Shapes Constraint Language:
Formalization, Expressiveness, and Provenance

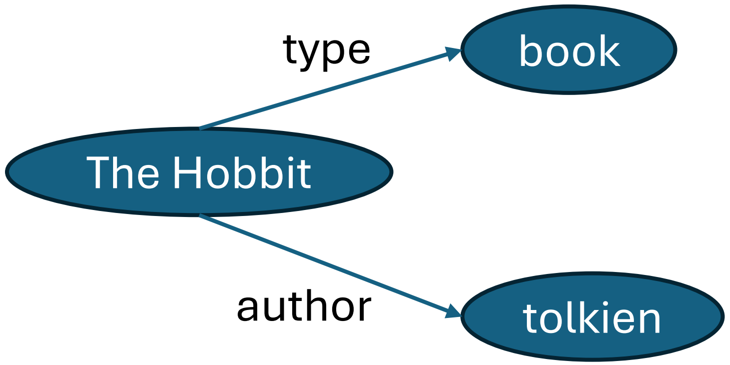
Maxime Jakubowski

Doctoral Defense

May 31st 2024

Knowledge Graphs

- Organising information in graphs
- Resource Description Framework (RDF) – RDF graphs



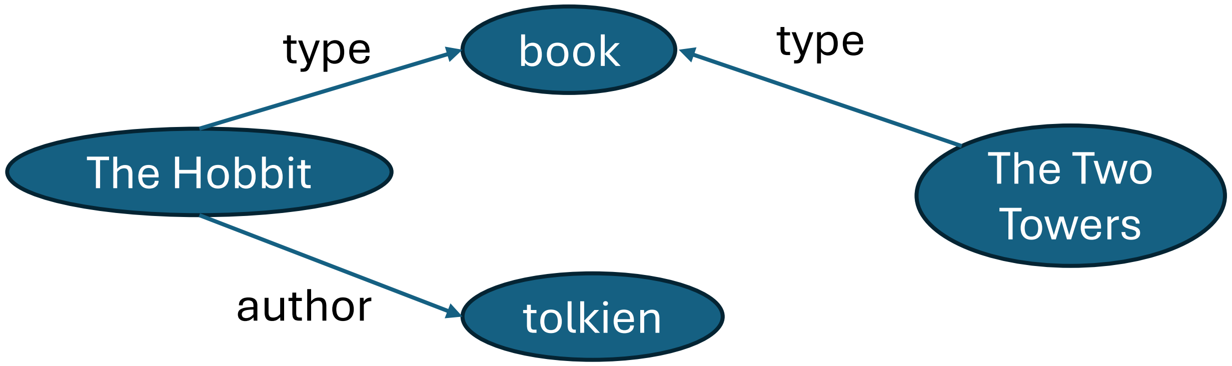
OWL Knowledge

"every **author** is a **human**"

- What to do with this information?
  - Querying (SPARQL): "who is the author of **The Hobbit**?"
  - Reasoning (OWL): "is **tolkien** human?"

Shapes **C**onstraint **L**anguage: SHACL

- RDF graphs can be very large
- We want to ensure good *quality* of these graphs
  - Knowing what to expect
  - Detecting errors in our data
  - Ensuring consistency
- For example: “every **book** must have an **author**”

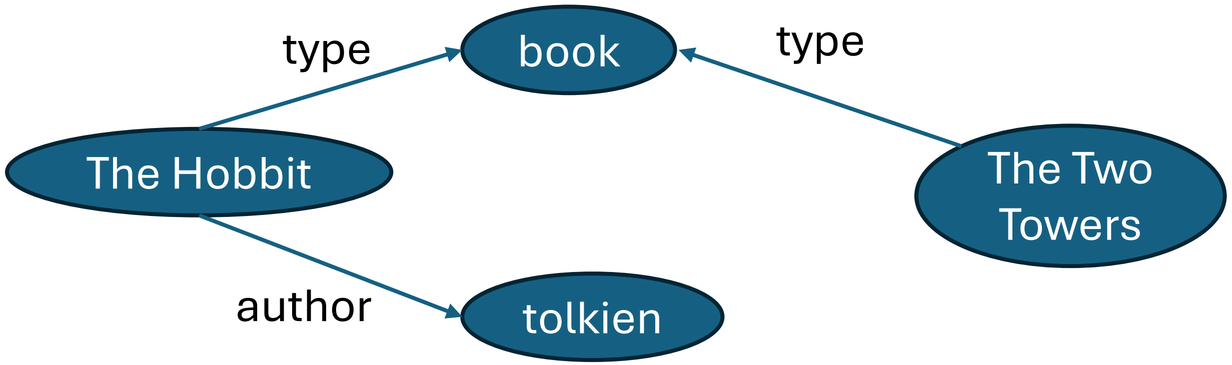


Shapes **C**onstraint **L**anguage: SHACL

- *Shapes* are constraints on a graph, and consists of two parts:
  - Constraint on a node: the *shape expression*
  - Specification which nodes must satisfy the constraint: *target declaration*
- In our example:
  - “has an **author**” is a shape expression
  - “every **book**” is a target declaration

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

:BookShape sh:targetClass :Book .
```



Formalization

“What is a good abstraction for SHACL?”

Bart Bogaerts, Maxime Jakubowski, Jan Van den Bussche:
SHACL: A Description Logic in Disguise. LPNMR 2022: 75-88

Bart Bogaerts, Maxime Jakubowski:
Fixpoint Semantics for Recursive SHACL. ICLP Tech. Comm. 2021: 41-47

Formal semantics for SHACL

- SHACL is defined by the World Wide Web Concortium in a document called the “recommendation”
  - This document is mainly written for software engineers that implement SHACL software
  - For academic understanding of SHACL, more abstraction is needed
- Academics soon formalized SHACL:
  - Julien Corman, Juan L. Reutter, Ognjen Savkovic:
    Semantics and Validation of Recursive SHACL. ISWC (1) 2018: 318-336
  - Medina Andresel, Julien Corman, Magdalena Ortiz, Juan L. Reutter, Ognjen Savkovic, Mantas Simkus:
    Stable Model Semantics for Recursive SHACL. WWW 2020: 1570-1580

SHACL – The Logical Perspective

Let N, P and S be disjoint universes of node names, property names and shape names.

Shape Expressions

$$\phi \coloneqq \top \mid hasValue(c) \mid hasShape(s) \mid eq(E,p) \mid disj(E,p)$$

$$\mid closed(Q) \mid \geq_n E.\phi \mid \phi \wedge \phi \mid \neg\phi$$

$$E \coloneqq id \mid p \mid p^- \mid E \cup E \mid E/E \mid E^*$$

with $c \in N, p \in P, s \in S$ and $Q \subseteq P$

E are regular path queries with inverse

Semantics of Shape Expressions

- Given through an interpretation I:
  - with a domain: Δ^I
  - interprets node names $c \in N$: $[\![c]\!]^I$
  - Interprets shape names $s \in S$: $[\![s]\!]^I$
  - Interprets property names $p \in P$: $[\![p]\!]^I$

- We define the *evaluation* of:
  - a path expression E, denoted $[\![E]\!]^I$, as a subset of $\Delta^I \times \Delta^I$
  - a shape expression ϕ, denoted $[\![\phi]\!]^I$, as a subset of Δ^I

Semantics of Shape Expressions

- We define the *evaluation* of:
  - a path expression E, denoted $[\![E]\!]^I$, as a subset of $\Delta^I \times \Delta^I$
  - a shape expression ϕ, denoted $[\![\phi]\!]^I$, as a subset of Δ^I

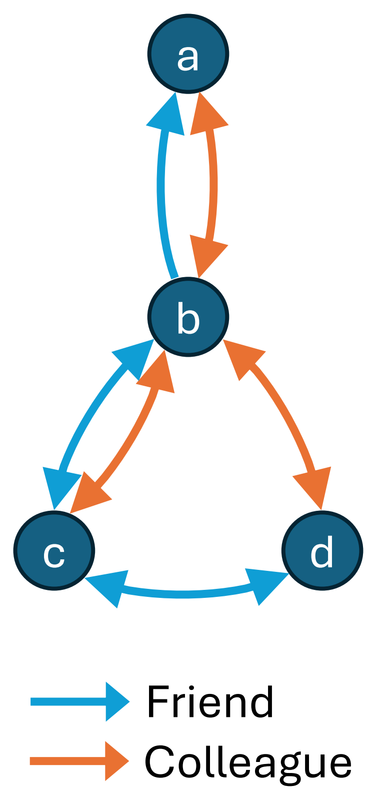
$\boldsymbol{\phi}$	$[\![\boldsymbol{\phi}]\!]^{\boldsymbol{I}}$
$\top$	Δ^I
$hasValue(c)$	$[\![c]\!]^I$
$hasShape(s)$	$[\![s]\!]^I$
$eq(E,p)$	$\{a \in \Delta^I \mid [\![E]\!]^I(a) = [\![p]\!]^I(a)\}$
$disj(E,p)$	$\{a \in \Delta^I \mid [\![E]\!]^I(a) \cap [\![p]\!]^I(a) = \emptyset\}$
$closed(Q)$	$\{a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

What’s in an RDF Graph?

- *Real SHACL* is not about interpretations, but about *graphs*
- An RDF graph is a finite set of facts $p(a, b)$ representing the edges
- This gives us a specific type of interpretation where:
  - The domain is the universe of all nodes: $\Delta^I = N$
  - Constants, i.e., nodes, are interpreted as themselves : $[\![c]\!]^I = \{c\}$
  - The interpretation of properties is given by the graph: $[\![p]\!]^I = [\![p]\!]^G$

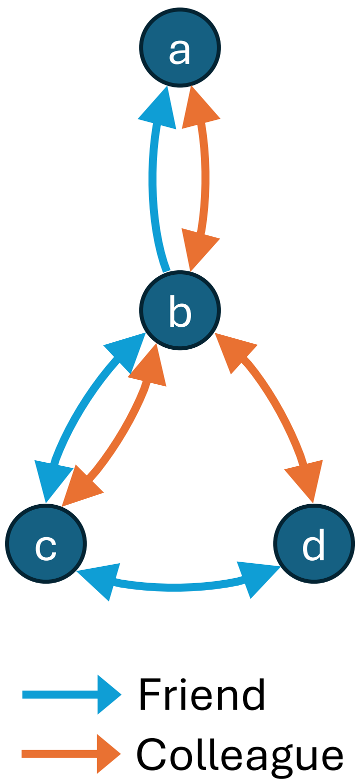
Example shapes

- *“Through a path of **friend** edges, the node can reach node d”*
  - $\exists friend^{*}.\,hasValue(d)$
  - b, c, and d satisfy this shape in G
- *“Nodes where **friend**ship is mutual”*
  - $eq(friend, friend^{-})$
  - c and d satisfy this shape in G
- *“Nodes who have at least one **colleague** who is also a **friend**”*
  - $\neg disj(friend, colleague)$
  - b and c satisfy this shape in G



Shape schemas

- The main task is to check whether a **graph** conforms to some constraints, not single nodes.
  - Shape definition: $s \leftarrow \phi$
  - Target statement: $\phi_t \subseteq \phi_s$
- Example schema (Def, T):
  - Def: $\mathbf{FriendOfD} \leftarrow \exists \mathit{friend}^*.\, hasValue(d)$
  - T: $\exists \mathit{friend}.\top \subseteq hasShape(\mathbf{FriendOfD})$



Correspondence with the Recommendation

- How does this logic exactly compare to SHACL?
- We want to study SHACL, **and** this logic:

Theorem	Every formal SHACL schema can be written as a real SHACL shapes graph and vice versa

Recursion in SHACL

- Recursion: defining a shape in terms of itself
- Some shapes are easily expressed with recursion

“You are a **GoodFriendOfD** if there is a *friend*-path of *Good* nodes to *d*”

GoodFriendOfD $\leftarrow hasValue(d)$
GoodFriendOfD $\leftarrow \exists friend.\, hasShape(\textbf{GoodFriendOfD}) \land \exists type.\, hasValue(Good)$

```
:GoodFriendOfD a sh:NodeShape ;
  sh:class :Good ;
  sh:or (
  [ sh:hasvalue :d ]
  [ sh:path :friend ;
    sh:qualifiedValueShape :GoodFriendOfD ;
    sh:qualifiedMinCount 1 ] ) .
```

Applying Approximation Fixpoint Theory

- Algebraic Framework to study fixpoints, defines:
  - Supported semantics
  - Stable semantics
  - Well-founded semantics
  - …
- We only need to:
  - Agree on an order of interpretations: the standard truth order
  - Agree on how to evaluate shapes in the three-valued logical setting: Kleene
- We get:
  - Well defined semantics for recursive SHACL
  - Theorethical body of results coming from AFT, now applicable to SHACL

$I_1 \leq I_2$ iff for every $s \in S$: $[\![s]\!]^{I_1} \subseteq [\![s]\!]^{I_2}$

p	q	$p \wedge q$	$p \vee q$
True	unknown	unknown	true
False	unknown	false	unknown

Existing Semantics

[Corman 2018] defined supported model semantics (CRS-supported)

- Already defined the three-valued semantic operator Φ_{Def}

Theorem	CRS-supported models coincide with the AFT-supported models

[Andreşel 2020] defined stable model semantics (ACORSS-stable)

- Defined in terms of 'level-mappings'

Theorem	Every AFT-stable model is a ACORSS-stable model. If Def is in *shape normal form*, the converse also holds.

Expressiveness

“What can we express with SHACL?”

Bart Bogaerts, Maxime Jakubowski, Jan Van den Bussche:

Expressiveness of SHACL Features and Extensions for Full Equality and Disjointness Tests. Log. Methods Comput. Sci. 20(1) (2024)

Relative Expressiveness

- What features of SHACL are *essential* ?
- We already have the classical logical equivalences for free:
  - $\forall friend.\,hasShape(\mathbf{GoodPerson}) \equiv \neg\exists friend.\,\neg hasShape(\mathbf{GoodPerson})$
- Three uncommon features of SHACL:
  - Equality: $eq(E, p)$
  - Disjointness: $disj(E, p)$
  - Closed: $closed(Q)$
- Can we express these features with other constructs?

SHACL Features

Simple Shape Expressions

$\phi \coloneqq \top \mid hasValue(c) \mid \geq_n E.\phi \mid \phi \wedge \phi \mid \neg\phi$

$E \coloneqq id \mid p \mid E^- \mid E \cup E \mid E/E \mid E^*$

with $c \in N, p \in P, s \in S$ and $Q \subseteq P$

- We call this the language L, with *none* of the interesting features
- $F \subseteq \{eq, disj, closed\}$
- $L(eq, disj, closed)$ is the logical core of SHACL
- With $L(F)$ we can write *generalized shape schemas*
- Set of inclusion statements: $\phi_t \subseteq \phi_s$ with ϕ_t and ϕ_s in $L(F)$

Main Result: all are primitive

Theorem

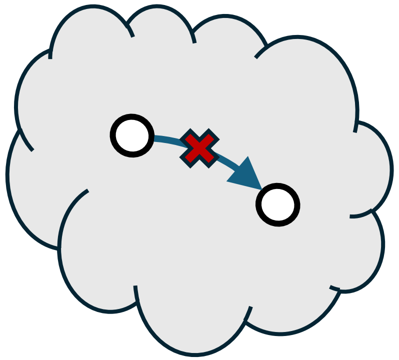
For each feature $X \in \{ eq,\ disj,\ closed\}$ we define a class of graphs Q_X such that:

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

Proving Primitivity of Equality

Equality	Q_{eq} is the class of symmetric graphs: $$\exists r.\top \subseteq eq(r, r^{-})$$ For any shape ϕ not using equality: $[\![\phi]\!]^{G} = [\![\phi]\!]^{G'}$

G



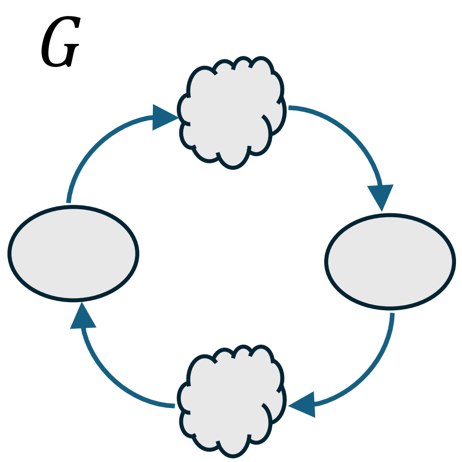
A complete directed graph with one edge removed

G'

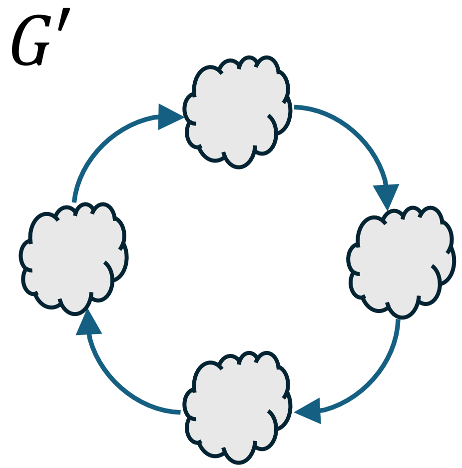
A complete directed graph

Proving Primitivity of Disjointness

Disjointness	Q_{disj} is the class of graphs where all nodes have at least one symmetric edge: $$\exists r.\top \subseteq \neg disj(r, r^-)$$ For any shape ϕ not using equality: $[\![\phi]\!]^G = [\![\phi]\!]^{G'}$



An alternating cycle of cliques



A cycle of cliques

Extending the Primitivity Result

- Equality and disjointness seem artificially restricted!
- Full-Equality: $eq(E_1, E_2)$
- Full-Disjointness: $disj(E_1, E_2)$

Theorem

For both features $X \in \{\, full\text{–}eq,\ full\text{–}disj \,\}$ we define a class of graphs Q_X such that:

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

Provenance:
“What subset of the graph is really relevant?”

Thomas Delva, Anastasia Dimou, Maxime Jakubowski, Jan Van den Bussche:

Data Provenance for SHACL. EDBT 2023: 285-297

Provenance & Neighborhoods

Goal:

- Provide **provenance** of a shape schema as a subgraph
- This **subgraph** only contains triples that are "relevant"

We define the **neighborhood**: $B(G, v, \phi)$

- G a graph
- v a node
- ϕ a shape

What part of G is relevant to decide that v satisfies ϕ in G?

Sufficiency and design principles

Sufficiency Property	If a node v satisfies a shape ϕ in a graph G, then: v also satisfies ϕ in G' for any subgraph G' with $B(G, v, \phi) \subseteq G' \subseteq G$.

"Strong" sufficiency: it holds for all G' where $B(G, v, \phi) \subseteq G' \subseteq G$.

- Technical necessity
- Allows for leniency in implementations of neighborhoods

When defining neighborhoods we want to be both **deterministic** and **minimal**

Neighborhood definition

Shapes in **negation normal form** (and no path expressions):

$\phi \;\coloneqq \top \mid \bot \mid hasValue(c) \mid \neg hasValue(c) \mid eq(p,q) \mid \neg eq(p,q)$

$\quad\mid disj(p,q) \mid \neg disj(p,q) \mid closed(Q) \mid \neg closed(Q)$

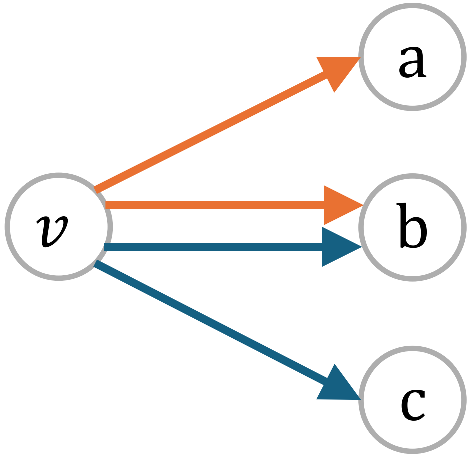
$\quad\mid \phi \wedge \phi \mid \phi \vee \phi \mid \;\geq_n p.\phi \mid\; \leq_n p.\phi$

Neighborhood of a node v according to a shape ϕ in graph G: $B(G, v, \phi)$

- When the node v does **not** satisfy ϕ in G, the neighborhood is empty
- Shapes that do not use any properties, also have an empty neighborhood

Nonequality: $\neg \boldsymbol{eq}(\boldsymbol{p}, \boldsymbol{q})$

G

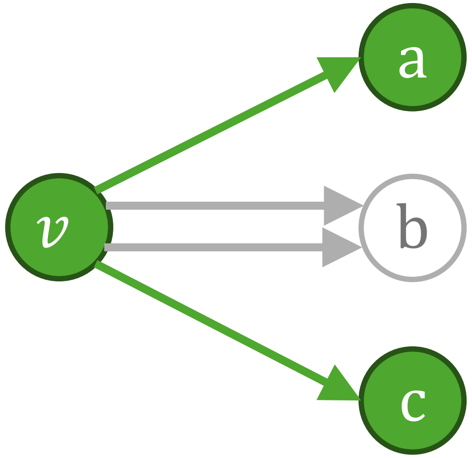


Options:

- Only edges to a and/or c
- All edges

$B(G, v, \phi)$

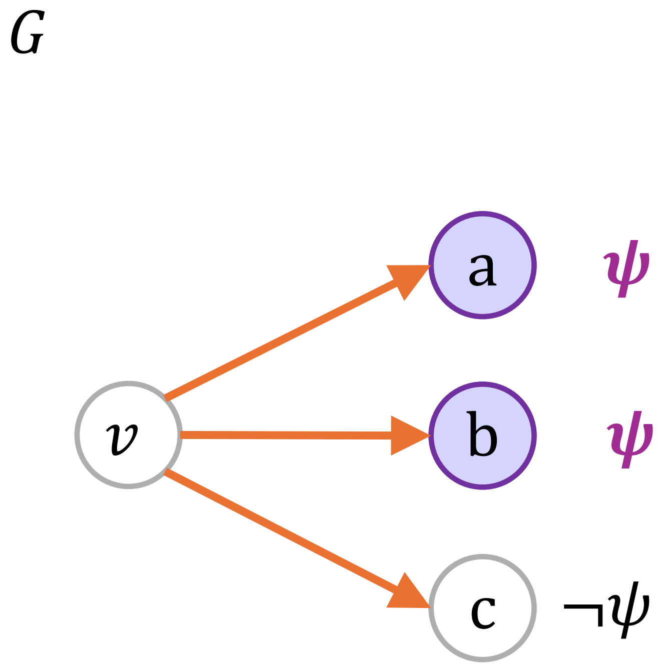
“symetric difference”



Reasons:

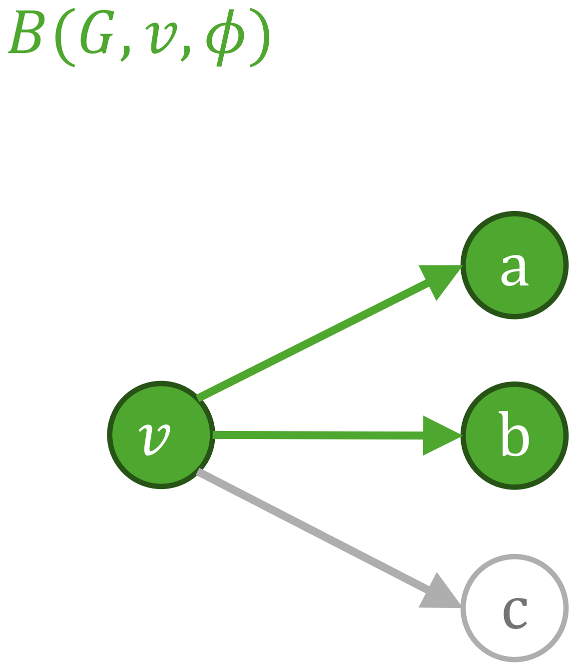
- Determinism
- Somehow minimal

Quantifiers: $\geq_1 \, p.\psi$



Options:

- Only edges to a and/or b
- All edges

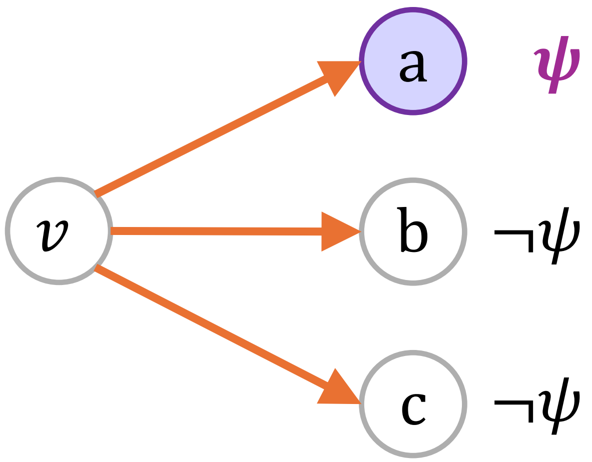


Reasons:

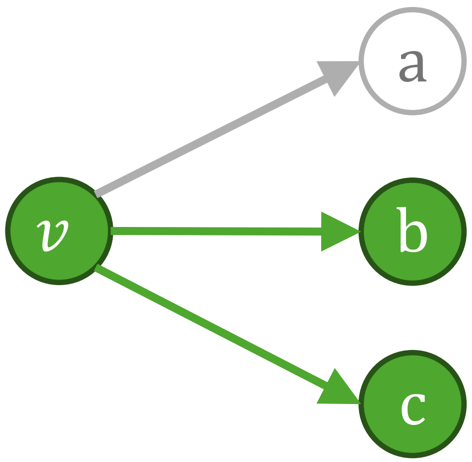
- Determinism
- Somehow minimal

Quantifiers: $\leq_1 \boldsymbol{p}.\boldsymbol{\psi}$

G



$B(G, v, \phi)$



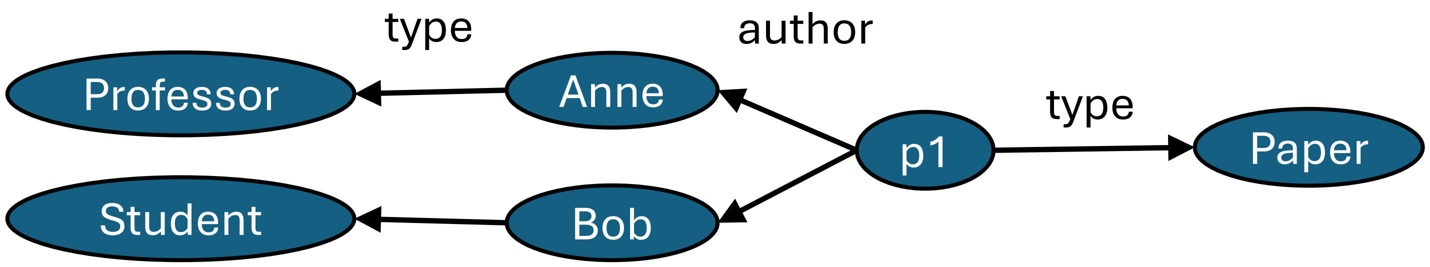
Options:

- No edges
- Edge to a
- All edges

Reasons:

- Determinism
- Somehow minimal
- Adding edges to the neighborhood may not break sufficiency

Example

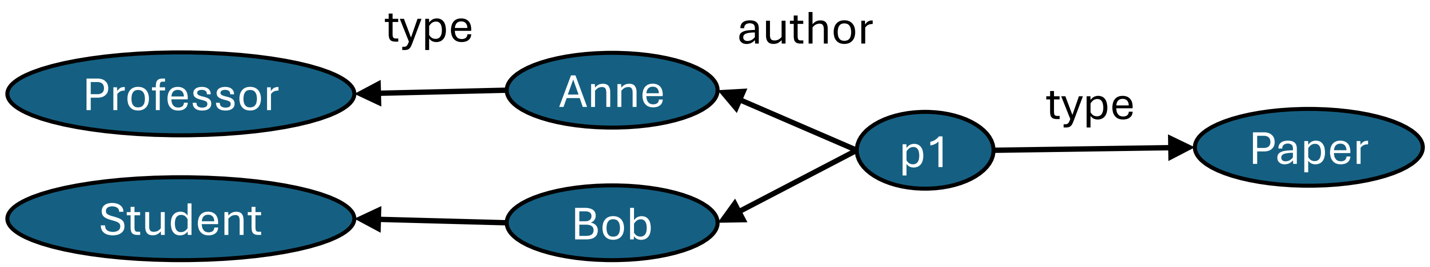


$$\phi \equiv \geq_1 author.\top \;\wedge\; \leq_1 author.\neg\geq_1 type.hasValue(Student)$$

"The node has an author and at most one author is not a student."

$$B(G, p1, \phi)$$

Example

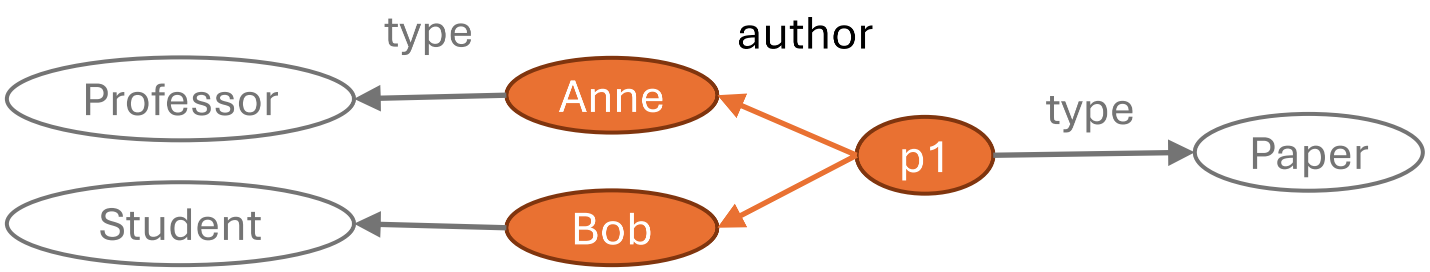


$$\phi \equiv \geq_1 author.\top \;\wedge\; \leq_1 author. \leq_0 type.hasValue(Student)$$

"The node has an author and at most one author is not a student."

$$B(G, p1, \phi)$$

Example

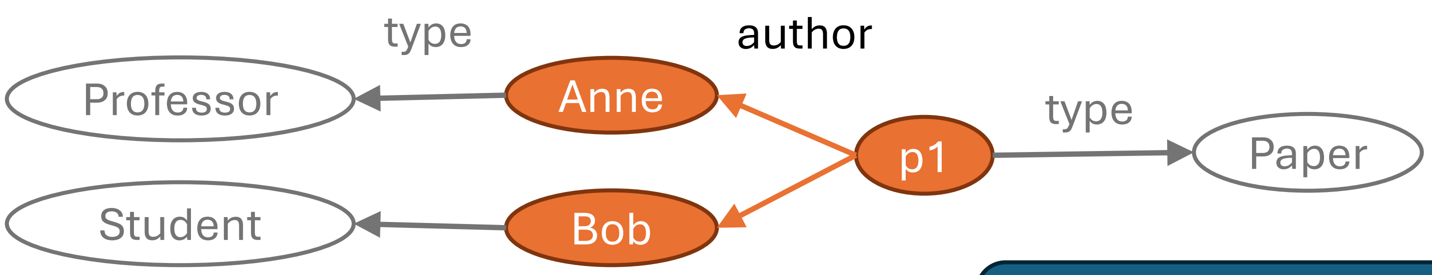


$$\phi \equiv \geq_1 author.\top \ \wedge \ \leq_1 author. \leq_0 type.hasValue(Student)$$

"The node has an author and at most one author is not a student."

$$B(G, p1, \phi)$$

Example



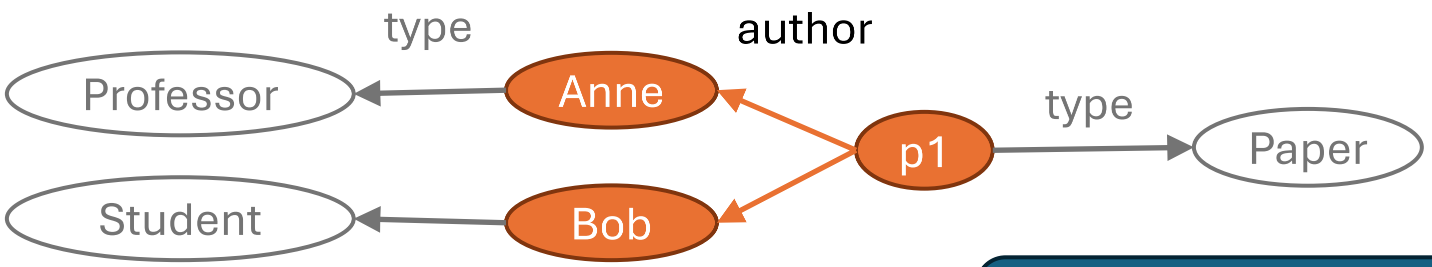
$B(G, \textbf{Anne}, \neg{\leq_0}\ type.\,hasValue(Student))$
$B(G, \textbf{Bob}, \neg{\leq_0}\ type.\,hasValue(Student))$

$$\phi \equiv\ \geq_1 author.\top\ \wedge\ \leq_1 author.\boxed{\leq_0 type.\,hasValue(Student)}$$

“The node has an author and at most one author is not a student.”

$$B(G, p1, \phi)$$

Example



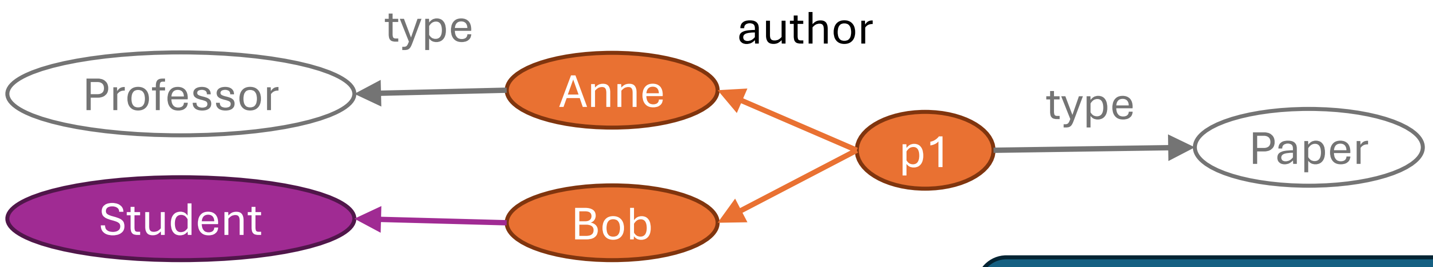
$B(G, \textbf{Anne}, \exists type.\, hasValue(Student))$
$B(G, \textbf{Bob}, \exists type.\, hasValue(Student))$

$$\phi \equiv \geq_1 author.\top \;\wedge\; \leq_1 author.\boxed{\leq_0 type.\, hasValue(Student)}$$

"The node has an author and at most one author is not a student."

$$B(G, p1, \phi)$$

Example



$\cancel{B(G, \textbf{Anne}, \exists type.\, hasValue(Student))}$

$B(G, \textbf{Bob}, \exists type.\, hasValue(Student))$

$$\phi \equiv \geq_1 author.\top \;\wedge\; \leq_1 author.\, \boxed{\leq_0 type.\, hasValue(Student)}$$

"The node has an author and at most one author is not a student."

$$B(G, p1, \phi)$$

Applications and Implementations

https://github.com/MaximeJakubowski/sls_project

https://github.com/Shape-Fragments

Shape Fragments

… as an application of neighborhoods.

We define $\mathbf{Frag}(\boldsymbol{G}, \boldsymbol{S})$ as the union of all neighborhoods of nodes satisfying the shapes from S in G.

Let H be a shape schema, we define:

$$\mathbf{Frag}(\boldsymbol{G}, \boldsymbol{H}) \coloneqq \mathrm{Frag}(G, S)$$

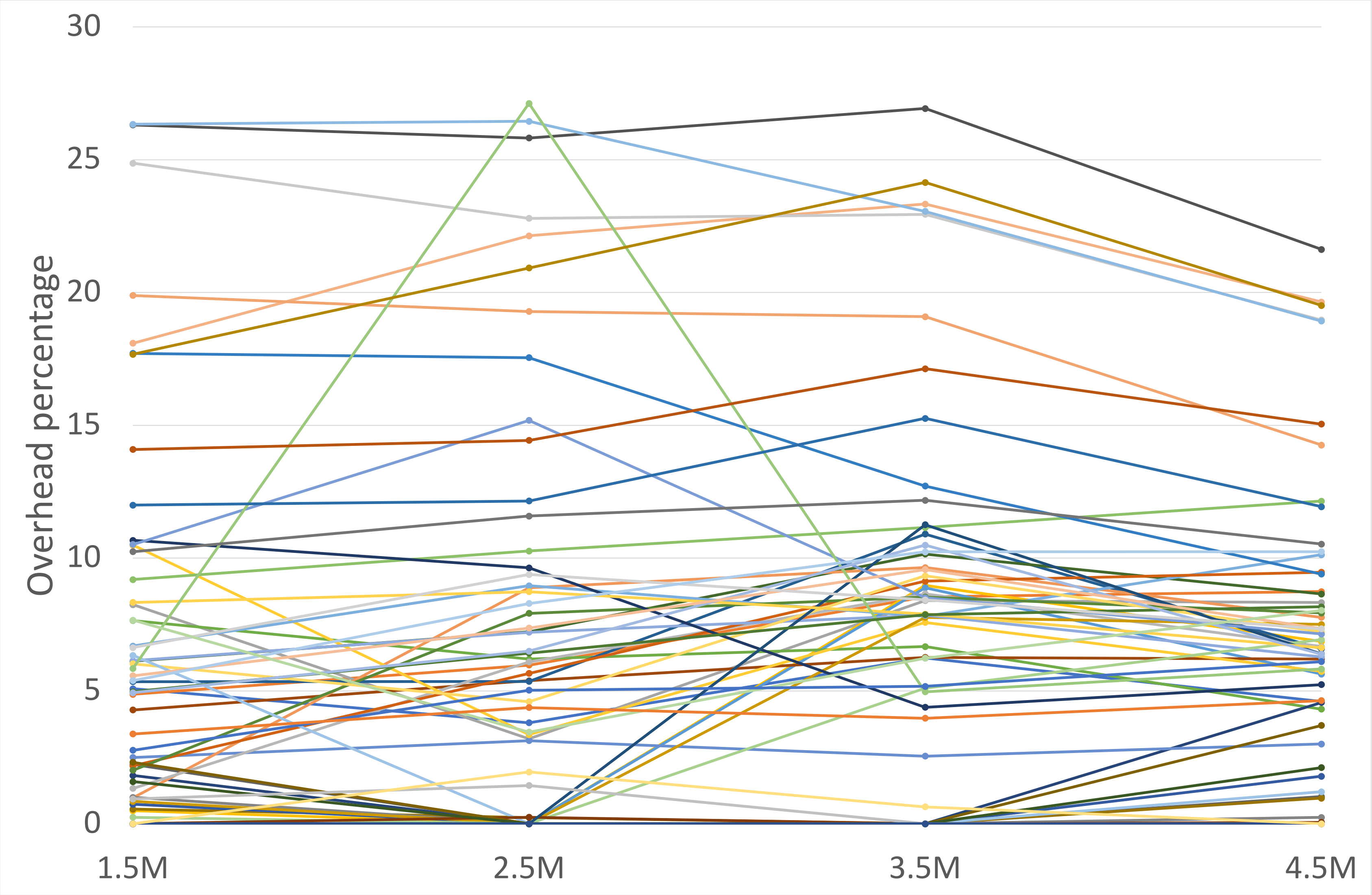
where $S = \{\phi \land \tau \mid \tau \textit{ is the target of } \phi \textit{ in } H\}$

Conformance Property	If a graph G satisfies a schema H, then $\mathrm{Frag}(G, H)$ also conforms to H.

Tools

- PySHACL implementation
- Translation to SPARQL
  - Conformance queries
  - Neighborhood queries

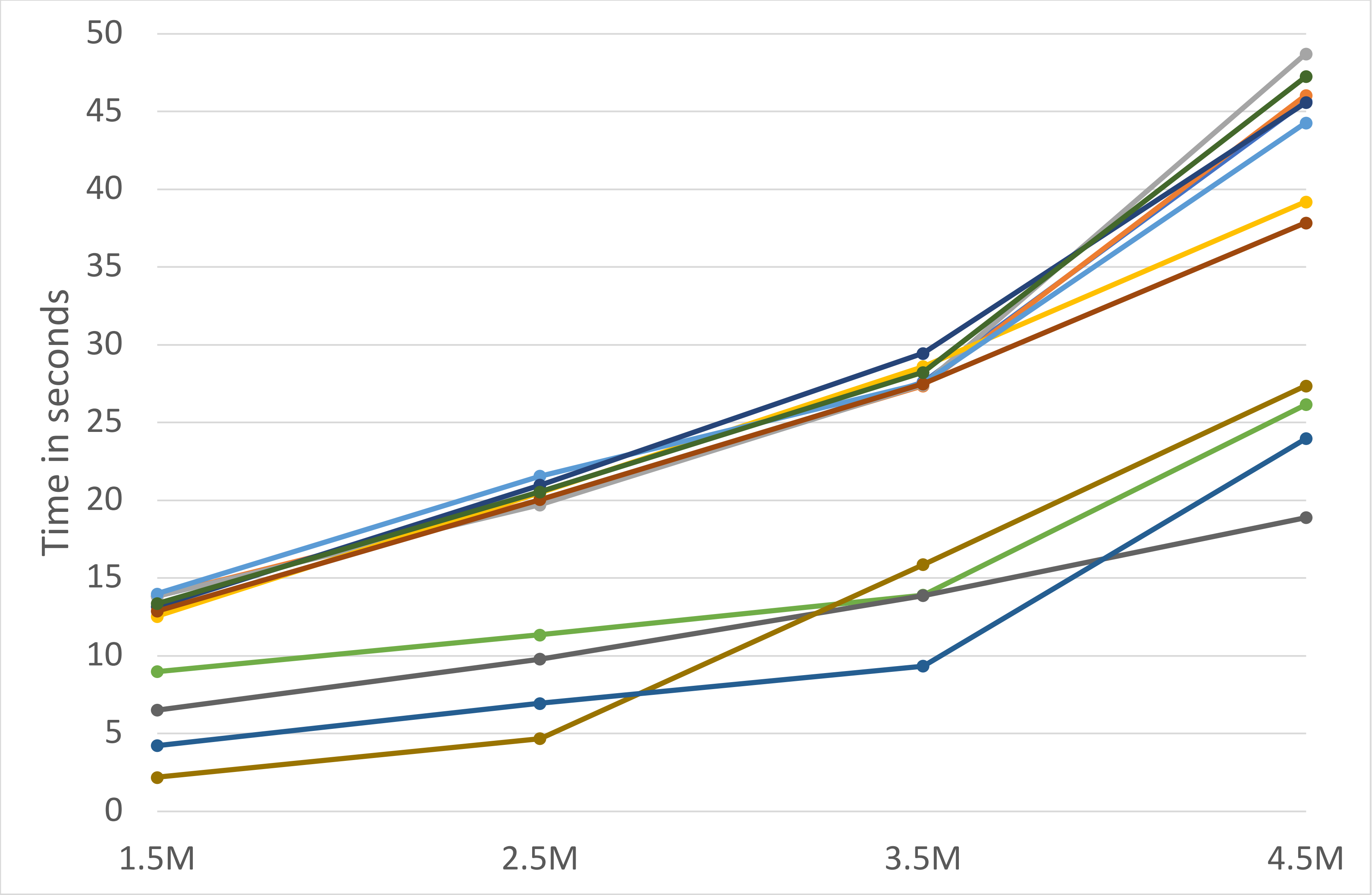
PySHACL Overhead: checking vs retrieval



- 56 shapes
- 1.5M → 4.5M triples

- Average: 10%
- Average ≥ 1s: 15,6%

Retrieving neighborhoods in SPARQL



- 13 shapes
- 1.5M → 4.5M triples

Conclusions

Recommendations and future work

SHACL, SPARQL, and OWL

- Different “views” on what an RDF (data) graph is:
  - OWL sees it as an ABox: a logical theory
  - SPARQL and SHACL see it as a specific interpretation
- SHACL can be formalized in a purely logical fashion, like OWL
  - Both can be viewed as a Description Logic, but with different tasks
  - OWL is mostly concerned with entailment and consistency
  - SHACL is mostly concerned with model checking

Suggestions for the Recommendation (1)

- The W3C leaves recursive shapes undefined
- Semantics from the literature (mostly) agree on:
  - Minimal model semantics for purely *positive* recursion
  - Minimal model semantics for shape definitions using negation in a *stratified* manner

Suggestion	Adopt “stratified negation” in the recommendation

Suggestions for the Recommendation (2)

- There is interest in extending SHACL with new constraint components (see DASH)
- The suggested extensions seem to add some expressive power

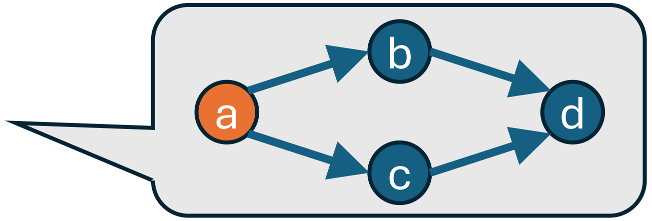
Suggestion	Allow for full property paths in the Property Pair Constraint Components, e.g., full equality and disjointness

Applications for Provenance

- The neighborhood can be used to *describe* a node in a graph
  - You get the *relevant* information
  - Suffiency guarantees that the description fits the shape, even when adding information
  - No need to write additional queries in SPARQL
- Neighborhoods can give an indication of the *coverage* of the schema

Specific Open Questions

- Expressiveness under the different recursive semantics: are the features still primitive?
- Expressiveness of "zero-or-one" paths as a feature: $E? \equiv E \cup id$
- Can we express the "diamond shape"?



- Provenance: formalizing or correcting our notion of minimality and determinism

Research Directions

- The relation between SHACL, ShEx, and PG-Schema
- Extending SHACL to capture the “full” power of RDF as a triple relation
- Exploring alternative SHACL semantics by translation to other languages:
  - Retrieval with SQL
  - Reasoning with IDP
- Usage of SHACL in the wild

Thank you