Data Provenance for SHACL

Thomas Delva (UGent),

Anastasia Dimou (KULeuven),

Maxime Jakubowski &

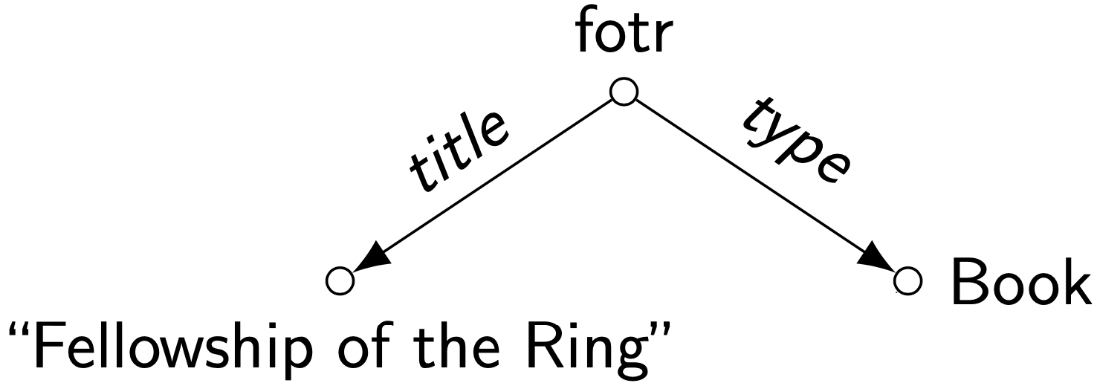
Jan Van den Bussche (UHasselt)

DSI DATA SCIENCE INSTITUTE UHASSELT

WWW.UHASSELT.BE/DSI

SHACL

- **Sha**pes **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.

$\geq_1 type.Book \subseteq \geq_1 title.\top$

Shapes

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

$$\phi \coloneqq \top \mid \{c\} \mid s \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \forall E.\phi \mid \;\geq_n E.\phi$$

$$\mid eq(E, p) \;\mid disj(E, p) \mid closed(Q)$$

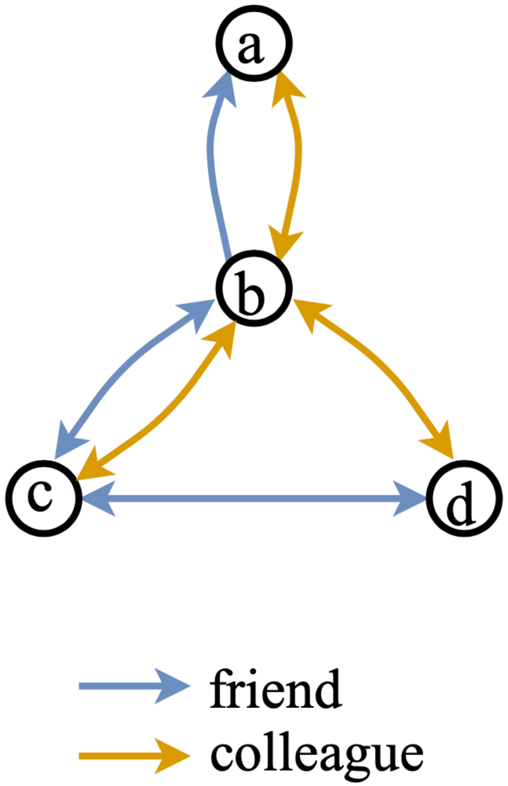
$$E \;\coloneqq p \mid p^- \mid E \cup E \mid E/E \mid E^* \mid E?$$

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

E are regular path queries with inverse and zero-or-one paths

Example shapes

- “Through a path of **friend** edges, the node can reach node d”
  - $\phi \equiv \geq_1 friend^*.\{d\}$
  - b, c, and d satisfy ϕ in G
- “Nodes where **friend**ship is mutual”
  - $\phi \equiv eq(friend, friend^-)$
  - c and d satisfy ϕ in G
- “Nodes who have at least one **colleague** who is also a **friend**”
  - $\phi \equiv \neg disj(friend, colleague)$
  - b and c satisfy ϕ in G



Shape schemas

The main task is to check whether a **graph** conforms to some constraints, not single nodes.

A shape definition is a statement of the form: $s \leftarrow \phi$

A shape schema consists of shape definitions and inclusion statements

$$\phi_t \subseteq \phi_s$$

SHACL allows only the following target shapes ϕ_t :

- Node targets: $\{c\}$
- Class-based targets: $\geq_1 \text{subclassOf}^*. \geq_1 \text{type}. \{c\}$
- Objects-of targets: $\geq_1 p^-. \top$
- Subjects-of targets: $\geq_1 p. \top$

We show that real SHACL can be translated to our formalism

Provenance & Neighborhoods

- Our goal: Provide **provenance** of a shape schema
- Provide a **subgraph** of the data that is 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?

Neighborhood definition

Negation is handled by considering the shapes in **negation normal form**

Simplified shapes (no path expressions):

$$\phi \coloneqq \top \mid \{c\} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \forall p.\phi \mid \geq_n p.\phi \mid eq(p,q) \mid disj(p,q) \mid closed(Q)$$

$$\mid \bot \mid \leq_n p.\phi \mid \neg eq(p,q) \mid \neg disj(p,q) \mid \neg closed(Q)$$

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

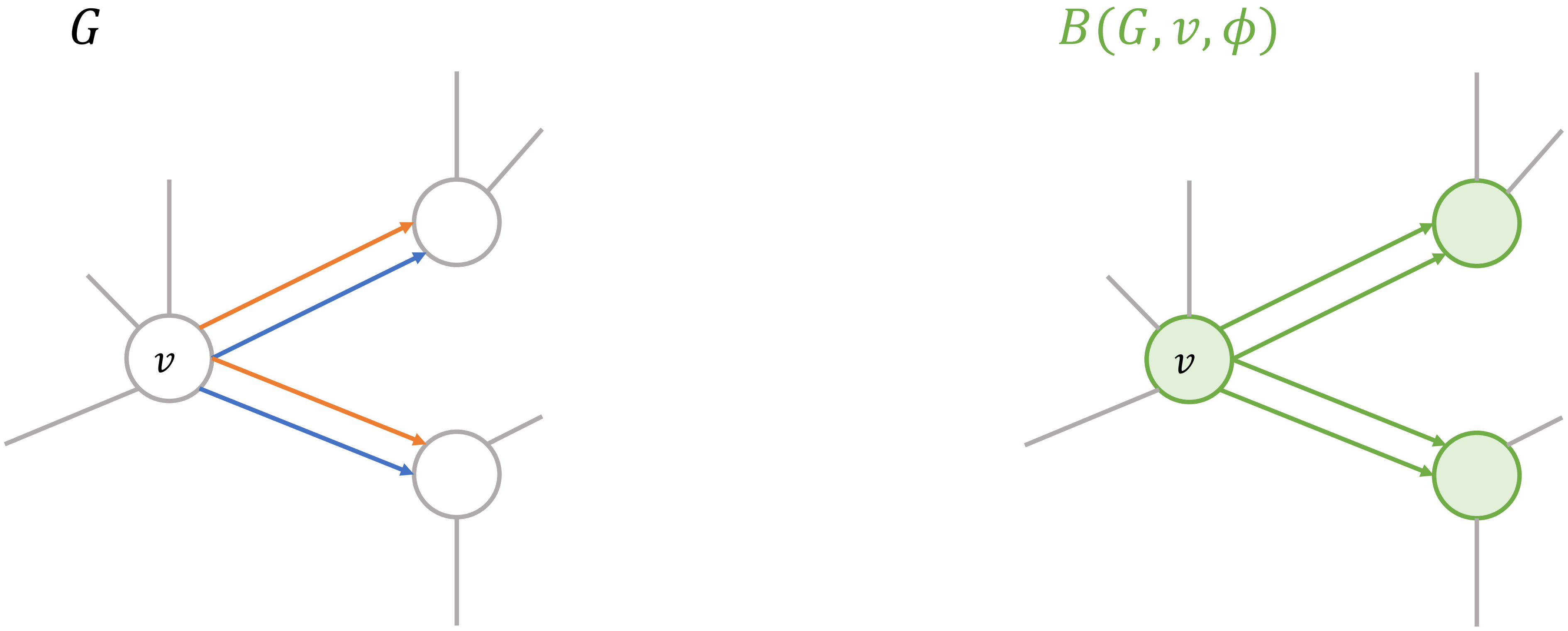
- When the node v does **not** satisfy ϕ in G, the neighborhood is empty
- Shapes that do not use any properties, also have an empty neighborhood

Conjunction and disjunction

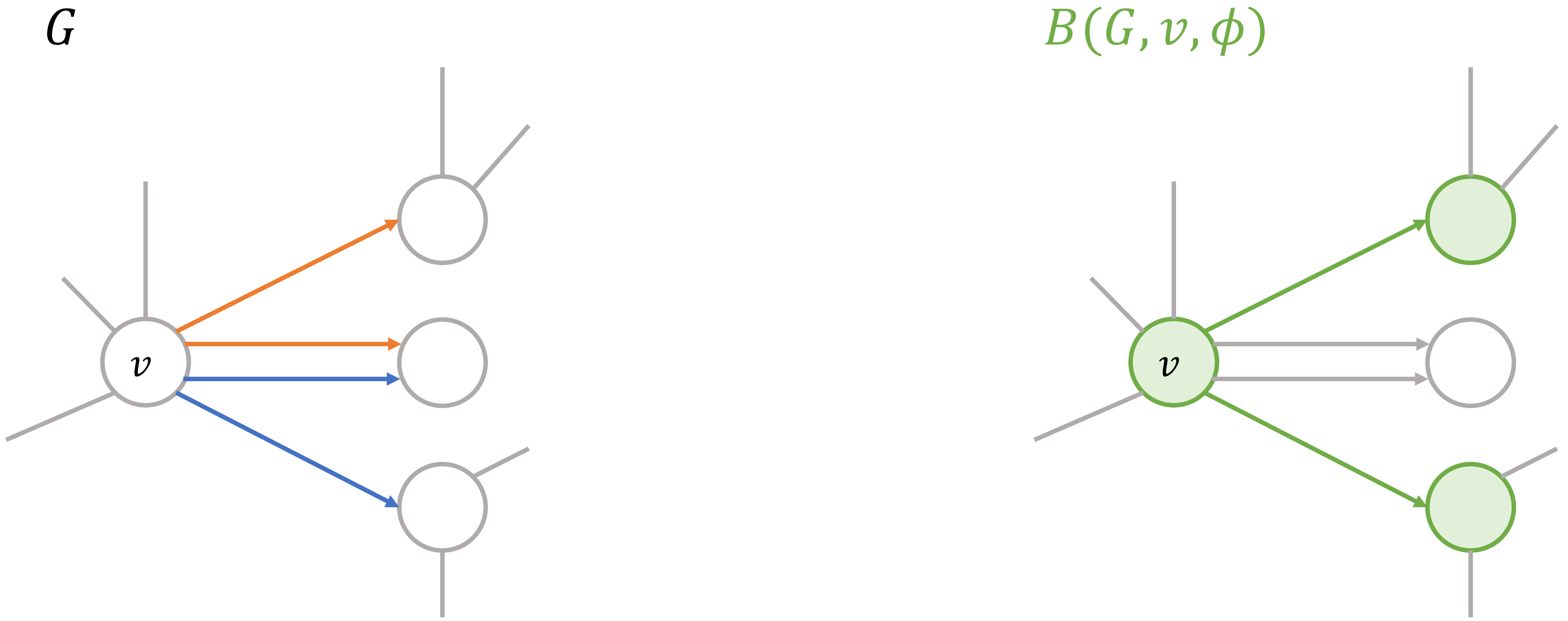
… are defined as the union of neighborhoods

- $B(G, v, \boldsymbol{\phi_1} \wedge \boldsymbol{\phi_2}) = B(G, v, \phi_1) \cup B(G, v, \phi_2)$
- $B(G, v, \boldsymbol{\phi_1} \vee \boldsymbol{\phi_2}) = B(G, v, \phi_1) \cup B(G, v, \phi_2)$

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



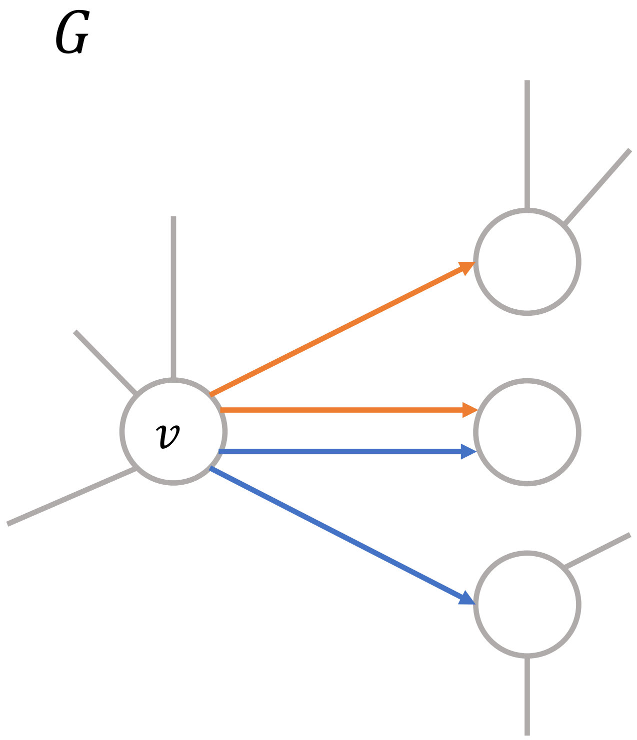
Nonequality: $\neg eq(p, q)$

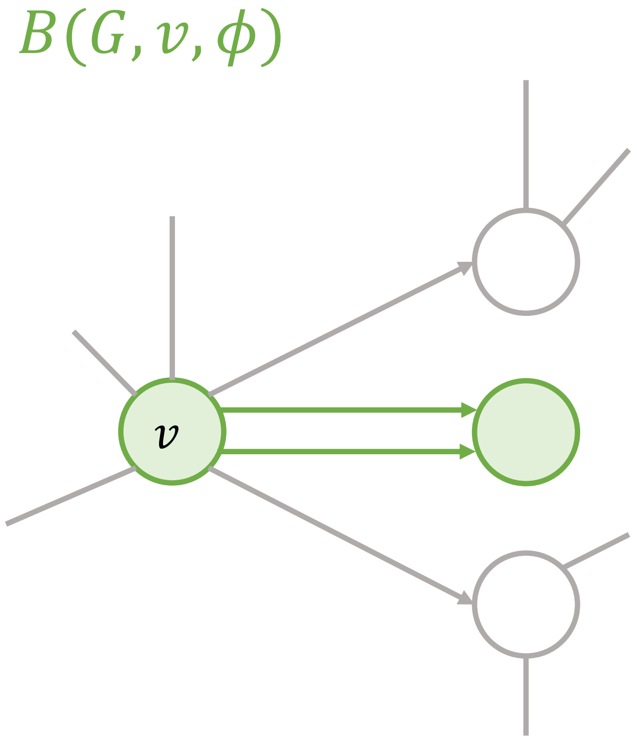


Disjointness: $\boldsymbol{disj(p,q)}$

- Empty neighborhood satisfies the disjointness shape
- Relaxing the definition to add p and q edges does not violate the correctness properties

Nondisjointness: $\neg disj(\boldsymbol{p}, \boldsymbol{q})$

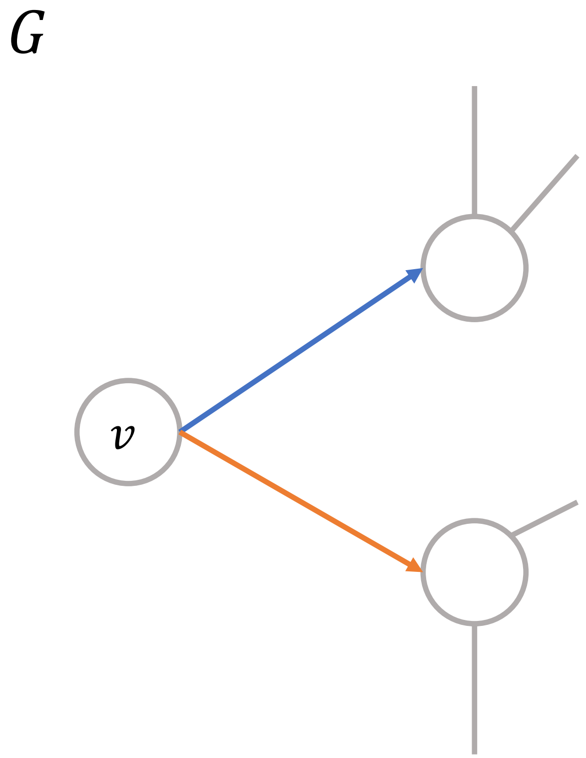


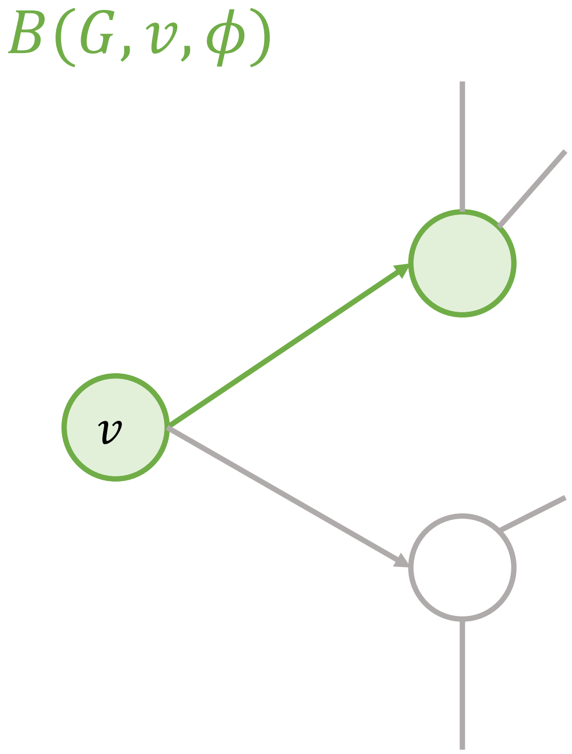


Closedness: $closed(Q)$

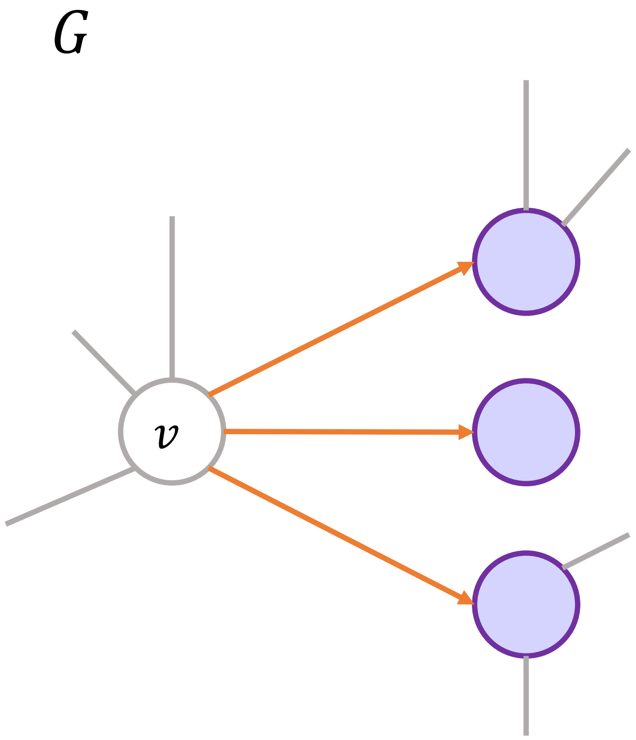
- Empty neighborhood satisfies the closedness shape
- Relaxing the definition to add all edges from Q does not violate the correctness properties

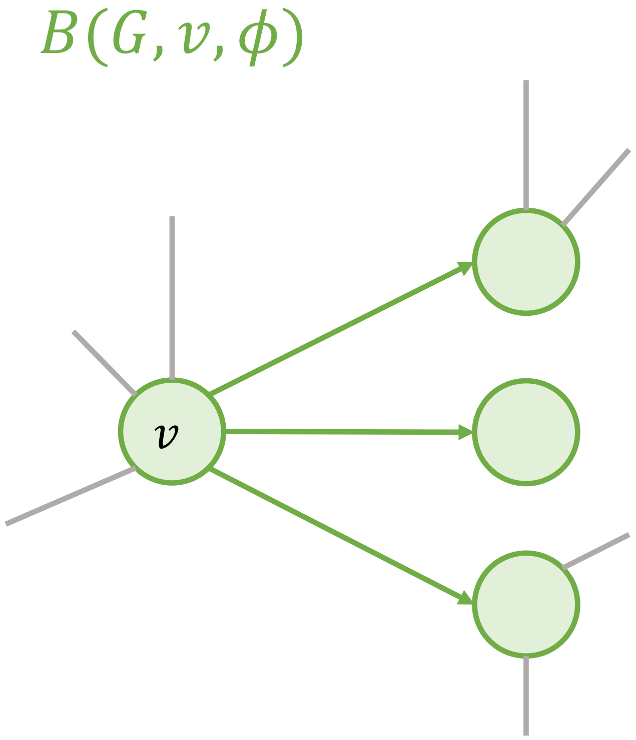
Nonclosure: $\neg closed(\{p\})$





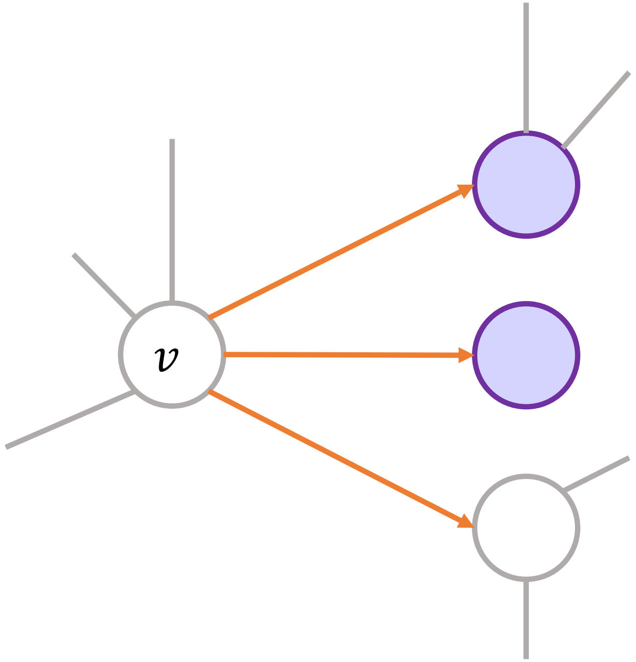
Quantifiers: $\forall \boldsymbol{p}.\,\boldsymbol{\psi}$



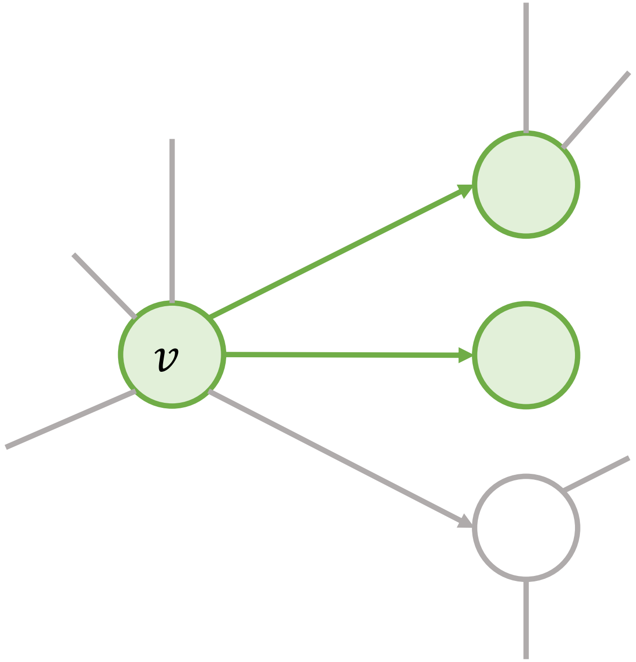


Quantifiers: $\geq_1 \boldsymbol{p}.\,\boldsymbol{\psi}$

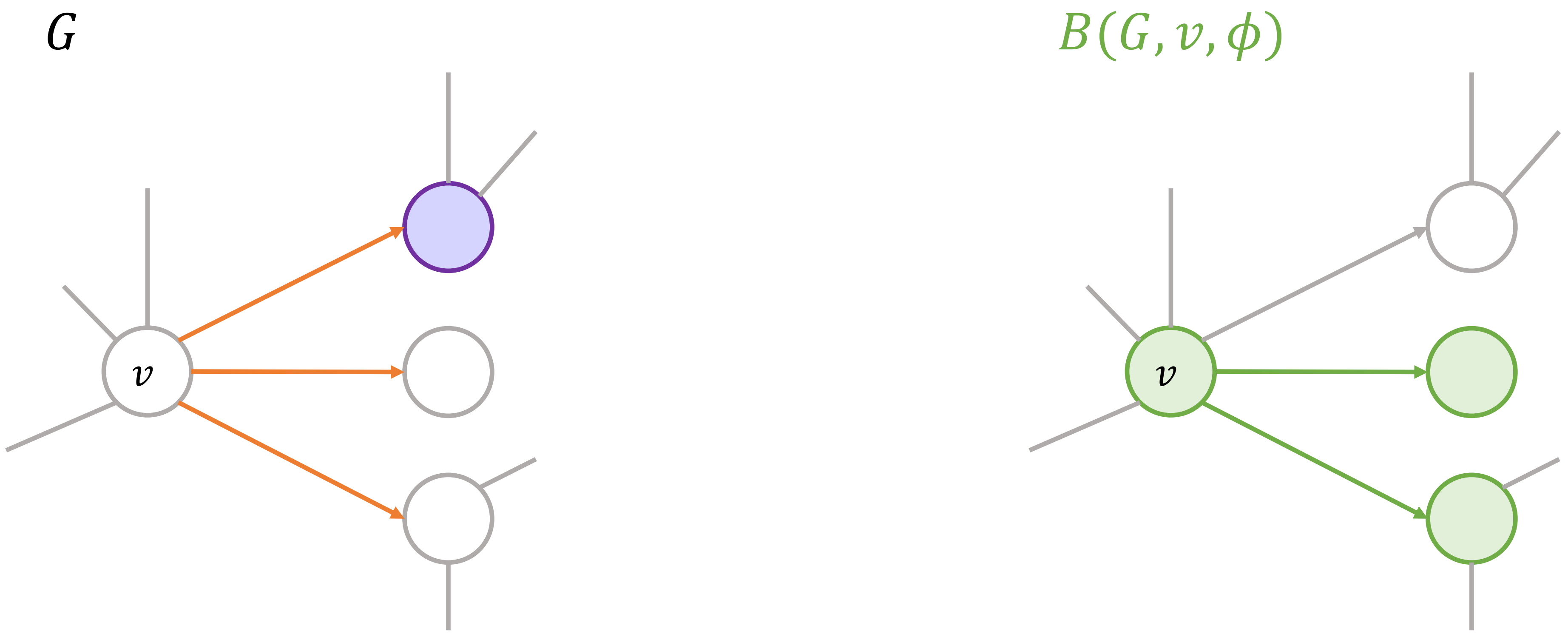
G



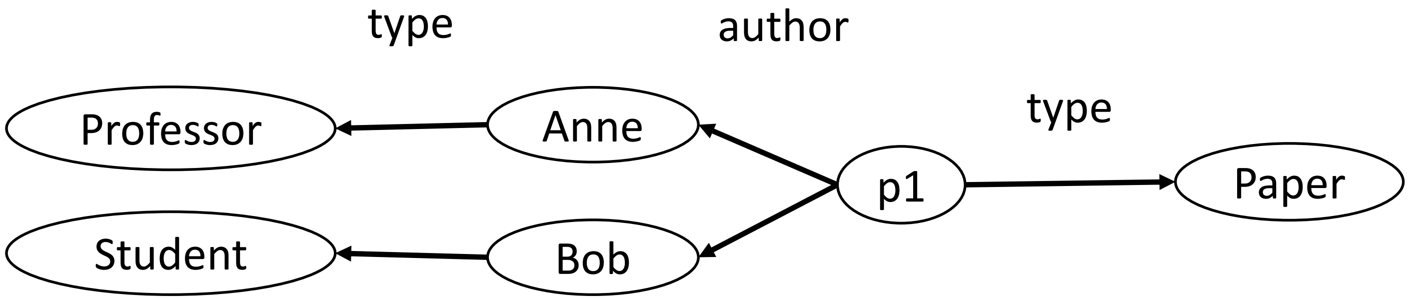
$B(G, v, \phi)$



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



Example

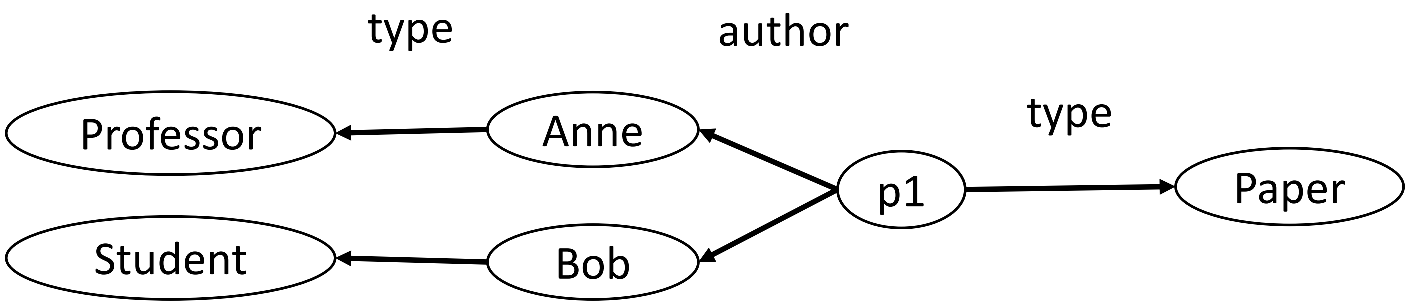


$$\phi \equiv\ \geq_1 author.\top\ \ \wedge\ \ \leq_1 author.\neg\geq_1 type.\{Student\}$$

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

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

Example

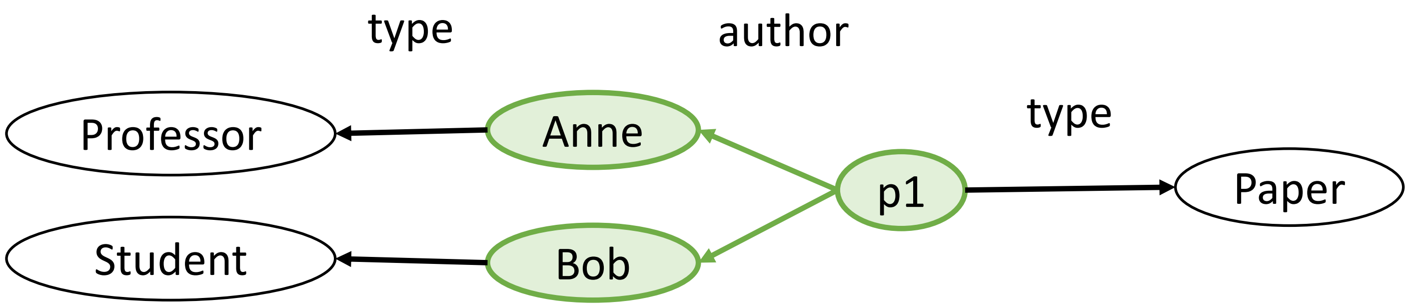


$$\phi \equiv \geq_1 author.\top \;\wedge\; \leq_1 author.\leq_0 type.\{Student\}$$

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

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

Example

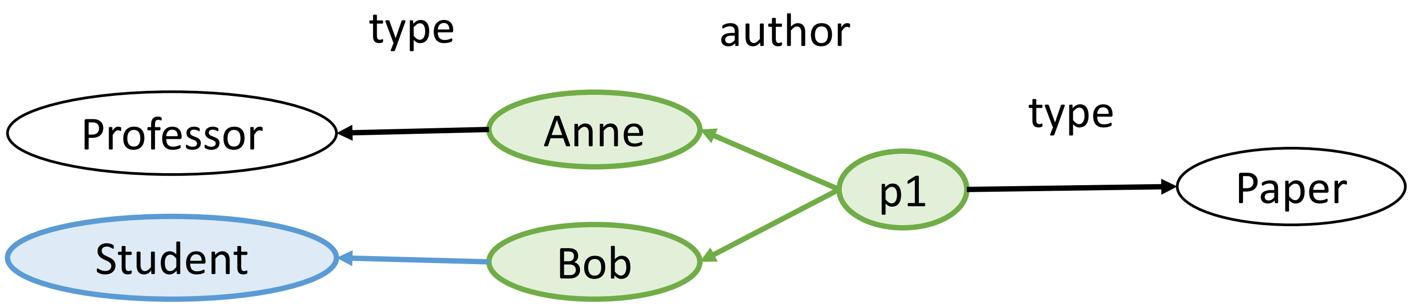


$$\phi \equiv \geq_1 author.\top \ \land \ \leq_1 author.\leq_0 type.\{Student\}$$

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

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

Example



$$\phi \equiv \geq_1 author.\top \;\wedge\; \leq_1 author.\leq_0 type.\{Student\}$$

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

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

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

Correctness properties

We have established:

Sufficiency Theorem. If a node v satisfies a shape ϕ in a graph G, then:

$$v \text{ also satisfies } \phi \text{ in } G' \text{ for any subgraph } G' \subseteq G \text{ s.t. } B(G, v, \phi) \subseteq G'.$$

Conformance Theorem. If a graph G satisfies a schema H, then:

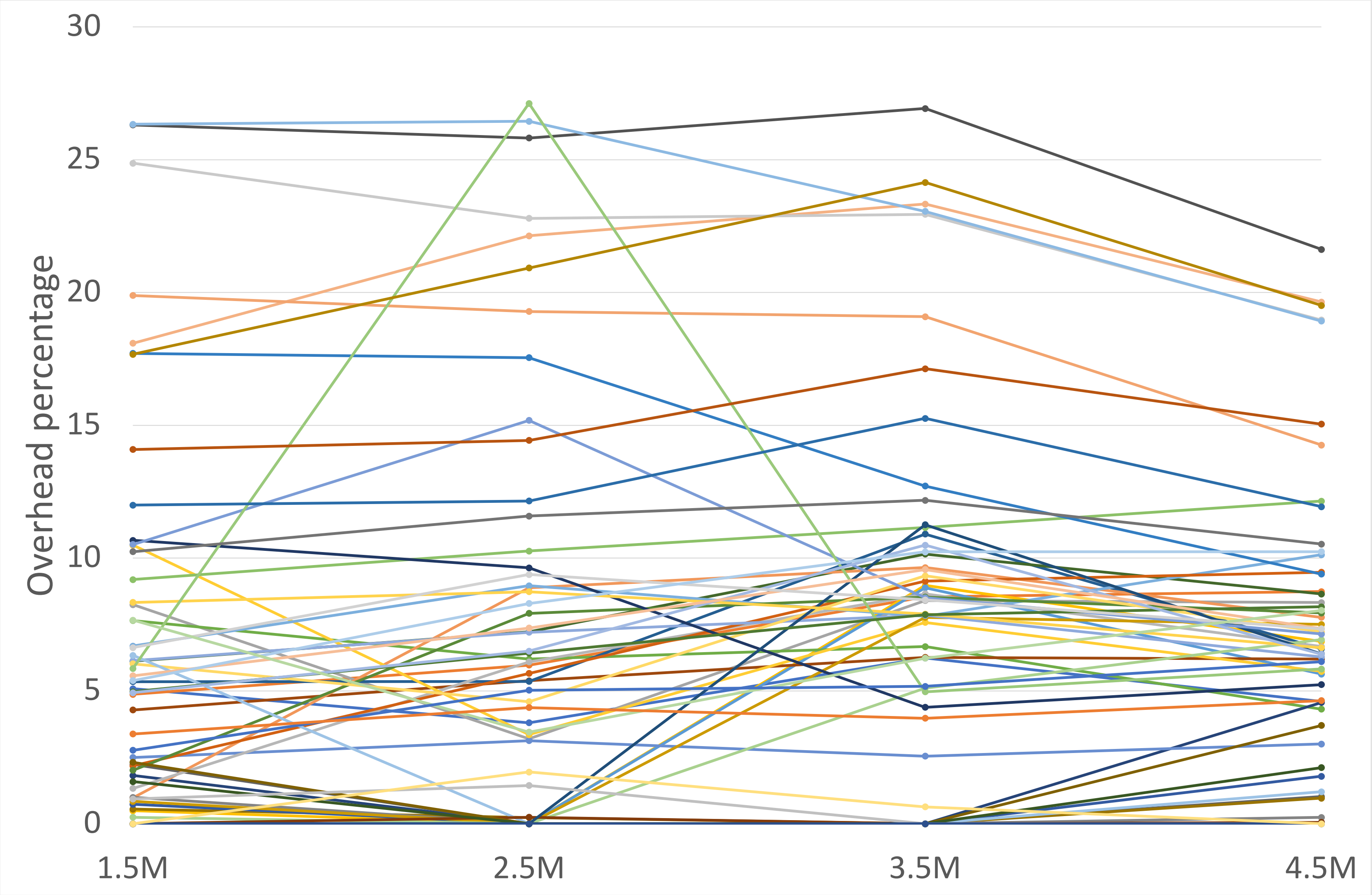
$$\mathrm{Frag}(G, H) \text{ also conforms to } H.$$

Tools

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

https://github.com/shape-fragments

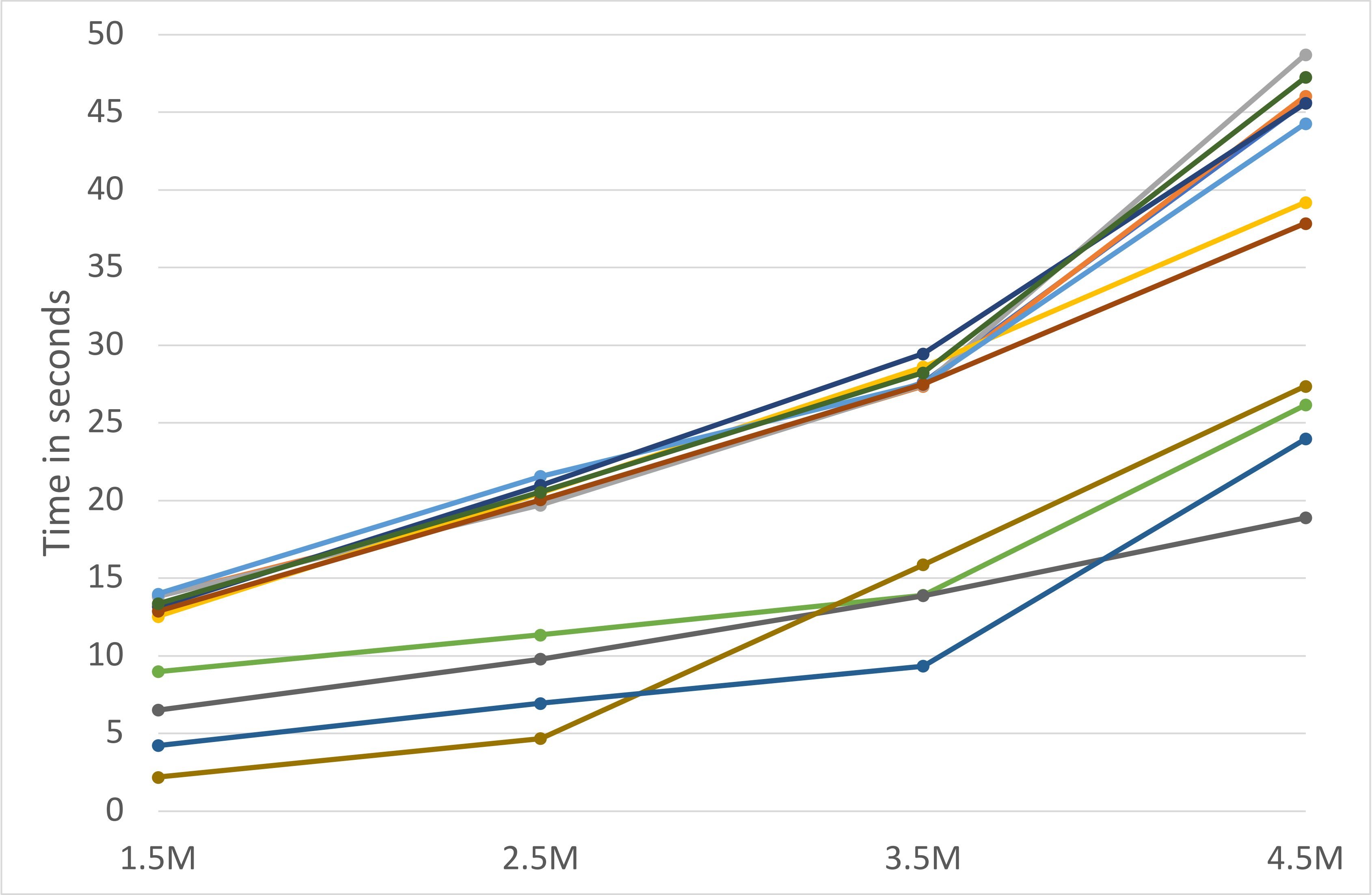
PySHACL overhead



- 56 shapes
- 1.5 → 4.5M triples

- Average: 10%
- Average ≥ 1s: 15,6%

SPARQL query run time



- 13 shapes
- 1.5 → 4.5M triples

Paths

SHACL supports (regular) path expressions:

$$E \coloneqq p \mid p^{-} \mid E \cup E \mid E/E \mid E^{*} \mid E?$$

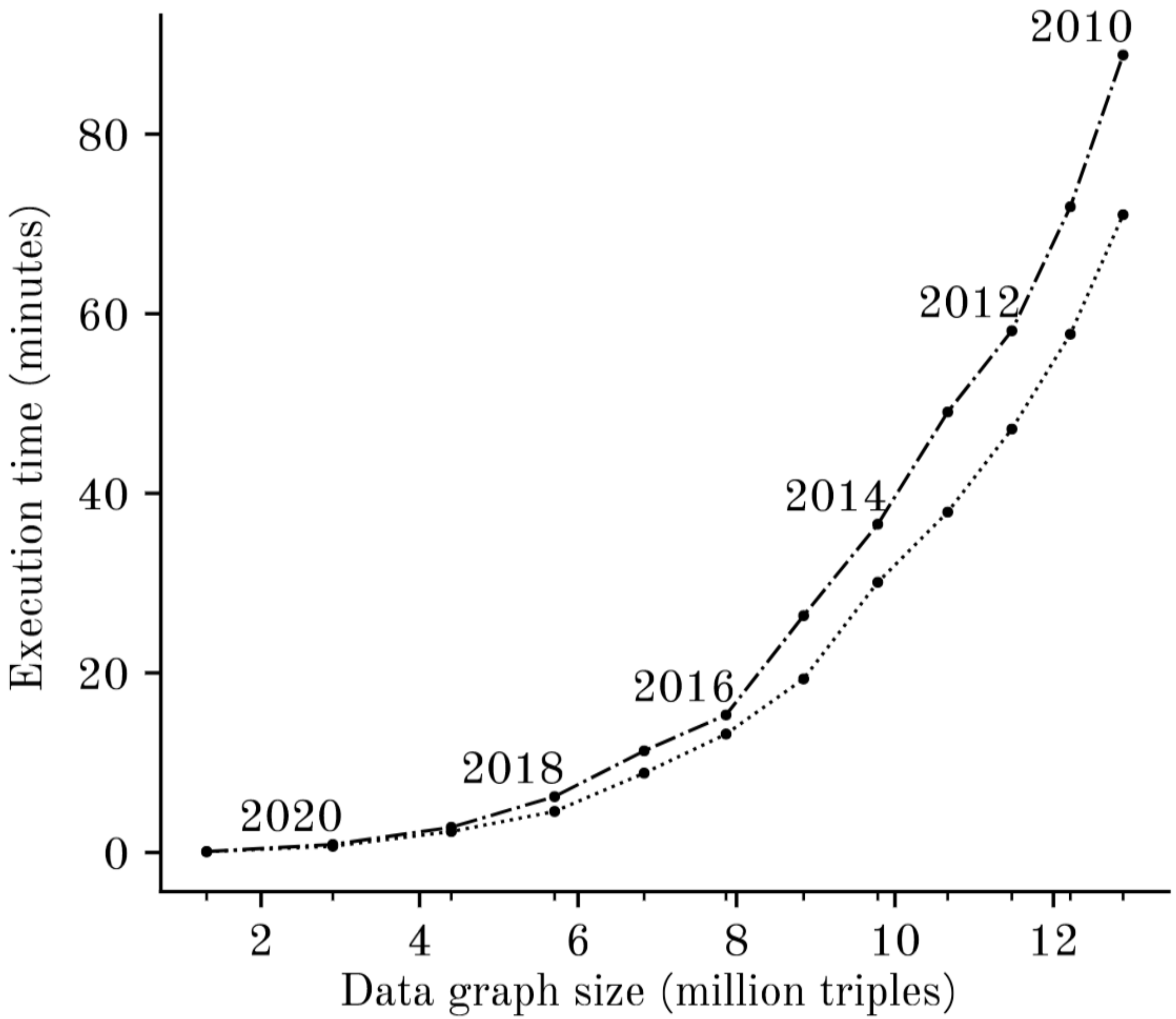
The neighborhood collects all triples on a path.

Example:

$$\geq_1 a^{-}/a/a^{-}/a/a^{-}/a.\{\mathrm{MYV}\}$$

→ retrieves all authors of distance 3 from $\{\mathrm{MYV}\}$, **and** all triples on that path.

Path shape with SPARQL



- Executed on DBLP RDF data
- Run on two SPARQL engines:
  - Jena ARQ (dotted)
  - GraphDB (dashed)

Concluding remarks

- There are many different ‘reasonable’ ways to define subgraphs from a shape
- Different definitions have different properties

- Sufficiency is a well-known property
- What properties can a subgraph have?

  … e.g., can we define subgraphs that are minimally sufficient and unique?

- What do we do with subinstance provenance in presence of negation?