Deciding SHACL Shape Containment Through Description Logics Reasoning

Martin Leinberger[1(✉)], Philipp Seifer[2], Tjitze Rienstra[1], Ralf Lämmel[2], and Steffen Staab[3,4]

[1] Institute for Web Science and Technologies, University of Koblenz-Landau, Koblenz, Germany
{mleinberger,rienstra}@uni-koblenz.de

[2] The Software Languages Team, University of Koblenz-Landau, Koblenz, Germany
{pseifer,laemmel}@uni-koblenz.de

[3] Institute for Parallel and Distributed Systems, University of Stuttgart, Stuttgart, Germany
steffen.staab@ipvs.uni-stuttgart.de

[4] Web and Internet Science Research Group, University of Southampton, 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

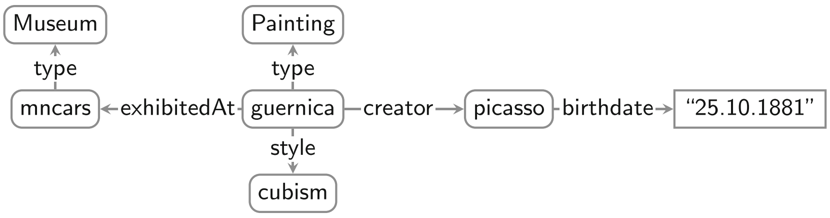
RDF has been designed as a flexible, semi-structured data format. To ensure data quality and to allow for restricting its large flexibility in specific domains, the W3C has standardized the *Shapes Constraint Language (SHACL)*[1]. A set of SHACL shapes are represented in a *shape graph*. A shape graph 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 $\boxed{\text{Painting}}$. It requires the presence of at least one $\xrightarrow{\text{exhibitedAt}}$ property (line 3) as well as that each node reachable via the $\xrightarrow{\text{creator}}$ property from a $\boxed{\text{Painting}}$ conforms to the `PainterShape` (line 4). The

[1] 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).

`PainterShape` (lines 5–8) requires all incoming $\xrightarrow{\text{creator}}$ properties to conform to `PaintingShape` (line 6) as well as the presence of exactly one $\xrightarrow{\text{birthdate}}$ property. Lastly, the shapes define a `CubistShape` (lines 9–11) which must have an incoming $\xrightarrow{\text{creator}}$ property from a node that has an outgoing $\xrightarrow{\text{style}}$ property to the node $\boxed{\text{cubism}}$. 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 [1,5] or program verification [16]. 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 $\boxed{?X}$ and $\boxed{?Y}$ such that $\boxed{?X} \xrightarrow{\text{style}} \boxed{\text{cubism}}$,, $\boxed{?X} \xrightarrow{\text{creator}} \boxed{?Y}$ and $\boxed{?X} \xrightarrow{\text{exhibitedAt}} \boxed{?Z}$ can be optimized. Since nodes that are results for $\boxed{?Y}$ must conform to `CubistShape` and `CubistShape` is contained in `PainterShape`, nodes that are results for $\boxed{?X}$ must conform to `PaintingShape`.

Subsequently, the pattern ?x $\xrightarrow{\text{exhibitedAt}}$?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 conforms to s' but not to s. A similar problem is concept subsumption in description logics (DL). For DL, efficient tableau-based approaches [4] 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 [10] and the Tarski-style semantics used by description logics, we leverage concept subsumption 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 [10] 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 Sect. 2. We describe how sets of SHACL shapes are translated into a DL knowledge base in Sect. 3. Section 4 investigates how to use standard DL entailment for deciding shape containment. Finally, we discuss related work in Sect. 5 and summarize our results. An extended version of this paper including full proofs and additional explanations is available on Arxiv[2].

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

[2] https://arxiv.org/abs/2008.13603.

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 [10].

Data Graphs. We assume familiarity with RDF. We abstract away from concrete RDF syntax, 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. 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:

$$\phi ::= \top \mid s \mid v \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \geqslant_n \rho.\phi \tag{1}$$

$$\rho ::= p \mid \hat{}\rho \mid \rho_1/\rho_2 \tag{2}$$

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 $\geqslant_n \rho.\phi$ 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 $\neg(\neg\phi_1 \wedge \neg\phi_2)$, $\leqslant_n \rho.\phi$ for $\neg(\geqslant_{n+1} \rho.\phi)$,$=_n \rho.\phi$ for $(\leqslant_n \rho.\phi) \wedge (\geqslant_n \rho.\phi)$, and $\forall \rho.\phi$ for $\leq_0 \rho.\neg\phi$. As an example, the constraint of `CubistShape` (see Fig. 1) can be expressed as $\geqslant_1 (\hat{}\texttt{creator/style}).\texttt{cubism}$.

Evaluation of constraints is rather straightforward with the exception of reference cycles. To highlight this issue, consider a shape name `Local` with its constraint $\forall$ `knows.Local`. In order to fulfill the constraint, any graph node reachable through $\xrightarrow{\text{knows}}$ must conform to `Local`. Consider a graph with a single vertex $\boxed{b_1}$ whose $\xrightarrow{\text{knows}}$ property points to itself Intuitively, there are two possible solutions. If $\boxed{b_1}$ is assumed to conform to `Local`, then the constraint is satisfied and it is correct to say that $\boxed{b_1}$ conforms to `Local`. If $\boxed{b_1}$ is assumed to not conform to `Local`, then the constraint is violated and it is correct to say that $\boxed{b_1}$ does not conform to `Local`. We follow the proposal of [10] 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 `Local` $\in \sigma(\boxed{b_1})$ and a different one where `Local` $\notin \sigma(\boxed{b_1})$. 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, [10] also defines partial assignments, lifting this restriction. Due to the lack of space, we refer to [10] 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* $\mathrm{Names}(S)$*. An assignment σ is a total function $\sigma : V_G \to 2^{\mathrm{Names}(S)}$ 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 $[\![\phi]\!]^{v,G,\sigma}$, is defined as shown in Fig. 2.

$$[\![\top]\!]^{v,G,\sigma} = \text{true}$$

$$[\![\neg\phi]\!]^{v,G,\sigma} = \begin{cases} \text{true if} \\ \quad [\![\phi]\!]^{v,G,\sigma} = \text{false} \\ \text{false otherwise} \end{cases}$$

$$[\![\phi_1 \wedge \phi_2]\!]^{v,G,\sigma} = \begin{cases} \text{true if } [\![\phi_1]\!]^{v,G,\sigma} = \text{true} \wedge \\ \quad [\![\phi_2]\!]^{v,G,\sigma} = \text{true} \\ \text{false otherwise} \end{cases}$$

$$[\![v']\!]^{v,G,\sigma} = \begin{cases} \text{true if } v = v' \\ \text{false otherwise} \end{cases}$$

$$[\![\geqslant_n \rho.\phi]\!]^{v,G,\sigma} = \begin{cases} \text{true if } |\{v_2 \mid (v_1, v_2) \in [\![r]\!]^G \wedge \\ \quad [\![\phi]\!]^{v_2,G,\sigma} = \text{true}\}| \geq n \\ \text{false otherwise} \end{cases}$$

$$[\![s]\!]^{v,G,\sigma} = \begin{cases} \text{true if } s \in \sigma(v) \\ \text{false otherwise} \end{cases}$$

$$\begin{aligned} [\![p]\!]^G &= \{(v_1, v_2) \mid \exists\, p : (v_1, p, v_2) \in E_G\} \\ [\![{}^\wedge\rho]\!]^G &= \{(v_2, v_1) \mid (v_1, v_2) \in [\![\rho]\!]^G\} \\ [\![\rho_1/\rho_2]\!]^G &= \{(v_1, v_2) \mid \exists\, v : (v_1, v) \in [\![\rho_1]\!]^G \wedge (v, v_2) \in [\![\rho_2]\!]^G\} \end{aligned}$$

Fig. 2. 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:

$$q ::= \bot \mid \{v_1, \ldots, v_n\} \mid \texttt{class}\ v \mid \texttt{subjectsOf}\ p \mid \texttt{objectsOf}\ p \qquad (3)$$

where $\bot$ represents a query that targets no nodes, $\{v_1 \ldots v_n\}$ targets all explicitly listed nodes with $v_1, \ldots, v_n \in \mathcal{V}$, `class` v targets all instances of the class represented by v where $v \in \mathcal{V}_C$, `subjectsOf` p targets all subjects of the property p and `objectsOf` p targets all objects of p. We use $\mathcal{Q}$ to refer to the set of all possible queries and $[\![q]\!]_G$ to denote the set of nodes in the RDF graph G targeted by the query q (c.f. Fig. 3).

$$[\![\bot]\!]_G = \emptyset \qquad [\![\{v_1, \ldots, v_n\}]\!]_G = \{v_1, \ldots, v_n\}$$

$$\begin{aligned} [\![\texttt{class } v_2]\!]_G &= \{v_1 \mid (v_1, \texttt{type}, v_2) \in E_G\} \\ [\![\texttt{subjectsOf } p]\!]_G &= \{v_1 \mid \exists v_2 : (v_1, p, v_2) \in E_G\} \\ [\![\texttt{objectsOf } p]\!]_G &= \{v_2 \mid \exists v_1 : (v_1, p, v_2) \in E_G\} \end{aligned}$$

Fig. 3. Evaluation of target node queries.

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 [10], we refer to the language represented by the definitions above as $\mathcal{L}$. As an example, Fig. 4 shows the shape graph defined in Fig. 1a as a set of shapes.

$$\begin{aligned} S_1 = \{ & \\ & (\texttt{PaintingShape}, \geqslant_1 \texttt{exhibitedAt}.\top \wedge \forall\, \texttt{creator}.\texttt{PainterShape}, \texttt{class Painting}), \\ & (\texttt{PainterShape}, =_1 \texttt{birthdate}.\top \wedge \forall\, {}^{\wedge}\texttt{creator}.\texttt{PaintingShape}, \bot), \\ & (\texttt{CubistShape}, \geqslant_1 {}^{\wedge}\texttt{creator/style}.\texttt{cubism}, \bot) \\ \} & \end{aligned}$$

Fig. 4. Representation of the shape graph shown 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:*

- $s \in \sigma(v) \Leftrightarrow [\![\phi]\!]^{v,G,\sigma}$.
- $v \in [\![q]\!]_G \Rightarrow s \in \sigma(v)$.

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* 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 $\boxed{\text{guernica}}$ and both `PainterShape` and `CubistShape` to

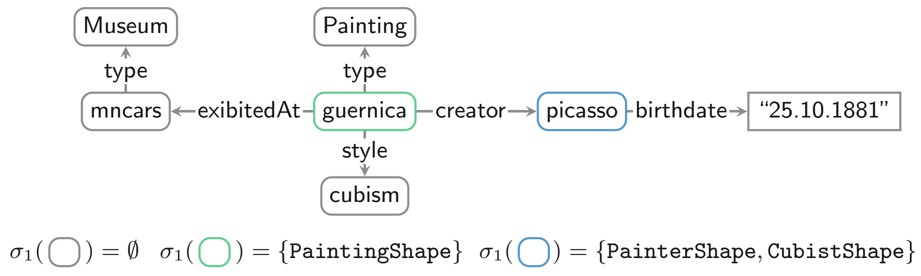


Fig. 5. Faithful assignment σ_1 for S_1 and the data graph shown in Fig. 1b.

picasso (see Fig. 5). 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

We focus on the highly-expressive DL $\mathcal{ALCOIQ}(\circ)$ as well as decidable subsets of this logic. We follow routine syntax and interpretation-based semantics (c. f. [3,4,13]). $\mathrm{Sig}(K) = (N_A, N_P, N_O)$ is the signature of a knowledge base K comprising of a set of atomic concept names N_A that is a subset of the set of all possible atomic concept names $\mathcal{N}_A$, a set of atomic property names N_P (a subset of $\mathcal{N}_P$) and a set of object names N_O (a subset of $\mathcal{N}_O$). From these, more complex role expressions, denoted by r, and concept expressions, denoted by C and D, are built (see Fig. 6) whereby $\mathcal{C}$ denotes the set of all possible concept expressions and $\mathcal{R}$ the set of all possible role expressions.

Constructor Name	**Syntax**	**Semantics**
atomic property name	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	$r_1 \circ r_2$	$\{(o_1, o_2) \mid (o_1, o) \in {r_1}^I \land (o, o_2) \in {r_2}^I\}$
atomic concept name	A	$A^I \subseteq \Delta^I$
nominal concept	$\{o_1, \ldots o_n\}$	$\{o_1^I, \ldots, o_n^I\}$
top	$\top$	Δ^I
negation	$\neg C$	$\Delta^I \setminus C^I$
conjunction	$C \sqcap D$	$C^I \cap D^I$
qualified number restriction	$\geq_n r.C$	$\{o_1 \mid \lvert\{o_1 \mid (o_1, o_2) \in r^I \land o_2 \in C^I\}\rvert \geq n\}$

Fig. 6. Syntax and semantics of roles r (above the line) and concept expressions C, D (below the line).

Axioms are either concept inclusions, concept assertions or role assertions (see Fig. 7). We use $C \equiv D$ as a shorthand for the two axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. In a given interpretation $I = (\Delta, \cdot^I)$ comprised of a universe Δ and an interpretation function $\cdot^I$, an axiom ψ is either true or false. An interpretation in which all

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. 7. Syntax and semantics of axioms.

axioms of K are true is a model of K. We use $\mathrm{Mod}(K)$ to denote the set of all models of K. An axiom ψ is entailed by K written $K \models \psi$ if it is true in all models of K. Lastly, we use $\mathcal{K}$ for the set of all possible knowledge bases.

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

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

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 [10]. 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 $K^{<S>}$ using four auxiliary functions (see Fig. 8): 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 $K^{<S>} \models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$.

To prove this property, we show that every finite model of $K^{<S>}$ 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 $K^{<S>}$ can be constructed from an assignment that is faithful with respect to S and any given RDF graph G.

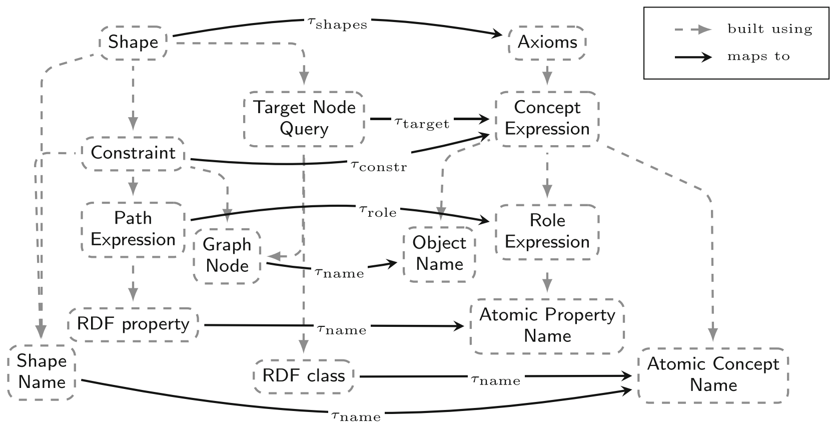


Fig. 8. Syntactic translation of SHACL to description logics.

3.1 Syntactic Mapping

We map the set of shapes S into a knowledge base $K^{<S>}$ 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 5 (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 6 (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 7 (Mapping constraints to DL concept expressions). *The* constraint mapping τ_{constr} *: $\Phi \rightarrow \mathcal{C}$ is defined as follows:*

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

Definition 8 (Mapping target node queries to DL concept expressions). *The* target node mapping $\tau_{\text{target}} : \mathcal{Q} \rightarrow \mathcal{C}$ *is defined as follows:*

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

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 9 (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 $\tau_{\text{shapes}}(S_1) = K^{<S_1>}$ is shown in Fig. 9.

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

Fig. 9. Translation $\tau_{\text{shapes}}(S_1) = K^{<S_1>}$ 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 10 (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 [7]. 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 $I^{G,\sigma}$ that is a finite model for the knowledge base $K^{<S>}$.

Definition 11 (Construction of the finite model $I^{G,\sigma}$). *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 $I^{G,\sigma}$ for the knowledge base $\tau_{\text{shapes}}(S) = K^{<S>}$ 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:* $\forall p \in N_P : \forall o_1, o_2 \in N_O : (o_1^{I^{G,\sigma}}, o_2^{I^{G,\sigma}}) \in p^{I^{G,\sigma}}$ *if* $(\tau_{\text{node}}(o_1), p, \tau_{\text{node}}(o_2)) \in (E_G \setminus \{(v_1, \texttt{type}, v_2) \in E_G\})$.
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:* $\forall A_v \in N_A : \forall o \in N_O : o^{I^{G,\sigma}} \in A_v^{I^{G,\sigma}}$ *if* $(\tau_{\text{node}}(o), \texttt{type}, \tau_{\text{node}}(A_v)) \in E_G$.
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:* $\forall A_s \in N_A : \forall o \in N_O : o^{I^{G,\sigma}} \in A_s^{I^{G,\sigma}}$ *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 $\forall (o_1, o_2) \in \tau_{\text{role}}(\rho)^{I^{G,\sigma}} \Rightarrow (\tau_{\text{node}}(o_1), \tau_{\text{node}}(o_2)) \in [\![\rho]\!]^G$ 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. 2) and semantics of role expressions (c. f. Fig. 6). □

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 (Sketch). $I^{G,\sigma}$ is finite because the RDF graph G has only a finite number of graph nodes. Furthermore, $I^{G,\sigma}$ satisfying the axioms created by τ_{shapes} can be shown via induction over the mapping rules for τ_{constr} and τ_{target}. □

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 σ^I such that σ^I is faithful with respect to S and G^I. We construct G^I and σ^I in the following manner:

Definition 12 (Construction of G^I and σ^I). *Let S be a set of shapes and $K^{<S>}$ a knowledge base constructed via $\tau_{\text{shapes}}(S)$. Furthermore, let $I \in \text{Mod}^{\text{fin}}(K^{<S>})$ be a finite model of $K^{<S>}$. The RDF graph $G^I = (V_G^I, E_G^I)$ and the assignment σ^I can then be constructed as follows:*

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

An assignment σ^I constructed in this manner is faithful with respect to the constructed RDF graph G^I and the set of shapes S.

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

Proof (Sketch). The two axioms that are generated by τ_{shapes} coincide with the two conditions for faithful assignments (c. f. Definitions 2 and 9). This can be shown by induction over the translation rules. □

3.3 Deciding Shape Containment Using Concept Subsumption

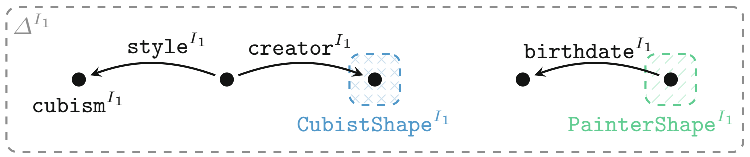
Given the translation rules and semantic equivalence between finite models of a description logic knowledge base and assignments for SHACL shapes, we can leverage description logics for deciding shape containment. Assume a set of shapes S containing definitions for two shapes s and s'. Those shapes are 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 $K^{<S>}$ using finite models.

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

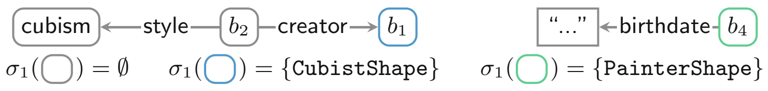
$$s <:_S s' \Leftrightarrow K^{<S>} \models_{\text{fin}} \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$$

Proof (Sketch). Using Theorems 1 and 2, any counterexample for one side can always be translated to a counterexample for the other side. □

As an example, reconsider the translation of the set of shapes $\tau_{\text{shapes}}(S_1) = K^{<S_1>}$ (see Fig. 9). From $K^{<S_1>}$ follows that $K^{<S_1>} \not\models \texttt{CubistShape} \sqsubseteq \texttt{PainterShape}$



(a) Model of $K^{<S_1>}$ showing that `CubistShape` $\not\sqsubseteq$ `PainterShape`.


cubism
style
b_2
creator
b_1
"..."
birthdate
b_4
$\sigma_1(\ ) = \emptyset$
$\sigma_1(\ ) = \{$CubistShape$\}$
$\sigma_1(\ ) = \{$PainterShape$\}$

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

Fig. 10. Counterexamples for `CubistShape` $<:_{S_1}$ `PainterShape`.

as there is a finite model $I_1 \in \mathrm{Mod}^{\mathrm{fin}}(K^{<S_1>})$ in which the concept expression `CubistShape` $\sqcap \neg$`PainterShape` is satisfiable (see Fig. 10).

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 (Sketch). Given constraints ϕ and ϕ', it is possible to introduce unique shape names s_C and s_D as well as an RDF class v_C. Constraint ϕ is then extended with v_C, allowing shape s_D to target v_C. □

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:

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

Finite satisfiability is known to be decidable for $\mathcal{SROIQ}$ [14] 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 [13]. 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 13. (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 [4].*

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* $\mathcal{ALCOIQ}$ *[7] 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.*

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 $K^{<S>} \models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$. 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 $K^{<S>} \not\models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$, 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 $K^{<S>} \not\models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$ 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, if the decision procedure terminates.

Theorem 4. *Let* S *be a set of shapes of the language* $\mathcal{L}^{restr}$*. 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 corrseponding DL is $\mathcal{SROIQ}$ for which the finite model property does not hold. If $K^{<S>} \models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$, 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. □

However, the approach is incomplete as it may be that $s <:_S s'$ but $K^{<S>} \not\models \tau_{\text{name}}(s) \sqsubseteq \tau_{\text{name}}(s')$ 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:

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

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 [17].*

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 $\mathcal{L}^{non\text{-}inv}$. Let $K^{<S>}$ be the knowledge base constructed through $\tau_{\text{shapes}}(S)$. Then it holds that*

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

Proof (Sketch). Due to $\mathcal{ALCOQ}$ having the finite model property, it is always possible to construct counterexamples for either side (c. f. Theorem 3). □

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

SHACL containment has also been studied by [19], whereas this work studies theoretical shape satisfiability (and thus containment) by defining an equisatisfiable FOL language. In contrast, our approach focuses on practical applicability by leveraging standard entailment of description logic reasoners. Before SHACL, several constraint-based schema languages for RDF have been proposed before SHACL. Among those are [2,12]. To the best of our knowledge, containment has not been investigated for those languages. Additionally, SPIN[3] proposed the usage of SPARQL queries as constraints. Then, the containment problem for constraints is equivalent to query containment. ShEx [6] 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 papers 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.

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 [18,23], epistemic operators [11], 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 [8,15]. 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. There is a non-recursive subset of SHACL that is known to be expressible as SPARQL queries [9]. 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 [14], which comprises of a translation of $\mathcal{SROIQ}$ into a fragment of first-order logic for which finite

[3] http://spinrdf.org/.

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. 11. 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.

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. 11. Soundness and completeness for deciding shape containment through description logics reasoning using standard entailment.

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.: SPARQL query containment with ShEx constraints. In: Kirikova, M., Nørvåg, K., Papadopoulos, G.A. (eds.) ADBIS 2017. LNCS, vol. 10509, pp. 343–356. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66917-5_23
2. Akhtar, W., Cortés-Calabuig, Á., Paredaens, J.: Constraints in RDF. In: Schewe, K.-D., Thalheim, B. (eds.) SDKB 2010. LNCS, vol. 6834, pp. 23–39. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23441-5_2
3. 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, Cambridge (2003)
4. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press, New York (2017)
5. Beneventano, D., Bergamaschi, S., Sartori, C.: Semantic query optimization by subsumption in OODB. In: Proceedings of the Flexible Query-Answering Systems (FQAS), pp. 167–187. Roskilde University (1996)
6. Boneva, I., Labra Gayo, J.E., Prud'hommeaux, E.G.: Semantics and validation of shapes schemas for RDF. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10587, pp. 104–120. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68288-4_7

7. Calvanese, D.: Finite model reasoning in description logics. In: Proceedings of the KR, pp. 292–303. Morgan Kaufmann (1996)
8. Chaudhuri, S., Vardi, M.: Optimization of real conjunctive queries. In: Proceedings of the PODS, pp. 59–70. ACM (1993)
9. Corman, J., Florenzano, F., Reutter, J.L., Savković, O.: Validating Shacl constraints over a Sparql endpoint. In: Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I., Hogan, A., Song, J., Lefrançois, M., Gandon, F. (eds.) ISWC 2019. LNCS, vol. 11778, pp. 145–163. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30793-6_9
10. Corman, J., Reutter, J.L., Savković, O.: Semantics and validation of recursive SHACL. In: Vrandečić, D., Bontcheva, K., Suárez-Figueroa, M.C., Presutti, V., Celino, I., Sabou, M., Kaffee, L.-A., Simperl, E. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 318–336. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_19
11. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM TOCL **3**(2), 177–225 (2002)
12. Fischer, P.M., Lausen, G., Schätzle, A., Schmidt, M.: RDF constraint checking. In: Proceedings of the EDBT/ICDT, pp. 205–212. CEUR-WS.org (2015)
13. Grandi, F.: On expressive description logics with composition of roles in number restrictions. In: Baaz, M., Voronkov, A. (eds.) LPAR 2002. LNCS (LNAI), vol. 2514, pp. 202–215. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36078-6_14
14. Kazakov, Y.: RIQ and SROIQ are harder than SHOIQ. In: Proceedings of the KR, pp. 274–284. AAAI Press (2008)
15. Klug, A.: On conjunctive queries containing inequalities. J. ACM **35**, 146–160 (1988)
16. Leinberger, M., Seifer, P., Schon, C., Lämmel, R., Staab, S.: Type checking program code using SHACL. In: Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I., Hogan, A., Song, J., Lefrançois, M., Gandon, F. (eds.) ISWC 2019. LNCS, vol. 11778, pp. 399–417. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30793-6_23
17. Lutz, C., Areces, C., Horrocks, I., Sattler, U.: Keys, nominals, and concrete domains. J. Artif. Intell. Res. **23**, 667–726 (2004)
18. Motik, B., Horrocks, I., Sattler, U.: Adding integrity constraints to OWL. In: Proceedings of the OWLED. CEUR Workshop Proceedings, vol. 258. CEUR-WS.org (2007)
19. Pareti, P., Konstantinidis, G., Magavero, F., Norman, T.J.: SHACL satisfiability and containment. In: Proceedings of the ISWC. LNCS. Springer (2020)
20. Patel-Schneider, P.F., Franconi, E.: Ontology constraints in incomplete and complete data. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 444–459. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35176-1_28
21. Rudolph, S.: Foundations of description logics. In: Polleres, A., et al. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 76–136. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23032-5_2
22. Staworko, S., Wieczorek, P.: Containment of shape expression schemas for RDF. In: Proceedings of the PODS, pp. 303–319. ACM (2019)
23. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: Proceedings of the AAAI. AAAI Press (2010)