Reconciling SHACL and Ontologies: Semantics and Validation via Rewriting

Shqiponja Ahmetaj[1], Magdalena Ortiz[2], Anouk Oudshoorn[1,*] and Mantas Šimkus[2]

[1]*Technical University of Vienna, Wiedner Hauptstraße 11, 1040 Vienna, Austria*
[2]*Umeå University, Universitetstorget 4, 901 87 Umeå, Sweden*

Abstract
This poster summarizes our recent work [1] on SHACL validation in the presence of OWL 2 QL ontologies. To overcome the challenge posed by the non-monotonic behavior of SHACL constraints, we propose a new intuitive validation semantics and a rewriting algorithm that embeds the effects of the ontological axioms into the SHACL constraints. We analyze the complexity of validation in this setting.

Keywords
SHACL, OWL 2 QL, validation, rewriting, complexity

1. Introduction

SHACL and OWL are two prominent W3C standards for managing RDF data, specifically designed to target two different issues. OWL gives semantics to RDF data and addresses its possible incompleteness by means of ontological axioms that complete the data with implicit information. OWL and its profiles are based on *Description Logics (DLs)* and make the open-world assumption (OWA); data not present can still be true. RDF was soon adopted in many applications and making decisions based on correct data became particularly crucial. Therefore, the W3C proposed the *Shapes Constraint Language (or SHACL)*, [2] a machine-readable language for describing and validating RDF graphs. Unlike OWL, SHACL operates under the closed-world assumption (CWA) and assumes the completeness of data. SHACL specifies the notion of a *shapes graph*; a set of (possibly recursive) constraints and targets for these constraints. However, the semantics of recursive constraints is not given in the standard. This led to recent works that propose semantics based on first-order logic and logic programming [3, 4, 5].

Combining SHACL and OWL, and specifically performing data validation while considering ontological knowledge, is a relevant but challenging problem. It is envisioned by the SHACL standard, but no guidance is given in the specificaiton as to how this should be realized.

To illustrate the potential benefits of leveraging an ontology when performing validation, consider a simplistic database of pet owners containing the following facts:

$$hasWingedPet(linda, blu), Bird(blu), PetOwner(john), hasPet(john, ace)$$

ISWC 2023 Posters and Demos: 22nd International Semantic Web Conference, November 6–10, 2023, Athens, Greece
*Main author.

shqiponja.ahmetaj@tuwien.ac.at (S. Ahmetaj); magdalena.ortiz@cs.umu.se (M. Ortiz); anouk.oudshoorn@tuwien.ac.at (A. Oudshoorn); simkus@cs.umu.se (M. Šimkus)

and the simple SHACL constraint

$$petOwnerShape \leftarrow PetOwner \vee \exists hasPet$$

with the target $petOwnerShape(linda)$, which asks to verify whether $linda$ is a pet owner. As $linda$ has a winged pet, we would like to see her as a pet owner. Currently, we cannot conclude this as we are missing the background knowledge that owning a winged pet implies owning a pet. In [1], we define a semantics that can reason with such background knowledge. Such a semantics requires integrating the OWA of OWL and the CWA of SHACL.

The main contributions of our work can be summarized as follows:

○ We present a novel notion of SHACL validation in the presence of a DL-Lite$_{\mathcal{R}}$ ontology, the logic underlying OWL 2 QL. Specifically, we consider *stratified* SHACL constraints, which do not allow the full combination of recursion and negation. Our notion of stratification is derived from the well-known class of stratified logic programs [6].

○ As SHACL constraints can contain negation, we have to be very careful when defining a semantics for validation. We use knowledge stemming from the ontology to complete the data graph with additional facts in an *austere* way: only a minimal amount of new successor nodes is added at each step of the procedure, with only merging successors if the axioms give us a reason to. Validation of constraints over a data graph in the presence of an ontology is defined as validation of the constraints in the possibly infinite *austere canonical model* that we introduce. We discuss this in more detail in Section 2.

○ Validation in the presence of ontologies requires reasoning over a structure that may be much larger than the data graph, even potentially infinite, and the complexity of the decision problem is not obvious. We discuss both combined complexity and data complexity. We prove that validation is decidable and is PTime-complete in data complexity. This coincides with the complexity of stratified constraints over plain data graphs [5], showing that the addition of a DL-Lite$_{\mathcal{R}}$ ontology does not increase worst-case data complexity. In combined complexity, in contrast, we see an increase to ExpTime-completeness. This is somewhat surprising since standard reasoning in DL-Lite$_{\mathcal{R}}$ ontologies and validation of stratified SHACL constraints over plain data graphs on its own are both tractable in combined complexity.

○ Our upper bounds on complexity follow from a *constraint rewriting technique* that we introduce in [1]. This procedure takes as input an ontology $\mathcal{T}$ and a set $\mathcal{C}$ of stratified constraints, and outputs an updated set $\mathcal{C}_{\mathcal{T}}$ of stratified constraints such that $\mathcal{C}_{\mathcal{T}}$ and $(\mathcal{T}, \mathcal{C})$ behave the same on any input data graph. Thus to perform validation in the presence of ontologies, we do not need to complete the data graph and we can reuse standard SHACL validators over the original graph. This technique joins the ranks of other rewriting-based methods for reasoning with infinite structures (see, e.g., [7]). Previous work [8] introduced a rewriting technique for *positive* SHACL, but we address SHACL with negation, which is *non-monotonic* and significantly more challenging.[1]

[1]The impossibility of a rewriting for SHACL with negation given in Theorem 1 of [8] does not apply to our semantics, nor to their own minimal-model semantics, as acknowledged by the authors in personal communication.

2. Semantics of SHACL Validation with Ontologies

In this section, we describe in more detail the semantics we propose in [1]. For a given ontology $\mathcal{T}$, data graph $\mathcal{A}$, and a shapes graph $(\mathcal{C}, \mathcal{G})$, we need to define when $(\mathcal{A}, \mathcal{T})$ validates $(\mathcal{C}, \mathcal{G})$. The usual open-world semantics for ontologies would check for validation over all models of $\mathcal{A}$ and $\mathcal{T}$. However, this only works for positive constraints, as the following example shows.

Let $\mathcal{A} = \{hasWingedPet(linda, blu), Bird(blu)\}$, and $\mathcal{T} = \{\}$. Let $(\mathcal{C}, \{s(linda)\})$ be a shapes graph, where $\mathcal{C}$ only contains the constraint $s \leftarrow \exists hasWingedPet \land \neg\exists hasPet.Dog$. We have an empty ontology, so we can use the usual setting for SHACL validation to conclude that $\mathcal{A}$ indeed validates $(\mathcal{C}, \{s(linda)\})$, as $linda$ has a winged pet and not a pet that is a dog. However, if we consider all possible models of $\mathcal{A}$ and $\mathcal{T}$, we must conclude non-validation since there is a model of $\mathcal{A}$ and $\mathcal{T}$ that includes $hasPet(linda, b)$, $Dog(b)$, for some b. The problem we encounter here is the *non-monotonicity* of SHACL, that is, adding facts to the data may cause a previously valid setting to become invalid.

We want an intuitive semantics that coincides with the usual validation in case the ontology is empty. As done in related settings (see e.g., [9, 10]) we rely on the *chase* procedure [11] known from Knowledge Representation and Database Theory. Roughly speaking, a chase procedure takes as input a data graph and an ontology and iteratively applies the axioms of the ontology to the data by adding atoms over possibly fresh individuals until all the axioms in the ontology are satisfied. The result of the chase is a so-called *canonical* or *universal* model, and can be used as a representative of all the models. For DL-Lite$_\mathcal{R}$ ontologies, such chase procedures may not terminate and result in infinite models. There are several chase variants producing different canonical models [11]. While for positive constraints these differences do not matter, constraints with negation can distinguish between them, resulting in different validation answers, as illustrated in the example below.

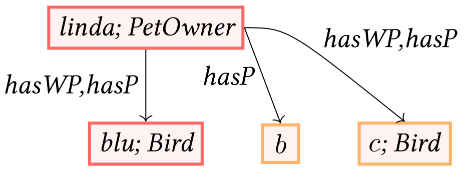
The semantics we propose in [1] is based on a special chase procedure that constructs an *austere* canonical model. The main ingredient is an auxiliary notion of a *good successor configuration*, which, for each object and its *type*, determines a set of successors that allows us to satisfy the axioms with as few fresh objects as possible while preserving the universality of the model. Our notion of austere canonical model is closely related to the *core* chase [11]. It will typically create fewer fresh successors than the *oblivious* chase, which, roughly speaking, applies the axioms of the ontology without first checking whether the axiom is already satisfied. It may also create fewer successors than the *restricted chase*, which may be sensitive to the order of rule applications.

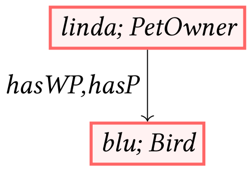
To illustrate the austere canonical model construction, consider the data graph $\mathcal{A}$ we introduced above, but now with $\mathcal{T}$ containing four axioms:

$$\begin{array}{ll} PetOwner \sqsubseteq \exists hasPet & hasWingedPet \sqsubseteq hasPet \\ PetOwner \sqsubseteq \exists hasWingedPet & \exists hasWingetPet^{-} \sqsubseteq Bird \end{array}$$

The good successor configuration will not generate a fresh successor for $linda$, since she has $blue$ as a winged pet, but also as a pet due to the second axiom. The austere canonical model (figure on the right) will only add a $hasPet$-role from $linda$ to $blue$. In contrast, the canonical model obtained from the oblivious chase (figure on the left) will introduce two fresh objects b,

c, to satisfy the two existential axioms. Note that this larger model is nevertheless *minimal* in the sense of [10, 8]. In the figure, we use $hasP$ ($hasWP$) instead of $hasPet$ ($hasWingedPet$).





Now, consider the shapes graph $(\mathcal{C}, \mathcal{G})$ with $\mathcal{C} = \{s \leftarrow \exists hasPet.\neg Bird\}$ and $\mathcal{G} = \{s(linda)\}$. The shapes graph asks to validate whether $linda$ has a pet that is not a bird. Clearly, the austere canonical model provides the expected answer, as it does not validate $(\mathcal{C}, \mathcal{G})$. In contrast, the canonical model on the left-hand-side of the figure—and thus the semantics of [10] adopted for SHACL in [8]—results in the unintended validation of $(\mathcal{C}, \mathcal{G})$.

3. Validation via Rewriting

We define a class of *stratified constraints* in SHACL that allows for both recursion and negation, but restricts their interaction. Analogously as in logic programming [6], we define the *shape dependency graph* for a set of constraints $\mathcal{C}$, and define $\mathcal{C}$ to be stratified if this graph does not have cycles involving negation [1].

For such stratified constraints, we propose a *rewriting technique* for validation in the presence of *DL-Lite*$_\mathcal{R}$ontologies. Given an ontology $\mathcal{T}$ and a set of stratified constraints $\mathcal{C}$, we compile $\mathcal{T}$ and $\mathcal{C}$ into a new set $\mathcal{C}_\mathcal{T}$ of stratified constraints so that for every data graph $\mathcal{A}$ that is consistent with $\mathcal{T}$, and every target $\mathcal{G}$, we have that $(\mathcal{A}, \mathcal{T})$ validates $(\mathcal{C}, \mathcal{G})$ iff $\mathcal{A}$ validates $(\mathcal{C}_\mathcal{T}, \mathcal{G})$. This is achieved by means of an inference procedure that uses a collection of inference rules to capture the possible "propagation" of shape names in the anonymous part of the canonical model. We define in [1] a procedure for positive SHACL, and then show how this can be extended into an algorithm for SHACL with stratified negation.

To illustrate the core ideas of the constraint rewriting algorithm, consider $\mathcal{C}$ containing the following two constraints:

$$birdShape \leftarrow Bird \qquad birdOwnerShape \leftarrow \exists hasPet.birdShape$$

and the same four axioms in $\mathcal{T}$ as above. The obtained set of constraints $\mathcal{C}_\mathcal{T}$ contains all constraints of which the right-hand side suffices to conclude that the structure of the original constraint will appear in an austere canonical model based on the given $\mathcal{T}$. For instance,

$$birdShape \leftarrow \exists hasWingedPet^- \qquad birdOwnerShape \leftarrow \exists hasWingedPet$$

are both contained in $\mathcal{C}_\mathcal{T}$. Note that this rewriting is data-independent: for instance, if $\mathcal{A}' = \{hasWingedPet(linda, blu), hasPet(john, ace)\}$, we can directly conclude that $(\mathcal{A}', \mathcal{T})$ validates $(\mathcal{C}, \{birdOwnerShape(linda), birdShape(blu)\}$, but not $(\mathcal{C}, \{birdOwnerShape(john), birdShape(ace)\}$, as only knowing that someone has a pet does not suffice to get a pet owner shape under this $\mathcal{T}$.

The rewriting algorithm allows us to obtain an ExpTime upper bound on the combined complexity of validation. We show in [1] that this bound is tight.

4. Outlook

There are several directions for future work. Our rewriting algorithm considers a restricted SHACL fragment, and we plan to extend it to support more syntactic features like *complex path expressions* and *counting* (number restrictions on paths). We believe the mentioned features can be incorporated in principle, but it requires a substantial extension. Another direction is to support ontology languages that go beyond OWL 2 QL. Our approach seems to generalize nicely to ontologies in $Horn\text{-}\mathcal{SHIQ}$, but non-Horn ontology languages appear more challenging. An implementation of our approach also remains for future work. The rewriting algorithm was meant to demonstrate the principle feasibility of the approach. Our rewriting is best-case exponential; in particular, there is a rule (rule 3 in Definition 5.3 of [1]) that forces us to add exponentially many new constraints. A way to avoid this problem will be needed in order to achieve an efficient implementation of the rewriting. Extending the SHACL fragment to consider *unstratified* negation is also an interesting direction for future work.

Acknowledgments

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

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

References

[1] Reconciling SHACL and ontologies: Semantics and validation via rewriting, 2023. ECAI, to appear.
[2] W3C, Shape constraint language (SHACL), Technical Report. (2017). https://www.w3.org/TR/shacl.
[3] J. Corman, J. L. Reutter, O. Savkovic, Semantics and validation of recursive SHACL, in: Proc. ISWC 2018, volume 11136 of *LNCS*, Springer, 2018, pp. 318–336.
[4] P. Pareti, G. Konstantinidis, F. Mogavero, Satisfiability and containment of recursive SHACL, J. Web Semantics 74 (2022).
[5] M. Andresel, J. Corman, M. Ortiz, J. L. Reutter, O. Savkovic, M. Šimkus, Stable model semantics for recursive SHACL, in: The Web Conference 2020, 2020.
[6] K. R. Apt, H. A. Blair, A. Walker, Towards a Theory of Declarative Knowledge, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988, p. 89–148.
[7] T. Eiter, M. Ortiz, M. Simkus, T. Tran, G. Xiao, Query rewriting for Horn-SHIQ plus rules, in: Proc. AAAI 2012, AAAI Press, 2012.
[8] O. Savkovic, E. Kharlamov, S. Lamparter, Validation of SHACL constraints over KGs with OWL 2 QL ontologies via rewriting, in: Proc. ESWC 2019, 2019.
[9] S. Borgwardt, W. Forkel, Closed-world semantics for conjunctive queries with negation over ELH-bottom ontologies, in: S. Kraus (Ed.), Proc. IJCAI 2019, 2019.
[10] B. Motik, I. Horrocks, U. Sattler, Bridging the gap between OWL and relational databases, in: Proc. WWW 2007, 2007.
[11] A. Deutsch, A. Nash, J. B. Remmel, The chase revisited, in: Proc. PODS 2008, ACM, 2008, pp. 149–158.