that the interpretation of role expressions constructed through τ_{role} contains the same nodes in the interpretation $I^{(G,\sigma)}$ as the evaluation of the path expression.

Lemma 1. *Let S be a set of shapes, G an RDF data graph and σ an assignment that is faithful with respect to S and G . Furthermore, let $I^{(G,\sigma)}$ be an interpretation for $K^{(S)}$. It holds that $\forall(o_1, o_2) \in \tau_{\text{role}}(\rho)^{I^{(G,\sigma)}} \Rightarrow (\tau_{\text{node}}(o_1), \tau_{\text{node}}(o_2)) \in \llbracket \rho \rrbracket^G$ for any path expression ρ .*

Proof. The interpretation $I^{(G,\sigma)}$ contains all properties of the RDF graph. The result is then immediate from the evaluation rules of path expressions (c. f. Fig. 3) and semantics of role expressions (c. f. Fig. 7). \square

Theorem 1. *Let S be a set of shapes, G an RDF data graph and σ an assignment that is faithful with respect to S and G . Furthermore, let $K^{(S)}$ be a knowledge base that is constructed through $\tau_{\text{shapes}}(S)$. The interpretation $I^{(G,\sigma)}$ is a finite model of $K^{(S)}$ ($I^{(G,\sigma)} \models K^{(S)}$).*

Proof. $I^{(G,\sigma)}$ is a finite model of $K^{(S)}$ iff $\forall \psi \in K^{(S)} : I^{(G,\sigma)} \models \psi$ and $\Delta^{I^{(G,\sigma)}}$ is finite. For each shape $(s, \phi, q) \in S$, there are two axioms in $K^{(S)}$. First, the axiom $\tau_{\text{target}}(q) \sqsubseteq \tau_{\text{name}}(s)$. Second, the axiom $\tau_{\text{constr}}(\phi) \equiv \tau_{\text{name}}(s)$. $I^{(G,\sigma)}$ must satisfy both axioms. We start by noting that $\Delta^{I^{(G,\sigma)}}$ is finite since the RDF graph G from which the set of objects N_O is constructed is finite. The axiom $\tau_{\text{target}}(q) \sqsubseteq \tau_{\text{name}}(s)$ is satisfied in $I^{(G,\sigma)}$ by examining each case of τ_{target} individually:

$\tau_{\text{target}}(\perp) = \perp$ Vacuously satisfied as \perp is a subset of every concept expression.

$\tau_{\text{target}}(\{\bar{v}\}) = \{\tau_{\text{name}}(\bar{v})\}$ Target nodes consist of an enumeration of nodes. σ is only faithful if the shape s is assigned to all those nodes. Likewise, $\{\tau_{\text{name}}(\bar{v})\}$ constitutes a concept expression that is an enumeration of graph nodes. As all nodes that are assigned to s in σ are also in the interpretation $\tau_{\text{name}}(s)^{I^{(G,\sigma)}}$ of $\tau_{\text{name}}(s)$, the axiom $\{\tau_{\text{name}}(\bar{v})\} \sqsubseteq \tau_{\text{name}}(s)$ must be satisfied in $I^{(G,\sigma)}$.

$\tau_{\text{target}}(\text{class } v) = \tau_{\text{name}}(v)$ The assignment σ is only faithful if the shape s is assigned to all instances of v . Due to the construction of $I^{(G,\sigma)}$, all instances of v are in the interpretation $\tau_{\text{name}}(s)^{I^{(G,\sigma)}}$ of $\tau_{\text{name}}(s)$. Subsequently, $\tau_{\text{name}}(v) \sqsubseteq \tau_{\text{name}}(s)$ must be satisfied in $I^{(G,\sigma)}$.

$\tau_{\text{target}}(\text{subjectsOf } p) = \exists \tau_{\text{role}}(p). \top$ The assignment σ is faithful if shape s is assigned to all nodes that have the given property p . Since the interpretation $I^{(G,\sigma)}$ is constructed using σ , all nodes having that property must be in the interpretation $\tau_{\text{name}}(s)^{I^{(G,\sigma)}}$ of $\tau_{\text{name}}(s)$. Subsequently, $\exists \tau_{\text{role}}(p). \top \sqsubseteq \tau_{\text{name}}(s)$ must be satisfied in $I^{(G,\sigma)}$.

$\tau_{\text{target}}(\text{objectsOf } p) = \exists p^-. \top$ The assignment σ is faithful if shape s is assigned to all nodes that have the given incoming p relation. Due to the construction of $I^{(G,\sigma)}$ (c. f. Definition 12), all nodes that have an incoming relation via the property must be in the interpretation $\tau_{\text{name}}(s)^{I^{(G,\sigma)}}$ of $\tau_{\text{name}}(s)$. Subsequently, the axiom $\exists p^-. \top \sqsubseteq \tau_{\text{name}}(s)$ must be satisfied in $I^{(G,\sigma)}$.

We continue by showing that $\tau_{\text{constr}}(\phi) \equiv \tau_{\text{name}}(s)$ is satisfied in $I^{(G, \sigma)}$ via induction over τ_{constr} :

- $\tau_{\text{constr}}(\top) = \top$ If $\phi = \top$, then $\llbracket \top \rrbracket^{o, G, \sigma}$ evaluates to true for all nodes. Therefore the shape s is assigned to all nodes and subsequently the concept $\tau_{\text{name}}(s)^{I^{(G, \sigma)}}$ contains all nodes due to the construction of $I^{(G, \sigma)}$. This is equivalent to the concept \top .
- $\tau_{\text{constr}}(s') = \tau_{\text{name}}(s')$ If $\phi = s'$, then $\llbracket s' \rrbracket^{o, G, \sigma}$ evaluates to true if $s' \in \sigma(o)$. Due to the construction of $I^{(G, \sigma)}$, all nodes for which this is true must also be in the interpretation $\tau_{\text{name}}(s)^{I^{(G, \sigma)}}$ of $\tau_{\text{name}}(s)$. Subsequently, the axiom $\tau_{\text{name}}(s') \equiv \tau_{\text{name}}(s)$ must be true in $I^{(G, \sigma)}$.
- $\tau_{\text{constr}}(v) = \{\tau_{\text{name}}(v)\}$ If $\phi = v$, then $\llbracket v \rrbracket^{v', G, \sigma}$ evaluates to true if $v = v'$. Therefore, the shape s is assigned to this node. Due to the construction of $I^{(G, \sigma)}$, $\tau_{\text{name}}(v)$ is the only node in the interpretation of $\tau_{\text{name}}(s)^{I^{(G, \sigma)}}$. The axiom $\{\tau_{\text{name}}(v)\} \equiv \tau_{\text{name}}(s)$ must therefore be true in $I^{(G, \sigma)}$.
- $\tau_{\text{constr}}(\phi_1 \wedge \phi_2) = \tau_{\text{constr}}(\phi_1) \sqcap \tau_{\text{constr}}(\phi_2)$ Evaluation of the constraint $\llbracket \phi_1 \wedge \phi_2 \rrbracket^{o, G, \sigma}$ evaluates to true for nodes where ϕ_1 and ϕ_2 evaluate to true. By induction hypothesis, $\tau_{\text{constr}}(\phi_1) = C$ is a concept expression that is equivalent to the set of nodes for which ϕ_1 evaluates to true. Likewise for $\tau_{\text{constr}}(\phi_2) = D$. The set of nodes for which both ϕ_1 and ϕ_2 evaluate to true must therefore be the intersection of $C \sqcap D$. Due to the construction of $I^{(G, \sigma)}$, those nodes must also be in the interpretation $\tau_{\text{name}}(s)^{I^{(G, \sigma)}}$ of $\tau_{\text{name}}(s)$. The axiom must therefore be true.
- $\tau_{\text{constr}}(\neg \phi_1) = \neg \tau_{\text{constr}}(\phi_1)$ By hypothesis, $\tau_{\text{constr}}(\phi_1) = C$ is a concept expression that is equivalent to the set of nodes for which ϕ_1 evaluates to true. Evaluation of the constraint $\llbracket \neg \phi_1 \rrbracket^{o, G, \sigma}$ evaluates to true for those nodes in which ϕ_1 evaluates to false. Since those nodes are assigned to s in σ , the interpretation $\tau_{\text{name}}(s)^{I^{(G, \sigma)}}$ of $\tau_{\text{name}}(s)$ must also be those nodes. This is equivalent to the interpretation of $\neg C$. The axiom $\neg C \equiv \tau_{\text{name}}(s)$ must therefore be satisfied in $I^{(G, \sigma)}$.
- $\tau_{\text{constr}}(\geq_n \rho. \phi_1) = \geq_n r. \tau_{\text{constr}}(\phi_1)$ By hypothesis, $\tau_{\text{constr}}(\phi_1) = C$ is a concept expression that represents the set of graph nodes for which ϕ_1 evaluates to true. $\llbracket \geq_n \rho. \phi_1 \rrbracket^{o, G, \sigma}$ evaluates to true for those nodes that have at least n successors via ρ and for which ϕ_1 evaluates to true. Those nodes must also be in the interpretation $\tau_{\text{name}}(s)^{I^{(G, \sigma)}}$ of $\tau_{\text{name}}(s)$. Due to the construction of $I^{(G, \sigma)}$, all graph nodes having n successor via r in G must also have n successors in $I^{(G, \sigma)}$. Subsequently, the axiom $\geq_n r. C \equiv \tau_{\text{name}}(s)$ must be true in $I^{(G, \sigma)}$. \square

Furthermore, we show that any finite model I of a knowledge base $K^{(S)}$ built from a set of shapes S can be transformed into an RDF graph $G^{(I)}$ and an assignment $\sigma^{(I)}$ such that $\sigma^{(I)}$ is faithful with respect to S and $G^{(I)}$. We construct $G^{(I)}$ and $\sigma^{(I)}$ in the following manner:

Definition 13 (Construction of $G^{(I)}$ and $\sigma^{(I)}$). *Let S be a set of shapes and $K^{(S)}$ a knowledge base constructed via $\tau_{\text{shapes}}(S)$. Furthermore, let $I \in$*

$\text{Mod}^{\text{fin}}(K^{\langle S \rangle})$ be a finite model of $K^{\langle S \rangle}$. The RDF graph $G^{\langle I \rangle} = (V_G^{\langle I \rangle}, E_G^{\langle I \rangle})$ and the assignment $\sigma^{\langle I \rangle}$ can then be constructed as follows:

1. The interpretations of all relations are interpreted as relations between graph nodes in the RDF graph:
 $\forall p \in N_P : (\sigma^I, \sigma'^I) \in p^I \Rightarrow (\tau_{\text{node}}(o), p, \tau_{\text{node}}(o')) \in E_G^{\langle I \rangle}$.
2. The interpretations of all concepts that are not shape names are triples indicating an instance in the RDF graph:
 $\forall A \in N_A : (\sigma^I \in A^I \wedge A \notin \text{Names}(S)) \Rightarrow (\tau_{\text{node}}(o), \text{type}, \tau_{\text{node}}(A)) \in E_G^{\langle I \rangle}$.
3. The interpretations of all concept names that are shape names are used to construct the assignment
 $\sigma^{\langle I \rangle} : \forall A \in N_A : (\sigma^I \in A^I \wedge A \in \text{Names}(S)) \Rightarrow \tau_{\text{node}}(A) \in \sigma^{\langle I \rangle}(\tau_{\text{node}}(o))$.

An assignment $\sigma^{\langle I \rangle}$ constructed in this manner is faithful with respect to the constructed RDF graph $G^{\langle I \rangle}$ and the set of shapes S .

Theorem 2. Let S be a set of shapes and $K^{\langle S \rangle}$ be a knowledge base constructed through $\tau_{\text{shapes}}(S)$. Furthermore, let $I \in \text{Mod}^{\text{fin}}(K^{\langle S \rangle})$ be a finite model for $K^{\langle S \rangle}$. The assignment $\sigma^{\langle I \rangle}$ is faithful with respect to S and $G^{\langle I \rangle}$.

Proof. $\sigma^{\langle I \rangle}$ is faithful with respect to S and $G^{\langle I \rangle}$ if two conditions hold:

1. Each shape is assigned to all of its target nodes.
2. If a shape is assigned to a node, then the constraint evaluates to true. Likewise, if the constraint evaluates to true for a node, then the shape is assigned to the node.

The knowledge base $\tau_{\text{shapes}}(S) = K^{\langle S \rangle}$ contains two axioms of the form $\tau_{\text{target}}(q) \sqsubseteq \tau_{\text{name}}(s)$ and $\tau_{\text{constr}}(\phi) \equiv \tau_{\text{name}}(s)$ for each (s, ϕ, q) . $I \in \text{Mod}(K^{\langle S \rangle})$ satisfies those axioms. We proceed by examining each case of τ_{target} individually:

$\tau_{\text{target}}(\perp) = \perp$ Vacuously satisfied as no target nodes exist.

$\tau_{\text{target}}(\{v_1, \dots, v_n\}) = \{\tau_{\text{name}}(v_1), \dots, \tau_{\text{name}}(v_n)\}$ The target nodes are an enumeration of nodes v_1, \dots, v_n . I is only a model if $\{\tau_{\text{name}}(v_1), \dots, \tau_{\text{name}}(v_n)\}^I \subseteq \tau_{\text{name}}(s)^I$ is true. Due to the construction of $G^{\langle I \rangle}$, all nodes $\tau_{\text{name}}(v_1), \dots, \tau_{\text{name}}(v_n)$ must exist in $G^{\langle I \rangle}$. Due to the construction of $\sigma^{\langle I \rangle}$, the shape s is assigned to all nodes in $\tau_{\text{name}}(s)^I$. As such, the shape is assigned to all of its target nodes.

$\tau_{\text{target}}(\text{class } v) = \tau_{\text{name}}(v)$ Target nodes are instances of a concept. I is only a model if $\tau_{\text{name}}(v)^I \subseteq \tau_{\text{name}}(s)^I$ is true. Due to the construction of $G^{\langle I \rangle}$, the concept $\tau_{\text{name}}(v)$ and its instances $\tau_{\text{name}}(v)^I$ must exist in $G^{\langle I \rangle}$. Due to the construction of $\sigma^{\langle I \rangle}$, s is assigned to all nodes in $\tau_{\text{name}}(v)^I$. As such, the shape is assigned to all of its target nodes.

$\tau_{\text{target}}(\text{subjectsOf } p) = \exists p. \top$ Target nodes are all subjects of a property. I is only a model if $\exists p. \top^I \subseteq \tau_{\text{name}}(s)^I$ is true. Due to the construction of $G^{\langle I \rangle}$, all nodes in $\exists p. \top^I$ must exist in $G^{\langle I \rangle}$ and have the property. Due to the construction of $\sigma^{\langle I \rangle}$, the shape s is assigned to all nodes in $\exists p. \top^I$. As such, the shape is assigned to all of its target nodes.

$\tau_{\text{target}}(\text{objectsOf } p) = \exists p^-. \top$ Target nodes are objects of a property. The case is similar to the previous case.

We continue by examining each case of τ_{constr} individually:

$\tau_{\text{constr}}(\top) = \top$ The constraint evaluates to true for all nodes. If I is a model, then $\top^I \equiv \tau_{\text{name}}(s)^I$ is true. Due to the construction of $\sigma^{(I)}$, s is assigned to all nodes. As such, the shape is assigned to all nodes for which the constraint evaluates to true.

$\tau_{\text{constr}}(s') = \tau_{\text{name}}(s')$ The constraint evaluates to true for all nodes o for which $s' \in \sigma^{(I)}(o)$. Since I is a model, $\tau_{\text{name}}(s')^I \equiv \tau_{\text{name}}(s)^I$ must be true. Due to the construction of $\sigma^{(I)}$, s is assigned to all nodes which are also assigned to s' . As such, the shape is assigned to all nodes for which the constraint evaluates to true.

$\tau_{\text{constr}}(v) = \{\tau_{\text{name}}(v)\}$ The constraint evaluates only for the node v to true. Since I is a model, $\{\tau_{\text{name}}(v)\}^I \equiv \tau_{\text{name}}(s)^I$ must be true. Due to the construction of $\sigma^{(I)}$, s is only assigned to the node v .

$\tau_{\text{constr}}(\phi_1 \wedge \phi_2) = \tau_{\text{constr}}(\phi_1) \sqcap \tau_{\text{constr}}(\phi_2)$ The constraint evaluates to true for all nodes for which ϕ_1 and ϕ_2 evaluate to true. By hypothesis, $\tau_{\text{constr}}(\phi_1) = C$ and $\tau_{\text{constr}}(\phi_2) = D$ represent the set of nodes for which ϕ_1 and ϕ_2 evaluate to true, respectively. Furthermore, $(C \sqcap D)^I \equiv \tau_{\text{name}}(s)^I$ is true if I is a model. Due to the construction of $\sigma^{(I)}$, the shape s is assigned to all nodes in $(C \sqcap D)^I$. The shape is therefore assigned to all nodes for which the constraint evaluates to true.

$\tau_{\text{constr}}(\neg\phi_1) = \neg\tau_{\text{constr}}(\phi_1)$ The constraint evaluates to true for all nodes for which ϕ_1 does not evaluate to true. By hypothesis, $\tau_{\text{constr}}(\phi_1) = C$ represent the set of nodes for which ϕ_1 evaluates to true. Furthermore, since I is a model, $(\neg C)^I = \tau_{\text{name}}(s)^I$ must be true. Due to the construction of $\sigma^{(I)}$, the shape s is assigned to all nodes for which ϕ_1 evaluates to false. The shape is therefore assigned to all nodes for which the constraint evaluates to true.

$\tau_{\text{constr}}(\geq_n \rho. \phi_1) = \geq_n r. \tau_{\text{constr}}(\phi_1)$ The constraint evaluates to true for all nodes that have n successors via the path expression ρ for which the constraint ϕ_1 evaluates to true. By hypothesis, $\tau_{\text{constr}}(\phi_1) = C$ is the set of nodes for which ϕ_1 evaluates to true. Furthermore, since I is a model, $(\geq_n r. C)^I = \tau_{\text{name}}(s)^I$. Due to the construction of $\sigma^{(I)}$, the shape s is assigned to all nodes that have n successors and that are in the interpretation of C^I . The shape is therefore assigned to all nodes for which the constraint evaluates to true. \square

3.3 Deciding Shape Containment using Concept Subsumption

Given the translation rules and semantic equivalence between finite models of a description logic knowledge base and assignments for SHACL shapes, we can leverage description logics for deciding shape containment. Assume a set of shapes S containing definitions for two shapes s and s' . Those shapes are represented by atomic concepts in the knowledge base $K^{(S)}$. As the following theorem proves, deciding whether the shape s is contained in the shape s' is equivalent

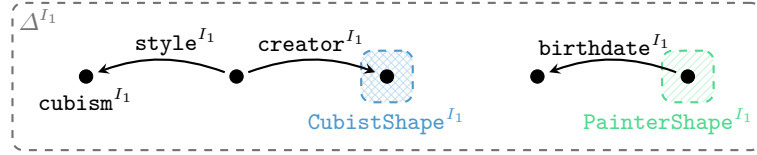
to deciding concept subsumption between s and s' in $K^{\langle S \rangle}$ using finite model reasoning.

Theorem 3 (Shape containment and concept subsumption). *Let S be a set of shapes and $K^{\langle S \rangle}$ the knowledge base constructed via $\tau_{\text{shapes}}(S)$. Let \models_{fin} indicate that an axiom is true in all finite models. It holds that:*

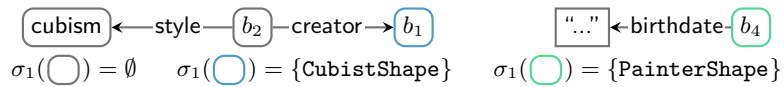
$$s <_S s' \Leftrightarrow K^{\langle S \rangle} \models_{\text{fin}} \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$$

Proof. Intuitively, the two problems are equivalent because any counterexample for one side of the equivalence relation could always be translated into a counterexample for the other side. If $K^{\langle S \rangle} \not\models_{\text{fin}} \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$, then there is a finite model of $K^{\langle S \rangle}$ in which $\tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$ is not true. Instead, there must be a model in which the concept expression $\tau_{\text{name}}(s) \sqcap \neg \tau_{\text{name}}(s')$ is true. Using Definition 13, this model can be translated into an RDF graph and an assignment (c.f. Theorem 2) that acts as a counterexample for s being contained in s' . If $K^{\langle S \rangle} \models_{\text{fin}} \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$, then there is no finite model in which $\tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$ is not true. Subsequently, there cannot be an RDF graph and an assignment that acts as a counterexample to s being contained in s' , because this could be translated into a finite model of $K^{\langle S \rangle}$ using Definition 12 (c.f. Theorem 1) and we know that no such model exists. \square

As an example, reconsider the translation of the set of shapes $\tau_{\text{shapes}}(S_1) = K^{\langle S_1 \rangle}$ (see Fig. 10). From $K^{\langle S_1 \rangle}$ follows that $K^{\langle S_1 \rangle} \not\models \text{CubistShape} \sqsubseteq \text{PainterShape}$ as there is a finite model $I_1 \in \text{Mod}^{\text{fin}}(K^{\langle S_1 \rangle})$ in which the concept expression $\text{CubistShape} \sqcap \neg \text{PainterShape}$ is satisfiable (see Fig. 11).



(a) Model of $K^{\langle S_1 \rangle}$ showing that $\text{CubistShape} \not\sqsubseteq \text{PainterShape}$.



(b) Graph and assignment showing that CubistShape is not contained in PainterShape .

Fig. 11: Counterexamples for $\text{CubistShape} <_{S_1} \text{PainterShape}$.

An important observation is that it is possible to express arbitrary concept subsumptions $C \sqsubseteq D$ despite the syntactic restrictions of τ_{shapes} .

Lemma 2. *For any axiom $C \sqsubseteq D$, one can define some $(s, \phi, q) \in S$ and $(s', \phi', q') \in S$ such that $\tau_{\text{constr}}(\phi) = C$ and $\tau_{\text{constr}}(\phi') = D$ and $\tau_{\text{shapes}}(S) \models C \sqsubseteq D$.*

Proof. The concepts C and D used in the concept subsumption axiom $C \sqsubseteq D$ follow no syntactic restrictions, but that can use the complete syntax defined in Fig. 7. The function τ_{shapes} on the other hand generates axioms of the following form:

$$C^q \sqsubseteq A^{\text{shape}}, D^\phi \equiv A^{\text{shape}}$$

where A^{shape} is a atomic concept that represents a shape name, D^ϕ is a concept expression and C^q is the translation of the target node query which adheres to the following grammar:

$$C^q ::= \perp \mid \{v_1, \dots, v_n\} \mid A^{\text{class}} \mid \exists p. \top \mid \exists p^-. \top$$

As constraints ϕ use the same syntactical connectors as concept expressions (c. f. Section 2 and Fig. 7), there is a ϕ_C such that $\tau_{\text{constr}}(\phi_C) = C$ and there is some ϕ_D such that $\tau_{\text{constr}}(\phi_D) = D$. Furthermore, for both C and D , we introduce shape names A_C^{shape} and A_D^{shape} . Thus, τ_{shapes} will produce the axioms $C \equiv A_C^{\text{shape}}$ and $D \equiv A_D^{\text{shape}}$. To represent the inclusion, target nodes must be used. However, a shape such as A_C^{shape} cannot act as a target node. Instead, we introduce an atomic concept A_C^{class} that represents an RDF class and modify the constraint for shape A_C^{shape} such that it includes the RDF class, giving us $C \sqcap A_C^{\text{class}} \equiv A_C^{\text{shape}}$. The shape A_D^{shape} can then target A_C^{class} , completing the subsumption.

In summary, the axiom $C \sqsubseteq D$ is semantically equivalent to the following set of axioms:

$$\{C \sqcap A_C^{\text{class}} \equiv A_C^{\text{shape}}, \perp \sqsubseteq A_C^{\text{shape}}, D \equiv A_D^{\text{shape}}, A_C^{\text{class}} \sqsubseteq A_D^{\text{shape}}\}$$

It is therefore possible to represent any concept subsumption $C \sqsubseteq D$ through a set of shapes. \square

For shapes belonging to the language \mathcal{L} , the corresponding description logic is $\mathcal{ALCOIQ}(\circ)$. To the best of our knowledge, finite satisfiability has not yet been investigated for $\mathcal{ALCOIQ}(\circ)$. Path concatenation can be restricted such that the fragment of SHACL corresponds to the description logic \mathcal{SROIQ} . The fragment for which constraints map to syntactical elements of \mathcal{SROIQ} , called $\mathcal{L}^{\text{restr}}$, uses the following constraint grammar:

$$\phi^{\text{restr}} ::= \top \mid s \mid v \mid \phi_1^{\text{restr}} \wedge \phi_2^{\text{restr}} \mid \neg \phi^{\text{restr}} \mid \exists \rho. \phi^{\text{restr}} \mid \geq_n p. \phi^{\text{restr}}$$

Finite satisfiability is known to be decidable for \mathcal{SROIQ} [15] and all its sublogics such as \mathcal{ALCOIQ} which completely removes role concatenation.

4 Deciding Shape Containment using Standard Entailment

While shape containment can be decided using finite model reasoning (c. f. Theorem 3), practical usability of our approach depends on whether existing reasoner

implementations can be leveraged. Implementations that are readily-available rely on standard entailment which includes infinitely large models. We therefore now focus on the soundness and completeness of our approach using the standard entailment relation.

Using standard entailment, the description logic $\mathcal{ALCOIQ}(\circ)$ which corresponds to the language \mathcal{L} , satisfiability of concepts, and thus concept subsumption, is undecidable [14]. First-order logic is semi-decidable. As $\mathcal{ALCOIQ}(\circ)$ can be translated to first-order logic through a straightforward extension of the translation rules for \mathcal{SROIQ} [21], $\mathcal{ALCOIQ}(\circ)$ is also semi-decidable. Therefore, a decision procedure can verify whether a formula is entailed in finite time, but may not terminate for non-entailed formula. More restricted description logics such as \mathcal{SROIQ} , which corresponds to $\mathcal{L}^{\text{restr}}$, are decidable, meaning that an answer by the decision procedure is guaranteed in finite time. However, the question arises whether the satisfiability of a concept implies the existence of a finite model.

Definition 14 (Finite Model Property). *A description logic has the finite model property if every concept that is satisfiable with respect to a knowledge base has a finite model [5].*

If C is a concept expression that is satisfiable with respect to some knowledge base K that belongs to a description logic having the finite model property, then there must be a finite model of K that shows the satisfiability of C . Thus, finite entailment and standard entailment are the same if a description logic has the finite model property.

Proposition 1. *The finite model property does not hold for the description logic \mathcal{ALCOIQ} [8] or more expressive description logics such as $\mathcal{ALCOIQ}(\circ)$ and \mathcal{SROIQ} . If a concept expression C is satisfiable with respect to a knowledge base K written in \mathcal{ALCOIQ} or a more expressive description logic, then it may be that there are only models with an infinitely large universe.*

To highlight Proposition 1, consider the following example adapted from [8]:

$$\begin{aligned} K_{\text{infinite}} = \{ & \text{Painting} \quad \equiv \exists \text{influences.Painting} \sqcap \\ & \leq 1 \text{influences}^-. \top, \\ & \text{NovelPainting} \equiv \text{Painting} \sqcap \leq 0 \text{influences}^-. \top \} \end{aligned}$$

Each **Painting** influences another **Painting**, but is influenced by at most one other **Painting**. A novel painting is a **Painting** that is not influenced by anything. The concept **NovelPainting** is satisfiable, but not finitely satisfiable. An object that is an instance of **NovelPainting** must have a second object which it influences. This second object must be an instance of **Painting** which means that it must influence another instance of **Painting**. This leads to an infinite sequence of paintings.

Given Proposition 1, it may be that there are only models with an infinitely large universe that show the satisfiability of a concept expression. There are three

different possibilities: (1) $\tau_{\text{name}}(s) \sqcap \neg \tau_{\text{name}}(s')$ is neither finitely nor infinitely satisfiable, meaning that $K^{\langle S \rangle} \models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$. It follows that $s <:_S s'$ is true, as there is no counterexample. (2) $\tau_{\text{name}}(s) \sqcap \neg \tau_{\text{name}}(s')$ is not finitely, but only infinitely satisfiable. It follows that $K^{\langle S \rangle} \not\models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$, but $s <:_S s'$ is true since the infinitely large model has no corresponding RDF graph. (3) $\tau_{\text{name}}(s) \sqcap \neg \tau_{\text{name}}(s')$ is both, finitely and infinitely, satisfiable. It follows that $K^{\langle S \rangle} \not\models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$ and indeed $s <:_S s'$ is false since the finite model can be translated into an RDF graph and a faithful assignment. Deciding shape containment for the shape languages that are translatable into $\mathcal{ALCOIQ}(\circ)$, \mathcal{SROIQ} or \mathcal{ALCOIQ} is therefore sound, provided that the decision procedure terminates.

Theorem 4. *Let S be a set of shapes of the language $\mathcal{L}^{\text{restr}}$. It then holds that:*

$$s <:_S s' \Leftarrow \tau_{\text{shapes}}(S) \models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$$

Proof. For $\mathcal{L}^{\text{restr}}$, the corresponding DL is \mathcal{SROIQ} for which the finite model property does not hold. If $K^{\langle S \rangle} \models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$, then there is neither a finitely nor an infinitely large model in which $\tau_{\text{name}}(s) \sqcap \neg \tau_{\text{name}}(s')$ is satisfiable. The shape s must therefore be contained in the shape s' as there is no RDF graph and assignment that acts as a counterexample. \square

However, the approach is incomplete as it may be that $s <:_S s'$ but $K^{\langle S \rangle} \not\models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$ because due to an infinitely large model in which $\tau_{\text{name}}(s) \sqcap \neg \tau_{\text{name}}(s')$ is satisfiable.

To restore the finite model property, inverse path expressions have to be removed. That is, the set of SHACL shapes S must belong to the language fragment $\mathcal{L}^{\text{non-inv}}$ that uses the following grammar:

$$\begin{aligned} \phi^{\text{non-inv}} &::= \top \mid v \mid s \mid \phi_1^{\text{non-inv}} \wedge \phi_2^{\text{non-inv}} \mid \neg \phi^{\text{non-inv}} \mid \geq_n p. \phi^{\text{non-inv}} \\ q^{\text{non-inv}} &::= \perp \mid \{v_1, \dots, v_n\} \mid \text{class } v \mid \text{subjectsOf } p \end{aligned}$$

As a result, the description logic that corresponds to $\mathcal{L}^{\text{non-inv}}$ is \mathcal{ALCOQ} .

Proposition 2. *The description logic \mathcal{ALCOQ} has the finite model property [18].*

Subsequently, for SHACL shapes that belong to $\mathcal{L}^{\text{non-inv}}$ shape containment and concept subsumption in the knowledge base constructed from the set of shapes are equivalent.

Theorem 5. *Let S be a set of shapes belonging to $\mathcal{L}^{\text{non-inv}}$. Let $K^{\langle S \rangle}$ be the knowledge base constructed through $\tau_{\text{shapes}}(S)$. Then it holds that*

$$s <:_S s' \Leftrightarrow K^{\langle S \rangle} \models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$$

Proof. If there is an RDF data graph and an assignment that acts as a counterexample for s being contained in s' , then it can be translated into a finite model that shows that $K^{\langle S \rangle} \not\models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$ (c.f. Theorem 3). On the other hand, there may be a model that acts as a counterexample showing

that $K^{\langle S \rangle} \not\models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$. Since \mathcal{ALCOQ} has the finite model property (c.f. Proposition 2), there must be a finite model that can be used as a counterexample. Therefore, a model exists that can be translated into an RDF data graph and an assignment such that s is not subsumed by s' . \square

In summary, using standard entailment our approach is sound and complete for the fragment of SHACL not using path concatenation or inverse path expressions. If inverse path expressions are used, then the approach is still sound although completeness is lost. Once arbitrary path concatenation is added, the resulting DL becomes semi-decidable. While an answer is not guaranteed in finite time, shape containment is still sound.

5 Related Work

Several constraint-based schema languages for RDF have been proposed before SHACL. Among those are [3,13]. To the best of our knowledge, containment has not been investigated for those languages. Additionally, SPIN⁶ proposed the usage of SPARQL queries as constraints. When queries are used to express constraints, the containment problem for constraints is equivalent to query containment. ShEx [7] is a constraint language for RDF that is inspired by XML schema languages. While SHACL and ShEx are similar approaches, the semantics of the latter is rooted in regular bag expressions. Validation of an RDF graph with ShEx therefore constructs a single assignment whereas the SHACL semantics used in this paper deals with multiple possible assignments. The containment problem of ShEx shapes has been investigated in [22]. Due to the specific definition of recursion in ShEx, any graph that conforms to the ShEx shapes will also conform to an equivalent SHACL definition. However, not all graphs that conform to SHACL shapes conform to equivalent ShEx shapes. It may be that a shape is contained in another shape in ShEx, but not in SHACL as there is a graph that can act as a counter-example for SHACL that does not conform to the ShEx shapes.

Similar to dedicated constraint languages, there have been proposals for the extension of description logics with constraints. While standard description logics adopts an open-world assumption not suited for data validation, extensions include special constraint axioms [23,19], epistemic operators [12], and closed predicates [20]. Constraints constitute T-Box axioms in these approaches, making constraint subsumption a routine problem.

Lastly, containment problems have been investigated for queries [16,9]. The query containment problem is slightly different as result sets of queries are typically sets of tuples whereas in SHACL we deal with conformance relative to faithful assignments. Given an RDF graph and a set of shapes there may be several, different faithful assignments. Operators available for SHACL are more expressive than operators found in query languages for which subsumption has been investigated. In particular, recursion is not part of most query languages.

⁶ <http://spinrdf.org/>

There is a non-recursive subset of SHACL that is known to be expressible as SPARQL queries [10]. When constraints are expressed as queries, containment of SHACL shapes becomes equivalent to query containment. Recursive fragments of SHACL, however, cannot be expressed as SPARQL queries.

6 Summary

In this paper, we have presented an approach for deciding SHACL shape containment by translating the problem into a description logic subsumption problem. Our translation allows for using efficient and well-known DL reasoning implementations when deciding shape containment. Thus, shape containment can be used, for example, in query optimization.

We defined a syntactic translation of a set of shapes into a description logic knowledge base. We then showed that finite models of this knowledge base and faithful assignments of RDF graphs can be mapped onto each other. Using finite model reasoning, this provides a sound and complete decision procedure for deciding SHACL shape containment, although the decidability of finite satisfiability in $\mathcal{ALCOIQ}(\circ)$ is still an open issue. As part of future work, we plan to adapt the proof used by [15], which comprises of a translation of \mathcal{SROIQ} into a fragment of first-order logic for which finite satisfiability is known. To ensure practical applicability, we also investigated the soundness and completeness of our approach using standard entailment. Our findings are summarized in Fig. 12.

SHACL Fragment	DL	Sound	Complete	Terminates
\mathcal{L}	$\mathcal{ALCOIQ}(\circ)$	Yes	No	Not guaranteed
$\mathcal{L}^{\text{restr}}$	\mathcal{SROIQ}	Yes	No	Yes
$\mathcal{L}^{\text{non-inv}}$	\mathcal{ALCOQ}	Yes	Yes	Yes

Fig. 12: Soundness and completeness for deciding shape containment through description logics reasoning using standard entailment.

Our approach is sound and complete for the SHACL fragment $\mathcal{L}^{\text{non-inv}}$ that uses neither path concatenation nor inverse roles, as the finite model property holds for the corresponding description logic \mathcal{ALCOQ} . Thus, finite entailment and standard entailment are the same for this description logic. The finite model property is lost as soon as inverse roles are added. Using standard entailment, our procedure is still sound for the fragment $\mathcal{L}^{\text{restr}}$ which translates into \mathcal{SROIQ} knowledge bases, but is incomplete due to the possibility of a knowledge base having only infinitely large models. Lastly, the SHACL fragment \mathcal{L} translates into $\mathcal{ALCOIQ}(\circ)$ knowledge bases. Our approach is sound, but incomplete. However, due to the semi-decidability of the description logic, it may be that the decision procedure does not terminate.

Acknowledgements. The authors gratefully acknowledge the financial support of project LISeQ (LA 2672/1-1) by the German Research Foundation (DFG).

References

1. Abbas, A., Genevès, P., Roisin, C., Layaïda, N.: Optimising SPARQL Query Evaluation in the Presence of ShEx Constraints. In: BDA - conférence sur la “Gestion de Données - Principes, Technologies et Applications”. pp. 1–12 (Nov 2017)
2. Abbas, A., Genevès, P., Roisin, C., Layaïda, N.: SPARQL Query Containment with ShEx Constraints. In: Proc. ADBIS. pp. 343–356. LNCS, Springer (2017)
3. Akhtar, W., Cortés-Calabuig, A., Paredaens, J.: Constraints in RDF. In: Proc. Semantics in Data and Knowledge Bases. p. 23–39. Springer (2010)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
5. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017)
6. Beneventano, D., Bergamaschi, S., Sartori, C.: Semantic Query Optimization by Subsumption in OODB. In: Proc. Flexible Query-Answering Systems (FQAS). pp. 167–187. Roskilde University (1996)
7. Boneva, I., Gayo, J., Prud’hommeaux, E.G.: Semantics and Validation of Shapes Schemas for RDF. In: Proc. ISWC. pp. 104–120. LNCS, Springer (2017)
8. Calvanese, D.: Finite Model Reasoning in Description Logics. In: Proc. KR. pp. 292–303. Morgan Kaufmann (1996)
9. Chaudhuri, S., Vardi, M.: Optimization of Real Conjunctive Queries. In: Proc. PODS. p. 59–70. ACM (1993)
10. Corman, J., Florenzano, F., Reutter, J., Savkovic, O.: Validating Shacl Constraints over a Sparql Endpoint. In: Proc. ISWC. pp. 145–163. LNCS, Springer (2019)
11. Corman, J., Reutter, J.L., Savkovic, O.: Semantics and Validation of Recursive SHACL. In: Proc. ISWC. pp. 318–336. LNCS, Springer (2018)
12. Donini, F.M., Nardi, D., Rosati, R.: Description Logics of Minimal Knowledge and Negation As Failure. ACM TOCL **3**(2), 177–225 (Apr 2002)
13. Fischer, P.M., Lausen, G., Schätzle, A., Schmidt, M.: RDF Constraint Checking. In: Proc. EDBT/ICDT. pp. 205–212. CEUR-WS.org (2015)
14. Grandi, F.: On expressive Description Logics with composition of roles in number restrictions. In: Proc. LPAR. pp. 202–215. LNCS, Springer (2002)
15. Kazakov, Y.: RIQ and SROIQ Are Harder than SHOIQ. In: Proc. KR. pp. 274–284. AAAI Press (2008)
16. Klug, A.: On Conjunctive Queries Containing Inequalities. J. ACM **35**(1), 146–160 (1988)
17. Leinberger, M., Seifer, P., Schon, C., Lämmel, R., Staab, S.: Type Checking Program Code Using SHACL. In: Proc. ISWC. pp. 399–417. LNCS, Springer (2019)
18. Lutz, C., Areces, C., Horrocks, I., Sattler, U.: Keys, nominals, and concrete domains. Journal of Artificial Intelligence Research **23**, 667–726 (2004)
19. Motik, B., Horrocks, I., Sattler, U.: Adding Integrity Constraints to OWL. In: Proc. OWLED. CEUR Workshop Proceedings, vol. 258. CEUR-WS.org (2007)
20. Patel-Schneider, P.F., Franconi, E.: Ontology Constraints in Incomplete and Complete Data. In: Proc. ISWC. pp. 444–459. LNCS, Springer (2012)
21. Rudolph, S.: Foundations of Description Logics, pp. 76–136. Springer (2011)

22. Staworko, S., Wiecezorek, P.: Containment of Shape Expression Schemas for RDF. In: Proc. PODS. pp. 303–319. ACM (2019)
23. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity Constraints in OWL. In: Proc. AAAI. AAAI Press (2010)