# What’s in a neighborhood?

## Describing nodes in RDF graphs using shapes

Maxime Jakubowski* & **Jan Van den Bussche**

Data Science Institute

Hasselt University, Belgium

*soon to join TU Wien

# What’s in a neighborhood?

## Describing nodes in RDF graphs using shapes

# Role of the **schema** in data management

- Traditional data modeling: **prescriptive** schema
  - data must conform
  - many advantages
- Web data, data integration: **descriptive** schema
  - express expected characteristics of data
  - in RDF graphs, such characteristics are known as **shapes**

# RDF graphs

- Directed graphs with labels on edges
- Edge $x \to y$ with label $p$: **triple** $(x, p, y)$
  - $x$ is called the **subject**
  - $y$ is called the **object**
  - $p$ is called the **property**
- Real RDF:
  - nodes can be of different kinds (IRI, blank, literal)
  - properties can also be nodes

# Shapes in graph data

- Shape:
  - a unary **query** over RDF graphs
    - returns a set of nodes
  - a **predicate** on nodes of RDF graphs
    - node under consideration is called **focus node**
- Examples: let $x$ denote the focus node
  - “$x$ has a phone property, but no email”
  - “$x$ has at least five managed-by edges”
  - “$x$ has a path of friend-edges to the CEO of Apple”
  - “$x$ has no other properties than name, address, and birthdate”

# Shape languages

- In principle, could simply use SPARQL to express shapes
- Yet, two dedicated shape languages:
- **SHACL**
  - Shapes Constraint Language
  - W3C Recommendation
  - logic-based, description logic style
- **ShEx**
  - Shape Expressions
  - shex.io
  - automata/regex based

# What's in a neighborhood?

## Describing nodes in RDF graphs using shapes

1. Introduction
2. **Motivation for neighborhoods**
3. Provenance polynomials
4. Causality
5. Desiderata for neighborhoods
6. Conclusion

# DESCRIBE queries in SPARQL

```
DESCRIBE <http://www.wikidata.org/entity/Q19660>
```

# DESCRIBE queries in SPARQL

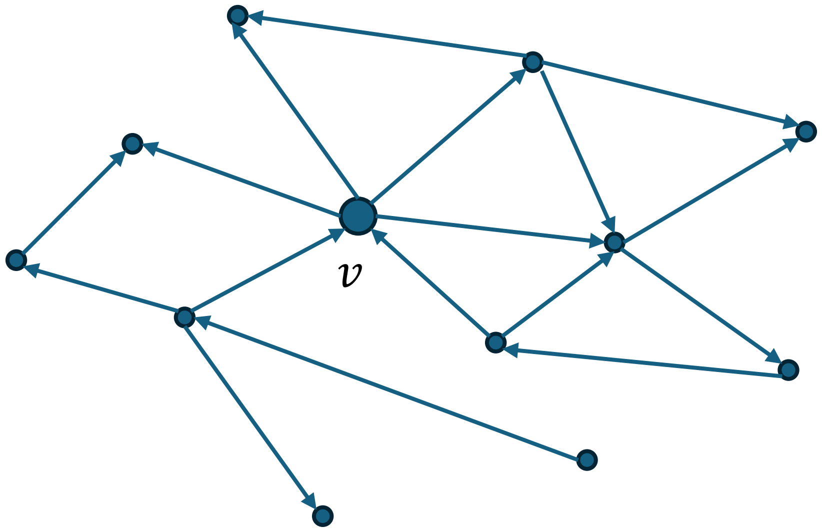
```
DESCRIBE <http://www.wikidata.org/entity/Q19660>
```

- Returns a **subgraph**: all edges to and from the node
- **“Ball of radius 1”**

# DESCRIBE queries in SPARQL

```
DESCRIBE <http://www.wikidata.org/entity/Q19660>
```

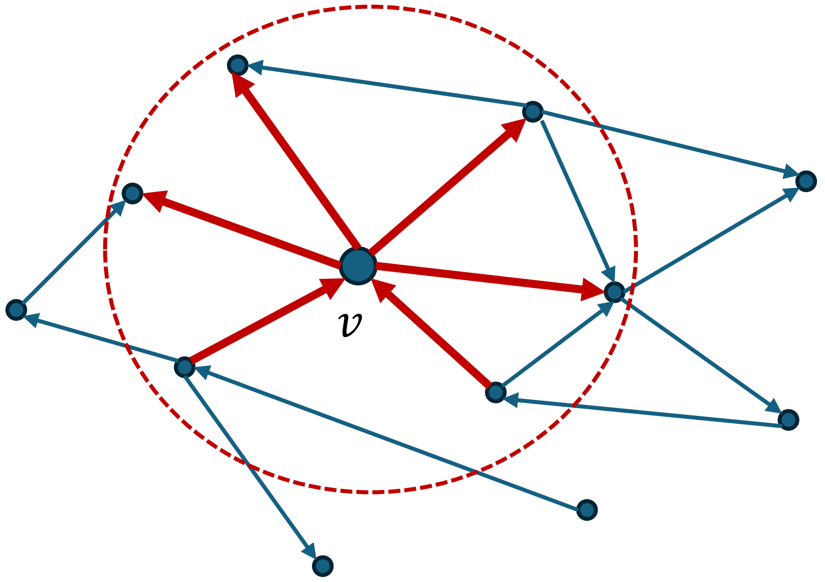
- Returns a **subgraph**: all edges to and from the node
- $B^G(v, 1)$: **"Ball of radius 1"** in graph $G$



# DESCRIBE queries in SPARQL

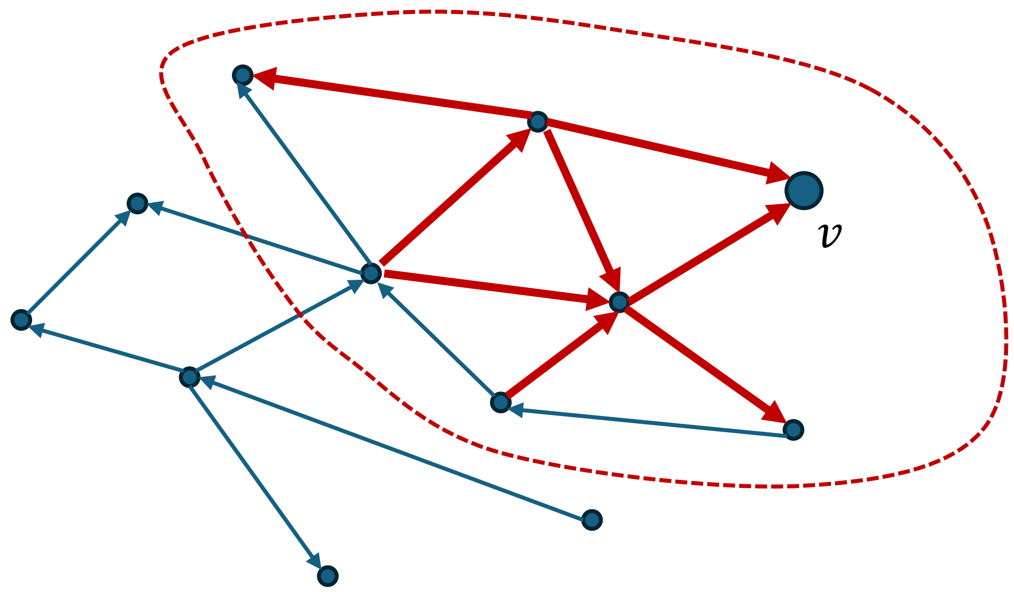
```
DESCRIBE <http://www.wikidata.org/entity/Q19660>
```

- Returns a **subgraph**: all edges to and from the node
- $B^G(v, 1)$: **“Ball of radius 1”** in graph $G$



# DESCRIBE queries in SPARQL

```
DESCRIBE <http://www.wikidata.org/entity/Q19660>
```




- $B^G(v, 2)$, etc.

# DESCRIBE USING SHAPE?

- Balls $B^G(v, k)$ where $k = 1, 2, \ldots$ give a concept of neighborhood that is **too crude**
- Using a shape $\sigma$, can we define a subgraph $B^G(v, \sigma)$?

**Example:**

- Let $\sigma$ be "$v$ has at least one email edge, and at most one name edge"
- What should $B^G(v, \sigma)$ consist of?
  - If $v$ does not satisfy $\sigma$: the empty graph
  - Otherwise: intuition: at least one of the email edges. **Anything else?**

# Motivations for neighborhoods

- **Provenance** for shapes: $B^G(v, \sigma)$ can serve an an **explanation** why $v$ satisfies $\sigma$
- **Repairing** shape violations: if $v$ does **not** satisfy $\sigma$, then $B^G(v, \neg\sigma)$ can point out edges that should be added
- Knowledge graph **subsets** [Labra Gayo et al.]: given a shape $\sigma$, build a subset of $G$ by taking union of all $B^G(v, \sigma)$
  - Also known as "shape fragments" [EDBT 2023 paper on provenance for SHACL]
  - Basically, using shapes as a **retrieval** mechanism

# What’s in a neighborhood?

## Describing nodes in RDF graphs using shapes

1. Introduction
2. Motivation for neighborhoods
3. **Provenance polynomials for SHACL**
4. Causality
5. Desiderata for neighborhoods
6. Conclusion

# SHACL as a logic [Corman et al.]

- Syntax of **shapes** $\phi$:

$$\begin{aligned}\phi ::= \;& \top \mid \bot \mid \mathit{hasValue}(c) \mid \mathit{test}(t) \mid \mathit{eq}(p,r) \mid \mathit{disj}(p,r) \mid \mathit{closed}(P) \\ \mid \;& \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \geq_k p.\phi \mid \leq_k p.\phi \mid \forall p.\phi\end{aligned}$$

- Semantics, node $a$ in graph $G$ satisfies $\phi$ if:

| $\phi$ | $G, a \models \phi$ if: |
|---|---|
| $\mathit{hasValue}(c)$ | $a = c$ |
| $\mathit{test}(t)$ | $a$ satisfies $t$ |
| $\mathit{eq}(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are equal |
| $\mathit{disj}(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are disjoint |
| $\mathit{closed}(P)$ | for all triples $(s,p,o) \in G$ with $s = a$ we have $p \in P$ |
| $\geq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \geq k$ |
| $\leq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \leq k$ |
| $\forall p.\psi$ | every $b \in [\![p]\!]^G(a)$ satisfies $G, b \models \psi$ |

# SHACL RDF syntax vs SHACL logical syntax

- W3C SHACL has an RDF syntax of **“shapes graphs”**
  - RDF syntax allows exchange and management of schema information using standard Web tools
- **Logical syntax** proposal by Corman et al.
  - More convenient for writing complex shapes, logical analysis
  - Extended to cover the **full** SHACL specification
  - [Delva, Dimou, Jakubowski, Van den Bussche EDBT 2023]
- Tool **SLS** developed

```
:WorkshopShape sh:property [
  sh:path :author ; sh:qualifiedMinCount 1 ;
  sh:qualifiedValueShape [ sh:class :Student ] ] .
```

Shapes graph in RDF

SLS tool

$\geq_1 \text{:author} \,.\, \geq_1 \text{rdf:type} \,.\, hasValue(\text{:Student})$

Logical syntax

https://github.com/MaximeJakubowski/sls_project

# SHACL as a logic [Corman et al.]

- Syntax of **shapes** $\phi$:

$$\begin{aligned}\phi ::= \;& \top \mid \bot \mid \mathit{hasValue}(c) \mid \mathit{test}(t) \mid \mathit{eq}(p,r) \mid \mathit{disj}(p,r) \mid \mathit{closed}(P) \\ \mid \;& \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \geq_k p.\phi \mid \leq_k p.\phi \mid \forall p.\phi\end{aligned}$$

- Semantics, node $a$ in graph $G$ satisfies $\phi$ if:

| $\phi$ | $G, a \models \phi$ if: |
|---|---|
| $\mathit{hasValue}(c)$ | $a = c$ |
| $\mathit{test}(t)$ | $a$ satisfies $t$ |
| $\mathit{eq}(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are equal |
| $\mathit{disj}(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are disjoint |
| $\mathit{closed}(P)$ | for all triples $(s,p,o) \in G$ with $s = a$ we have $p \in P$ |
| $\geq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \geq k$ |
| $\leq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \leq k$ |
| $\forall p.\psi$ | every $b \in [\![p]\!]^G(a)$ satisfies $G, b \models \psi$ |

# SHACL as a logic [Corman et al.]

- Syntax of **shapes** $\phi$: $c$: constant

$$\begin{aligned}\phi ::= {} & \top \mid \bot \mid \mathit{hasValue}(c) \mid \mathit{test}(t) \mid \mathit{eq}(p, r) \mid \mathit{disj}(p, r) \mid \mathit{closed}(P) \\ \mid {} & \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \geq_k p.\phi \mid \leq_k p.\phi \mid \forall p.\phi\end{aligned}$$

- Semantics, node $a$ in graph $G$ satisfies $\phi$ if:

| $\phi$ | $G, a \models \phi$ if: |
|---|---|
| $\mathit{hasValue}(c)$ | $a = c$ |
| $\mathit{test}(t)$ | $a$ satisfies $t$ |
| $\mathit{eq}(p, r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are equal |
| $\mathit{disj}(p, r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are disjoint |
| $\mathit{closed}(P)$ | for all triples $(s, p, o) \in G$ with $s = a$ we have $p \in P$ |
| $\geq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \geq k$ |
| $\leq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \leq k$ |
| $\forall p.\psi$ | every $b \in [\![p]\!]^G(a)$ satisfies $G, b \models \psi$ |

# SHACL as a logic [Corman et al.]

- Syntax of **shapes** $\phi$:

$t$: node test

$$\phi ::= \top \mid \bot \mid hasValue(c) \mid test(t) \mid eq(p,r) \mid disj(p,r) \mid closed(P)$$
$$\mid \quad \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \geq_k p.\phi \mid \leq_k p.\phi \mid \forall p.\phi$$

- Semantics, node $a$ in graph $G$ satisfies $\phi$ if:

| $\phi$ | $G, a \models \phi$ if: |
|---|---|
| $hasValue(c)$ | $a = c$ |
| $test(t)$ | $a$ satisfies $t$ |
| $eq(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are equal |
| $disj(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are disjoint |
| $closed(P)$ | for all triples $(s,p,o) \in G$ with $s = a$ we have $p \in P$ |
| $\geq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \geq k$ |
| $\leq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \leq k$ |
| $\forall p.\psi$ | every $b \in [\![p]\!]^G(a)$ satisfies $G, b \models \psi$ |

# SHACL as a logic [Corman et al.]

- Syntax of **shapes** $\phi$:

$p, r$: properties

$$
\begin{aligned}
\phi ::= \; & \top \mid \bot \mid \mathit{hasValue}(c) \mid \mathit{test}(t) \mid \mathit{eq}(p, r) \mid \mathit{disj}(p, r) \mid \mathit{closed}(P) \\
\mid \; & \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \geq_k p.\phi \mid \leq_k p.\phi \mid \forall p.\phi
\end{aligned}
$$

- Semantics, node $a$ in graph $G$ satisfies $\phi$ if:

| $\phi$ | $G, a \models \phi$ if: |
|---|---|
| $\mathit{hasValue}(c)$ | $a = c$ |
| $\mathit{test}(t)$ | $a$ satisfies $t$ |
| $\mathit{eq}(p, r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are equal |
| $\mathit{disj}(p, r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are disjoint |
| $\mathit{closed}(P)$ | for all triples $(s, p, o) \in G$ with $s = a$ we have $p \in P$ |
| $\geq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \geq k$ |
| $\leq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \leq k$ |
| $\forall p.\psi$ | every $b \in [\![p]\!]^G(a)$ satisfies $G, b \models \psi$ |

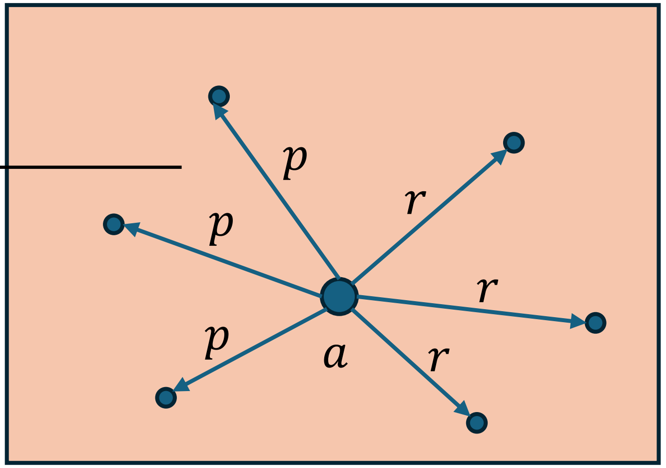
# SHACL as a logic [Corman et al.]

- Syntax of **shapes** $\phi$:

$$\begin{aligned}\phi ::= \; & \top \mid \bot \mid \mathit{hasValue}(c) \mid \mathit{test}(t) \mid \mathit{eq}(p,r) \mid \mathit{disj}(p,r) \mid \mathit{closed}(P) \\ \mid \; & \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \geq_k p.\phi \mid \leq_k p.\phi \mid \forall p.\phi\end{aligned}$$

- Semantics, node $a$ in graph $G$ satisfies $\phi$ if:

| $\phi$ | $G, a \models \phi$ if: |
|---|---|
| $\mathit{hasValue}(c)$ | $a = c$ |
| $\mathit{test}(t)$ | $a$ satisfies $t$ |
| $\mathit{eq}(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are equal |
| $\mathit{disj}(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are disjoint |
| $\mathit{closed}(P)$ | for all triples $(s,p,o) \in G$ with $s = a$ we have $p \in P$ |
| $\geq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \geq k$ |
| $\leq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \leq k$ |
| $\forall p.\psi$ | every $b \in [\![p]\!]^G(a)$ satisfies $G, b \models \psi$ |



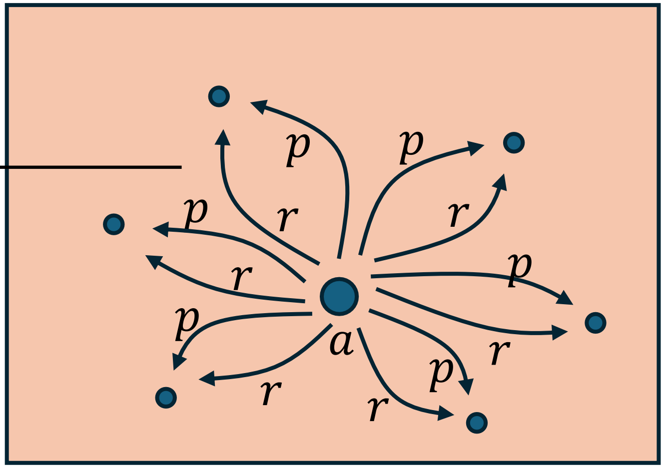
# SHACL as a logic [Corman et al.]

- Syntax of **shapes** $\phi$:

$$\begin{aligned}\phi ::= \; & \top \mid \bot \mid hasValue(c) \mid test(t) \mid eq(p,r) \mid disj(p,r) \mid closed(P) \\ \mid \; & \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \geq_k p.\phi \mid \leq_k p.\phi \mid \forall p.\phi\end{aligned}$$

- Semantics, node $a$ in graph $G$ satisfies $\phi$ if:

| $\phi$ | $G, a \models \phi$ if: |
|---|---|
| $hasValue(c)$ | $a = c$ |
| $test(t)$ | $a$ satisfies $t$ |
| $eq(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are equal |
| $disj(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are disjoint |
| $closed(P)$ | for all triples $(s,p,o) \in G$ with $s = a$ we have $p \in P$ |
| $\geq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \geq k$ |
| $\leq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \leq k$ |
| $\forall p.\psi$ | every $b \in [\![p]\!]^G(a)$ satisfies $G, b \models \psi$ |



# SHACL as a logic [Corman et al.]

- Syntax of **shapes** $\phi$:

$P$: set of properties

$$\begin{aligned}\phi ::= \ & \top \mid \bot \mid \mathit{hasValue}(c) \mid \mathit{test}(t) \mid \mathit{eq}(p,r) \mid \mathit{disj}(p,r) \mid \mathit{closed}(P) \\ \mid \ & \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \geq_k p.\phi \mid \leq_k p.\phi \mid \forall p.\phi\end{aligned}$$

- Semantics, node $a$ in graph $G$ satisfies $\phi$ if:

| $\phi$ | $G, a \models \phi$ if: |
|---|---|
| $\mathit{hasValue}(c)$ | $a = c$ |
| $\mathit{test}(t)$ | $a$ satisfies $t$ |
| $\mathit{eq}(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are equal |
| $\mathit{disj}(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are disjoint |
| $\mathit{closed}(P)$ | for all triples $(s,p,o) \in G$ with $s = a$ we have $p \in P$ |
| $\geq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \geq k$ |
| $\leq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \leq k$ |
| $\forall p.\psi$ | every $b \in [\![p]\!]^G(a)$ satisfies $G, b \models \psi$ |

*closed*(name,email,birthdate)

# SHACL as a logic [Corman et al.]

- Syntax of **shapes** $\phi$:

$$\phi ::= \top \mid \bot \mid hasValue(c) \mid test(t) \mid eq(p,r) \mid disj(p,r) \mid closed(P) \\ \mid \phi \wedge \phi \mid \phi \vee \phi \mid \neg\phi \mid \geq_k p.\phi \mid \leq_k p.\phi \mid \forall p.\phi$$

Description logics

- Semantics, node $a$ in graph $G$ satisfies $\phi$ if:

| $\phi$ | $G, a \models \phi$ if: |
|---|---|
| $hasValue(c)$ | $a = c$ |
| $test(t)$ | $a$ satisfies $t$ |
| $eq(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are equal |
| $disj(p,r)$ | the sets $[\![p]\!]^G(a)$ and $[\![r]\!]^G(a)$ are disjoint |
| $closed(P)$ | for all triples $(s,p,o) \in G$ with $s = a$ we have $p \in P$ |
| $\geq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \geq k$ |
| $\leq_k p.\psi$ | $\#\{b \in [\![p]\!]^G(a) \mid G, b \models \psi\} \leq k$ |
| $\forall p.\psi$ | every $b \in [\![p]\!]^G(a)$ satisfies $G, b \models \psi$ |

# Intermezzo: expressiveness of SHACL

- Each of $eq$, $disj$, $closed$ is **primitive**
  - cannot be expressed in terms of the other language constructs
- We simplified a bit: SHACL allows regular expression **property paths** $E$ and constraints $eq(E, p)$ and $disj(E, p)$
  - Allowing even more general $eq(E_1, E_2)$ would **increase** expressive power
  - Same for $disj(E_1, E_2)$
- [Bogaerts, Jakubowski, Van den Bussche, ICDT and LMCS]

- **Recursion** is left unspecified in W3C Recommendation
  - can be added as in logic programming

# Intermezzo: computing shape queries

- The **shape query** for a shape $\sigma$:
  - Input: an RDF graph $G$
  - Output: all nodes $v \in G$ such that $G, v \vDash \sigma$
- Dedicated SHACL engines exist, e.g., TopQuadrant
- SHACL can also be compiled to SPARQL [Corman et al.]
  - Without property paths, even to **SQL** [ISWC 2024]
- SHACL is **strictly** weaker than SPARQL
- **Not** expressible, focus node $x$:
  - "$x$ is part of a 4-clique"
  - "$x$ has more $p$-edges than $r$-edges"

# Provenance polynomials

- Provenance polynomials for **database queries**:
  - $v$ a result of a query $Q$ on database $D$
  - $\boldsymbol{pol(D,v,Q)}$: compact representation of all proofs why $v \in Q(D)$
- Multivariate polynomial, unknowns are **facts** in $D$

**Example:** let $Q$ be select R.A from R(A,B) join S(B)

R

| A | B |
|---|---|
| a | b |
| a | c |
| a | d |

S

| B |
|---|
| b |
| c |
| e |

$$pol(D,a,Q) = [a,b] \cdot [b] + [a,c] \cdot [c]$$

- Known for positive relational algebra, first-order logic, Datalog

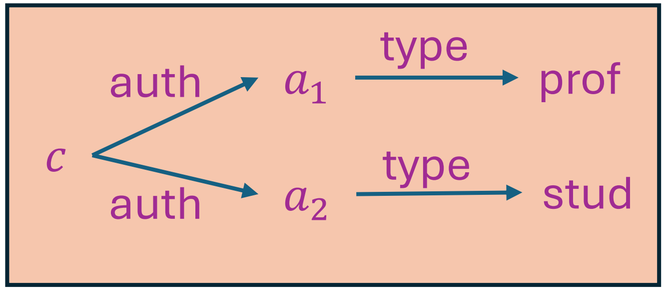
  [Green, Karvounarakis, Tannen], [Grädel, Tannen], [Deutch, Milo, Roy, Tannen]

# Provenance for SHACL

- [Dannert, Grädel]: provenance polynomials for ALC
  - Simplest description logic
  - Unknowns are **triples**
  - $pol(G, a, \phi_1 \wedge \phi_2) = pol(G, a, \phi_1) \cdot pol(G, a, \phi_2)$
  - $pol(G, a, \phi_1 \vee \phi_2) = pol(G, a, \phi_1) + pol(G, a, \phi_2)$
  - $pol(G, a, \exists p. \psi) = \sum_{(a,p,b) \in G} [a, p, b] \cdot pol(G, b, \psi)$
  - $pol(G, a, \forall p. \psi) = \prod_{(a,p,b) \in G} [a, p, b] \cdot pol(G, b, \psi)$
- **Crucial property:** $a$ satisfies $\phi$ iff polynomial **not** zero
- We must extend this:
  - to **counting** qualifiers $\geq_k p. \psi$ and $\leq_k p. \psi$
  - to $eq$, $disj$, $closed$

# Polynomials for $\geq_k p.\psi$ and $\leq_k p.\psi$

- Idea:
  - $\geq_1 p.\psi$ is same as $\exists p.\psi$
  - $\leq_0 p.\psi$ is same as $\forall p.\neg\psi$
  - Adapt to larger $k$ (see paper)



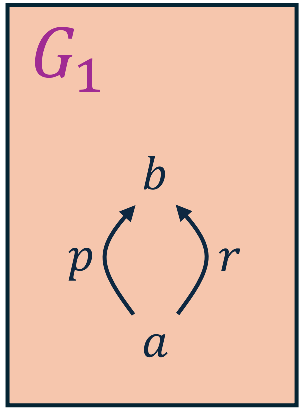

**Example:** $\phi = \leq_1 \text{auth}.\leq_0 \text{type}.\,hasValue(\text{stud})$

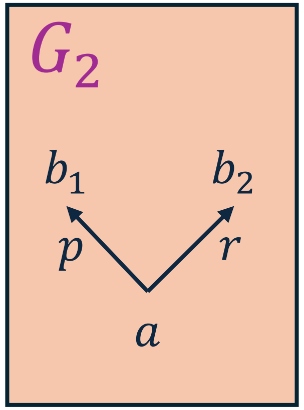
"$x$ has at most one author who is not a student"

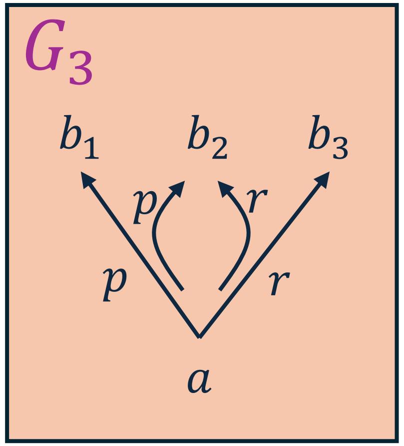
$$
\begin{aligned}
pol(G, c, \phi) = {} & [c,\text{auth}, a_1] \cdot [a_1,\text{type,prof}] \cdot 0 \\
& + [c,\text{auth}, a_2] \cdot [a_2,\text{type,stud}] \cdot 1 \\
= {} & [c,\text{auth}, a_2] \cdot [a_2,\text{type,stud}]
\end{aligned}
$$

# Polynomials for $eq, disj$

- For $disj$ and $\neg eq$, we also need negated triples (**absence** of triple)



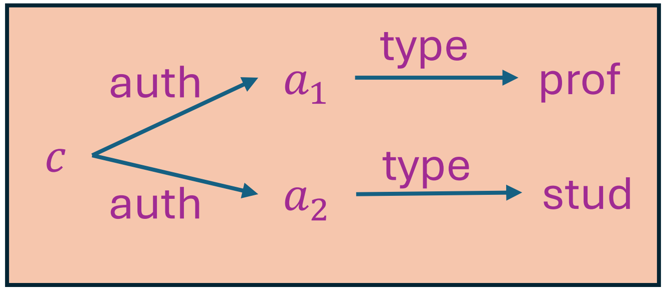




$$pol(G_1, a, eq(p,r)) = [a,p,b] \cdot [a,r,b] \cdot [a,r,b] \cdot [a,p,b]$$
$$pol(G_2, a, disj(p,r)) = [a,p,b_1] \cdot \overline{[a,r,b_1]} \cdot [a,r,b_2] \cdot \overline{[a,p,b_2]}$$
$$pol(G_3, a, \neg disj(p,r)) = [a,p,b_2] \cdot [a,r,b_2]$$
$$pol(G_1, a, eq(p,r)) = [a,p,b_1] \cdot \overline{[a,r,b_1]} + [a,r,b_3] \cdot \overline{[a,p,b_3]}$$

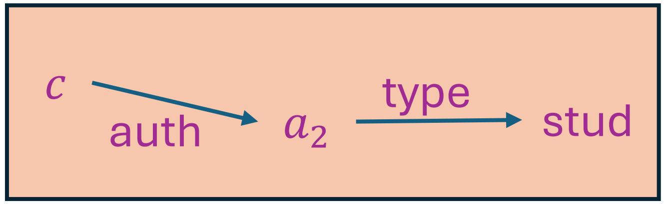
# From polynomials to neighborhoods

- Let $G, v \vDash \sigma$. How should we define $B(G, v, \sigma)$?
- $\boldsymbol{B_{tok}}$:
  - Calculate provenance polynomial $pol(G, v, \sigma)$
  - Return all **positive** triples occurring as unknowns (**tokens**) in the polynomial
  - We could also take **all** triples, both negative and positive
    - [Bogaerts, Jakubowski, Van den Bussche PODS 2024]
- $\boldsymbol{B_{mon}}$:
  - Pick a monomial (term) from the polynomial
  - Return all triples in there
  - **Non-deterministic!**

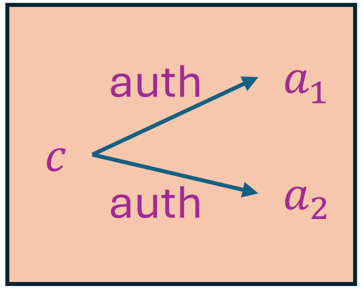
# $B_{tok}$ and $B_{mon}$



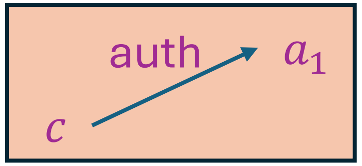

- **Example:** $\phi = \leq_1$auth.$\leq_0$type.$hasValue$(stud)
  - $pol(G, c, \phi) = [c, \text{auth}, a_2] \cdot [a_2, \text{type,stud}]$
  - So, $\boldsymbol{B_{tok}(G, c, \phi) = B_{mon}(G, c, \phi)} =$

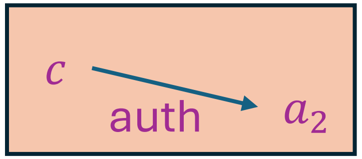



- **Example:** $\sigma = \geq_1$auth. T
  - $pol(G, c, \sigma) = [c\text{,auth}, a_1] + [c\text{,auth}, a_2]$ so $\boldsymbol{B_{tok}(G, c, \sigma)} =$



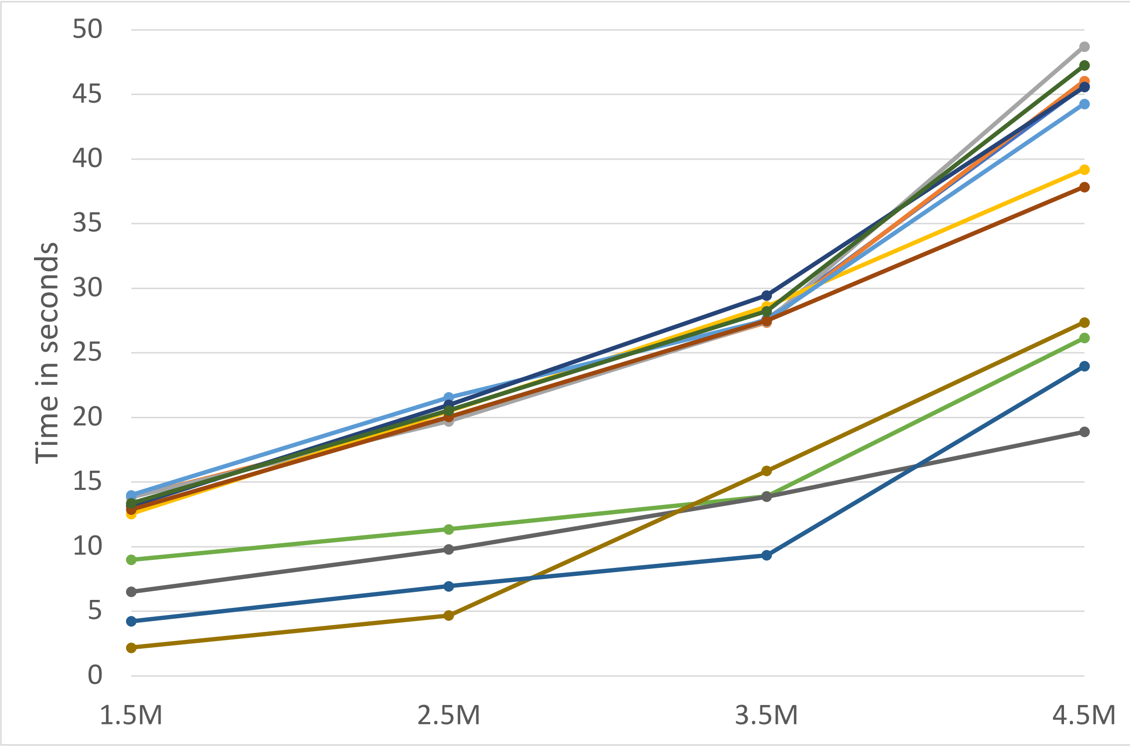

  - For $\boldsymbol{B_{mon}(G, c, \sigma)}$ two possibilities:



# Remark: computing neighborhoods

- Computing $B_{tok}$ can done in SPARQL
- Extends known translations from SHACL to SPARQL
- Works even with property paths
- [Delva, Dimou, Jakubowski, Van den Bussche, EDBT 2023]



# What’s in a neighborhood?

## Describing nodes in RDF graphs using shapes

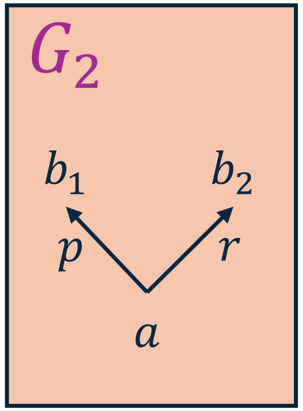
# Causality as alternative to provenance polynomials

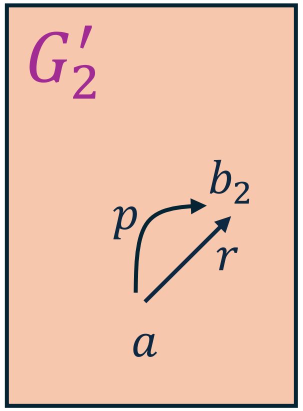
- Neighborhood $B(G, v, \phi)$ is supposed to **explain** why $G, v \vDash \phi$
- $B_{tok}$ and $B_{mon}$ do that, in a sense (see later)

- Halpern-Pearl causality: formal definition of **cause** for $G, v \vDash \phi$
- **Supercause:** set $C$ of positive, negative triples from $G$ such that after "flipping" $C$ in $G$, node $v$ no longer satisfies $\phi$
- **Cause:** minimal supercause

- **Note:** A repair for violating $\phi$ is the same as a cause of $\neg\phi$!
  - [Ahmetaj et al., ISWC 2022]

# Causality: example




- We have $G_2, a \vDash disj(p, r)$
- $\{\overline{[a,p,b_2]}, [a,p,b_1]\}$ is a **supercause**:
  - Flipping this in $G_2$ yields $\boldsymbol{G_2'}$: $p$ and $r$ no longer disjoint $a_3'$
  - Not a **cause**: deleted $[a,p,b_1]$ is unnecessary
  - The two (minimal) **causes** are:
    - $\{\overline{[a,p,b_2]}\}$ (insert $[a,p,b_2]$)
    - $\{\overline{[a,r,b_1]}\}$ (insert $[a,r,b_1]$)

# Neighborhoods by causality?

- **Tempting** to define a neighborhood to be a cause
  - We will see soon this is not "sufficient"
- What **does** work: from $B_{tok}$, only keep **causally relevant** triples
  - Belonging to some cause
  - High computational complexity

# What’s in a neighborhood?

## Describing nodes in RDF graphs using shapes

# Is there a “best” definition of neighborhood?

- **No.** Several **desiderata**, incompatible.
- **Sufficiency:** Natural desidaratum in provenance research [Glavic]
  - If $G, v \models \sigma$, then also $\boldsymbol{B(G, v, \sigma)}, v \models \sigma$
  - “Node $v$ should still satisfy the shape in its shape neighborhood”
- **Theorem:** $B_{tok}$, $B_{mon}$, and causally relevant restrictions, are sufficient
- **Determinism**
- **Minimality**, e.g., minimally sufficient neighborhoods
  - not deterministic: “focus node has at least an email **or** a phone property”
- [Bogaerts, Jakubowski, Van den Bussche PODS 2024]

# Conclusions and further research

- Shapes can be used for **more** than descriptive schemas
- Retrieve subgraphs!
- No single approach is “best”, but we can follow some **principles**

1. Extend SHACL to full RDF (where properties can be nodes)
    - Or even RDF-star?
2. Neighborhoods for **property paths**
    - Can become very large
3. Empirical research needed why SHACL is “better” than SPARQL
    - Theoretical complexity is lower
4. Compare SHACL neighborhoods to ShEx neighborhoods [Labra Gayo et al.]

# References—thanks Bart Bogaerts, Thomas Delva, Anastasia Dimou


- Maxime Jakubowski, Jan Van den Bussche: *Compiling SHACL into SQL*. **ISWC 2024**
- Bart Bogaerts, Maxime Jakubowski, Jan Van den Bussche: *Postulates for provenance: Instance-based provenance for first-order logic*. **PODS 2024**
- Thomas Delva, Anastasia Dimou, Maxime Jakubowski, Jan Van den Bussche: *Data provenance for SHACL*. **EDBT 2023**
- Bart Bogaerts, Maxime Jakubowski, Jan Van den Bussche: *Expressiveness of SHACL features and extensions for full equality and disjointness tests*. **ICDT 2022, LMCS 2024**
- Bart Bogaerts, Maxime Jakubowski, Jan Van den Bussche: *SHACL: A description logic in disguise*. **LPNMR 2022**
- Bart Bogaerts, Maxime Jakubowski: *Fixpoint semantics for recursive SHACL*. **ICLP 2021**