Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Reasoning about Explanations for Non-validation in SHACL

Shqiponja Ahmetaj[2*], **Robert David**[4], **Magdalena Ortiz**[1], **Axel Polleres**[2,3], **Bojken Shehu**[5], **Mantas Šimkus**[1]
[1]Technical University of Vienna, Austria
[2]Vienna University of Economics and Business, Austria
[3]Complexity Science Hub Vienna, Austria
[4]Semantic Web Company, Austria
[5]Polytechnic University of Tirana, Albania
{shqiponja.ahmetaj, axel.polleres}@wu.ac.at, robert.david@semantic-web.com, ortiz@kr.tuwien.ac.at, bshehu@fti.edu.al, simkus@dbai.tuwien.ac.at

Abstract

The Shapes Constraint Language (SHACL) is a recently standardized language for describing and validating constraints over RDF graphs. The SHACL specification describes 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. Specifically, explaining the reasons why the input graph does not satisfy the constraints is challenging. In fact, the current SHACL standard leaves it open on how such explanations can be provided to the users. In this paper, inspired by works on logic-based abduction and database repairs, we study the problem of explaining non-validation of SHACL constraints. In particular, 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. We define a collection of decision problems for reasoning about explanations, possibly restricting to explanations that are minimal with respect to cardinality or set inclusion. We provide a detailed characterization of the computational complexity of those reasoning tasks, including the combined and the data complexity.

1 Introduction

The SHApe Constraint Language (SHACL) is a recently standardized language for expressing constraints on RDF graphs. It is the result of industrial and academic efforts to provide solutions for checking the quality of RDF graphs and for declaratively describing (parts of) their structure. We recommend (Gayo et al. 2017) for an introduction to SHACL and it close relative *ShEx*. The SHACL standard provides a syntax for writing down constraints, and describes the way RDF graphs should be *validated* against such constraints. However, some aspects of validation are not completely specified in the standard, like the semantics of validation for constraints with cyclic dependencies. To address these shortcomings, several formalizations have emerged recently, which describe SHACL in terms of logic-based languages with clear semantics, like first-order logic (Corman, Reutter, and Savkovic 2018) and logic programming (Andresel et al. 2020). Connections between SHACL and Description Logics (DLs), which underlie the OWL standard for writing ontologies, have also been established (Leinberger et al. 2020). The key difference between SHACL and DLs is that SHACL makes the *closed-world assumption (CWA)*, while DLs use the *open-world assumption (OWA)*. To understand the difference, one can think of RDF graphs equipped with SHACL constraints as DL knowledge bases in which all roles and some concept names are treated as *closed predicates* (see (Franconi, Ibáñez-García, and Seylan 2011; Lutz, Seylan, and Wolter 2013; Ngo, Ortiz, and Simkus 2016)).

In SHACL, the basic computational problem is to check whether a given RDF graph G *validates* a SHACL document (C, T), where C is a specification of validation rules (*constraints*) and T is a specification of nodes to which the validation rules should apply (*targets*). In order to make SHACL truly useful and widely accepted, we need automated tools that implement not only validation, which results in "yes" or "no" answers, but also support the users in their efforts to understand the reasons *why* a given graph validates or not against a given document. The SHACL specification stresses the importance of explaining validation outcomes and introduces the notion of *validation reports* for this purpose. If an input graph does validate against a document, the standard has clear guidance how the validation reports should look like. However, the situation is different when the graph does not validate. The principles of validation reports in case of non-validation are left largely open in the standard, which specifies little beyond requiring that the node and constraint violated are indicated. It is not hard to see that, in general, there may be a very large number of possible reasons for a specific validation target to fail, and it is far from obvious what should be presented to the user in validation reports. This is precisely the topic of our study.

In this paper, we advocate explanations in the style of *database repairs* (Arenas, Bertossi, and Chomicki 1999) as one concrete way to provide explanations for the non-validation of SHACL constraints. This approach is closely related to subareas of KR&R like abductive reasoning, model-based diagnosis and counterfactuals, which have received very significant attention in last decades and ap-

[*]Work done while at the Technical University of Vienna.

plied to a range of similar problems requiring explanatory services (see, e.g., (Eiter and Gottlob 1995; Van Harmelen, Lifschitz, and Porter 2008; Calvanese et al. 2013; Ceylan et al. 2020)).

The main goal of this paper is to formalize the notion of explanations for SHACL, to define a collection of reasoning tasks for exploring explanations, and to characterize their computational complexity. In a nutshell, the contributions of this paper are as follows:

∘ To explain non-validation of a SHACL document (C, T) by an RDF graph G, we introduce the notion of a *SHACL Explanation Problem (SEP)*. A solution to a SEP is a pair (A, D) that describes a collection A of facts to be added to G and a collection D of facts to be deleted from G, so that the resulting graph does validate the document (C, T). We consider natural preference orders over explanations, and study also explanations that are minimal w.r.t. set inclusion or w.r.t. cardinality. We illustrate the use of explanations with some examples.

∘ We define a collection of inference services for reasoning about explanations for non-validation. We start with the basic tasks of recognizing whether a given candidate is indeed a (preferred) explanation, and deciding whether a (preferred) explanation exists. We also define the problems of checking whether a given atom is relevant (resp., necessary) as an addition or as deletion in some explanation (resp., all explanations). These task are reminiscent of basic reasoning problems in logic-based abduction (Eiter and Gottlob 1995).

∘ We study the computational complexity of the introduced reasoning tasks, characterizing both combined and data complexity. Our results range from tractability to completeness for the second level of the polynomial hierarchy.

∘ After studying the general setting, we turn our attention to *non-recursive* SHACL constraints. We show that with one exception, reasoning about explanations in the presence of non-recursive constraints does not become easier in terms of computational complexity. The exception is the problem of recognizing an explanation, which becomes tractable in the absence of a preference order. As a side result we show that SHACL validation in the presence of non-recursive constraints is P-complete.

∘ Finally we consider a generalization of SEPs with *restricted explanation signatures*. This useful feature allows, e.g., to specify that some classes and properties are *read-only*, prohibiting deletions from them during explanations. Also for this setting we establish a collection of complexity results, including the case of non-recursive constraints.

2 Preliminaries

We introduce here SHACL and the notion of *validation* by RDF graphs, the most important task in SHACL. We follow the formalization from (Andresel et al. 2020), which abstracts away from the concrete syntax of RDF and ignores concrete domains, IRIs, literals, and blank nodes.

Data Graphs. SHACL uses the term "data graph" for an RDF graph that is to be validated against some constraints. Let $\mathbf{N}$, $\mathbf{C}$, and $\mathbf{P}$ denote countably infinite, mutually disjoint sets of *nodes*, *class names*, and *property names*, respectively. A *data graph* G is a finite set of *atoms* of the form $B(c)$ and $p(c, d)$, where $B \in \mathbf{C}$, $p \in \mathbf{P}$, and $c, d \in \mathbf{N}$. The set of nodes appearing in G is denoted with $V(G)$.

Syntax of SHACL We assume a countably infinite set $\mathbf{S}$ of *shape names*, disjoint from $\mathbf{N} \cup \mathbf{C} \cup \mathbf{P}$. A *shape atom* is an expression of the form $s(a)$, where $s \in \mathbf{S}$ and $a \in \mathbf{N}$. A *path expression* E is a regular expression built using the usual operators $*, \cdot, \cup$ from symbols in $\mathbf{P}^+ = \mathbf{P} \cup \{p^- \mid p \in \mathbf{P}\}$. If $p \in \mathbf{P}$, then p^- is the *inverse property* of p. A *(complex) shape* is an expression ϕ obeying the syntax:

$$\phi, \phi' ::= \top \mid s \mid B \mid c \mid \phi \wedge \phi' \mid \neg\phi \mid \geq_n E.\phi \mid E = E',$$

where $s \in \mathbf{S}$, $B \in \mathbf{C}$, $c \in \mathbf{N}$, n is a positive integer, and E, E' are path expressions.

In what follows, we write $\phi \vee \phi'$ instead of $\neg(\neg\phi \wedge \neg\phi')$, $\exists E.\phi$ instead of $\geq_1 E.\phi$, and $\forall E.\phi$ instead of $\neg \geq_1 E.\neg\phi$.

A *(shape) constraint* is an expression $s \leftrightarrow \phi$ where $s \in \mathbf{S}$ and ϕ is a possibly complex shape. For example, the constraint $\mathsf{FamousSinger} \leftrightarrow \exists\mathsf{singerOf}.\mathsf{FamousSong}$ specifies that a node c validates the shape name FamousSinger exactly when c is the singer of at least one famous song.

In SHACL, *targets* are used to prescribe that certain nodes of the input data graph should validate certain shapes. W.l.o.g. we consider targets to be shape atoms of the form $s(a)$, where $s \in \mathbf{S}$ and $a \in \mathbf{N}$, stating that the shape name s must be validated at the node a of the input data graph. The SHACL specification allows for a richer specification of targets, e.g., to state that all nodes of a certain class must validate a certain shape name, but these do not affect our results (see Section 6 for a discussion).

SHACL also specifies the notion of *shape document*, which is simply a pair (C, T), where *(i)* C is a set of constraints such that each shape name appearing in C occurs exactly once on the left-hand side of a constraint in C, and *(ii)* T is a set of targets.

Evaluation of complex shapes The evaluation of a shape expression ϕ is given by assigning nodes of the data graph to (possibly multiple) shape names. More formally, a *(shape) assignment* for a data graph G is a set $I = G \cup L$, where L is a set of shape atoms such that $a \in V(G)$ for each $s(a) \in L$. The evaluation of a (complex) shape w.r.t. an assignment I is given in terms of a function $[\![\cdot]\!]^I$ that maps a (complex) shape expression ϕ to a sets of nodes, and a path expression E to a set of pairs of nodes. We refer to Table 1 for details on the evaluation of the various operators in complex shapes.

Validation In this paper, for validation we consider the semantics proposed in (Corman, Reutter, and Savkovic 2018), the first work to formalize the semantics of validation for SHACL in a logic-based language. Assume a SHACL document (C, T) and a data graph G such that each node that appears in C or T also appears in G. Then, an assignment I for G is a *(supported) model* of C if $[\![\phi]\!]^I = s^I$ for all $s \leftrightarrow \phi \in C$. The data graph G *validates* (C, T) if there exists an assignment $I = G \cup L$ for G such that (i) I is a model of C, and (ii) $T \subseteq L$. The VALIDATION problem consists

$$[\![\top]\!]^I = V(I) \qquad [\![c]\!]^I = \{c\} \qquad [\![B]\!]^I = \{c \mid B(c) \in I\}$$
$$[\![E \cup E']\!]^I = [\![E]\!]^I \cup [\![E']\!]^I \quad [\![p]\!]^I = \{(a,b) \mid p(a,b) \in I\}$$
$$[\![E \cdot E']\!]^I = [\![E]\!]^I \circ [\![E']\!]^I \quad [\![p^-]\!]^I = \{(a,b) \mid p(b,a) \in I\}$$
$$[\![E^*]\!]^I = \{(a,a) \mid a \in V(I)\} \cup [\![E]\!]^I \cup [\![E \cdot E]\!]^I \cup \cdots$$
$$[\![s]\!]^I = \{c \mid s(c) \in I\}$$
$$[\![\neg\phi]\!]^I = V(I) \setminus [\![\phi]\!]^I \qquad [\![\phi_1 \wedge \phi_2]\!]^I = [\![\phi_1]\!]^I \cap [\![\phi_2]\!]^I$$
$$[\![\geq_n E.\phi]\!]^I = \{c \mid |\{(c,d) \in [\![E]\!]^I \text{ and } d \in [\![\phi]\!]^I\}| \geq n\}$$
$$[\![E = E']\!]^I = \{c \mid \forall d : (c,d) \in [\![E]\!]^I \text{ iff } (c,d) \in [\![E']\!]^I\}$$

Table 1: Evaluation of complex shapes

in determining whether G *validates* the SHACL document (C, T). It can be solved in non-deterministic polynomial time by guessing an assignment, and then checking that it is a model; this is in fact worst case optimal.

Theorem 1 ((Corman, Reutter, and Savkovic 2018)). VALIDATION *is* NP*-complete in data and combined complexity.*

Here and in the remainder of the paper the combined complexity is measured in the combined size of the input parameters, while data complexity is measured in the size of G only, i.e., we assume the size of the rest of the input parameters is bounded by a constant.

Example 1. *Consider (C, T) and the data graph G:*

$$C = \{\mathsf{Human} \leftrightarrow \mathit{HumanBeing} \vee \exists \mathit{hasRelative}.\mathsf{Human}\}$$
$$T = \{\mathsf{Human}(\mathit{Ann}), \mathsf{Human}(\mathit{Ben})\},$$
$$G = \{\mathit{hasRelative}(\mathit{Ann}, \mathit{Ben}), \mathit{HumanBeing}(\mathit{Ben})\}$$

Intuitively, the single statement in C tells us that a node c validates the shape name Human *iff c belongs to the class $\mathit{HumanBeing}$ or it has a $\mathit{hasRelative}$ connection to a node c' that in turn validates the shape name* Human. *Intuitively, the two atoms in T tell us that our target is to check that Ann and Ben can be consistently assumed to be human. Clearly, G validates (C, T), as witnessed by the assignment $I = G \cup \{\mathsf{Human}(\mathit{Ann}), \mathsf{Human}(\mathit{Ben})\}$.*

We note that the above semantics can be considered as based on classical first-order logic, extended with predicates whose extension is fixed (closed). More specifically, we can observe a connection to terminologies expressed in a Description Logic (DL) extended with closed predicates (Franconi, Ibáñez-García, and Seylan 2011; Lutz, Seylan, and Wolter 2013). If we omit the path equality construct and all complex path expressions except the inverse properties, then (complex) shapes correspond to concept expressions in the DL $\mathcal{ALCOIQ}$, where both shape names and class names are seen as concept names. A data graph or a target set can be directly viewed as an ABox. A constraint set C simply corresponds to a terminology $\mathcal{T}_C$ that contains a definition $s \equiv \phi$ for each $s \leftrightarrow \phi \in C$. Then the validation of a data graph G against (C, T) corresponds to checking the satisfiability of a DL knowledge base $(\mathcal{T}_C, G \cup T)$ where only concept names corresponding to shape names are open, and all other concept names and role names are closed.

3 Explaining Non-Validation in SHACL

In this section, we formalize the idea of using repairs to explain non-validation of a SHACL document by a data graph. Explanations are given in terms of sets of facts that need to be added or removed from the original data graph, so that the resulting data graph validates the document.

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

We illustrate our notion of explanations for SHACL non-validation with an example.

Example 2. *Consider a SEP $\Psi = (G, C, T, H)$, where:*

$$C = \{\mathsf{Teacher} \leftrightarrow \exists \mathit{teaches}.\top,$$
$$\mathsf{Student} \leftrightarrow \exists \mathit{enrolledIn}.\mathit{Course} \wedge \neg\mathsf{Teacher}\}$$
$$T = \{\mathsf{Student}(\mathit{Ben}), \mathsf{Teacher}(\mathit{Ann})\}$$
$$G = \{\mathit{enrolledIn}(\mathit{Ben}, C_1), \mathit{teaches}(\mathit{Ann}, \mathit{Ben}),$$
$$\mathit{teaches}(\mathit{Ben}, \mathit{Ben}), \mathit{teaches}(\mathit{Ann}, \mathit{Peter})\}$$
$$H = \{\mathit{Course}(C_1), \mathit{Course}(C_2)\}$$

The constraints state that each Teacher *must teach someone, and each* Student *must be enrolled in some course and must not comply with the shape* Teacher. *Note that* Teacher *and* Student *are shape names, $\mathit{enrolledIn}$ is a property name, and Course is a class name. The data graph G validates $(C, \{\mathsf{Teacher}(\mathit{Ann})\})$, but does not validate (C, T). A possible explanation for non-validation is that G is missing the fact that C_1 is a Course, and moreover, it contains the possibly erroneous fact that $\mathit{teaches}(\mathit{Ben}, \mathit{Ben})$. Thus, validation is ensured by repairing G with the explanation (A, D), where $A = \{\mathit{Course}(C_1)\}$ and $D = \{\mathit{teaches}(\mathit{Ben}, \mathit{Ben})\}$.*

Note that the hypothesis set H in Definition 1 allows to introduce during repairs a limited number of nodes not occurring in the input set of constraints and data graph. This is useful in handling constructs such as existential restrictions and negation in constraints, which may sometimes enforce explanations to add atoms over fresh nodes. Furthermore, constructs such as negation or "at most" restrictions may require explanations to delete facts from the graph in order to ensure validation of the targets. Note that we do not make any validity assumptions on the input SEP, i.e. the data graph may already validate the input SHACL document, in which case $(\emptyset, \emptyset)$ would be a valid explanation.

Preferences among Explanations Considering all possible explanations for a SEP may not be desirable as there may be a large number (possibly exponentially many) of different explanations, and redundant facts that are not relevant for regaining validity may be present in those explanation. For instance, if the given set of hypothesis is rich enough, explanations may delete the entire input data graph and construct another one from the hypothesis atoms validating the SHACL document. To focus only on a subset of desirable solutions, we consider *preference relations* over explanations, given by a pre-order $\preceq$, that is, a reflexive and transitive relation on the set of explanations. We consider two kinds of preferred explanations under two typical preference orders: *subset-minimal* ($\subseteq$), and *cardinality-minimal* ($\leq$) explanations.

Definition 2. *Let Ψ be a SEP, and let $(A, D), (A', D')$ be two explanations for Ψ. We write $(A, D) \subseteq (A', D')$ if $A' \subseteq A$ and $D' \subseteq D$, and we write $(A, D) \leq (A', D')$ if $|A| + |D| \leq |A'| + |D'|$.*

A preferred explanation of a SEP Ψ under the pre-order $\preceq$, called $\preceq$-explanation, is an explanation ξ such that there is no explanation ξ' for Ψ with $\xi' \preceq \xi$ and $\xi \not\preceq \xi'$.

Observe that cardinality-minimal explanations are also subset-minimal, but the converse does not hold in general.

Decision Problems We next define the main decision problems for explanations, in the style of logic-based abduction (Eiter and Gottlob 1995; Calvanese et al. 2013).

Definition 3. *Let $\Psi = (G, C, T, H)$ be a SEP, let A, D be data graphs, let α be an atom in $G \cup H$, and let $\preceq$ be a (possibly empty) preorder. We define six decision problems:*

- $\preceq$-IsExpl*: is (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 omit $\preceq$ from the name of decision problems when $\preceq$ is empty, that is, there is no preference order, and write $(\preceq)$ when considering the variants with and without $\preceq$. In what follows we use $\preceq$ as a place holder for both $\subseteq$ and $\leq$.

IsExpl is sometimes called the *recognition problem*, which checks whether a given candidate explanation is indeed an explanation for the input SEP. *Existence* of an explanation, Exist, is a classical problem with a "yes" answer if there exists an explanation for the input SEP and "no" otherwise. For *necessity* and *relevance* of an atom, i.e. whether it occurs in *all* or *some* explanations, we separated the problems for additions and deletions. Testing whether an atom is a necessary addition or deletion provides valuable insights to the user that the particular atom is immediately related to the non-validation of the target shape atoms. That an atom appears in all explanations as a necessary deletion from the input data graph may also mean that it clashes with the constraints preventing the validation of any possible target set.

Note that an explanation $\xi = (A, D)$ provides the *symmetric difference* $A \cup D$ between a data graph G and the result $G' = (G \setminus D) \cup A$ of applying ξ to G, and that the application of a $\subseteq$-explanation to a data graph G results in a *repair* G' of G, as known from the seminal paper on repairs for databases (Arenas, Bertossi, and Chomicki 1999).

We present some examples to illustrate necessary additions and deletions as part of $\preceq$-explanations.

Example 3. *Recall the explanation (A, D) from Example 2, and observe that for any other explanation (A', D') for Ψ it must be the case that $A \subseteq A'$ and $D \subseteq D'$. It follows that $Course(C_1)$ is a $\preceq$-*necessary *addition for Ψ and the atom $teaches(Ben, Ben)$ is a $\preceq$-*necessary *deletion for Ψ. If we change the hypothesis set to $H = \{Course(C_1), enrolledIn(Ben, E), Course(E)\}$, then $Course(C_1)$ is not a $\subseteq$-*necessary *addition for Ψ because we introduce a new explanation that adds $A' = \{enrolledIn(Ben, E), Course(E)\}$, avoiding the addition of $Course(C_1)$. Note that even with the new H, the atom $Course(C_1)$ remains a $\leq$-*necessary *addition and $teaches(Ben, Ben)$ is still a $\preceq$-*necessary *deletion.*

We now illustrate relevant additions and deletions.

Example 4. *Consider the SEP Ψ from Example 2, which we modify to the SEP $\Psi' = (G', C', T, H)$, where C' is obtained from C by substituting the first constraint with* Teacher $\leftrightarrow \exists teaches.Person$*, and letting $G' = G \cup \{Person(Ben), enrolledIn(Ben, C_1)\}$. There are four $\leq$-explanations (A_i, D_i) with $i \in [1, 4]$ for Ψ':*

$$A_1 = \{Course(C_1)\}, D_1 = \{teaches(Ben, Ben)\}.$$
$$A_2 = \{Course(C_1)\}, D_2 = \{Person(Ben)\}.$$
$$A_3 = \{Course(C_2)\}, D_3 = \{teaches(Ben, Ben)\}.$$
$$A_4 = \{Course(C_2)\}, D_4 = \{Person(Ben)\}.$$

*The atoms $Course(C_1)$ and $Course(C_2)$ are both $\leq$-relevant additions for Ψ'. Note that $Course(C_1)$ is not anymore a $\leq$-necessary addition for Ψ' since there are $\leq$-explanations where $Course(C_1)$ does not appear, e.g., (A_3, D_3). Note also that $teaches(Ben, Ben)$ is not anymore a $\leq$-*necessary *deletion for Ψ', since there are $\leq$-explanations where $teaches(Ben, Ben)$ does not appear, e.g., (A_2, D_2). The atoms $teaches(Ben, Ben)$ and $Person(Ben)$ are both $\leq$-relevant deletions for Ψ'. Note that the statements here hold also if we replace "$\leq$" by "$\subseteq$".*

4 Complexity of Decision Problems

We now investigate the combined and data complexity of the decision problems for SEPs presented in Definition 3. We start by establishing some relations between these problems.

Proposition 1. *A SEP has an explanation iff it has a $\subseteq$-minimal explanation iff it has a $\leq$-minimal explanation.*

The following proposition also allows us to transfer complexity bounds between decision problems:

Proposition 2. *Assume a complexity class $\mathcal{C}$. The following are true for both combined and data complexity:*

1. NecAdd *is $\mathcal{C}$-complete iff* NecDel *is $\mathcal{C}$-complete iff* Exist *is co$\mathcal{C}$-complete.*
2. $\subseteq$-NecAdd *is $\mathcal{C}$-complete iff* NecAdd *is $\mathcal{C}$-complete iff* $\subseteq$-NecDel *is $\mathcal{C}$-complete iff* NecDel *is $\mathcal{C}$-complete.*

3. RELADD *is $\mathcal{C}$-complete iff* RELDEL *is $\mathcal{C}$-complete iff* EXIST *is $\mathcal{C}$-complete.*

We can show that rather simple reductions between the problems exist. For example, to reduce NECADD to the co-problem of EXIST, assume a SEP $\Psi = (G, C, T, H)$ and atom α that is a necessary addition for Ψ. Here, we show the reduction for the case where α has the form $B(a)$. We construct a SEP $\Psi' = (G, C', T', H')$ that forces $B(a)$ to not be added, by setting $C' = C \cup \{s \leftrightarrow B' \wedge \neg B\}$, $T' = T \cup \{s(a)\}$, and $H' = H \cup \{B'(a)\}$, for a fresh shape name s and fresh class name B' not occurring in Ψ. Then, $B(a)$ is a necessary add for Ψ iff Ψ' has no explanation. The converse reduction is also easy. Let $\Psi = (G, C, T, H)$ be a SEP. Then Ψ does not admit an explanation iff α is a necessary add for $\Psi' = (G, C, T, H')$, where $H' = H \cup \{\alpha\}$ and α is an arbitrary fresh atom.

Note that for the data complexity results, we need reductions that map the static part of the input instance to a static part of the target instance in a data independent way.

4.1 Complexity Results

We establish the data and combined complexity of the problems. Since the bounds we obtain for both complexity measures coincide, we only need to show the upper bound for combined and the lower bound for data complexity.

We start with the problem of recognizing explanations.

Theorem 2. *For both data and combined complexity:*

1. ISEXPL *is* NP*-complete, and*
2. $\subseteq$-ISEXPL *and* $\leq$-ISEXPL *are* DP*-complete.*

Sketch. For the upper bound of ISEXPL, to recognize whether (A, D) is an explanation for $\Psi = (G, C, T, H)$, we need to check that: 1) $A \subseteq H$, and $D \subseteq G$, which can be done in linear time in the size of G and H and 2) that the data graph resulting from applying (A, D) to G validates (C, T), that is, a validity check feasible in NP. NP-hardness in data complexity holds by Theorem 1 since VALIDATION is a special case of ISEXPL where the candidate pair is $(\emptyset, \emptyset)$.

The DP membership with preferences is inferred from the following simple algorithm, which checks that the given (A, D): (1) is indeed an explanation for Ψ, and (2) is ($\subseteq$ or $\leq$) minimal. Step (1) corresponds to solving ISEXPL, and (2) amounts to guessing a (subset or cardinality) smaller candidate and solving the co-problem ISEXPL. The guess is polynomial since it is bounded by the size of G and H.

It is left to argue the DP-hardness in data complexity; we argue it for $\subseteq$, but the proof works also for $\leq$-minimal explanations. We pick an NP-hard problem Q_1 and a coNP-hard problem Q_2. For Q_1 we take VALIDATION for a given document (C_1, T_1). For Q_2 we take the co-problem of 3-colorability NON3COL, that is, to decide whether a given graph $G_c = (V, E)$ is not colorable with 3 colors (green, blue, and red). Consider an arbitrary instance of Q_1, consisting of G_1, and an arbitrary instance $G_2 = (V, E)$ of Q_2. We define a SHACL document (C, T) such that for every instance G_1 of VALIDATION over a fixed (C_1, T_1) and every instance G_2 of NON3COL, we can construct in polynomial time a data graph G, a set of hypotheses H, and a tuple (A, D), such that (A, D) is an explanation for $\Psi = (G, C, T, H)$ iff G_1 is a positive instance of VALIDATION and G_2 of NON3COL.

We define G as $G_1 \cup G_{col}$, where G_{col} encodes G_2 and contains the atoms: (1) $edge(a, b)$ for each $(a, b) \in E$, and (2) $r(w, a)$ for each $a \in V$, where w is a fresh node, and r is a fresh property name. W.l.o.g. we assume $(b, a) \in E$ for each $(a, b) \in E$. The set C is defined as $C_1 \cup C_{col}$, where C_{col} contains the constraints:

$$\mathsf{red} \leftrightarrow \exists r^- \wedge \neg\mathsf{green} \wedge \neg\mathsf{blue} \wedge \neg\exists edge.\mathsf{red} \tag{1}$$

$$\mathsf{green} \leftrightarrow \exists r^- \wedge \neg\mathsf{red} \wedge \neg\mathsf{blue} \wedge \neg\exists edge.\mathsf{green} \tag{2}$$

$$\mathsf{blue} \leftrightarrow \exists r^- \wedge \neg\mathsf{green} \wedge \neg\mathsf{red} \wedge \neg\exists edge.\mathsf{blue} \tag{3}$$

$$\mathsf{colored} \leftrightarrow \forall r.(\mathsf{red} \vee \mathsf{blue} \vee \mathsf{green}) \tag{4}$$

$$\mathsf{valid} \leftrightarrow \mathsf{colored} \vee B \tag{5}$$

We use here a fresh class name B, the rest are shape names. We define $T = T_1 \cup \{\mathsf{valid}(w)\}$ and $H = \{B(w)\}$. Finally, we define the tuple (A, D), where $A = \{B(w)\}$ and $D = \emptyset$. We can prove that G_2 is not 3-colorable and G_1 validates (C_1, T_1) iff $(A, \emptyset)$ is a $\subseteq$-explanation for Ψ. ☐

We now consider the existence of an explanation.

Theorem 3. $\subseteq$-EXIST*,* $\leq$-EXIST*, and* EXIST *are* NP*-complete in data and combined complexity.*

Sketch. By Proposition 1, it suffices to show the claim for EXIST. The upper bound is shown by simply modifying the algorithm for VALIDATION: we guess a candidate explanation together with an assignment for the data graph that results from applying it. Then checking that the assignment witnesses validation can be done as usual (Corman, Reutter, and Savkovic 2018).

To show the NP-hardness in data complexity, we reduce from 3-colorability 3COL. Let $G = (V, E)$ be an instance of 3COL. If we take G_{col} as in the proof of Theorem 2 and C'_{col} has the constraints (1) to (4) of C_{col} above, then we obtain that G is 3-colorable iff $\Psi_c = (G_{col}, C'_{col}, \{\mathsf{colored}(w)\}, \emptyset)$ has an explanation of the form $(\emptyset, \emptyset)$. That is, we have a reduction from 3COL to to the existence of *an explanation that does not delete any atoms*. However, Ψ_c may also allow explanations that delete atoms from G_{col}, which amounts to ensuring validation by removing edges from G or leaving vertices uncolored. Note that in fact we can find an explanation $(\emptyset, D)$ for Ψ_c by just taking any singleton $D \subseteq G_{col}$.

To obtain a reduction from 3COL to EXIST we transform Ψ_c into a $\Psi'_c = (G, C, T, \emptyset)$ whose explanations correspond to the explanations of Ψ_c that don't delete atoms from G_{col}. We explain how this is done for the $edge$ atoms, r atoms are treated analogously. We also note that the technique is generic: it shows how to extend any SEP $\Psi = (G, C, T, H)$ into $\Psi = (G', C', T', H)$ with $G \subseteq G'$ and $C \subseteq C'$ and $T \subseteq T'$ in such a way that no elements from a given set $\alpha_1, \ldots \alpha_n$ of binary atoms are deleted in the explanations for Ψ'. A similar (but simpler) trick can be used to forbid the deletion of a given set of unary atoms.

First we use fresh symbols to add to G_{col} atoms that enforce an (arbitrary but fixed) linear order over the non-deletable atoms $edge(a_1, b_1)$, ..., $edge(a_m, b_m)$. We use a fresh node e_i for each atom $edge(a_i, b_i)$, a property name $enext$ and class names Fe and Le, and a property aux. Let G_{ord} be the set G' of atoms which contains: a) $enext(e_i, e_{i+1})$ for each $1 \leq i \leq m-1$, b) $aux(e_i, a_i)$, $aux(b_i, e_i)$ for each $edge(a_i, b_i)$ and c) $Fe(e_1)$ and $Le(e_m)$.

Using fresh shape names goode, edgepath and totaledge, we add the following constraints C_{ord}:

$$\begin{aligned} \mathsf{goode} \leftrightarrow & aux = aux^- \cdot edge^- \\ \mathsf{edgepath} \leftrightarrow & \exists enext.(\exists aux \wedge \mathsf{goode} \wedge (\mathsf{edgepath} \vee Le)) \\ \mathsf{totaledge} \leftrightarrow & \exists r.(\exists aux^-.(Fe \wedge \mathsf{goode} \wedge \mathsf{edgepath})) \end{aligned}$$

An assignment for a graph G' containing $\mathsf{goode}(e_i)$ can only be a model of C' if $edge(a_i, b_i)$ is in G', that is, if G' is not the result of applying an explanation that deletes $edge(a_i, b_i)$. Hence, if an assignment containing $\{Fe(e_1), \mathsf{goode}(e_1), \mathsf{edgepath}(e_1)\}$ is a model, then the sequence $Fe(e_1)$, $aux(e_1, a_1)$, $edge(a_1, b_1)$, $aux(b_1, e_1)$, $enext(e_1, e_2)$, $\ldots$, $aux(e_m, a_m)$, $edge(a_m, b_m)$, $aux(b_m, e_m)$, $Le(e_m)$ is in the graph and no edge atom has been deleted. It follows that $\Psi'_c = (G_{col} \cup G_{ord}, C'_{col} \cup C_{ord}, \{\mathsf{colored}(w), \mathsf{totaledge}(w)\}, H)$ has an explanation iff $\Psi_c = (G_{col}, C'_{col}, \{\mathsf{colored}(w)\}, H)$ has an explanation (A, D) with $D \cap \{edge(a_i, b_i), \ldots, edge(a_m, b_m)\} = \emptyset$. □

Now we study the problems of deciding whether an atom must be present in all explanations.

Theorem 4. *The following results are true in both combined and data complexity:*

1. NECADD*,* NECDEL*,* $\subseteq$-NECADD*, and* $\subseteq$-NECDEL *are co*NP*-complete, and*
2. $\leq$-NECADD *and* $\leq$-NECDEL *are* $P_{\|}^{\mathrm{NP}}$*-complete.*

Sketch. Item 1 follows from Proposition 1, Proposition 2 and Theorem 3. For $\leq$-NECADD and $\leq$-NECDEL, we show the upper bound in combined complexity and the lower bound in data complexity.

For the $P_{\|}^{\mathrm{NP}}$ membership of both problems, we build on ideas from (Calvanese et al. 2013). We show here the upper bound for $\leq$-NECADD; the arguments for $\leq$-NECDEL are analogous. Consider a SEP $\Psi = (G, C, T, H)$ and an atom $\alpha \in H$ (for $\leq$-NECDEL α must be in D); let $m = |G \cup H|$. Note that α is not a $\leq$-necessary addition for Ψ iff there is an $i \in [1, m]$ such that (a) Ψ has an explanation (A, D) of size i where $\alpha \notin A$, and (b) (A, D) is cardinality-minimal. We use two auxiliary problems:

- SIZEEPXL: given a SEP Γ, an atom α', and an integer n', decide whether there is an explanation (A, D) for Ψ' with $\alpha' \notin A$ and $|A| + |D| = n'$.
- NOSMALLER: given a SEP Ψ and an integer n', decide whether there is no explanation (A', D') for Ψ' such that $|A'| + |D'| < n'$.

SIZEEPXL is in NP and NOSMALLER is in coNP. We have that α is added in all $\leq$-explanations of Ψ iff for all $i \in [0, m]$, one of the following holds: (i) $\Gamma_i = (\Psi, \alpha, i)$ is a negative instance of SIZEEPXL, or (ii) $\Lambda_i = (\Psi, i)$ is a negative instance of NOSMALLER. One can traverse the set S of all Γ_i, Λ_i in polynomial time and test (i) and (ii) by making no more than $2m$ calls to an NP oracle.

We show the lower bound by a reduction from the $P_{\|}^{\mathrm{NP}}$-complete problem ODDMAX3SAT (Wagner 1987): given a 3-CNF propositional formula φ, decide whether there is an odd integer $k \in [1, |\varphi|]$ (where $|\varphi|$ is the number of clauses in φ) such that there exists a truth assignment that satisfies k clauses, and there is no assignment that satisfies $\ell > k$ clauses. We show the reduction for $\leq$-NECADD.

We construct a SEP $\Psi = (G, C, T, H)$ and atom α, where C, T, and α are fixed, such that φ is a positive instance of ODDMAX3SAT iff α is a $\leq$-necessary add for $\Psi = (G, C, T, H)$. We associate each clause in φ with a node c_i and let nodes $c_1, \ldots, c_n$ denote an order over the clauses. The data graph G contains the atoms $F(c_1)$, $L(c_n)$ storing the first and last clause, two auxiliary atoms $r(w, c_1)$, $r(c_n, w)$, $F(c_1)$, $L(c_n)$, and for each propositional variable p_i in φ, two atoms $N(p_i)$ and $isNeg(\bar{p_i}, p_i)$. The set of hypotheses H contains the atoms:

- $hasLit(c_i, \mathbf{p}_{ij})$ for each clause c_i and each literal $\mathbf{p}_{ij}$ occurring in c_i, where $\mathbf{p}_{ij} = p_{ij}$ if p_{ij} is a propositional variable, and $\mathbf{p}_{ij} = \bar{p}_{ij}$ if $\mathbf{p}_{ij}$ is of the form $\neg p_{ij}$,
- $prevBl(c_i, c_{i+1})$, $prevS(c_i, c_{i+1})$, for each $i \in [1, n-1]$,
- $B_1(c_i)$, $B_2(c_i)$, $B_3(c_i)$ for each $i \in [1, n]$, and
- $odd(w)$.

Intuitively, the atoms in H allow us to guess whether the maximal number of satisfied clauses is odd, and then to guess, for each clause c_i, either:
- an atom $hasLit(c_i, \mathbf{p}_{ij})$ that witnesses the satisfaction of the clause, together with an atom $prevS(c_i, c_{i+1})$ that propagates the satisfaction of c_i to the next clause c_{i+1}, or
- three 'blocking' atoms $B_1(c_i)$, $B_2(c_i)$ and $B_3(c_i)$ and an atom $prevBl(c_i, c_{i+1})$ that propagates this blocking (i.e., non-satisfaction) of c_i to the next clause c_{i+1}.

Importantly, blocking a clause needs more atoms than satisfying it; this will ensure that in cardinality minimal explanations the number of satisfied clauses is maximized.

The fixed set of constraints C contains:

$$\mathsf{True} \leftrightarrow N \wedge \neg\mathsf{False} \qquad \mathsf{False} \leftrightarrow N \wedge \neg\mathsf{True} \tag{6}$$

$$\mathsf{satC} \leftrightarrow \exists hasLit.(\mathsf{True} \vee (\exists isNeg.\mathsf{False})) \tag{7}$$

$$\mathsf{blockC} \leftrightarrow B_1 \wedge B_2 \wedge B_3 \tag{8}$$

$$\begin{aligned} \mathsf{path} \leftrightarrow & (\mathsf{satC} \wedge \exists prevS.(\mathsf{final} \vee \mathsf{path})) \vee \\ & (\mathsf{blockC} \wedge \exists prevBl.(\mathsf{final} \vee \mathsf{path})) \end{aligned} \tag{9}$$

$$\mathsf{Odd} \leftrightarrow \neg\mathsf{Even} \wedge (\exists prevBl^-.\mathsf{Odd} \vee \exists prevS^-.\mathsf{Even}) \tag{10}$$

$$\mathsf{Even} \leftrightarrow \neg\mathsf{Odd} \wedge (F \vee \exists prevBl^-.\mathsf{Even} \vee \exists prevS^-.\mathsf{Odd}) \tag{11}$$

$$\mathsf{evensat} \leftrightarrow (\mathsf{Even} \wedge \mathsf{blockC}) \vee (\mathsf{Odd} \wedge \mathsf{satC}) \tag{12}$$

$$\mathsf{oddsat} \leftrightarrow ((\mathsf{Odd} \wedge \mathsf{blockC}) \vee (\mathsf{Even} \wedge \mathsf{satC})) \wedge \exists r.odd \tag{13}$$

$$\mathsf{final} \leftrightarrow L \wedge (\mathsf{oddsat} \vee \mathsf{evensat}) \tag{14}$$

$$\mathsf{satisfied} \leftrightarrow \exists r.(F \wedge \mathsf{Even} \wedge \mathsf{path}) \tag{15}$$

We let $T = \{\mathsf{satisfied}(w)\}$, and $\alpha = odd(w)$.

Constraint (7) makes sure that the truth values represented in an assignment for G make true the clauses that were selected for satisfaction, and Constraints (7) to (9) propagate forward the (non-)satisfaction of clauses. The shapes Odd and Even are used for storing the parity of the total clauses that have been satisfied up to clause c_i: c_1 is always even (by the target and Constraint (15)), and then $prevS$ enforces the alternation of Even to Odd, while $prevBl$ keeps the same parity (Constraints (10) and (11)). Finally, constraints (12) to (14) take into account the last clause in the parity of the total satisfied clauses, and make sure that it matches the presence or absence of the atom $\alpha = odd(w)$. Overall, the atom $odd(w)$ will always be in an explanation if an odd number k of clauses was made true with a $hasLit$ atom, and this k will be maximal in the $\leq$-minimal ones (which will all be of the form $(A, \emptyset)$, since deleting atoms from G does not help us satisfy φ). $\square$

We finally study the problem of deciding whether an atom is in relevant for some explanation.

Theorem 5. *The following results hold both combined and data complexity,*

1. RELADD *and* RELDEL *are* NP*-complete,*
2. $\subseteq$-RELADD *and* $\subseteq$-RELDEL *are* Σ_2^P*-complete, and*
3. $\leq$-RELADD *and* $\leq$-RELDEL *are* $P_{\|}^{\text{NP}}$*-complete.*

Sketch. By Proposition 2 and Theorem 3 it can be concluded that the complexity of deciding RELADD and RELDEL is NP-complete. For the Σ_2^P upper bound of $\subseteq$-RELADD and $\subseteq$-RELDEL, suppose a SEP $\Psi = (G, C, T, H)$ and an atom $\alpha \in G \cup H$. We give the following algorithm: (1) Guess a pair (A, D) with $\alpha \in A$ or $\alpha \in D$, and guess an assignment I for $(G \setminus D) \cup A$. (2) Check that I is a model of C that includes T. (3) Call a coNP oracle to check that there is no $\subseteq$-smaller explanation (A', D') for Ψ containing α. Steps (1) and (2) can be done in NP and together with the coNP oracle show the Σ_2^P membership. The lower bound is shown by a reduction from the problem of checking whether a variable p is in a $\subseteq$-minimal model of a 3-CNF propositional formula φ, which is known to be Σ_2^P complete (Eiter and Gottlob 1993). In this reduction the formula is represented as atoms in the data graph G (of the form $hasLit(c, p)$, $hasLit(c, \overline{p})$ and $N(p)$ for clauses c and propositional variables p, with other auxiliary atoms), and it is important to make sure that these atoms are not deleted in explanations. This is achieved with the technique described in the proof of Theorem 3.

The $P_{\|}^{\text{NP}}$ upper bound for $\leq$-RELADD and $\leq$-RELDEL can be shown similarly as in the proof of Theorem 4. We modify the problem SIZEEPXL as follows: decide for a Γ consisting of a SEP Ψ', atom α', and an integer n', whether there exists an explanation (A, D) for Ψ' such that $|A| + |D| = n'$ and α' occurs in A for $\leq$-RELADD or in D for $\leq$-RELDEL. Now, α is added or deleted in *some* $\leq$-explanations of Ψ iff for *some* $i \in [0, m]$, one of the following holds: (i) Γ_i is a *positive* instance of SIZEEPXL, and (ii) Λ_i is a *positive* instance of NOSMALLER. For the lower bound, we use the reduction from ODDMAX3SAT in the proof of Theorem 4. In particular, it can be shown that $odd(w)$ is a $\leq$-necessary add for Ψ iff $odd(w)$ is a $\leq$-relevant add for Ψ, thus showing the claim for $\leq$-RELADD. Small modifications are needed for $\leq$-RELDEL. $\square$

5 Further Results

In this section, we discuss some further results for a restricted fragment of SHACL and the setting where some restrictions are imposed on the signature of explanations.

5.1 Non-Recursive SHACL

An important fragment of SHACL, that is fully described in the SHACL specification, is non-recursive SHACL. Assume a set C of constraints. We say a shape name s *directly refers* to a shape name s' in C, if C has a constraint $s \leftrightarrow \phi$ such that s' appears in ϕ. We say s *refers* s' in C, if s directly refers to s' in C, or there exists a shape name s'' such that s refers to s'' in C, and s'' directly refers to s' in C. A set of SHACL constraints C is *non-recursive* if there is no shape name in C that refers to itself.

In this section, we present some preliminary results for SEPs over non-recursive SHACL. We observe that the results for combined complexity do not differ from the recursive case. In particular, the upper bounds can be immediately inferred from the latter. For the lower bounds we need alternative proofs, as the hardness proofs provided for data complexity in Section 4 use recursion and cannot be exploited.

Theorem 6. *SEPs over recursive and non-recursive SHACL have the same combined complexity for* ($\preceq$)-EXIST, ($\preceq$)-NECADD, ($\preceq$)-NECDEL, ($\preceq$)-RELADD, ($\preceq$)-RELDEL.

The reductions for the lower bounds are not difficult. Roughly, for most problems we use an encoding of a 3-CNF propositional formula φ in a set of non-recursive constraints: the propositional variables of φ are viewed as class names and stored in the set of hypothesis, and the data graph will be the empty set. We then reduce the problem of checking whether there exists a model of φ to deciding explanation existence, and the problem of whether an atom p is in a $\subseteq$-minimal model of φ to deciding $\subseteq$-relevance. For $\leq$-relevance and $\leq$-necessity we again use ODDMAX3SAT. The proof follows similar ideas to that of Theorem 4, but now we can use one constraint to perform the necessary checks for each clause of φ.

Our next target is the problem of recognizing an explanation in the presence of non-recursive constraints. To show some of the results, we employ the problem VALIDATION. We first point out the combined complexity of VALIDATION in the presence of non-recursive SHACL constraints. The data complexity of this problem is known to be complete for NLOGSPACE (Corman et al. 2019), but to our knowledge, the combined complexity had not been established.

Theorem 7. VALIDATION *for non-recursive SHACL is* P*-complete in combined complexity.*

Proof (Sketch). A polynomial time algorithm is not difficult: simply compute the extensions of all shape names in a bottom-up fashion (starting from shape names that do not depend on other shape names), and then check whether the targets are contained in the uniquely generated assignment. Note that in polynomial time we can compute the extension of a given shape expression in a given graph, a task that we need to perform only linearly many times to compute the extension of all shape names of a given problem instance.

The lower bound can be shown using a reduction from the problem of checking whether a proposition g belongs to the least model of a propositional Horn logic program P. Assume such a program P and a goal proposition g (g is assumed to occur in P). For every proposition p of P, let $s_p^0, \ldots, s_p^k$ be shape names, where k is the number of propositions in P. Take the graph $G = \{N(a)\}$, and let C be the set of constraints that contains (a) $s_h^{i+1} \leftrightarrow N \wedge s_{b_1}^i \wedge \cdots \wedge s_{b_n}^i$ for all rules $h \leftarrow b_1, \ldots, n_n$ in P and all $0 \leq i < k$, and (b) $s_p^{i+1} \leftrightarrow s_p^i$ for all propositions p in P and all $0 \leq i < k$. Take the target $T = \{s_g^k(a)\}$. It is easy to see that g occurs in the least model of P iff G validates (C, T). □

We now discuss IsExpl, $\subseteq$-IsExpl and $\leq$-IsExpl.

Theorem 8. *The following results are true:*

1. IsExpl *is* P*-complete in combined complexity, and* NLogSpace*-complete in data complexity.*
2. $\subseteq$-IsExpl *and* $\leq$-IsExpl *are co*NP*-complete in both data and combined complexity.*

Proof (Sketch). The upper bounds are obtained using a simple algorithm. Given a candidate explanation (A, D) for a SEP (G, C, T, H), the explanation is applied to G and then we check if the resulting graph G' validates (C, T). Note that applying (A, D) on G is feasible in logarithmic space. From the complexity of Validation, we then obtain the upper bounds in point 1. For point 2, we need to check the non-existence of an explanation that is smaller w.r.t. set or cardinality inclusion, which trivially belongs to coNP.

The lower bounds in point 1 follow from the fact that Validation is a special case of IsExpl. The remaining coNP lower bound can be shown for $\subseteq$-IsExpl and $\leq$-IsExpl already in data complexity. We discuss this only for $\subseteq$-IsExpl. We provide a reduction from the problem of checking whether a set of propositional variables M is a $\subseteq$-minimal model of a 3-CNF formula φ. We create the following instance of $\subseteq$-IsExpl. We start with $\Psi = (G, C, T, H)$. The data graph G contains the following atoms:

- $\mathit{Clause}(c)$ for every clause c in φ, where Clause is a fresh class name.
- $\mathit{hasClause}(\mathit{root}, c)$ for each clause c in φ, where $\mathit{hasClause}$ is a fresh property name,
- $N(p)$, $\mathit{isNeg}(\overline{p}, p)$ for each propositional variable p occurring in φ, where N is a fresh class name,
- $\mathit{hasLit}(c, p)$ for each clause c in φ and each propositional variable p that occurs positively in c, and
- $\mathit{hasLit}(c, \overline{p})$ for every clause c in φ and every literal $\neg p$ that occurs in c.

We let $H = \{\mathit{True}_{prop}(p) \mid p \text{ occurs in } \varphi\}$, where True_{prop} is a fresh class name. The set C contains the following constraints:

$$\begin{aligned}\mathsf{True}_{form} &\leftrightarrow \forall \mathit{hasClause}.\mathsf{True}_{clause}\\ \mathsf{True}_{clause} &\leftrightarrow \exists \mathit{hasLit}.\mathsf{True}_{lit}\\ \mathsf{True}_{lit} &\leftrightarrow \mathit{True}_{prop} \vee \exists \mathit{isNeg}.\neg \mathit{True}_{prop}\end{aligned}$$

Finally, we define $T = \{\mathsf{True}_{form}(\mathit{root})\}$. This finishes the construction of Ψ. We define the explanation (A, D) obtained from M by letting $D = \emptyset$ and $A = \{\mathit{True}_{prop}(p) \mid p \in M\}$. It is not hard to see that (A, D) is a $\subseteq$-explanation for Ψ iff M is a minimal-model of φ. □

We do not know the data complexity of deciding relevance and necessity in the presence of non-recursive constraints. The proofs of their lower bounds in the recursive setting exploit recursive constraints to traverse a linear order. It remains to be seen whether alternative reductions can be found, or whether non-recursiveness leads to a drop in data complexity. In the next section we observe, however, that such a drop does not happen if we restrict the signature of the explanations.

5.2 Explanations over Restricted Signatures

A user may want to guide the explanations by specifying that data about certain class or property names in the data graph is correct and thus should not be affected by repairs (*read-only*). More generally, for a given class or property name a user may want to consider only additions of facts (*add-only*), or only deletions of facts (*delete-only*). As a result of such specification, some undesired explanations may be filtered out. This more general setting is discussed next.

Definition 4 (Σ-SEP)**.** *Let $\Psi = (G, C, T, H)$ be a SEP, let Σ_A, Σ_D be finite sets of class and property names. We call $\Psi_\Sigma = (G, C, T, H, \Sigma_A, \Sigma_D)$ a Σ-SEP. An explanation for Ψ_Σ is an explanation (A, D) for Ψ, where A, D do not use class and property names from Σ_A and Σ_D, respectively.*

We observe that a SEP is a special case of a Σ-SEP, where $\Sigma_A = \Sigma_D = \emptyset$. Therefore, all the lower bounds proved in Sections 4 and 5.1 hold also for Σ-SEPs. The upper bounds can be easily modified to additionally check whether a given explanation conforms with Σ_A and Σ_D.

Theorem 9. *All complexity results for SEPs obtained in the previous sections also hold for Σ-SEPs.*

We observe next that the more general setting of Σ-SEPs can be simulated in the original setting of SEPs. The prohibition of additions via a set Σ_A can be simulated by properly choosing the set H of hypothesis. The more tricky part is to simulate the prohibition of deletions that are given by a set Σ_D. This can be simulated in original SEP in polynomial time, but using recursive constraints. To prohibit removal of atoms over Σ_D in explanations, we can perform similar tricks as in the proof of Theorem 3. In a nutshell, we modify the input data graph by adding fresh atoms that enforce a linear order over the atoms that use signature from Σ_D, and add constraints that enforce the traversal of this linear order.

Lemma 1. *Let Ψ_Σ be a Σ-SEP. Then, there exists a SEP Ψ over recursive SHACL such that (A, D) is an explanation for Ψ iff (A, D) is an explanation for Ψ_Σ.*

We next present our results for the data complexity of reasoning over Σ-SEPs with non-recursive constraints. These results cover most of the problems, including some for which we left open the data complexity without signature restrictions (see the last paragraph of Section 5.1). Those gaps are a consequence of our use of recursive constraints to prohibit the deletion of certain atoms, but this can be done easily—with no need for recursion—if we restrict the deletion signature. For non-recursive Σ-SEPs we only leave open $\leq$-RELADD and $\leq$-RELDEL, for which the hardness proofs of Section 4 use recursion to traverse the clauses of a formula, and we do not know yet whether the absence of recursion may cause a drop in complexity.

Theorem 10. *SEPs over recursive SHACL and Σ-SEPs over non-recursive SHACL have the same data complexity for* ($\preceq$)-EXIST, ($\subseteq$)-NECADD, ($\subseteq$)-NECDEL, ($\subseteq$)-RELADD, *and* ($\subseteq$)-RELDEL.

Proof (Sketch). The upper bounds follow from Theorem 9. We briefly discuss the lower bounds.

To show NP-hardness for EXIST, we reduce the problem of checking whether a 3-CNF formula has a model and use the construction provided in the proof of Theorem 8 with a small modification. For a formula φ, we construct a Σ-SEP $\Psi_\Sigma = (G, C, T, H, \Sigma_D, \emptyset)$, where G, C, T, H are defined as in the proof of Theorem 8. To prohibit deletions from G we set Σ_D to be the set of all class and property names occurring in G. It is easy to see that φ has a model iff Ψ_Σ has an explanation. This shows that EXIST is NP-complete. The result for EXIST together with Proposition 1 and Proposition 2 readily imply results for most of the other problems: (a) $\subseteq$-EXIST and $\leq$-EXIST are NP-complete, (b) RELADD and RELDEL are NP-complete, and (c) NECADD, NECDEL, $\subseteq$-NECADD, and $\subseteq$-NECDEL are coNP-complete. Note that the reductions for Proposition 2 are such that if the input instance is non-recursive then so is the target instance. It is left to argue the results for $\subseteq$-RELADD and $\subseteq$-RELDEL.

The Σ_2^P lower bound for $\subseteq$-RELADD can be shown by reducing the problem of deciding whether an atom p belongs to some $\subseteq$-minimal model of a 3-CNF formula φ. To this end, we can use the same Ψ_Σ described above for EXIST. It is easy to see that p belongs to a $\subseteq$-minimal model of φ iff $\mathit{True}_{prop}(p)$ belongs to some $\subseteq$-minimal explanation for Ψ_Σ. For $\subseteq$-RELDEL, we reduce the problem of checking whether a variable p is not in a subset-maximal model of a formula φ. We construct $\Psi'_\Sigma = (G', C, T, \emptyset, \Sigma_D, \emptyset)$, where $G' = G \cup H$, and G, H, C, T are as in Ψ_Σ. It is easy to see that p is not in a subset-maximal model of φ iff $\mathit{True}_{prop}(p)$ is deleted in a $\subseteq$-explanation for Ψ'_Σ. $\square$

6 Discussion

Richer Targets In this paper, the targets in a SHACL document (C, T) are shape atoms of the form $s(c)$. The SHACL standard also describes validation targets that are specified using class and property names. Concretely, given a graph G, we may be required to validate a shape name s at every node that is an instance of a class B (or, in the domain or in the range of some given property p). All the upper bounds of this paper can be immediately updated to this richer settings. We only remark here that in this setting it is important to recompute the targeted nodes after an update is applied. For an example, consider a graph $G = \{B(c), p(c, d)\}$, a constraint set $C = \{s \leftrightarrow \exists p.B\}$, and let $T = \{B\}$, i.e., we want every instance of B to validate the shape name s. Since c is an instance of B, we may consider adding $B(d)$ to G as a repair so that a validates the shape name s. However, this does not fully repair G, because adding $B(d)$ to G now requires also d to validate the shape name s.

Unrestricted H Recall that a SEP includes a set H of atoms that are allowed to be added to the input graph. The set H of hypothesis may be omitted from the input SEP whenever H is viewed as the set of all atoms over the class names, property names, and nodes that explicitly occur in the input graph G. Another natural option is to assume that such H is simply the set of all possible atoms, which would allow unrestricted introduction of fresh nodes into the input graph. In such setting, finding an explanation for a SEP $\Psi = (\emptyset, C, T)$ corresponds to checking *satisfiability* of a SHACL document (C, T), which is known to be undecidable (Pareti et al. 2020). The undecidability proof in (Pareti et al. 2020) is for non-recursive constraints but using class targets, which can be easily updated for recursive constraints with atom-based targets. This means that EXIST is undecidable, which also extends to all other problems discussed here except $\preceq$-ISEXPL. The proof in (Pareti et al. 2020) exploits interactions between complex path expressions and number restrictions, which is a known dangerous combination in DLs. It remains to be seen if fragments of constraints can be identified where the reasoning services discussed here remain decidable, even if arbitrary atoms can be added during repairs. Some promising results exploring the connection to DLs have appeared in (Leinberger et al. 2020).

7 Conclusion

In this paper, we have provided an initial study of the problem of explaining non-validation in SHACL. We did not explicitly consider blank nodes, but since SHACL treats blank nodes as unique objects, they do not require special treatment (as long as no RDF entailment is considered, which the SHACL standard, however, leaves optional/undefined). Following intuitions from areas like explaining negative query answers in ontology-based data access, database repairs, and more broader areas like abductive reasoning and counterfactuals, we proposed that non-validation is explained by means of repairs to the offending graph. As an initial step towards an actual implementation of explanation services, in this paper we have defined a collection of reasoning tasks and provided a detailed overview of their computational complexity. We are currently working on an implementation of some of these reasoning services, employing tools from Answer Set Programming; this work will be presented in the future.

The notion of (minimal) repairs considered in this paper does not take into account the possible user preferences regarding additions or deletions of concrete facts. E.g., a user might prefer deletions of facts that are older or are coming from less reliable sources. Such setting has led, e.g., to the notions of *global*, *Pareto* and *completion* optimality of repairs (Staworko, Chomicki, and Marcinkowski 2012). Adapting these and other notions (see (Bertossi 2019)) to SHACL is an important task for future work.

Acknowledgements

This work was partially supported by the Vienna Business Agency and the Austrian Science Fund (FWF) projects P30360 and P30873. Axel Polleres' work is supported by funding in the European Commission's Horizon 2020 Research Programme under Grant Agreement Number 957402.

References

Andresel, M.; Corman, J.; Ortiz, M.; Reutter, J. L.; Savkovic, O.; and Šimkus, M. 2020. Stable model semantics for recursive SHACL. In *Proc. of The Web Conference 2020*, WWW '20, 1570–1580. ACM.

Arenas, M.; Bertossi, L. E.; and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *Proc. of PODS*, 68–79. ACM Press.

Bertossi, L. E. 2019. Database repairs and consistent query answering: Origins and further developments. In *Proc. of PODS 2019*, 48–58. ACM.

Calvanese, D.; Ortiz, M.; Simkus, M.; and Stefanoni, G. 2013. Reasoning about explanations for negative query answers in DL-Lite. *J. Artif. Intell. Res.* 48:635–669.

Ceylan, İ. İ.; Lukasiewicz, T.; Malizia, E.; Molinaro, C.; and Vaicenavicius, A. 2020. Explanations for negative query answers under existential rules. In *Proc. of KR 2020*, 223–232.

Corman, J.; Florenzano, F.; Reutter, J. L.; and Savkovic, O. 2019. Validating shacl constraints over a sparql endpoint. In *ISWC*. Springer.

Corman, J.; Reutter, J. L.; and Savkovic, O. 2018. Semantics and validation of recursive SHACL. In *Proc. of ISWC'18*. Springer.

Eiter, T., and Gottlob, G. 1993. Propositional circumscription and extended closed-world reasoning are Π_2^P-complete. *Theor. Comput. Sci.* 114(2):231–245.

Eiter, T., and Gottlob, G. 1995. The complexity of logic-based abduction. *J. ACM* 42(1):3–42.

Franconi, E.; Ibáñez-García, Y. A.; and Seylan, I. 2011. Query answering with DBoxes is hard. *Electr. Notes Theor. Comput. Sci.*

Gayo, J. E. L.; Prud'hommeaux, E.; Boneva, I.; and Kontokostas, D. 2017. *Validating RDF Data*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers.

Leinberger, M.; Seifer, P.; Rienstra, T.; Lämmel, R.; and Staab, S. 2020. Deciding SHACL shape containment through description logics reasoning. In *Proc. of ISWC 2020*, volume 12506 of *Lecture Notes in Computer Science*, 366–383. Springer.

Lutz, C.; Seylan, I.; and Wolter, F. 2013. Ontology-based data access with closed predicates is inherently intractable(sometimes). IJCAI/AAAI.

Ngo, N.; Ortiz, M.; and Simkus, M. 2016. Closed predicates in description logics: Results on combined complexity. In *Proc. of KR*, 237–246. AAAI Press.

Pareti, P.; Konstantinidis, G.; Mogavero, F.; and Norman, T. J. 2020. SHACL satisfiability and containment. In *Proc. of ISWC 2020*. Springer.

Staworko, S.; Chomicki, J.; and Marcinkowski, J. 2012. Prioritized repairing and consistent query answering in relational databases. *Ann. Math. Artif. Intell.* 64(2-3):209–246.

Van Harmelen, F.; Lifschitz, V.; and Porter, B. 2008. *Handbook of knowledge representation*. Elsevier.

Wagner, K. W. 1987. More complicated questions about maxima and minima, and some closures of NP. *Theor. Comput. Sci.* 51:53–80.