SHACL: A Description Logic in Disguise

Bart Bogaerts[1], Maxime Jakubowski[2,1], and Jan Van den Bussche[2]

[1] Vrije Universiteit Brussel, Brussels, Belgium
{bart.bogaerts,maxime.jakubowski}@vub.be
[2] Universiteit Hasselt, Hasselt, Belgium
{maxime.jakubowski,jan.vandenbussche}@uhasselt.be

Abstract. SHACL is a W3C-proposed language for expressing structural constraints on RDF graphs. In recent years, SHACL's popularity has risen quickly. This rise in popularity comes with questions related to its place in the semantic web, particularly about its relation to OWL (the de facto standard for expressing ontological information on the web) and description logics (which form the formal foundations of OWL). We answer these questions by arguing that *SHACL is in fact a description logic*. On the one hand, our answer is surprisingly simple, some might even say obvious. But, on the other hand, our answer is also controversial. By resolving this issue once and for all, we establish the field of description logics as the solid formal foundations of SHACL.

Keywords: Shapes, SHACL, Description Logics, Ontologies

1 Introduction

The Resource Description Framework (RDF [21]) is a standard format for publishing data on the web. RDF represents information in the form of directed graphs, where labeled edges indicate properties of nodes. To facilitate more effective access and exchange, it is important for a consumer of an RDF graph to know what properties to expect, or, more generally, to be able to rely on certain structural constraints that the graph is guaranteed to satisfy. We therefore need a declarative language in which such constraints can be expressed formally.

Two prominent proposals in this vein have been ShEx [8] and SHACL [23]. In both approaches, a formula expressing the presence (or absence) of certain properties of a node (or its neighbors) is referred to as a "shape". In this paper, we adopt the elegant formalization of shapes in SHACL proposed by Corman, Reutter and Savkovic [9]. That work has revealed a striking similarity between *shapes* and *concept expressions*, familiar from description logics (DLs) [5].

The similarity between SHACL and DLs runs even deeper when we account for *named shapes* and *targeting*, which is the actual mechanism to express constraints on an RDF graph using shapes. A *shape schema* is essentially a finite list of shapes, where each shape ϕ_s is given a name s and additionally associated with a target query q_s. The shape–name combinations in a shape schema specify, in DL terminology, an *acyclyc TBox* consisting of all the formulas

$$s \equiv \phi_s.$$

Given an RDF graph G, this acyclic TBox determines a unique interpretation of sets of nodes to shape names s. We then say that G *conforms* to the schema if for each query q_s, each node v returned by q_s on G satisfies s in the extension of G.

Now interestingly, the types of target queries q considered for this purpose in SHACL as well as in ShEx, actually correspond to simple cases of shapes ϕ_{q_s} and the actual integrity constraint thus becomes

$$\phi_{q_s} \sqsubseteq s.$$

As such, in description logic terminology, a shape schema consists of two parts: an acyclic TBox (defining the shapes in terms of the given input graph) and a general TBox (containing the actual integrity constraints).

2 The Wedge

Despite the strong similarity between SHACL and DLs, and despite the fact that in a couple of papers, SHACL has been formalized in a way that is extremely similar to description logics [9,3,14], this connection is not recognized in the community. In fact, some important stakeholders in SHACL recently even wrote the following in a blog post explaining why they use SHACL, rather than OWL:

> “OWL was inspired by and designed to exploit 20+ years of research in Description Logics (DL). This is a field of mathematics that made a lot of scientific progress right before creation of OWL. I have no intention of belittling accomplishments of researchers in this field. However, there is little connection between this research and the practical data modeling needs of the common real world software systems. — [19]”

thereby suggesting that SHACL and DLs are two completely separated worlds and as such contradicting the introductory paragraphs of this paper. On top of that, SHACL is presented by some stakeholders [25] as an alternative to the Web ontology language OWL [16], which is based on the description logic SROIQ [10].

This naturally begs the question: which misunderstanding is it that drives this wedge between communities? How can we explain this discrepancy from a mathematical perspective (thereby patently ignoring strategic, economic, social, and other aspects that play a role).

3 SHACL, OWL, and Description Logics

Our answer is that there are two important differences between OWL and SHACL that deserve attention. These differences, however, do not contradict the central thesis of this paper, which is that *SHACL is a description logic.*

1. The first difference is that **in SHACL, the data graph (implicitly) represents a first-order interpretation, while in OWL, it represents a first-order theory (an ABox)**. Of course, viewing the same syntactic structure (an RDF graph) as an interpretation is very different from viewing it as a theory. While this is a discrepancy between OWL and SHACL, theories as well as interpretations exist in the world of description logic and as such, this view is perfectly compatible with our central thesis. There is, however, one caveat with this claim that deserves some attention, and that is highlighted by the use of the world "implicitly". Namely, to the best of our knowledge, it is never mentioned that the data graph simply represents a standard first-order interpretation, and it has not been made formal what *exactly* the interpretation is that is associated to a graph. Instead, SHACL's language features are typically evaluated *directly* on the data graph. There are several reasons why we believe it is important to make this translation of a graph into an interpretation *explicit.*
   – This translation makes *the assumptions SHACL makes about the data* explicit. For instance, it is often informally stated that "SHACL uses closed-world assumptions" [13]; we will make this statement more precise: SHACL uses closed-world assumptions with respect to the relations, but open-world assumptions on the domain.
   – Once the graph is eliminated, we are in familiar territory. In the field of description logics a plethora of language features have been studied. It now becomes clear how to add them to SHACL, if desired. The 20+ years of research mentioned in [19] suddenly become directly applicable to SHACL.
2. The second difference, which closely relates to the first, is that **OWL and SHACL have a different (default) inference task**: the standard inference task at hand in OWL is *deduction*, while in SHACL, the main task is validation of RDF graphs against shape schemas. In logical terminology, this is evaluating whether a given interpretation satisfies a theory (TBox), i.e., this is the task of *model checking.*
   Of course, the fact that a different inference task is typically associated with these languages does not mean that their logical foundations are substantially different. Furthermore, recently, other researchers [14,17,18] have started to investigate tasks such as *satisfiability* and *containment* (which are among the tasks typically studied in DLs) for SHACL, making it all the more obvious that the field of description logics has something to offer for studying properties of SHACL.

In the next section, we develop our formalization of SHACL, building on the work mentioned above. Our formalization differs form existing formalizations of SHACL in a couple of small but important ways. First, as we mentioned, we explicitly make use of a first-order interpretation, rather than a graph, thereby indeed showing that SHACL is in fact a description logic. Second, the semantics for SHACL we develop would be called a "natural" semantics in database theory [1]: variables always range over the universe of all possible nodes. The use of the natural semantics avoids an anomaly that crops up in the definitions of Andreşel et al. [3], where an "active-domain" semantics is adopted instead, in which variables range only over the set of nodes actually occurring in the input graph. Unfortunately, such a semantics does not work well with constants. The problem is that a constant mentioned in a shape may or may not actually occur in the input graph. As a result, the semantics adopted by Andreşel et al. violates familiar logic laws like De Morgan's law. This is troublesome, since automated tools (and humans!) that generate and manipulate logic formulas may reasonably and unwittingly assume these laws to hold. Also other research papers (see Remark 4) contain flaws related to not taking into account nodes that *do not* occur in the graph. This highlights the importance of taking a logical perspective on SHACL.

A minor caveat with the natural semantics is that decidability of validation is no longer totally obvious, since the universe of nodes is infinite. A solution to this problem is well-known from relational databases [1, Theorem 5.6.1]. Using an application of solving the first-order theory of equality, one can reduce, over finite graphs, an infinite domain to a finite domain, by adding symbolic constants [4,11]. It turns out that in our case, just a single extra constant suffices.

In this paper, we will not give a complete syntactic translation of SHACL shapes to logical expressions. In fact, such a translation has already been developed by Corman et al. [9], and was later extended to account for all SHACL features by Jakubowski [12]. Instead, we show very precisely how the data graph at hand can be viewed as an interpretation, and that after this small but crucial step, we are on familiar grounds and know well how to evaluate expressions.

As already mentioned before, our formalization of SHACL differs in a couple of ways from existing work. These design choices are grounded in true SHACL: with each of them we will provide actual SHACL specifications that prove that SHACL validators indeed behave in the way we expected. All our examples have been tested on three SHACL implementations: Apache Jena SHACL[3] (using their Java library) TopBraid SHACL[4] (using their Java library as well as their online playground), and Zazuko[5] (using their online playground). The raw files encoding our examples (SHACL specifications and the corresponding graphs) are available online.[6] All our SHACL examples will assume the following prefixes are defined:

```
@prefix ex: <http://www.example.org/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
```

[3] https://jena.apache.org/documentation/shacl/index.html
[4] https://shacl.org/playground/
[5] https://shacl-playground.zazuko.com/
[6] https://vub-my.sharepoint.com/:f:/g/personal/bart_bogaerts_vub_be/Eicv10DwSnVEnT0BWNwEW8QBFuQjYTbwYYct1WYrkoefKQ?e=XhE8o0.

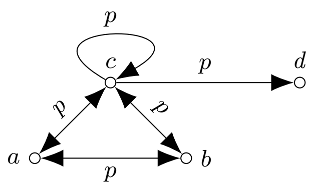

Fig. 1. An example graph to illustrate language features of SHACL.

4 SHACL: The Logical Perspective

In this section of the paper we begin with the formal development. We define shapes, shape schemas, and validation. Our point of departure is the treatment by Andreşel et al. [3], which we adapt and extend to our purposes.

From the outset we assume three disjoint, infinite universes N, S, and P of *node names*, *shape names*, and *property names*, respectively.[7] We define *path expressions* E and *shapes* ϕ by the following grammar:

$$E ::= p \mid p^- \mid E \cup E \mid E \circ E \mid E^* \mid E?$$
$$\phi ::= \top \mid s \mid \{c\} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \geq_n E.\phi \mid \mathit{eq}(p, E) \mid \mathit{disj}(p, E) \mid \mathit{closed}(Q)$$

where p, s, and c stand for property names, shape names, and node names, respectively, n stands for nonzero natural numbers, and Q stands for finite sets of property names. In description logic terminology, a node name c is a *constant*, a shape name is a *concept name* and a property name is a *role name*.

As we will formalize below, every property/role name evaluates to a binary relation, as does each path expression. In the path expressions, p^- represents the inverse relation of p, $E \circ E$ represents composition of binary relations, E^* the reflexive-transitive closure of E and $E?$ the reflexive closure of E. As we will see, shapes (which represent unary predicates) will evaluate to a subset of the domain. The three last expressions are probably the least familiar. Equality ($\mathit{eq}(p, E)$) means that there are outgoing p-edges (edges labeled p) exactly to those nodes for which there is a path satisfying the expression E (defined below). Disjointness ($\mathit{disj}(p, E)$) means that there are *no* outgoing p-edges to which there is also a path satisfying E. For instance in the graph in Figure 1, $\mathit{eq}(p, p^*)$ would evaluate to $\{c\}$, since c is the only node that has direct outgoing p-edge to all nodes that are reachable using only p-edges, and $\mathit{disj}(p, p^-)$ would evaluate to $\{d\}$ since d is the only node that has no symmetric p-edges. Closedness is also a typical SHACL feature: $\mathit{closed}(Q)$ represents that there are no outgoing edges about any predicates other than those in Q. In our example figure $\mathit{closed}(\{p\})$ would evaluate to $\{a, b, c, d\}$ and $\mathit{closed}(\{q\})$ to the empty set.

[7] In practice, node names, shape names, and property names are IRIs [21], hence the disjointness assumption does not hold. However, this assumption is only made for simplicity of notation.

E	$[\![E]\!]^I$
p^-	$\{(a,b) \mid (b,a) \in [\![p]\!]^I\}$
$E_1 \cup E_2$	$[\![E_1]\!]^I \cup [\![E_2]\!]^I$
$E_1 \circ E_2$	$\{(a,b) \mid \exists c : (a,c) \in [\![E_1]\!]^I \wedge (c,b) \in [\![E_2]\!]^I\}$
E^*	the reflexive-transitive closure of $[\![E]\!]^I$
$E?$	$[\![E]\!]^I \cup \{(a,a) \mid a \in \Delta^I\}$

Table 1. Semantics of a path expression E in an interpretation I over Σ.

Remark 1. Andreşel et al. [3] also have the construct $\forall E.\phi$, which can be omitted (at least for theoretical purposes) as it is equivalent to $\neg \geq_1 E.\neg\phi$. In our semantics, the same applies to $\phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$, of which we need only one as the other is then expressible via De Morgan's laws. However, here we keep both for the sake of our later Remark 3. In addition to the constructors of Andreşel et al. [3], we also have $E?$, *disj*, and *closed*, corresponding to SHACL features that were not included there. □

A *vocabulary* Σ is a subset of $N \cup S \cup P$. A path expression or shape is said to be *over* Σ if it only uses symbols from Σ. On the most general logical level, shapes are evaluated in *interpretations*. We recall the familiar definition: An interpretation I over Σ consists of

1. a set Δ^I, called the *domain* of I;
2. for each constant $c \in \Sigma$, an element $[\![c]\!]^I \in \Delta^I$;
3. for each shape name $s \in \Sigma$, a subset $[\![s]\!]^I$ of Δ^I; and
4. for each property name $p \in \Sigma$, a binary relation $[\![p]\!]^I$ on Δ^I.

On any interpretation I as above, every path expression E over Σ evaluates to a binary relation $[\![E]\!]^I$ on Δ^I, and every shape ϕ over Σ evaluates to a subset of Δ^I, as defined in Tables 1 and 2.

ϕ	$[\![\phi]\!]^I$
$\top$	Δ^I
$\{c\}$	$\{c^I\}$
$\phi_1 \wedge \phi_2$	$[\![\phi_1]\!]^I \cap [\![\phi_2]\!]^I$
$\phi_1 \vee \phi_2$	$[\![\phi_1]\!]^I \cup [\![\phi_2]\!]^I$
$\neg\phi_1$	$\Delta^I \setminus [\![\phi_1]\!]^I$
$\geq_n E.\phi_1$	$\{a \in \Delta^I \mid \sharp([\![\phi_1]\!]^I \cap [\![E]\!]^I(a)) \geq n\}$
$eq(p, E)$	$\{a \in \Delta^I \mid [\![p]\!]^I(a) = [\![E]\!]^I(a)\}$
$disj(p, E)$	$\{a \in \Delta^I \mid [\![p]\!]^I(a) \cap [\![E]\!]^I(a) = \emptyset\}$
$closed(Q)$	$\{a \mid [\![p]\!]^I(a) = \emptyset$ for every $p \in \Sigma \setminus Q\}$

Table 2. Semantics of a shape ϕ in an interpretation I over Σ. For a set X, we use $\sharp X$ to denote its cardinality. For a binary relation R and an element a, we use $R(a)$ to denote the set $\{b \mid (a,b) \in R\}$.

As argued above, we define a *shape schema* $\mathcal{S}$ over Σ as a tuple (D, T), where

- D is an *acyclic TBox* [5], i.e., a finite set of expressions of the form $s \equiv \phi_s$ with s a shape name in Σ and ϕ_s a shape over Σ and where
  1. each s occurs exactly once as the left-hand-side of such an expression and
  2. the transitive closure of the relation $\{(s,t) \mid t \text{ occurs in } \phi_s\}$ is acyclic.
- T is a TBox, i.e., a finite set of statements of the form $\phi_1 \sqsubseteq \phi_2$, with ϕ_1 and ϕ_2 shapes.

If $\mathcal{S} = (D, T)$ is a shape schema over Σ and I an interpretation over $\Sigma \setminus S$, then there is a unique interpretation $I \diamond D$ that agrees with I outside of S and that satisfies D, i.e., such that for every expression $s \equiv \phi_s \in D$, $[\![s]\!]^{I\diamond D} = [\![\phi_s]\!]^{I\diamond D}$. We say that I *conforms to* $\mathcal{S}$, denoted by $I \models \mathcal{S}$, if $[\![\phi_1]\!]^{I\diamond D}$ is a subset of $[\![\phi_2]\!]^{I\diamond D}$, for every statement $\phi_1 \sqsubseteq \phi_2$ in T. In other words, I conforms to $\mathcal{S}$ if there exists an interpretation that satisfies $D \cup T$ that coincides with I on $N \cup P$.

Remark 2. In real SHACL, a shape schema is called a "shapes graph". There are some notable differences between shapes graphs and our shape schemas.

First, we take abstraction of some features of real SHACL, such as checking data types like numbers and strings.

Second, in real SHACL, the left-hand side of an inclusion statement in T is called a "target" and is actually restricted to shapes of the following forms: a constant ("node target"); $\exists r.\{c\}$ ("class-based target", where r is 'rdf:type'); $\exists r.\top$ ("subjects-of target"); or $\exists r^-.\top$ ("objects-of target"). Our claims remain valid if this syntactic restriction imposed.

Third, in real SHACL not every shape name needs to occur in the left-hand side of a defining rule. The default that is taken in real SHACL is that shapes without a definition are *always satisified*. On the logical level, this means that for every shape s name that has no explicit definition, a definition $s \equiv \top$ is implicitly assumed. The example that illustrates that our chosen default indeed corresponds to actual SHACL. □

Example 1. The following SHACL shape `ex:MyShape` states that all nodes with an `ex:r`-edge must conform to the `ex:NoDef` and `ex:AlsoNoDef` shapes which we do not define.

```
ex:MyShape a sh:NodeShape ;
    sh:and ( ex:NoDef ex:AlsoNoDef ) .
ex:MyShape sh:targetSubjectsOf ex:r .
```

In our formal notation, this shapes graph corresponds to the shape schema

$$\texttt{ex:MyShape} \equiv \texttt{ex:NoDef} \land \texttt{ex:AlsoNoDef}$$
$$\exists \texttt{ex:r}.\top \sqsubseteq \texttt{ex:MyShape}$$

where the first line is the definition of `ex:MyShape`, and the second line its target.

When validating a graph containing only the triple `ex:a ex:r ex:b` (as we will show later, this corresponds to an interpretation in which the property name `ex:r` has the interpretation $\{(\texttt{ex:a}, \texttt{ex:b})\}$ and the interpretation of all other property names is empty), and thus targeting the node `ex:a`, it validates without violation. This supports our observation that **shapes without an explicit definition are assumed to be satisfied by all nodes (i.e., are interpreted as $\top$)**.

To further strengthen this claim, if instead we consider the SHACL shapes graph

```
ex:MyShape a sh:NodeShape ;
    sh:not ex:NoDef .
ex:MyShape sh:targetSubjectsOf ex:r .
```

i.e., the shape schema

$$\texttt{ex:MyShape} \equiv \neg\texttt{ex:NoDef}$$
$$\exists\texttt{ex:r}.\top \sqsubseteq \texttt{ex:MyShape}$$

validation on the same graph yields the validation error that "node `ex:a` does not satisfy `ex:MyShape` since it has shape `ex:NoDef`". □

5 From Graphs to Interpretations

Up to this point, we have discussed the logical semantics of SHACL, i.e., how to evaluate a SHACL expression in a standard first-order interpretation. However, in practice, SHACL is not evaluated on interpretations but on RDF graphs. In this section, we show precisely and unambiguously how to go from a graph to a logical interpretation (in such a way that the actual SHACL semantics coincides with what we described above). A *graph* is a finite set of *facts*, where a fact is of the form $p(a, b)$, with p a property name and a and b node names. We refer to the node names appearing in a graph G simply as the *nodes* of G; the set of nodes of G is denoted by N_G. A pair (a, b) with $p(a, b) \in G$ is referred to as an *edge*, or a *p-edge*, in G. The set of p-edges in G is denoted by $[\![p]\!]^G$ (this set might be empty).

We want to be able to evaluate *any* shape on *any* graph (independently of the vocabulary the shape is over). Thereto, we will unambiguously associate, to any given graph G, an interpretation I over $N \cup P$ as follows:

- Δ^I equals N (the universe of all node names).
- $[\![c]\!]^I$ equals c itself, for every node name c.
- $[\![p]\!]^I$ equals $[\![p]\!]^G$, for every property name p.

If I is the interpretation associated to G, we use $[\![E]\!]^G$ and $[\![\phi]\!]^G$ to mean $[\![E]\!]^I$ and $[\![\phi]\!]^I$, respectively.

RDF also has a model-theoretic semantics [20]. These semantics reflect the view of an RDF graph as a basic ontology or logical theory, as opposed to the view of an RDF graph as an interpretation. Since the latter view is the one followed by SHACL, it is thus remarkable that SHACL effectively ignores the W3C-recommended semantics of RDF.

Remark 3. Andreşel et al. [3] define $[\![\phi]\!]^G$ a bit differently. For a constant c, they define $[\![\{c\}]\!]^G = \{c\}$ like we do. For all other constructs, however, they define $[\![\phi]\!]^G$ to be $[\![\phi]\!]^I$, but with the domain of I taken to be N_G, rather than N. In that approach, if $c \notin N_G$, $[\![\neg\neg\{c\}]\!]^G$ would be empty rather than $\{c\}$ as one would expect. For another illustration, still assuming $c \notin N_G$, $[\![\neg(\neg\phi \wedge \neg\{c\})]\!]^G$ would be $[\![\phi]\!]^G$ rather than $[\![\phi]\!]^G \cup \{c\}$, so De Morgan's law would fail. The next examples shows that actual SHACL implementations indeed coincide with our semantics. □

Example 2. The following SHACL shape `ex:MyShape` states that it cannot be so that the node `ex:MyNode` is different from itself (i.e., that it must be equal to itself, but specified with a double negation).

```
ex:MyShape a sh:NodeShape ;
    sh:not [ sh:not [ sh:hasValue ex:MyNode ] ] .
ex:MyShape sh:targetNode ex:MyNode .
```

In our formal notation, this shapes graph corresponds to the shape schema

$$\begin{aligned}&\texttt{ex:MyShape} \equiv \neg\neg\{\texttt{ex:MyNode}\}\\&\{\texttt{ex:MyNode}\} \sqsubseteq \texttt{ex:MyShape}\end{aligned}$$

Clearly, this shape should validate every graph, also graphs in which the node `ex:MyNode` is not present and it indeed does so in all SHACL implementations we tested. This supports our **choice of the natural semantics**, rather than the active domain semantics of [3]. Indeed, in that semantics, this shape will never validate any graph because the right-hand side of the inclusion will be evaluated to be the empty set. □

Example 3. Another example in the same vein as the previous, to show that the **natural semantics** correctly formalizes is the one where [3]'s semantics does not respect the De Morgan's laws, as follows:

```
ex:MyShape a sh:NodeShape ;
    sh:not [
        sh:and (
            [ sh:not [
                  sh:path ex:r ;
                  sh:minCount 1 ] ]
            [ sh:not [ sh:hasValue ex:MyNode ] ] ) ] .
ex:MyShape sh:targetNode ex:MyNode .
```

This shapes graph corresponds to the shape schema

$$\begin{aligned}&\texttt{ex:MyShape} \equiv \neg(\neg\exists\texttt{ex:r}.\top \wedge \neg\{\texttt{ex:MyNode}\})\\&\{\texttt{ex:MyNode}\} \sqsubseteq \texttt{ex:MyShape}\end{aligned}$$

In the formalism of Andreşel et al. [3], this schema does not validate on graphs that do not mention the node `ex:MyNode`, but in our formalism (and all SHACL implementations), it does validate. □

Remark 4. The use of active domain semantics has also introduced some errors in previous work. For instance [14, Theorem 1] is factually incorrect. The problem originates with the notion of *faithful assignment* introduced by Corman et al. [9] and adopted by Leinberger et al. This notion is defined in an active-domain fashion, only considering nodes actually appearing in the graph. For a concrete counterexample to that theorem, consider a single shape named s defined as $\exists r.\top$, with target $\{b\}$. In our terminology, this means that

$$D = \{s \equiv \exists r.\top\}, \text{ and}$$
$$T = \{\{b\} \sqsubseteq s\}.$$

On a graph G in which b does not appear, we can assign $\{s\}$ to all nodes from G with an outgoing r-edge (meaning that all these nodes satisfy s and no other shape (names)), and assign the empty set to all other nodes (meaning that all other nodes do not satisfy any shape). According to the definition, this is a faithful assignment. However, the inclusion $\{b\} \sqsubseteq s$ is not satisfied in the interpretation they construct from this assignment, thus violating their Theorem 1. $\square$

The bug in [14], as well as the violation of De Morgan's laws will only occur in corner cases where the shape schema mentions nodes that not occur in the graph. After personal communications, Leinberger et al. [14] included an errata section where they suggest to fix this by demanding that (in order to conform) the target queries do not mention any nodes not in the graph. While technically, this indeed resolves the issue (under that condition, Theorem 1 indeed holds), this solution in itself has weaknesses as well. Indeed, shape schemas are designed to validate graphs not known at design-time, and it should be possible to check conformance of *any* graph with respect to *any* shape schema. As the following example shows, it makes sense that a graph should conform to a schema in case a certain node does *not* occur in the graph (or does not occur in a certain context), and that — contrary to the existing SHACL formalizations — the natural semantics indeed coincides with the behaviour of SHACL validators in such cases.

Example 4. Consider a schema with $D = \emptyset$ and T consisting of a single inclusion

$$\{\mathit{MarcoMaratea}\} \sqsubseteq \neg\exists(\mathit{author} \circ \mathit{venue}).\{\mathit{LPNMR22}\},$$

which states that Marco Maratea (one of the LPNMR PC chairs) does not author any LPNMR paper. If Marco Maratea does not occur in the list of of accepted papers, this list should clearly[8] conform to this schema. This example can be translated into actual SHACL as follows:

```
ex:NotAnAuthor a sh:NodeShape ;
    sh:not [
```

[8] Technically, the standard is slightly ambiguous with respect to nodes not occurring in the data graph.

```
        a sh:PropertyShape ;
        sh:path (ex:author ex:venue) ;
        sh:qualifiedValueShape [ sh:hasValue ex:LPNMR22 ] ;
        sh:qualifiedMinCount 1 ] .
ex:NotAnAuthor sh:targetNode ex:MarcoMaratea .
```

where we simply give the name `ex:NotAnAuthor` to the shape that holds for all nodes that do not author any LPNMR paper and subsequently enforce that Marco Maratea satisfy this shape. We see that indeed, in accordance with our proposed semantics, graphs without a node `ex:MarcoMaratea` validate with respect to this SHACL specification. The fix in the erratum of Leinberger et al. [14], on the other hand, specifies that this does not validate. □

The definition of I makes — completely independent of the actual language features of SHACL — a couple of assumptions explicit. First of all, SHACL uses unique names assumptions (UNA): each constant is interpreted in I as a different domain element. Secondly, if $p(a, b)$ does not occur in the graph, it is assumed to be *false*. However, if a node c does not occur anywhere in the graph, it is not assumed to not exist: the domain of I is infinite! Rephrasing this: SHACL makes the Closed World Assumption (CWA) on predicates, but not on objects.

Effective evaluation Since the interpretation defined from a graph has the infinite domain N, it is not immediately clear that shapes can be effectively evaluated over graphs. As indicated above, however, we can reduce to a finite interpretation. Let $\Sigma \subseteq N \cup P$ be a finite vocabulary, let ϕ be a shape over Σ, and let G be a graph. From G we define the interpretation $I_\star$ over Σ just like I above, except that the domain of $I_\star$ is not N but rather

$$N_G \cup (\Sigma \cap N) \cup \{\star\},$$

where $\star$ is an element not in N. We use $[\![\phi]\!]^G_\star$ to denote $[\![\phi]\!]^{I_\star}$ and find:

Theorem 1. *For every $x \in N_G \cup (\Sigma \cap N)$, we have $x \in [\![\phi]\!]^G$ if and only if $x \in [\![\phi]\!]^G_\star$. For all other node names x, we have $x \in [\![\phi]\!]^G$ if and only if $\star \in [\![\phi]\!]^G_\star$.*
Hence, I conforms to $\mathcal{S}$ if and only if $I_\star$ does.

Theorem 1 shows that conformance can be performed by finite model checking, but other tasks typically studied in DLs are not decidable; this can be shown with a small modification of the proof of undecidability of the description logic $\mathcal{ALRC}$, as detailed by Schmidt-Schauß [22].

Theorem 2. *Consistency of a shape schema (i.e., the question whether or not some I conforms to $\mathcal{S}$) is undecidable.*

Following description logic traditions, decidable fragments of SHACL have been studied already; for instance Leinberger et al. [14] disallow equality, disjointness, and closedness in shapes, as well as union and Kleene star in path expressions.

6 Related Work and Conclusion

Formal investigations of SHACL have started only relatively recently. We already mentioned the important and influential works by Corman et al. [9] and by Andreşel et al. [3], which formed the starting point for the present paper. The focus of these papers is mainly on the extending the semantics to *recursive* SHACL schemas, which are not present in the standard yet, and which we also do not consider in the current paper.

The connection between SHACL and description logics has also been observed by several other groups of researchers [14,17,18,2]. There, the focus is on typical reasoning tasks from DLs applied to shapes, and on reductions of these tasks to decidable description logics or decidable fragments of first-order logic. In its most general form, this cannot work (see Theorem 2), but the addressed works impose restrictions on the allowed shape expressions.

Next to shapes, other proposals for adding integrity constraints to the semantic web have been proposed, for instance by integrating them in OWL ontologies [24,15]. There, the entire ontology is viewed as an incomplete database.

None of the discussed works takes the explicit viewpoint that a data graph represents a standard first-order interpretation or that SHACL validation is model checking. We took this viewpoint and in doing so formalized precisely how SHACL relates to the field of description logics. There are (at least) three reasons why this formalization is important. First, it establishes a bridge between two communities, thereby allowing to exploit the many years of research in DLs also for studying SHACL. Second, our formalization of SHACL clearly separates two orthogonal concerns:

1. *Which information does a data graph represent?* This is handled in the translation of a graph into its *natural interpretation*.
2. *What is the semantics of language constructs?* This is handled purely in the well-studied logical setting.

Third, as we showed above, our formalization corresponds closer to actual SHACL than existing formalizations, respects well-known laws (such as De Morgan's) and avoids issues with nodes not occurring in the graph requiring special treatment. As such, we believe that by rooting SHACL in the logical setting, we have devised solid foundations for future studies and extensions of the language. We already build on the logical foundations of the current paper in our work on extending the semantics to *recursive* shape schemas [6], as well as in an analysis of the primitivity of the different language features of SHACL [7].

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Ahmetaj, S., David, R., Ortiz, M., Polleres, A., Shehu, B., Simkus, M.: Reasoning about explanations for non-validation in SHACL. In: Proceedings 18th International Conference on Principles of Knowledge Representation and Reasoning. pp. 12–21. IJCAI Organization (2021)

3. Andreşel, M., Corman, J., Ortiz, M., Reutter, J., Savkovic, O., Simkus, M.: Stable model semantics for recursive SHACL. In: Proceedings of WWW. pp. 1570–1580 (2020)
4. Aylamazyan, A., Gilula, M., Stolboushkin, A., Schwartz, G.: Reduction of the relational model with infinite domains to the case of finite domains. Doklady Akademii Nauk SSSR 286(2), 308–311 (1986), in Russian
5. Baader, F., Calvanese, D., McGuiness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press (2003)
6. Bogaerts, B., Jakubowski, M.: Fixpoint semantics for recursive SHACL. In: Technical Communications of ICLP. pp. 41–47 (2021)
7. Bogaerts, B., Jakubowski, M., Van den Bussche, J.: Expressiveness of SHACL features. In: Proceedings of ICDT. pp. 15:1–15:16 (2022)
8. Boneva, I., Gayo, J.L., Prud'hommeaux, E.: Semantics and validation of shape schemas for RDF. In: Proceedings of ISWC. pp. 104–120 (2017)
9. Corman, J., Reutter, J., Savkovic, O.: Semantics and validation of recursive SHACL. In: Proceedings of ISWC. pp. 318–336 (2018), extended version, technical report KRDB18-01
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Proceedings of KR. pp. 57–67 (2016)
11. Hull, R., Su, J.: Domain independence and the relational calculus. Acta Informatica 31, 513–524 (1994)
12. Jakubowski, M.: Formalization of SHACL. https://www.mjakubowski.info/files/shacl.pdf, accessed: 2021-06-16
13. Knublauch, H.: SHACL and OWL compared. https://spinrdf.org/shacl-and-owl.html, accessed: 2021-06-16
14. Leinberger, M., Seifer, P., Rienstra, T., Lämmel, R., Staab, S.: Deciding SHACL shape containment through description logics reasoning. In: Proceedings of ISWC. pp. 366–383 (2020)
15. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. J. Web Semant. 7(2), 74–89 (2009)
16. OWL 2 Web ontology language: Structural specification and functional-style syntax. W3C Recommendation (Dec 2012)
17. Pareti, P., Konstantinidis, G., Mogavero, F., Norman, T.J.: SHACL satisfiability and containment. In: Proceedings of ISWC. pp. 474–493 (2020)
18. Pareti, P., Konstantinidis, G., Mogavero, F.: Satisfiability and containment of recursive SHACL. Journal of Web Semantics (2022), https://arxiv.org/abs/2108.13063
19. Polikoff, I.: Why I don't use OWL anymore – Top Quadrant blog. https://www.topquadrant.com/owl-blog/, accessed: 2021-06-04
20. RD 1.1 semantics. W3C Recommendation (Feb 2014)
21. RDF 1.1 primer. W3C Working Group Note (Jun 2014)
22. Schmidt-Schauß, M.: Subsumption in KL-ONE is undecidable. In: Proceedings of KR. pp. 421–431 (1989)
23. Shapes constraint language (SHACL). W3C Recommendation (Jul 2017)
24. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: Proceedings of AAAI (2010)
25. TopQuadrant: An overview of SHACL: A new W3C standard for data validation and modeling. https://www.topquadrant.com/an-overview-of-shacl/ (2017), webinar slides