Umeå University, Sweden

A Short Introduction to SHACL for Logicians

Magdalena Ortiz
magdalena.ortiz@umu.se

UMEÅ UNIVERSITY

July 13, 2023

Contents

Data on the Web

On the web, data is stored as **RDF graphs**

triples (subject, predicate, object)

```
@prefix : <http://example.com/> .
:cervantes :authorOf :elQuixote .
```

Data on the Web

On the web, data is stored as **RDF graphs**

triples (subject, predicate, object)

```
@prefix : <http://example.com/> .
:cervantes :authorOf :elQuixote .
```

DBPedia hundreds of millions Wikipedia facts
https://dbpedia.org/

Yago high-quality knowledge graph (people, countries, organizations, ...) > 2 billion facts, 50 million nodes
https://yago-knowledge.org/

Knowledge Graphs

edge-labeled graphs

set of **node names** $\mathbf{N}$

set of **property names** $\mathbf{P}$

Data on the Web

Querying Knowledge graphs

Dedicated query language called **SPARQL**, access via **endpoints**

```
PREFIX : <http://example.com/>

SELECT ?author
WHERE {
  ?author :authorOf ?book .
  ?book :hasTitle "El Quixote" .
}
```

Data on the Web

Querying Knowledge graphs

Dedicated query language called **SPARQL**, access via **endpoints**

```
PREFIX : <http://example.com/>

SELECT ?author
WHERE {
  ?author :authorOf ?book .
  ?book :hasTitle "El Quixote" .
}
```

But what can I query for?

Formalisms for Describing RDF Graphs

OWL, Web Ontology Languages since 2009

Sharable knowledge to **infer implicit facts**

SHACL, Shapes Constraint Language since 2017

Validate constraints on the graph

Formalisms for Describing RDF Graphs

OWL, Web Ontology Languages since 2009

Sharable knowledge to **infer implicit facts**

SHACL, Shapes Constraint Language since 2017

Validate constraints on the graph

(Ann) Person

$\text{Person} \sqsubseteq \exists \text{hasNumber}.\text{PersNumber}$

In OWL, we infer that Ann has a PersonNumber

Formalisms for Describing RDF Graphs

OWL, Web Ontology Languages since 2009
Sharable knowledge to **infer implicit facts**

SHACL, Shapes Constraint Language since 2017
Validate constraints on the graph

(Ann) Person

$\text{Person} \sqsubseteq \exists \text{hasNumber}.\text{PersNumber}$

In OWL, we infer that Ann has a PersonNumber

$PersonShape \equiv \exists \text{hasNumber}.PersNumber$

In SHACL, Ann does not validate the $PersonShape$

Logics for Knowledge Graphs

The logics behind

OWL is based on **Description Logics**

What about SHACL? `https://www.w3.org/TR/shacl/`

For example, the component `sh:MinCountConstraintComponent` declares the parameter `sh:minCount` to represent the restriction that a node has at least a minimum number of values for a particular property.

For a constraint component `C` with mandatory parameters `p1`, ... `pn`, a shape `s` in a shapes graph `SG` *declares* a ***constraint*** that has ***kind*** `C` with mandatory parameter values `<p1,v1>`, ... `<pn,vn>` in `SG` when `s` has `vi` as a value for `pi` in `SG`. For constraint components with optional parameters, the constraint declaration consists of the values that the shape has for all mandatory and optional parameters of that component.

- Corman, Reuter & Slavkovic (2018) converted the 'core' of the standard into an **abstract, logic-like syntax**
- Others have built on it

Syntax of SHACL

Recall: set of **property names** $\mathbf{P}$, set of **node names** $\mathbf{N}$.
We also have a set of **shape names** $\mathbf{S}$.

We write **shape constraints** of the form

$$s \equiv \varphi$$

using **shape expressions** φ and **path expressions** E

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

where $s \in \mathbf{S}$, $a \in \mathbf{N}$, $p \in \mathbf{P}$, and $n \geq 0$.

We can express

$$\varphi_1 \vee \varphi_2 \qquad \exists E.\varphi \qquad \forall E.\varphi \qquad \leq_{n-1} E.\varphi$$

SHACL Constraints

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

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

Syntax of SHACL (cont.)

A **shapes specification** is a pair

$$(C, T)$$

- C is a set of **shape constraints**
  (usually, one constraint $s \equiv \varphi$ for each s)
- T is a set of **target** atoms $s(a)$ with $s \in \mathbf{S}$ and $a \in \mathbf{N}$

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

$$T = \{Pizza(\mathsf{capricciosa}), VeggiePizza(\mathsf{margherita})\}$$

Interpretations

We consider $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ with a non-empty **domain** $\Delta \subseteq \mathbf{N}$ and an **interpretation function** $\cdot^{\mathcal{I}}$ that assigns

to node names $a \in \mathbf{N}$,	an element $a^{\mathcal{I}} \in \Delta$,
to shape names $s \in \mathbf{S}$,	a set $s^{\mathcal{I}} \subseteq \Delta$,
to property names $p \in \mathbf{P}$,	a set of pairs $p^{\mathcal{I}} \subseteq \Delta \times \Delta$

Interpretations

We consider $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ with a non-empty **domain** $\Delta \subseteq \mathbf{N}$ and an **interpretation function** $\cdot^{\mathcal{I}}$ that assigns

to node names $a \in \mathbf{N}$,	an element $a^{\mathcal{I}} \in \Delta$,
to shape names $s \in \mathbf{S}$,	a set $s^{\mathcal{I}} \subseteq \Delta$,
to property names $p \in \mathbf{P}$,	a set of pairs $p^{\mathcal{I}} \subseteq \Delta \times \Delta$

$\cdot^{\mathcal{I}}$ is extended to complex expressions as expected:

- **path expressions** E are **binary relations** $E^{\mathcal{I}}$ over Δ

Interpretations

We consider $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ with a non-empty **domain** $\Delta \subseteq \mathbf{N}$ and an **interpretation function** $\cdot^{\mathcal{I}}$ that assigns

to node names $a \in \mathbf{N}$,	an element $a^{\mathcal{I}} \in \Delta$,
to shape names $s \in \mathbf{S}$,	a set $s^{\mathcal{I}} \subseteq \Delta$,
to property names $p \in \mathbf{P}$,	a set of pairs $p^{\mathcal{I}} \subseteq \Delta \times \Delta$

$\cdot^{\mathcal{I}}$ is extended to complex expressions as expected:

- **path expressions** E are **binary relations** $E^{\mathcal{I}}$ over Δ
- **shape expressions** φ interpreted as **sets** $\varphi^{\mathcal{I}} \subseteq \Delta$

$$\{a\}^{\mathcal{I}} = \{a\} \qquad \top^{\mathcal{I}} = \Delta \qquad (\neg\varphi)^{\mathcal{I}} = \Delta \setminus \varphi^{\mathcal{I}}$$

$$(\varphi_1 \wedge \varphi_2)^{\mathcal{I}} = {\varphi_1}^{\mathcal{I}} \cap {\varphi_2}^{\mathcal{I}}$$

$$(\geq_n E.\varphi)^{\mathcal{I}} = \{d \in \Delta \mid \text{there exist distinct } d_1, \ldots, d_n \text{ with } (d, d_i) \in E^{\mathcal{I}} \text{ land } d_i \in \varphi^{\mathcal{I}} \text{ for each } 1 \leq i \leq n\}$$

$$(E = E')^{\mathcal{I}} = \{d \in \Delta \mid \text{ for all } c \in \Delta : (c, d) \in E^{\mathcal{I}} \text{ iff } (c, d) \in {E'}^{\mathcal{I}}\}$$

SHACL Semantics 1.0

In SHACL, the graph itself is an interpretation that we can adorn with the shape names

Let $G = (N, E, \ell)$ be a **data graph**

An **shape adorment** of of G is an $\mathcal{I}$ with $p^{\mathcal{I}} = \{(a, b) \in E \mid p \in \ell(a, b)\}$ for each property p.

$\mathcal{I}$ **satisfies a constraint** $s \equiv \varphi$ if $s^{\mathcal{I}} = \varphi^{\mathcal{I}}$

G **validates** $(\mathcal{C}, T)$ if there is an adorment satisfies all constraints in $\mathcal{C}$ and has $a \in s^{\mathcal{I}}$ for every $s(a) \in T$.

SHACL Validation

Definition (SHACL validation)

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

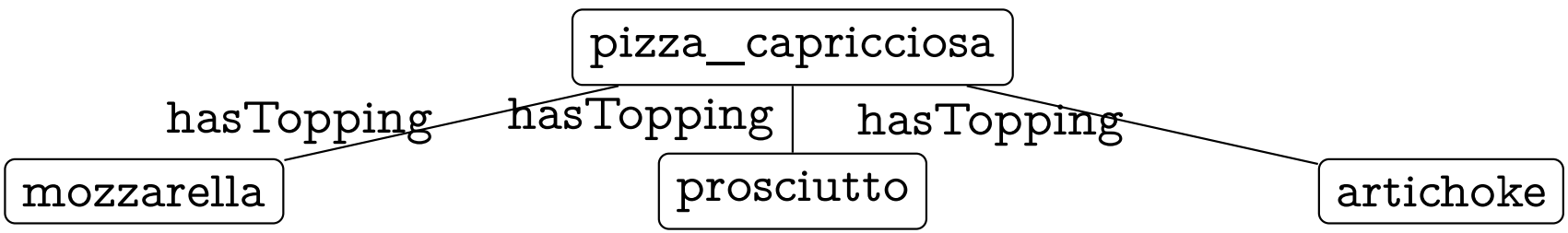
SHACL Validation

Definition (SHACL validation)

The **SHACL validation problem** consists on deciding, for a given G and (C, T), whether G validates (C, T).

$$C = \{ \quad Pizza \equiv \geq_2 \text{hasTopping}.\top,$$
$$VeggiePizza \equiv Pizza \wedge \forall \text{hasTopping}.\, VeggieTopping,$$
$$VeggieTopping \equiv \{\text{mozzarella}\} \vee \{\text{tomato}\} \vee \{\text{basil}\} \vee \{\text{artichoke}\} \quad \}$$

$$T = \{Pizza(\text{pizza\_\_capricciosa})\} \qquad T' = \{VeggiePizza(\text{pizza\_\_capricciosa})\}$$



OWL and Description Logics

If we call $\mathbf{S}$ **concept names**, then the same syntax defines **concept expressions** in the description logic $\mathcal{ALCOIQ}^{=}_{reg}$.

OWL and Description Logics

If we call $\mathbf{S}$ **concept names**, then the same syntax defines **concept expressions** in the description logic $\mathcal{ALCOIQ}^{=}_{reg}$. An $\mathcal{ALCOIQ}^{=}_{reg}$ **ontology** is a set of **concept inclusions**

$$\varphi_1 \sqsubseteq \varphi_2$$

In the special case of **terminologies** we have **definitions**

$$s \equiv \varphi$$

OWL and Description Logics

If we call $\mathbf{S}$ **concept names**, then the same syntax defines **concept expressions** in the description logic $\mathcal{ALCOIQ}^{=}_{reg}$. An $\mathcal{ALCOIQ}^{=}_{reg}$ **ontology** is a set of **concept inclusions**

$$\varphi_1 \sqsubseteq \varphi_2$$

In the special case of **terminologies** we have **definitions**

$$s \equiv \varphi$$

Semantics of concept expressions in $\mathcal{I}$ as for SHACL

- $\mathcal{I}$ **satisfies an inclusion** $\varphi_1 \sqsubseteq \varphi_2$ if $\varphi_2^{\mathcal{I}} = \varphi_2^{\mathcal{I}}$
- $\mathcal{I}$ is a **model** of an ontology $\mathcal{O}$ if it satisfies all inclusions.

OWL and Description Logics

If we call $\mathbf{S}$ **concept names**, then the same syntax defines **concept expressions** in the description logic $\mathcal{ALCOIQ}^{=}_{reg}$. An $\mathcal{ALCOIQ}^{=}_{reg}$ **ontology** is a set of **concept inclusions**

$$\varphi_1 \sqsubseteq \varphi_2$$

In the special case of **terminologies** we have **definitions**

$$s \equiv \varphi$$

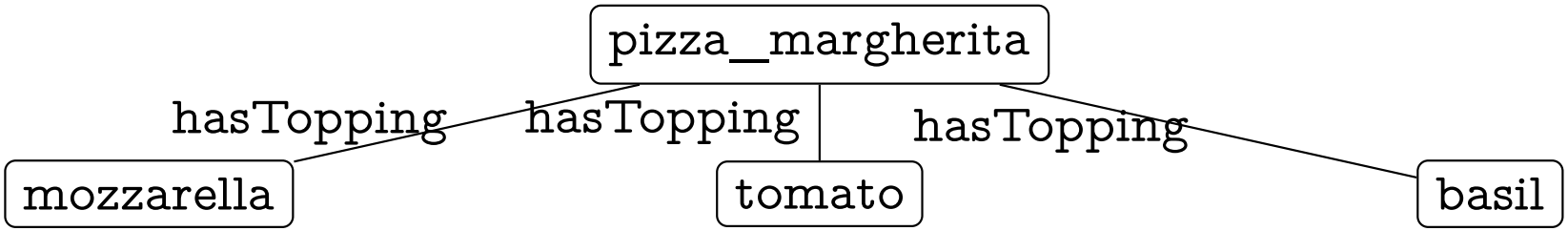
Semantics of concept expressions in $\mathcal{I}$ as for SHACL

- $\mathcal{I}$ **satisfies an inclusion** $\varphi_1 \sqsubseteq \varphi_2$ if $\varphi_2^{\mathcal{I}} = \varphi_2^{\mathcal{I}}$
- $\mathcal{I}$ is a **model** of an ontology $\mathcal{O}$ if it satisfies all inclusions.

Relaxed definition of model: $\mathcal{I}$ is a model of G if it contains G

- G **is consistent with** $\mathcal{O}$ if there exists **a model** of G and $\mathcal{O}$
- G **and** $\mathcal{O}$ **entail a fact** α if it's true in **every model** of G and $\mathcal{O}$

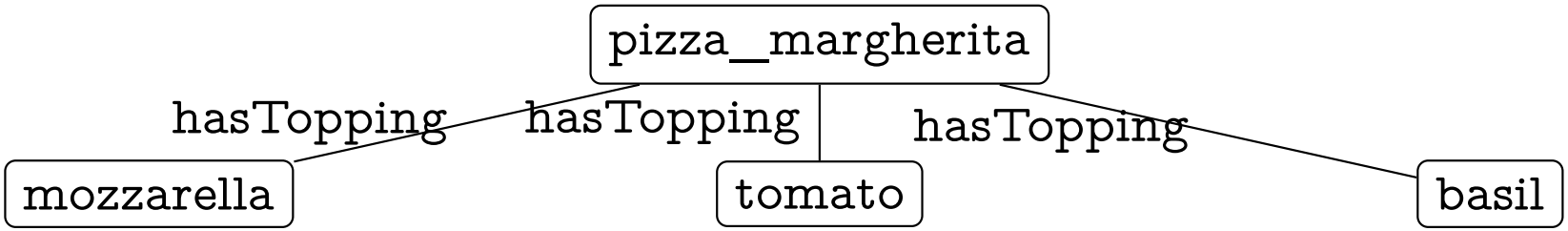
SHACL validation vs. DL entailment



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

G_{Pizza} validates $\mathit{VeggiePizza}(\texttt{pizza\_margherita})$

SHACL validation vs. DL entailment



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

G_{Pizza} validates $\mathit{VeggiePizza}(\texttt{pizza\_margherita})$

Assuming that $\mathit{VeggieTopping}$ and $\mathit{VeggiePizza}$ are concept names, consider the ontology:

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

$\mathcal{O}_{\text{Pizza}}, G_{\text{Pizza}}$ does not entail $\mathit{VeggiePizza}(\texttt{pizza\_margherita})$

DL resoning and SHACL satisfiability

In DLs, **complexity** of **consistency and entailment** known

DL resoning and SHACL satisfiability

In DLs, **complexity** of **consistency and entailment** known

$$\varphi ::= s \mid \top \mid \{a\} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \leq_n E.\varphi \mid E = E$$
$$E ::= p \mid p^- \mid E \cup E \mid E \circ E \mid E^*$$

DL resoning and SHACL satisfiability

In DLs, **complexity** of **consistency and entailment** known

$$\varphi ::= s \mid \top \mid \{a\} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \leq_n E.\varphi \mid E = E$$
$$E ::= p \mid p^- \mid E \cup E \mid E \circ E \mid E^*$$

- In $\mathcal{ALCOIQ}$ they are NExpTime complete

$$\varphi ::= s \mid \top \mid \{a\} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \geq_n p.\varphi$$

OWL is an extension of this DL called $\mathcal{SHOIQ}$

- In $\mathcal{ALCOIQ}_{reg}$ they are **long-standing open problems**

$$\varphi ::= s \mid \top \mid \{a\} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \leq_n p.\varphi \mid \exists E.\varphi$$

Extensions with $\geq_1 p.\varphi$ and $\mu\mathcal{ALCOIQ}$ are undecidable

SHACL **satisfiability and containment** are **undecidable**

Validation of Recursive SHACL

$$Director \equiv \exists \mathsf{creatorOf}.Movie$$
$$Movie \equiv \exists \mathsf{creatorOf}^{-}.Director$$

Validation of Recursive SHACL

$$Director \equiv \exists \mathsf{creatorOf}.Movie$$
$$Movie \equiv \exists \mathsf{creatorOf}^{-}.Director$$

Validation of Recursive SHACL

$$Director \equiv \exists \mathsf{creatorOf}.Movie$$
$$Movie \equiv \exists \mathsf{creatorOf}^{-}.Director$$

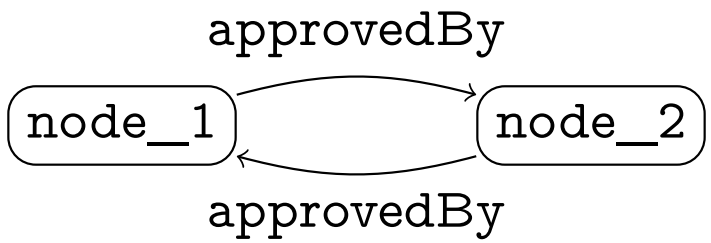
We can validate $Director(\texttt{Shakespaere})$!?!

Validation of Recursive SHACL 2

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

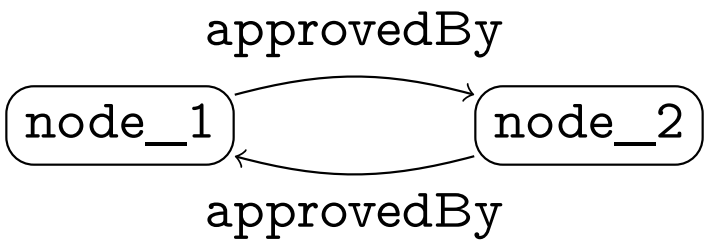
Validation of Recursive SHACL 2

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



Validation of Recursive SHACL 2

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



We can validate $certifiedNode(\texttt{node\_1})$

Semantics of Recursive SHACL

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

SHACL Recommendation, §3.4.3

KR to the rescue

Two robust semantics that avoid dubious validations

Stable model semantics minimal models of the Gelfond-Lifschitz reduct

Can be defined using **levels**

NP-complete

Well-founded semantics 3-valued approximation of stable models

P-complete

KR to the rescue

Two robust semantics that avoid dubious validations

Stable model semantics minimal models of the Gelfond-Lifschitz reduct
Can be defined using **levels**
NP-complete

Well-founded semantics 3-valued approximation of stable models
P-complete

Not only intuitive, also computationally more manageable

KR to the rescue

Two robust semantics that avoid dubious validations

Stable model semantics minimal models of the Gelfond-Lifschitz reduct
Can be defined using **levels**
NP-complete

Well-founded semantics 3-valued approximation of stable models
P-complete

Not only intuitive, also computationally more manageable

For **stratified constraints** (only **positive** recursion cycles)

- Perfect model = stable model = well-founded model
- **P-complete** validation

KR to the rescue

Two robust semantics that avoid dubious validations

Stable model semantics minimal models of the Gelfond-Lifschitz reduct
Can be defined using **levels**
NP-complete

Well-founded semantics 3-valued approximation of stable models
P-complete

Not only intuitive, also computationally more manageable

For **stratified constraints** (only **positive** recursion cycles)

- Perfect model = stable model = well-founded model
- **P-complete** validation
- In contrast, supported validation is **NP-complete**

Sowing more logic benefits

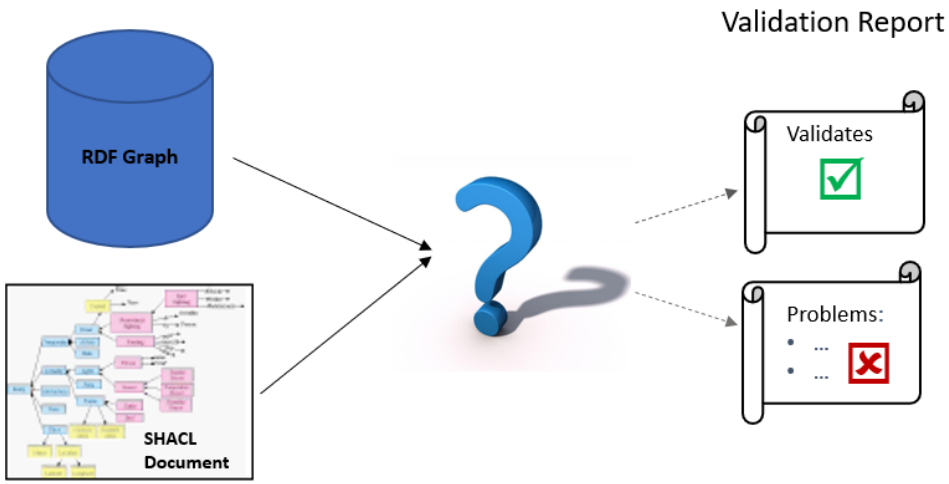
Ideas from Logic (Programming) also for:

- validating large graphs
  - Magic sets
  - combines advantages of bottom-up and top-down validation
  - only 'relevant' neighbourhood of potentially huge graphs
- inconsistency tolerant validation
  - disregard inconsistencies that are relevant to my targets

Explanations

SHACL specification describes **validation reports**

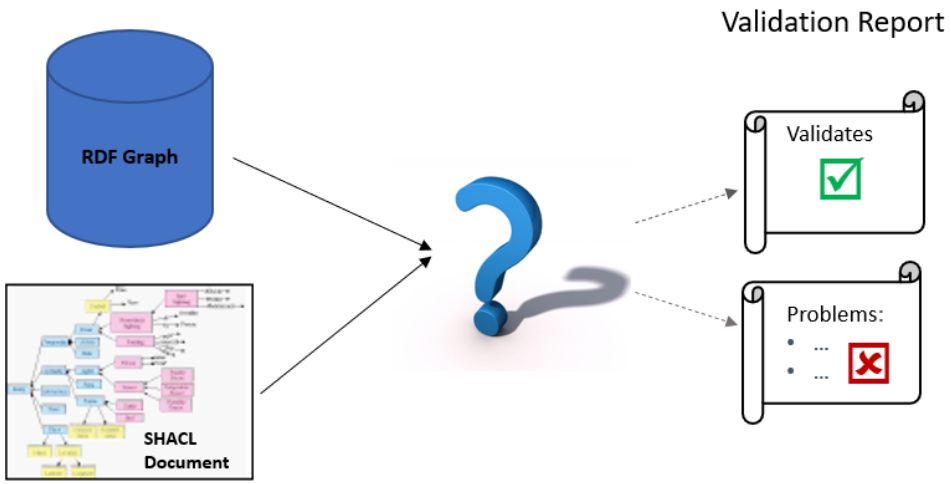
- explain the outcome of validating an RDF graph against a shapes graph
- explaining why the input graph does not satisfy the constraints is challenging.



Explanations

SHACL specification describes **validation reports**

- explain the outcome of validating an RDF graph against a shapes graph
- explaining why the input graph does not satisfy the constraints is challenging.



The standard leaves it **open how to provide such explanations to the users!**

Abductive Explanations for SHACL

We propose a notion of explanations based on logic-based abduction and database repairs!

Definition

Let G be a graph, (C, T) a SHACL specification, and the set of hypotheses H a 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 consider 2 typical preference orders over explanations:

- Subset-minimal explanations ($\subseteq$)
- Cardinality-minimal explanations ($\leq$)

Example

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

$$G = \{enrolledIn(Ben, C_1)\}$$

$$C = \{\text{Student} \leftrightarrow \exists enrolledIn.Course\}$$

$$T = \{\text{Student}(Ben)\}$$

$$H = \{Course(C_1), Course(C_2)\}$$

Example

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

$$G = \{enrolledIn(Ben, C_1)\}$$

$$C = \{\text{Student} \leftrightarrow \exists enrolledIn.Course\}$$

$$T = \{\text{Student}(Ben)\}$$

$$H = \{Course(C_1), Course(C_2)\}$$

G does not validate (C, T).

Example

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

$$G = \{enrolledIn(Ben, C_1)\}$$

$$C = \{\text{Student} \leftrightarrow \exists enrolledIn.Course\}$$

$$T = \{\text{Student}(Ben)\}$$

$$H = \{Course(C_1), Course(C_2)\}$$

G does not validate (C, T).

Explanation (A, D), where $A = \{Course(C_1)\}$ and $D = \emptyset$.

Example

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

$$C = \{\text{Student} \leftrightarrow \exists enrolledIn.Course \wedge =_1 hasID\}$$

$$T = \{\text{Student}(Ben)\}$$

$$G = \{enrolledIn(Ben, C_1), hasID(Ben, id1), hasID(Ben, id2)\}$$

$$H = \{Course(C_1), Course(C_2), enrolledIn(Ben, C_2)\}$$

There are 2 $\preceq$-explanations for Ψ:

$$A_1 = \{Course(C_1)\}, D_1 = \{hasID(Ben, id1)\}.$$

$$A_2 = \{Course(C_1)\}, D_2 = \{hasID(Ben, id2)\}.$$

Reasoning Tasks

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.

1. $\preceq$-IsExpl: is (A, D) a $\preceq$-explanation for Ψ?
2. $\preceq$-Exist: does there exist a $\preceq$-explanation for Ψ?
3. $\preceq$-NecAdd: is α a $\preceq$-necessary addition for Ψ, that is does α occur in A in every $\preceq$-explanation (A, D) for Ψ?
4. $\preceq$-NecDel: is α a $\preceq$-necessary deletion for Ψ, that is does α occur in D in every $\preceq$-explanation (A, D) for Ψ?
5. $\preceq$-RelAdd: is α a $\preceq$-relevant addition for Ψ, that is does α occur in A in some $\preceq$-explanation (A, D) for Ψ?
6. $\preceq$-RelDel: is α a $\preceq$-relevant deletion for Ψ, that is does α occur in D in some $\preceq$-explanation (A, D) for Ψ?

Complexity Results

	$\emptyset$	$\subseteq$	$\leq$
IsExpl	NP-c	DP-c	DP-c
Exist	NP-c	NP-c	NP-c
NecAdd	coNP-c	coNP-c	$P^{\|\mathsf{NP}}$-c
NecDel	coNP-c	coNP-c	$P^{\|\mathsf{NP}}$-c
RelAdd	NP-c	Σ_2^P-c	$P^{\|\mathsf{NP}}$-c
RelDel	NP-c	Σ_2^P-c	$P^{\|\mathsf{NP}}$-c

All these results hold in **data and combined complexity**

Other Results

We have also studied

- the **non-recursive case** same complexity
- a more general setting with **explanation signatures**
  - we can specify predicates that are
    read-only add-only delete-only
  - fine-grianed filtering of explanations
  - all the results hold also for this setting

Using ASP to explain non-validation

Simple prototype for providing explanations using **Answer Set Programming (ASP)**

Conclusions and Outlook

Some of our ongoing efforts:

- validation in the presence of ontologies
- validation when the graph is subjected to updates
- better and more useful explanations for validation reports

Many open problems for logicians to tackle!

SHACL is a relatively young field that can use insights from well-established areas of logic

Conclusions and Outlook

Some of our ongoing efforts:

- validation in the presence of ontologies
- validation when the graph is subjected to updates
- better and more useful explanations for validation reports

Many open problems for logicians to tackle!

SHACL is a relatively young field that can use insights from well-established areas of logic

Thanks to all collaborators!

In particular: Mantas Šimkus, Shqiponja Ahmetaj, Medina Andresel, Bianca Löhnert, Anouk Oudshoorn, Juan Reutter, Julien Corman.

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