Expressiveness of SHACL Features

ICDT 2021

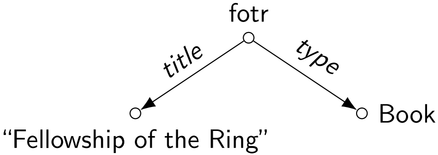
Bart Bogaerts, *Maxime Jakubowski* & Jan Van den Bussche

Vrije Universiteit Brussel & Universiteit Hasselt

March 2022

SHACL

- **Sh**apes **C**onstraint **L**anguage
- Constraint language for RDF graphs
- Conformance checking



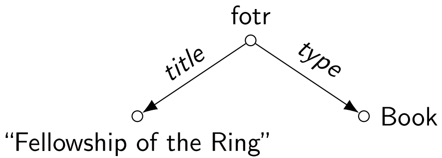
SHACL

- **Sh**apes **C**onstraint **L**anguage
- Constraint language for RDF graphs
- Conformance checking

```
:BookShape
  a sh:PropertyShape;
  sh:path :title;
  sh:minCount 1.

:BookShape sh:targetClass :Book
```

$$\geq_1 \mathit{type}.\mathit{Book} \subseteq \geq_1 \mathit{title}.\top$$



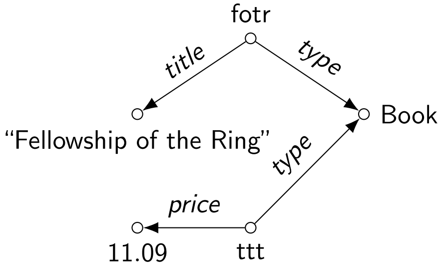
SHACL

- **Sh**apes **C**onstraint **L**anguage
- Constraint language for RDF graphs
- Conformance checking

```
:BookShape
  a sh:PropertyShape;
  sh:path :title;
  sh:minCount 1.

:BookShape sh:targetClass :Book
```

$$\geq_1 \mathit{type}.\mathit{Book} \subseteq \geq_1 \mathit{title}.\top$$



SHACL shapes

The language $\mathcal{L}$

$$\phi ::= \top \mid \{c\} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \forall E.\phi \mid \geq_n E.\phi$$
$$E ::= p \mid p^- \mid E \cup E \mid E/E \mid E^*$$

E are regular path queries with inverse

SHACL shapes

The language $\mathcal{L}$

$$\phi ::= \top \mid \{c\} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \forall E.\phi \mid \geq_n E.\phi$$
$$E ::= p \mid p^- \mid E \cup E \mid E/E \mid E^*$$

E are regular path queries with inverse

An interpretation I:

- domain Δ^I
- interprets *node names*
- interprets *property names*

ϕ	$I, a \vDash \phi$ if:
$\{c\}$	$a = [\![c]\!]^I$
$\geq_n E.\psi$	$\sharp\{b \in [\![E]\!]^I(a) \mid I, b \vDash \psi\} \geq n$
$\forall E.\psi$	every $b \in [\![E]\!]^I(a)$ must $I, b \vDash \psi$

SHACL shapes

The language $\mathcal{L}(eq, disj, closed, ?)$

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

E are regular path queries with inverse

Distinctive features:

- Equality
- Disjointness
- Closure
- Zero-or-one path

ϕ	$I, a \vDash \phi$ if:
$\{c\}$	$a = [\![c]\!]^I$
$\geq_n E.\psi$	$\sharp\{b \in [\![E]\!]^I(a) \mid I, b \vDash \psi\} \geq n$
$eq(E, p)$	the sets $[\![E]\!]^I(a)$ and $[\![p]\!]^I(a)$ are equal
$disj(E, p)$	the sets $[\![E]\!]^I(a)$ and $[\![p]\!]^I(a)$ are disjoint
$closed(R)$	$[\![p]\!]^I(a)$ is empty for each $p \in \Sigma - R$

SHACL shapes

The language $\mathcal{L}(eq, disj, closed, ?)$

$$\phi ::= \top \mid \{c\} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \forall E.\phi \mid \geq_n E.\phi \mid eq(p, E) \mid disj(p, E) \mid closed(Q)$$

$$E ::= p \mid p^- \mid E \cup E \mid E/E \mid E^* \mid E?$$

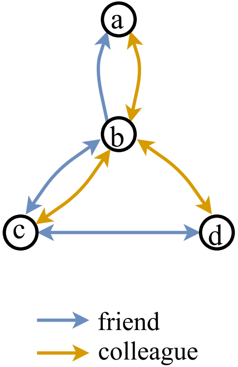
E are regular path queries with inverse

Do we need these features in the language?

Example shapes

"Through a path of **friend** edges, the node can reach node *d* "

$$\phi \equiv \geq_1 friend^*.\{d\}$$

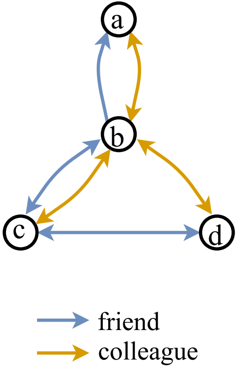


Example shapes

"Through a path of **friend** edges, the node can reach node d"

$$\phi \equiv \geq_1 \mathit{friend}^*.\{d\}$$

$$[\![\phi]\!]^G = \{b, c, d\}$$



Example shapes

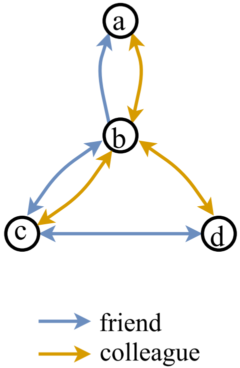
"Through a path of **friend** edges, the node can reach node *d*"

$$\phi \equiv \geq_1 \mathit{friend}^*.\{d\}$$

$$[\![\phi]\!]^G = \{b, c, d\}$$

"Nodes where the **friend**ship is mutual"

$$\phi \equiv eq(\mathit{friend}, \mathit{friend}^-)$$



Example shapes

"Through a path of **friend** edges, the node can reach node d"

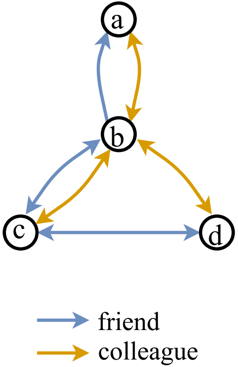
$$\phi \equiv \geq_1 friend^*.\{d\}$$

$$[\![\phi]\!]^G = \{b, c, d\}$$

"Nodes where the **friend**ship is mutual"

$$\phi \equiv eq(friend, friend^-)$$

$$[\![\phi]\!]^G = \{c, d\}$$



Example shapes

"Through a path of **friend** edges, the node can reach node d"

$$\phi \equiv \geq_1 friend^*.\{d\}$$

$$[\![\phi]\!]^G = \{b, c, d\}$$

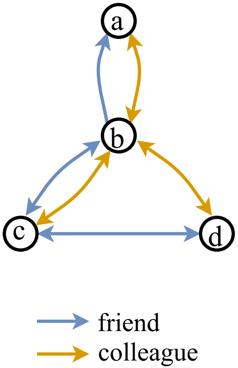
"Nodes where the **friend**ship is mutual"

$$\phi \equiv eq(friend, friend^-)$$

$$[\![\phi]\!]^G = \{c, d\}$$

"Nodes who have at least one **colleague** who is also a **friend**"

$$\phi \equiv \neg disj(friend, colleague)$$



Example shapes

"Through a path of **friend** edges, the node can reach node d"

$$\phi \equiv \geq_1 friend^*.\{d\}$$

$$[\![\phi]\!]^G = \{b, c, d\}$$

"Nodes where the **friend**ship is mutual"

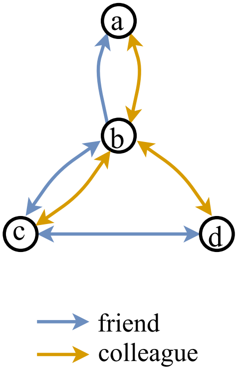
$$\phi \equiv eq(friend, friend^-)$$

$$[\![\phi]\!]^G = \{c, d\}$$

"Nodes who have at least one **colleague** who is also a **friend**"

$$\phi \equiv \neg disj(friend, colleague)$$

$$[\![\phi]\!]^G = \{b, c\}$$



Graphs and interpretations

- A graph is a finite set of *facts*
- A fact is of the form $p(a, b)$ with p a property name and a, b nodes from the graph

We associate to any given graph an interpretation I:

- The domain is the universe of *all nodes*
- Every constant is interpreted as itself
- The interpretation of a property name is fixed by the facts

Shape schemas

A *shape schema* is a set of inclusion statements

$$\phi_t \subseteq \phi_s$$

A shape schema *defines* a class of graphs.

Shape schemas

A *shape schema* is a set of inclusion statements

$$\phi_t \subseteq \phi_s$$

A shape schema *defines* a class of graphs.

Example: the class of symmetric graphs.

$$\geq_1 r.\top \subseteq eq(r, r^-)$$

Main result: primitivity of the features

For each feature $X \in \{eq, disj, closed, ?\}$ we define a class of graphs Q_X such that:

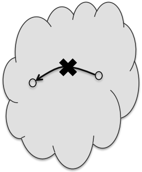
- Q_X is definable by a simple inclusion using only the feature X
- Q_X is **not** definable without X

Proving primitivity of equality

Equality

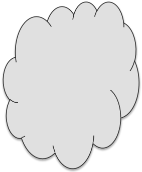
Q_{eq} is the class of symmetric graphs: $\geq_1 r.\top \subseteq eq(r, r^-)$

Graph G



A complete directed graph
with one edge removed

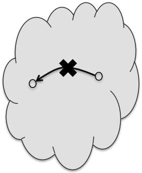
Graph G'



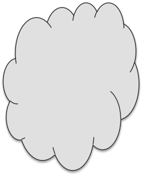
A complete directed graph

Proving primitivity of equality

Graph G



Graph G'



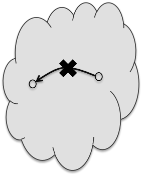
Lemma

Let H be G or G'. For every path expression E, we have:

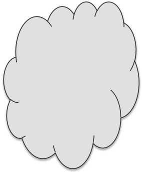
- $[\![E]\!]^H \supseteq [\![r]\!]^H$
- $[\![E]\!]^H \supseteq [\![r^-]\!]^H$
- $[\![E]\!]^H \supseteq V \times V$

Proving primitivity of equality

Graph G



Graph G'



Proposition

For any shape ϕ not using eq: $[\![\phi]\!]^G = [\![\phi]\!]^{G'}$.

Proving primitivity of disjointness

Disjointness

Q_{disj} is the class of graphs where all nodes have at least one symmetric edge:

$$\geq_1 r.\top \subseteq \neg disj(r, r^-)$$

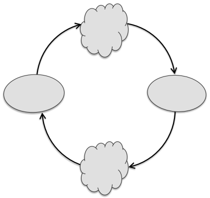
Proving primitivity of disjointness

Disjointness

Q_{disj} is the class of graphs where all nodes have at least one symmetric edge:

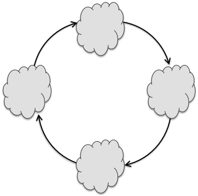
$$\geq_1 r.\top \subseteq \neg disj(r, r^-)$$

Graph G



An alternating cycle of cliques

Graph G'

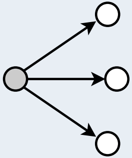


A cycle of cliques

Proving primitivity of zero-or-one paths

Zero-or-one paths

$Q_?$ is the class of graphs where all nodes have at least three outgoing edges not to themselves:



$$\geq_1 r.\top \subseteq \geq_4 r?.\top$$

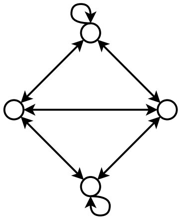
Proving primitivity of zero-or-one paths

Zero-or-one paths

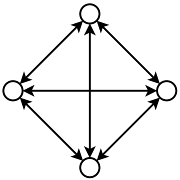
$Q_?$ is the class of graphs where all nodes have at least three outgoing edges not to themselves:

$$\geq_1 r.\top \subseteq \geq_4 r?.\top$$

Graph G



Graph G'



Proving primitivity of closure

Closure

Q_{closed} is the class of graphs where the only edge label allowed is r.

Proposition

Shapes without using closed do not distinguish between graphs that are equal on all edge labels mentioned in the shape.

Conclusion & Future Work

We established the primitivity of equality, disjointness, zero-or-one paths and closure in SHACL.

Conclusion & Future Work

We established the primitivity of equality, disjointness, zero-or-one paths and closure in SHACL.

Real SHACL has some hidden features:

- $eq(id, p)$ which is expressible as $eq(p?, p) \wedge \geq_1 p.\top \wedge \leq_1 p.\top$
- $disj(id, p)$ which is expressible as $\neg eq(p?, p)$
- Is zero-or-one path still primitive?

Conclusion & Future Work

We established the primitivity of equality, disjointness, zero-or-one paths and closure in SHACL.

Real SHACL has some hidden features:

- $eq(id, p)$ which is expressible as $eq(p?, p) \wedge \geq_1 p.\top \wedge \leq_1 p.\top$
- $disj(id, p)$ which is expressible as $\neg eq(p?, p)$
- Is zero-or-one path still primitive?

Natural extensions of the shape language:

- allow shapes of the form $eq(E_1, E_2)$ and $disj(E_1, E_2)$
- allow path expressions with *tests* (as in PDL)
- expressiveness under recursive semantics (stable model, well-founded)