A Short Introduction to SHACL for Logicians

Magdalena Ortiz[0000−0002−2344−9658]

Umeå University
magdalena.ortiz@cs.umu.se

Abstract. The Shapes Constraint Language (SHACL) was recommended by the W3C in 2017 for describing constraints on web data (specifically, on the so-called RDF graphs) and validating them. At first glance, it may not seem to be a topic for logicians, but as it turns out, SHACL can be approached as a formal logic, and actually quite an interesting one. In this paper, we give a brief introduction to SHACL tailored towards logicians and frame key uses of SHACL as familiar logic reasoning tasks. We discuss how SHACL relates to description logics, which are the basis of the OWL Web Ontology Languages, a related yet orthogonal standard for web data. Finally, we summarize some of our recent work in the SHACL world, hoping that this may shed light on how ideas, results, and techniques from well-established areas of logic can advance the state of the art in this emerging field.

Keywords: SHACL · semantic web · description logics

1 What is SHACL and why do we need it?

One of the most fundamental changes the world has seen in the last decades is the emergence and ultrafast growth of the Web, where almost inconceivable amounts of data of all shapes and forms is shared and interconnected. The World Wide Web Consortium (W3C) has been a key player in this growth: an international community that develops open standards that are used for building the web from documents and data. A particularly influential set of standards are those developed within the *semantic web* initiative, which aims to build a useful *web of data* that is interoperable and understandable to both humans and machines.

The big bulk of shared data on the web uses the RDF standard [28]. In a nutshell, labelled graphs where nodes represent web resources and data values, connected by arrows labelled with different kinds of properties with a standardized meaning. RDF datasets are often described as sets of triples (s, p, o) where the *subject* s and the object o are data items of resources with a unique identifier, and they are connected by property p. For our purposes, it is enough to think of graphs whose edges are labeled with *properties* from a dedicated alphabet.[1] There are enormous repositories of such data on the web containing millions

[1] In RDF, properties are not necessarily disjoint from the nodes, which can be either web identifiers called IRIs, data values given as *literals*, and the so-called *blank nodes*; we omit RDF details from here and refer to [28].

of nodes and edges; famous examples include *knowledge graphs* like DBPedia[2] which contains several hundred million of facts extracted from Wikipedia, and Yago, a high-quality knowledge base about people, cities, countries, and organizations, containing more than 2 billion facts about 50 million entities.[3] Once RDF became widespread for sharing data on the web, accompanying standards were proposed, like the OWL Web Ontology Languages for describing knowledge domains and for inferring implicit relationships and facts from data on the web [26], and a dedicated query language for RDF data called SPARQL [27]. Web interfaces called *SPARQL end-points* allow any person to ask questions and obtain interesting facts from these knowledge graphs.

A crucial feature of the RDF data format is its *flexibility*: if something can be represented as a labelled graph, then it can be published on the web using RDF. But so much flexibility is a two-edged sword. Users that want to query a source like DBpedia easily find themselves lost and do not know where to start: how do I formulate my query? Which properties may connect a country to its capital city? Is there information about the family relationships of celebrities? Which facts about rivers could I query for?

As the web of linked data kept growing, the pressing need for a *normative* standard emerged: a language that can be used for describing and validating the structure and content of RDF graphs. This language is SHACL, the Shapes Constraint Language, recommended in 2017 by the W3C [29]. It allows users to define "shapes" which may, for example, say that a person has a name and exactly one date of birth, and may be married to another person. Like other W3C standards, SHACL is defined in a long 'specification' document that is very hard to read for anybody that is not familiar with the semantic web jargon. While OWL was standardized on the basis of over two decades of research in *description logics*, a well-understood family of decidable logics [6, 26], the younger SHACL did not come to the world equipped with such a robust logic foundation. However, a handful of logic-minded people from the knowledge representation and database theory community have been developing a solid logic-based foundation for SHACL. As it turns out, SHACL can be seen as a simple and elegant logic, and its fundamental *validation* problem is a model checking task very familiar in logic and computer science.

2 SHACL as a Logic

The challenge of extracting from the specification a formal syntax and semantics was tackled by Corman et al. [13], and the majority of the later SHACL works have built on their formalization, e.g. [4, 1, 3, 11, 10]. We do the same, and like most of them, we focus on the 'core' of SHACL. Some authors have extended this formalization to cover more SHACL features; see, for example, the extended formalization compiled by Jakubowski [16].

[2] `http://dbpedia.org/`
[3] `https://yago-knowledge.org/`

2.1 Syntax

The main syntactic object in SHACL are the so-called *shape constraints*, which assign possibly complex *shape expressions* to special predicates called *shape names*.

For writing these, we use an alphabet consisting of three countably infinite pairwise disjoint sets: the set of *shape names* $\mathbf{S}$, the set of *property names* $\mathbf{P}$, and the set of *node names* $\mathbf{N}$. Then *shape expressions* φ and *path expressions* ρ obey the following grammar.

$$\rho ::= p \mid p^- \mid \rho \cup \rho \mid \rho \cdot \rho \mid \rho^*$$
$$\varphi ::= s \mid \top \mid \{a\} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \geq_n \rho.\varphi \mid \rho = \rho$$

where $s \in \mathbf{S}$, $a \in \mathbf{N}$, $p \in \mathbf{P}$, and $n \geq 0$ is a natural number. We may write $\leq_{n-1} \rho.\varphi$ in place of $\neg(\geq_n \rho.\varphi)$, and write $\exists\rho.\varphi$ and $\forall\rho.\varphi$ in place of $\geq_1 \rho.\varphi$ and $\leq_0 \rho.\neg\varphi$, respectively. Those familiar with modal logic will recognise the syntax of multidimensional modal formulas extended with *nominals* $\{a\}$ from hybrid logics [5], graded modalities $\geq_n \rho.\varphi$, converse ρ^-, and reflexive, transitive closure as in propositional dynamic logic [14]. We will revisit this relationship from the perspective of description logics in the next section.

We can now write *shape constraints* of the form

$$s \equiv \varphi$$

where $s \in \mathbf{S}$ and φ is a *shape expression*. In SHACL, shape constraints come together with a set of *targets*, indicating at which nodes of the graph the shapes of interest are to be validated. We focus here on atomic targets of the form $s(a)$ with $s \in \mathbf{S}$ and $a \in \mathbf{N}$. Since shape names are unary predicates, we can read $s(a)$ as 'a is in the interpretation of s'. Then we define a *shapes graph* as a pair $(\mathcal{C}, T)$ of a set $\mathcal{C}$ of shape constraints where each $s \equiv \varphi$ has a different s in the head, and a set T of target atoms.

Example 1. Consider the following shape constraints:

$$\begin{aligned} \mathit{Pizza} &\equiv \geq_2 \texttt{hasTopping}.\top, \\ \mathit{VeggiePizza} &\equiv \mathit{Pizza} \wedge \forall \texttt{hasTopping}.\mathit{VeggieTopping}, \\ \mathit{VeggieTopping} &\equiv \{\texttt{mozzarella}\} \vee \{\texttt{tomato}\} \vee \{\texttt{basil}\} \vee \{\texttt{artichoke}\} \end{aligned}$$

Intuitively, these constraints define the shape 'pizza' as having at least two toppings, and vegetarian pizzas as pizzas having only vegetarian toppings. A shape can be defined by directly listing the nodes in it, as done here for vegetarian toppings. To give a rough impression of the way this is written in usual SHACL documents, we show in Figure 1 an incomplete declaration of the first constraint in SHACL machine-readable syntax.

```
a sh:NodeShape ;
sh:property [
    sh:path pizza:hasTopping ;
    sh:minCount 2    ] .
```

Fig. 1. A SHACL shape in machine-readable syntax

2.2 Semantics

We now define the so-called *supported model semantics* for SHACL, and discuss other semantics in Section 4.1.

Like other logic formalisms, the semantics of SHACL can be elegantly defined using interpretations. For our purposes, it will be enough to consider interpretations whose domains are nodes from $\mathbf{N}$. We consider interpretations $\mathcal{I}$ consisting of a non-empty domain $\Delta \subseteq \mathbf{N}$, and an interpretation function $\cdot^{\mathcal{I}}$ that maps

- each shape name $s \in \mathbf{S}$ a set $s^{\mathcal{I}} \subseteq \Delta$, and
- each property name to a set of pairs $s^{\mathcal{I}} \subseteq \Delta \times \Delta$.

The interpretation function $\mathcal{I}$ is inductively extended to complex expressions, see Figure 2. Note that path expressions ρ are interpreted as binary relations $\rho^{\mathcal{I}}$ over Δ, while shape expressions φ are interpreted as sets $\varphi^{\mathcal{I}} \subseteq \Delta$.

$$
\begin{aligned}
(p^{-})^{\mathcal{I}} &= \{(d', d) \mid (d, d') \in p^{\mathcal{I}}\} \\
(\rho \cup \rho')^{\mathcal{I}} &= \rho^{\mathcal{I}} \cup \rho'^{\mathcal{I}} \\
(\rho \cdot \rho')^{\mathcal{I}} &= \{(c, d) \in \Delta \times \Delta \mid \text{there is some } d' \text{ with } (c, d') \in \rho^{\mathcal{I}}, (d', d) \in \rho^{\mathcal{I}}\} \\
(\rho^{*})^{\mathcal{I}} &= \{(d, d) \mid d \in \Delta \times \Delta\} \cup (\rho)^{\mathcal{I}} \cup (\rho \cdot \rho)^{\mathcal{I}} \cup \cdots
\end{aligned}
$$

$$
\begin{aligned}
\{a\}^{\mathcal{I}} &= \{a\} \\
\top^{\mathcal{I}} &= \Delta \\
(\neg\varphi)^{\mathcal{I}} &= \Delta \setminus \varphi^{\mathcal{I}} \\
(\varphi_1 \wedge \varphi_2)^{\mathcal{I}} &= {\varphi_1}^{\mathcal{I}} \cap {\varphi_2}^{\mathcal{I}} \\
(\varphi_1 \vee \varphi_2)^{\mathcal{I}} &= {\varphi_1}^{\mathcal{I}} \cup {\varphi_2}^{\mathcal{I}} \\
(\geq_n \rho.\varphi)^{\mathcal{I}} &= \{d \in \Delta \mid \text{there exist distinct } d_1, \ldots, d_n \\
&\qquad \text{with } (d, d_i) \in \rho^{\mathcal{I}} \text{ and } d_i \in \varphi^{\mathcal{I}} \text{ for each } 1 \leq i \leq n\} \\
(\rho = \rho')^{\mathcal{I}} &= \{d \in \Delta \mid \text{ for all } d' \in \Delta, (d, d') \in \rho^{\mathcal{I}} \text{ iff } (d,) \in \rho'^{\mathcal{I}}\}
\end{aligned}
$$

Fig. 2. Interpretation of path and shape expressions

We say that $\mathcal{I}$ *satisfies a constraint* $s \equiv \varphi$ if $s^{\mathcal{I}} = \varphi^{\mathcal{I}}$, and $\mathcal{I}$ *satisfies a shapes graph* $(\mathcal{C}, T)$ if it satisfies all constraints in $\mathcal{C}$ and additionally it contains all the targets, that is, $a \in s^{\mathcal{I}}$ for every $s(a) \in T$.

SHACL shapes graphs are intended to be validated over an input *data graph* (essentially a knowledge graph or RDF graph). A *data graph* $G = (N, E, \ell)$ is defined as a graph with vertices $N \subseteq \mathbf{N}$ and a labelling function $\ell : E \to 2^{\mathbf{P}}$, that is, edges are labelled with sets of property names from $\mathbf{P}$, and a graph is just a collection of $\mathbf{P}$-indexed binary relations on N.

Given such a graph $G = (N, E, \ell)$, we say that the interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ is a *shape adornment for* G if $\Delta = \mathbf{N}$ and $p^{\mathcal{I}} = \{(a, b) \in E \mid p \in \ell((a, b))\}$ for each property p. That is, $\mathcal{I}$ is a shape adornment of G if properties p are interpreted as relations as specified by G. Note that G does not determine the interpretation of the shape names. In modal logic terms, we can call G a multi-relational Kripke frame, and $\mathcal{I}$ is a multi-relational Kripke model.

In SHACL, the main problem of interest is the *validation* of given constraints and targets on a given graph.

Definition 1 (SHACL validation). *We say that* a data graph G validates a shapes graph $(\mathcal{C}, T)$ *if there exists a shape adornment for* G *that satisfies* $(\mathcal{C}, T)$.

The SHACL validation problem *consists of deciding, for a given* G *and* $(\mathcal{C}, T)$, *whether* G *validates* $(\mathcal{C}, T)$.

Example 2. Consider the following shape constraints:

$$\begin{aligned} \mathcal{C}_{\texttt{Pizza}} = \{ & \mathit{Pizza} \equiv \geq_2 \texttt{hasTopping}.\top, \\ & \mathit{VeggiePizza} \equiv \mathit{Pizza} \wedge \forall \texttt{hasTopping}.\mathit{VeggieTopping}, \\ & \mathit{VeggieTopping} \equiv \{\texttt{mozzarella}\} \vee \{\texttt{tomato}\} \vee \{\texttt{basil}\} \vee \{\texttt{artichoke}\} \quad \} \end{aligned}$$

The graph $G_{\texttt{Pizza}}$ in Figure 3 validates the target $\mathit{Pizza}(\texttt{pizza\_capricciosa})$, but it does not validate the target $\mathit{VeggiePizza}(\texttt{pizza\_capricciosa})$ since there is no shape adornment $\mathcal{I}$ satisfying $\mathcal{C}_{\texttt{Pizza}}$ where $\texttt{prosciutto} \in \mathit{VeggieTopping}^{\mathcal{I}}$.

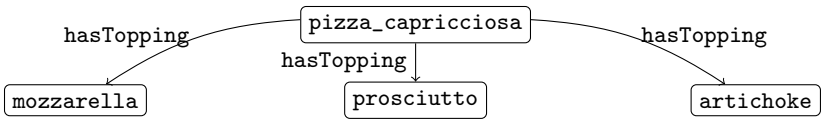


Fig. 3. A pizza data graph $G_{\texttt{Pizza}}$

3 SHACL, OWL and Description Logics

The syntax of shape expressions is more than familiar to those acquainted with *description logics*, a well-studied family of decidable logics tailored for knowledge

representation and reasoning [6, 7], and the logics underpinning the OWL standard. In fact, if instead of shape names we call the symbols in **S** *concept names*, then exactly the same grammar defines *concept expressions* φ in a description logic that we will call $\mathcal{ALCOIQ}^{=}_{reg}$. It extends the well-known description logic $\mathcal{ALCOIQ}$ with regular role expressions and equalities thereof. *Concept definitions* take the form

$$s \equiv \varphi$$

Hence, there is basically no difference between SHACL constraints and sets of $\mathcal{ALCOIQ}^{=}_{reg}$ concept definitions, at least syntactically.

In description logics, one often considers not only concept definitions but a more general form of *concept inclusions* $\varphi_1 \sqsubseteq \varphi_2$, and a set of such inclusions is called an *ontology*. The variant of OWL called OWL-DL allows writing ontologies in a DL called $\mathcal{SHOIQ}$ that is quite similar to $\mathcal{ALCOIQ}^{=}_{reg}$ [15]. The main differences are that the concept constructor $\rho = \rho'$ is omitted, and instead of regular path expressions ρ we can only use property names p and their inverses p^- inside concept expressions. $\mathcal{SHOIQ}$ also allows for subproperty relations, not supported in SHACL, and for declaring a set of property names that must be interpreted as transitive relations, which can be used inside concept expressions of the form $\exists p.\varphi$ and $\forall p.\varphi$.

Semantically they are closely related too. However, we must pay attention to some subtle details. The semantics of DLs is also defined in terms of interpretations $\mathcal{I}$ as above, and the satisfaction of concept definitions is just as for SHACL constraints. For the more general concept inclusions $\varphi_1 \sqsubseteq \varphi_2$ we have satisfaction if $\varphi_1^{\mathcal{I}} \subseteq \varphi_2^{\mathcal{I}}$, as expected. An interpretation $\mathcal{I}$ that satisfies all the inclusions in an ontology $\mathcal{O}$ is called a *model* of $\mathcal{O}$, in symbols, $\mathcal{I} \models \mathcal{O}$. When we pair an ontology with a data graph G (typically called an ABox in description logics jargon), then we require $\mathcal{I}$ to model G as well, that is, to contain all the facts given in the graph. Formally, we say that $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ models $G = (N, E, \ell)$ if $N \subseteq \Delta$ and $\{(a, b) \in E \mid p \in \ell(a, b)\} \subseteq p^{\mathcal{I}}$ for each property p. $\mathcal{I}$ is a model of $(\mathcal{O}, G)$ if $\mathcal{I}$ models both $\mathcal{O}$ and G. We say that $(\mathcal{O}, G)$ is *consistent* if it admits a model, and we say that a fact $s(a)$ (resp. $p(a, b)$) is *entailed by* $(\mathcal{O}, G)$ if $a \in s^{\mathcal{I}}$ (resp. $(a, b) \in p^{\mathcal{I}}$) for every model $\mathcal{I}$ of $(\mathcal{O}, G)$.

We stress here that for reasoning in description logics, we typically consider *all models*, in contrast to SHACL, where we restrict our attention to shape adornments. In fact, SHACL can be seen as a special instance of $\mathcal{ALCOIQ}^{=}_{reg}$ with *closed predicates* [24, 20]. Closed predicates are a well-known extension of description logics for reasoning in settings where complete and incomplete information co-exist. A selected set of concepts and roles is declared to be *closed*, and the models of the ontology are restricted to those that interpret the closed predicates exactly as in the data. Shape adornments in SHACL precisely coincide with the models in $\mathcal{ALCOIQ}^{=}_{reg}$ when *all roles are closed*.

The following example illustrates the difference between SHACL validation and description logic entailment.

Example 3. Consider the graph $G'_{\texttt{Pizza}}$ in Figure 4, and the following shape constraints:

$$\mathcal{C}'_{\texttt{Pizza}} = \{\, \mathit{VeggiePizza} \equiv \forall \texttt{hasTopping}.\, \mathit{VeggieTopping},$$
$$\mathit{VeggieTopping} \equiv \{\texttt{mozzarella}\} \vee \{\texttt{tomato}\} \vee \{\texttt{basil}\} \vee \{\texttt{artichoke}\}\}$$

Then $G'_{\texttt{Pizza}}$ validates the target $\mathit{VeggiePizza}(\texttt{pizza\_margherita})$, as witnessed by the shape adornment $\mathcal{I}$ that assigns the nodes `mozzarella`, `tomato` and `basil` to the shape *VeggieTopping* and the node `pizza_margherita` to *VeggiePizza*.

Assume now that *VeggieTopping* and *VeggiePizza* are concept names, and consider the description logic ontology:

$$\mathcal{O}_{\texttt{Pizza}} = \{\, \mathit{VeggiePizza} \equiv \forall \texttt{hasTopping}.\, \mathit{VeggieTopping},$$
$$\mathit{VeggieTopping} \equiv \{\texttt{mozzarella}\} \vee \{\texttt{tomato}\} \vee \{\texttt{basil}\} \vee \{\texttt{artichoke}\}\}$$

The interpretation $\mathcal{I}$ above is a model of $\mathcal{O}_{\texttt{Pizza}}, G'_{\texttt{Pizza}}$. However, the following $\mathcal{I}'$ is also a model:

$$\begin{aligned} \mathit{VeggieTopping}^{\mathcal{I}'} &= \mathit{VeggieTopping}^{\mathcal{I}} \\ \mathit{hasTopping}^{\mathcal{I}'} &= \mathit{hasTopping}^{\mathcal{I}} \cup \{(\texttt{pizza\_margherita}, \texttt{new\_topping})\} \\ \mathit{VeggiePizza}^{\mathcal{I}'} &= \emptyset \end{aligned}$$

where `pizza_margherita` has an additional non-vegetarian topping. This shows that $\mathcal{O}_{\texttt{Pizza}}, G'_{\texttt{Pizza}}$ does not entail $\mathit{VeggiePizza}(\texttt{pizza\_margherita})$.

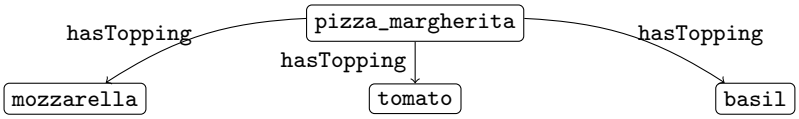


Fig. 4. Another pizza data graph $G'_{\texttt{Pizza}}$

3.1 Reasoning in SHACL and in OWL

The fundamental difference in purpose between SHACL and OWL means that their reasoning problems are also different. In SHACL we are interested in whether the input graph validates the constraints, which is essentially a *model checking* problem. In contrast, in description logics we focus on *inferring* information: determining whether facts and inclusions are *entailed* by the ontology, that is, whether they are true in *all models*. Moreover, models may extend the graph and make it arbitrarily large. In fact, even for significantly less expressive description logics than $\mathcal{ALCOIQ}^{=}_{reg}$—such as $\mathcal{ALCIF}$—there are ontologies that only have models with an infinite domain.

The different nature of SHACL reasoning vs traditional description logic reasoning matters because, as we know very well in logic, entailment is computationally much more challenging than model checking. One can naturally define logic reasoning problems for SHACL, like *satisfiability* or *containment* of constraints [21, 19]. But the connection to DLs allows us to immediately observe that such problems are practically always undecidable. Consider the *satisfiability of SHACL constraints*: given a set of constraints $\mathcal{C}$, is there a graph for which an interpretation satisfying the constraints exists? This problem is precisely the $\mathcal{ALCOIQ}^{=}_{reg}$ satisfiability problem, which has been known to be undecidable for decades. Indeed, the equality between regular role expressions $\rho = \rho'$ is a variation of the well-known *role-value maps* $p_1 \cdot \dots \cdot p_n \subseteq p'_1 \cdot \dots \cdot p'_m$ that were present already in the very early description logic KL-ONE. Schmidt-Schauß proved in 1989 that role-value maps make inference in KL-ONE undecidable [22], one of the oldest undecidability results in the field. It is widely known that without strong restrictions on the role-value maps, not even the weakest of DLs remain decidable [8]. Even without path equalities, allowing path expressions in the counting constructors $\geq_n \rho.\varphi$ and $\leq_n \rho.\varphi$ is another well-known cause of undecidability [18]. If we only allow property names and their inverses in counting concepts, and restrict complex property paths to allowing expressions of the forms $\exists r^*.\varphi$ and $\forall r^*.\varphi$, then we end up with the description logic $\mathcal{ALCOIQ}^*$: the decidability of which is a very long-standing open problem in description and modal logic [17].

These straightforward observations already make clear that SHACL satisfiability and containment can only be decidable for rather restricted fragments. For instance, we can restrict ourselves to $\mathcal{ALCOIQ}$, which only allows property names and their inverses, and obtain decidability. (We note that this logic is closely related to $\mathcal{SHOIQ}$ mentioned above, and their satisfiability problems are interreducible). A detailed study of fragments of SHACL with (un)decidable satisfiability and containment problems has been done by Pareti et al. [21], while Leinberger et al. [19] have shown cases where containment is decidable by reducing the problem to description logic reasoning.

Description logics also tell us a lot about the complexity of satisfiability and containment in SHACL fragments. But the news is not particularly positive and their worst-case complexity is high. Indeed, in the $\mathcal{ALCOIQ}$ fragment, satisfiability is complete for NExpTime [25]. (Here the ontology is considered as input, that is, we are talking of *ontology complexity* or *combined complexity*). On the positive side, $\mathcal{ALCOIQ}$ and $\mathcal{SHOIQ}$ are supported by off-the-shelf reasoners which can handle efficiently large real-world ontologies, and these reasoners can be directly deployed for deciding SHACL satisfiability and containment in the corresponding fragments.

4 What has Logic done for SHACL?

Viewing SHACL as a logic allows us to transfer important insights, techniques and results from other areas of logic. We already illustrated how we can obtain

some (un)decidability and complexity results directly from description logics. In this section we briefly summarize a few recent results of our research group that also illustrate how the logic view of SHACL can be a stepping stone to providing robust solutions to some SHACL open problems.

4.1 Semantics of Recursive SHACL

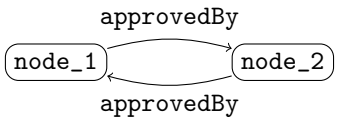
Our definition of SHACL expressions above imposes no constraints on the occurrences of shape names in the definitions of other shape names, that is, it allows for *recursion*. Given a set of SHACL constraints $\mathcal{C}$, its *dependency graph* has a node for each shape name occurring in $\mathcal{C}$, and there is an arc from s to s' if s' occurs in the body φ of a constraint $s \equiv \varphi$ for s. $\mathcal{C}$ is called *recursive* if this graph contains a cycle.[4] The SHACL specification does not disallow such recursion, but when the semantics of validation is defined, one encounters a surprise:

> "The validation with recursive shapes is not defined in SHACL and is left to SHACL processor implementations. For example, SHACL processors may support recursion scenarios or produce a failure when they detect recursion." *SHACL Recommendation [29], §3.4.3*

Naturally, this lack of proper validation semantics in the presence of recursion was one of the first SHACL open problems to be addressed by means of formal logic. The semantics that we have presented here is often called *supported model semantics*, and it was proposed already by Corman et al. when they first formalized SHACL [13]. However, this semantics has not been free of criticism.

Example 4. Consider the following constraint, stating that a node may be certified if it either has a certificate or if it has been approved by a node that is certified. Consider the graph $G_{\texttt{cc}}$ below.

$$\mathit{certifiedNode} \equiv \exists \texttt{hasCertificate}.\mathit{Certificate} \vee \exists \texttt{approvedBy}.\mathit{certifiedNode}$$



$G_{\texttt{cc}}$ validates the target $\mathit{certifiedNode}(\texttt{node\_1})$, as witnessed by the adornment

$$\mathit{certifiedNode}^{\mathcal{I}} = \{\texttt{node\_1}, \texttt{node\_2}\}.$$

Intuitively, `node_1` can be considered certified because it was approved by `node_2`, which in turn is certified because it was approved by `node_1`, although no node has any legitimate certification.

[4] Note that this *monadic* recursion over shape names is orthogonal to the linear recursion over properties present in the path expressions ρ.

The stable model semantics [4] and the well-founded semantics [12] were both proposed for avoiding these dubious validations and instead only allow for validations that are based on proper well-founded assignments. Both semantics are not only intuitive, but they are also computationally more manageable. Unlike supported validation, stable validation can be decided in polynomial time if the constraints are *stratified*, which intuitively allows only for *positive* recursion cycles. Here we refer to *data complexity*, which assumes that the constraints are fixed and measures the complexity in terms of the size of the data graph only. Well-founded validation is particularly interesting since it is *always* computable in polynomial time; it yields a three-valued approximation of stable models, and coincides with it whenever there is no recursion involving negation. These results witness the value of building on the decades of experience of the logic programming and non-monotonic reasoning community when defining proper semantics for full recursive SHACL, an aspect emphasized in [9]. Techniques for efficient goal-oriented validation in the presence of recursion, like *magic sets*, have been successfully applied to SHACL [3].

4.2 Explaining Non-Validation

The SHACL specification calls for the so-called *validation reports*, which are meant to explain to the users the outcome of validating an RDF graph against a collection of constraints. The specification gives some details about how these reports should look, e.g., which fields they should contain (e.g., the node(s) and value(s) that caused the failure of some target), but it does not address the problem of what does it mean to 'cause' a failure, and how to find the causes when a test does not succeed. These questions are far from obvious.

In our recent work [1] we draw inspiration from logic-based abduction and database repairs to study the problem of explaining non-validation of SHACL constraints. In our framework non-validation is explained using the notion of a *repair*, i.e., a collection of additions and deletions whose application on an input graph results in a repaired graph that does satisfy the given SHACL constraints.

Example 5. Consider the following shapes graph $(\mathcal{C}_t, T_t)$ and data graph G_t.

$$\begin{aligned}\mathcal{C}_t =& \{ \mathit{Teacher} \leftrightarrow \exists \mathtt{teaches}.\top\ ,\\ & \quad \mathit{Student} \leftrightarrow \exists \mathtt{enrolledIn}.\top \wedge \neg \mathit{Teacher}\}\\ T_t =& \{\mathit{Student}(\mathtt{Ben}),\ \mathit{Teacher}(\mathtt{Ann})\}\\ G_t =& \{\mathtt{enrolledIn}(\mathtt{Ben}, \mathtt{c}_1), \mathtt{teaches}(\mathtt{Ben}, \mathtt{c}_2)\}\end{aligned}$$

G_t does not validate either target. To validate $\mathit{Student}(\mathtt{Ben})$ we need to remove $\mathtt{teaches}(\mathtt{Ben}, \mathtt{c}_2)$, while to validate $\mathit{Teacher}(\mathtt{Ann})$ we need to add a fact $\mathtt{teaches}(\mathtt{Ann}, \mathtt{c})$ for some $c \in \mathbf{N}$. We call the pair (A, D) of *Additions* $A = \{\mathtt{teaches}(\mathtt{Ann}, \mathtt{c})\}$ and *Deletions* $D = \{\mathtt{teaches}(\mathtt{Ben}, \mathtt{c}_2)\}$ an *explanation* for the SHACL validation problem above.

Since sometimes we need to introduce fresh nodes, in order to keep things computationally manageable, we assume that the set of acceptable additions is somehow given as an (implicit or explicit) data graph H. Then we can define our SHACL explanations as follows.

Definition 2 ([1]). *Let G be a data graph, let $(\mathcal{C}, T)$ be a shapes graph, and let the set of* hypotheses *H be a data graph disjoint from G. Then $\Psi = (G, \mathcal{C}, T, H)$ is a* SHACL Explanation Problem (SEP). *An* explanation *for Ψ is a pair (A, D), such that (a) $D \subseteq G$, $A \subseteq H$, and (b) $(G \setminus D) \cup A$ validates $(\mathcal{C}, T)$.*

A preference order *is a preorder $\preceq$ on the set of explanations for Ψ. A* preferred explanation *of a SEP Ψ under the $\preceq$, or a $\preceq$-*explanation *for short, is an explanation ξ such that there is no explanation ξ' for Ψ with $\xi' \preceq \xi$ and $\xi \not\preceq \xi'$.*

The following decision problems for explanations are defined:

- $\preceq$-IsExpl*: is a given pair (A, D) a $\preceq$-explanation for Ψ?*
- $\preceq$-Exist*: does there exist a $\preceq$-explanation for Ψ?*
- $\preceq$-NecAdd*: is α a* $\preceq$-necessary *addition for Ψ, that is does α occur in A in* every *$\preceq$-explanation (A, D) for Ψ?*
- $\preceq$-NecDel*: is α a* $\preceq$-necessary *deletion for Ψ, that is does α occur in D in* every *$\preceq$-explanation (A, D) for Ψ?*
- $\preceq$-RelAdd*: is α a* $\preceq$-relevant *addition for Ψ, that is does α occur in A in* some *$\preceq$-explanation (A, D) for Ψ?*
- $\preceq$-RelDel*: is α a* $\preceq$-relevant *deletion for Ψ, that is does α occur in D in* some *$\preceq$-explanation (A, D) for Ψ?*

We studied all these decision problems for the following preorders $\preceq$:

- the *subset relation* $(A, D) \subseteq (A', D')$, defined as $A' \subseteq A$ and $D' \subseteq D$,
- the *cardinality relation* $(A, D) \leq (A', D')$, defined as $|A| + |D| \leq |A'| + |D'|$,
- the identity; in this case, we may talk of 'no preference order' and omit $\preceq$.

We characterized the computational complexity of all of them, in the general case and in the non-recursive case. We also analyzed the effect on the complexity of restricting the set of predicates that can be added or removed. Most of the problems turned out to be intractable, up to the second level of the polynomial hierarchy, but some problems can be solved in polynomial time. The results are summarised in Table 1, see [1] for details.

pref. order	IsExpl	Exist	NecAdd	NecDel	RelAdd	RelDel
$\emptyset$	NP	NP	coNP	coNP	NP	NP
$\subseteq$	DP	NP	coNP	coNP	Σ_2^P	Σ_2^P
$\leq$	DP	NP	$P^{\|\text{NP}}$	$P^{\|\text{NP}}$	$P^{\|\text{NP}}$	$P^{\|\text{NP}}$

Table 1. The complexity of SHACL Explanation Problems (completeness results). The bounds hold also under signature restrictions and for non-recursive SHACL.

Our algorithms can be a stepping stone for automatically computing such explanations, even in intractable cases. For instance, a follow-up work of some co-authors used Answer Set Programming (ASP) to implement a prototype tool for repairing SHACL specifications [2].

5 Conclusions and Outlook

The recent emergence of SHACL as a standard for web data provides fresh evidence that, in the complex data landscape of today's world, increasingly flexible new formalisms for describing, validating, and managing data are still needed, and approaches grounded in formal logic are as important as ever. SHACL has profound connections to well-established research fields of logic in computer science, in particular to two related communities: description logics on the one hand, and logic programming and non-monotonic reasoning on the other. The debates about the semantics of negation are reminiscent of the challenges that the latter community faced already in the 1990s, and in retrospect, maybe more active communication between these two fields could have helped avoid the tortuous road towards the different semantics for validation of recursive SHACL.

It is hard to overstate the similarity between SHACL and description logics. We gave a few examples of (un)decidability and complexity results that can be immediately transferred to the SHACL world, but there is much more to leverage in this connection. The vast trove of computational complexity and decidability results accumulated by the community can guide further SHACL developments, and the use of SHACL may bring forward new problems that have not yet been addressed in description logics.

The open- versus the closed-world assumption is often emphasized as a key difference between OWL and SHACL. And discussed, SHACL can be seen as a DL where properties are *closed predicates*. This points once more towards an ever-present challenge in knowledge representation and reasoning: **combining open- and closed-world reasoning**. Neither of them is enough on its own, since more often than not complete and incomplete information co-exist, and useful inferences call for leveraging this partial completeness.

In our ongoing work we are studying, for example, how to do validation in the presence of ontologies, that is, taking implicit facts into account in the validation. We are also exploring techniques for SHACL validation when the graph is subjected to updates, and we are continuing our work on explanations for SHACL in order to devise better and more useful validation reports.

Many open problems remain ahead, and SHACL is a relatively young field where technologies are still being constructed. We hope that this short journey through SHACL and some of its challenges may inspire more logicians to explore SHACL and other emerging data management solutions and to try to contribute to that field by bringing insights from well-established areas of logic [23].

Acknowledgments The work summarized here has been carried out with many collaborators, to whom I want to express my sincere thanks. I want to thank in particular the current and former members of our team Mantas Šimkus, Shqiponja Ahmetaj, Anouk Oudshoorn, Medina Andresel and Bianca Löhnert. Thank you also to Juan Reutter and Julien Corman.

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. It was also partially supported by the Austrian Science Fund (FWF) projects P30360 and P30873.

References

1. Ahmetaj, S., David, R., Ortiz, M., Polleres, A., Shehu, B., Simkus, M.: Reasoning about explanations for non-validation in SHACL. In: KR. pp. 12–21 (2021)
2. Ahmetaj, S., David, R., Polleres, A., Simkus, M.: Repairing SHACL constraint violations using answer set programming. In: ISWC. Lecture Notes in Computer Science, vol. 13489, pp. 375–391. Springer (2022)
3. Ahmetaj, S., Löhnert, B., Ortiz, M., Simkus, M.: Magic shapes for SHACL validation. Proc. VLDB Endow. **15**(10), 2284–2296 (2022)
4. Andresel, M., Corman, J., Ortiz, M., Reutter, J.L., Savkovic, O., Šimkus, M.: Stable model semantics for recursive SHACL. In: Proc. of The Web Conference 2020. p. 1570–1580. ACM (2020). https://doi.org/10.1145/3366423.3380229
5. Areces, C., ten Cate, B.: Hybrid logics. In: Blackburn, P., van Benthem, J.F.A.K., Wolter, F. (eds.) Handbook of Modal Logic, Studies in logic and practical reasoning, vol. 3, pp. 821–868. North-Holland (2007). https://doi.org/10.1016/s1570-2464(07)80017-6, https://doi.org/10.1016/s1570-2464(07)80017-6
6. 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)
7. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017)
8. Baader, F., Théron, C.: Role-value maps and general concept inclusions in the minimal description logic with value restrictions or revisiting old skeletons in the DL cupboard. Künstliche Intell. **34**(3), 291–301 (2020)
9. Bogaerts, B., Jakubowski, M.: Fixpoint semantics for recursive SHACL. In: ICLP Technical Communications. EPTCS, vol. 345, pp. 41–47 (2021)
10. Bogaerts, B., Jakubowski, M., den Bussche, J.V.: Expressiveness of SHACL features. In: ICDT. LIPIcs, vol. 220, pp. 15:1–15:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
11. Bogaerts, B., Jakubowski, M., den Bussche, J.V.: SHACL: A description logic in disguise. In: LPNMR. Lecture Notes in Computer Science, vol. 13416, pp. 75–88. Springer (2022)
12. Chmurovic, A., Simkus, M.: Well-founded semantics for recursive SHACL. In: Datalog. CEUR Workshop Proceedings, vol. 3203, pp. 2–13. CEUR-WS.org (2022)
13. Corman, J., Reutter, J.L., Savkovic, O.: Semantics and validation of recursive SHACL. In: Proc. of ISWC'18. Springer (2018). https://doi.org/10.1007/978-3-030-00671-6_19
14. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)

15. Horrocks, I., Sattler, U.: A tableau decision procedure for *SHOIQ*. J. Autom. Reason. **39**(3), 249–276 (2007)
16. Jakubowski, M.: Formalization of SHACL. Tech. rep., (unpublished) (2021), `https://www.mjakubowski.info/files/shacl.pdf`, accessed April 15, 2023
17. Kaminski, M., Smolka, G.: A goal-directed decision procedure for hybrid PDL. J. Autom. Reason. **52**(4), 407–450 (2014)
18. Kazakov, Y., Sattler, U., Zolin, E.: How many legs do I have? Non-simple roles in number restrictions revisited. In: LPAR. Lecture Notes in Computer Science, vol. 4790, pp. 303–317. Springer (2007)
19. Leinberger, M., Seifer, P., Rienstra, T., Lämmel, R., Staab, S.: Deciding SHACL shape containment through description logics reasoning. In: ISWC (1). Lecture Notes in Computer Science, vol. 12506, pp. 366–383. Springer (2020)
20. Ngo, N., Ortiz, M., Simkus, M.: Closed predicates in description logics: Results on combined complexity. In: KR. pp. 237–246. AAAI Press (2016)
21. Pareti, P., Konstantinidis, G., Mogavero, F.: Satisfiability and containment of recursive SHACL. J. Web Semant. **74**, 100721 (2022)
22. Schmidt-Schauß, M.: Subsumption in KL-ONE is undecidable. In: Brachman, R.J., Levesque, H.J., Reiter, R. (eds.) Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR'89). Toronto, Canada, May 15-18 1989. pp. 421–431. Morgan Kaufmann (1989)
23. Schneider, T., Simkus, M.: Ontologies and data management: A brief survey. Künstliche Intell. **34**(3), 329–353 (2020)
24. Seylan, I., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over DBoxes. In: IJCAI. pp. 923–925 (2009)
25. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. J. Artif. Intell. Res. **12**, 199–217 (2000)
26. World Wide Web Consortium: OWL 2 Web Ontology Language Primer. World Wide Web Consortium (W3C) (2009), `https://www.w3.org/TR/owl2-primer/`
27. World Wide Web Consortium: Sparql 1.1 query language (2013), `https://www.w3.org/TR/sparql11-query/`
28. World Wide Web Consortium: RDF 1.1 Primer. World Wide Web Consortium (W3C) (2014), `https://www.w3.org/TR/rdf11-primer/`
29. World Wide Web Consortium: Shape Constraint Language (SHACL). W3C World Wide Web Consortium (2017), `https://www.w3.org/TR/shacl`