

## What's in a Neighborhood? Describing Nodes in RDF Graphs Using Shapes

Peer-reviewed author version

JAKUBOWSKI, Maxime & VAN DEN BUSSCHE, Jan (2024) What's in a Neighborhood? Describing Nodes in RDF Graphs Using Shapes. In: Kirrane, S.; Simkus, M.; Soylu, A.; Roman, D. (Ed.). Rules and reasoning, RULEML+RR 2024, SPRINGER INTERNATIONAL PUBLISHING AG, p. 1 -13.

DOI: 10.1007/978-3-031-72407-7\_1

Handle: <http://hdl.handle.net/1942/44783>

# What’s in a neighborhood?

## Describing nodes in RDF graphs using shapes

Maxime Jakubowski<sup>1</sup>[0000–0002–7420–1337] and Jan Van den  
Bussche<sup>1</sup>[0000–0003–0072–3252]

Data Science Institute, Universiteit Hasselt  
www.mjakubowski.info  
vdbuss.github.io

**Abstract.** There are several situations where it is desirable to be able to extract a subgraph from an RDF graph, based on a node in the graph, and given a shape that the node conforms to. Such a subgraph can be called a neighborhood. We discuss desiderata for neighborhoods, and compare different possible definitions. We show connections with data provenance and causality. We also show how to obtain provenance polynomials for the shape constraint language SHACL from the work of Danert and Grädel.

## 1 Introduction

A lot of data on the Web is present in the form of RDF graphs [23, 1]. For our purposes, we can think of an RDF graph simply as an edge-labeled, directed graph [25]. In reality the RDF data model is more general [11]; notably, it does not distinguish between nodes and edge labels.

RDF graphs can be interpreted from two perspectives. One perspective considers an RDF graph to be a description of knowledge: a logical theory. Thereto, appropriate vocabularies have been introduced for representing logical constructs: RDFS for relationships and class hierarchies, and OWL for more expressive description logic formulas [3]. The other perspective is not in opposition to the first, but is more primitive in that it does not really try to interpret the graph. The only interpretation that is still done is that the graph nodes are not only merely abstract (blank nodes) but can also represent concrete data values (literal nodes) or can identify actual resources on the Web (IRI nodes).

Here we are taking this second perspective. In this perspective, RDF is a schemaless data model: unlike in classical data modeling, one does not first design and prescribe a complex schema, and then only considers instances conforming to that schema. Instead, every edge-labeled graph is a potential instance. Of course, there are still many advantages to having a schema, or structural information about the data, even if it is only partial. These advantages are well known and lie in efficiency, usability, data quality, etc.

For this reason, schema languages were soon proposed for RDF graphs, notably, SHACL and ShEx [21, 26, 7, 23]. The emphasis is on *descriptive* rather than

*prescriptive* schemas: not the form or structure of the entire data is intended to be prescribed completely, but rather, one describes various *shapes* that can be expected to be present in the data.

Shapes can be thought of as possibly complex conditions on nodes  $v$  in a graph: which labels can, or must, appear on edges involving  $v$ ? Which atomic data types should literal nodes have when they are linked to  $v$ ? Which further shapes should the neighbors of  $v$  satisfy? Indeed, in SHACL, shapes are essentially logic formulas, which are always evaluated on nodes, as in description logics [5, 24]. In ShEx, shapes are expressed more in a manner reminiscent of regular expressions and tree automata. In this paper, we will work with the convenient logical formalisation of SHACL started by Corman et al. [10] and extended to full SHACL in our work with Delva and Dimou [14].

We note that a SHACL schema (known as a shapes graph) does not only contain definitions of shapes, but it also allows to pair these shapes to simply node-selecting queries (so-called targets). An RDF graph  $G$  is then said to conform to a schema if for each shape–target pair  $\sigma$ – $\tau$ , every node  $v$  selected by  $\tau$  in  $G$  satisfies  $\sigma$  in  $G$ . The task of checking this (known as validation) will be less of a focus in the present paper; we will focus more on shapes in themselves.

## 2 Neighborhoods

In various situations, it is desirable to be able to extract, from an RDF graph  $G$ , a *subgraph*  $B$  of  $G$ , based on some given node  $v$ , and in accordance to some given shape  $\sigma$ . We discuss some of these next.

**DESCRIBE queries** The RDF query language SPARQL has a DESCRIBE query form which, given a node  $v$ , returns some RDF graph purporting to describe that node. Such queries are used often in practice (e.g., [8]). The SPARQL standard leaves open how these queries should actually be answered. Yet, most engines return all edges of the graph  $G$  in which  $v$  is involved, effectively returning a subgraph of  $G$  which is often called the *neighborhood* of  $v$ . We might denote it by  $B^G(v, 1)$ .

The 1 in the above notation stands for the distance the edges go from  $v$  in the graph, which indeed equals one; the  $B$  stands for ball, following terminology and notation from metric topology. More generally, for any natural number  $k$ , we could define  $B^G(v, k)$  to be the subgraph of  $G$  consisting of all edges lying on paths of length at most  $k$  that contain  $v$ .

It seems more useful, however, if we could use a describing shape  $\sigma$  instead of just some distance  $k$ , something that has been discussed within the community [27]. The idea would be to have a notion of neighborhood  $B^G(v, \sigma)$ , defined for any given node  $v$  that satisfies  $\sigma$  in graph  $G$ . Intuitively, we would like this to be a subgraph of  $G$ , containing the relevant edges from  $G$  that cause  $v$  to satisfy  $\sigma$ , but preferably not more.

As a simple example, suppose  $\sigma$  is the shape “ $v$  has at least one outgoing email edge, and at most one outgoing name edge”. If  $v$  satisfies  $\sigma$ , it seems

intuitively clear that  $B^G(v, \sigma)$  will need to contain at least one of the email edges emanating from  $v$ . Should the neighborhood contain additional edges as well? That is less clear.

**Provenance** Beyond giving a semantics to DESCRIBE queries using shapes, neighborhoods can also serve to provide *provenance* for shapes. In databases, many notions of provenance have been considered [9, 15]. The common pattern to all approaches is that given a database  $D$ , a query  $Q$ , and a query result  $\nu$ , the provenance should explain why  $\nu$  is indeed a result of  $Q$  in  $D$ . Our setting clearly matches this pattern, with  $G$ ,  $v$  and  $\sigma$  playing the role of  $D$ ,  $\nu$  and  $Q$ , respectively.

With this motivation of providing provenance for SHACL, we proposed a concrete definition of neighborhoods in our work with Delva and Dimou [14]. We will indicate that definition by  $B_{\text{prov}}^G(v, \sigma)$ , to discriminate it from the general idea of neighborhood, and from other concrete proposals.

**Knowledge graph subsets** Many large knowledge graphs consulted in practice are presented in RDF format and support a SPARQL endpoint. In principle, any desired graph or subgraph, including one’s favorite definition of neighborhoods, could be constructed from a large data graph using a SPARQL query. Yet, it may be easier and more natural to extract subgraphs using shapes: given a shape  $\sigma$ , retrieve all nodes satisfying  $\sigma$ , together with their neighborhoods.

With this motivation of subset extraction, Labra Gayo and collaborators have proposed another concrete definition of neighborhoods, this time using shapes expressed in ShEx rather than SHACL [22, 20].

**Shape fragments** Independently, the same idea of using shapes as a retrieval mechanism was proposed in our above-cited work on provenance for SHACL; we called it *shape fragments* to make the link with the already known mechanism of triple pattern fragments [29]. We proved the following correctness property for  $B_{\text{prov}}$ : when a graph conforms to a SHACL schema, then so does the shape fragment formed by taking the neighborhoods of all target nodes of all shapes in the schema. Informally speaking, this means that the shape fragment still contains all information that is important for the schema; yet, the fragment can be much smaller than the original graph.

**Repairs** When a graph is validated against a schema, a violation report is generated, listing all target nodes that do not satisfy the shape associated to the target. When we are faced with a node  $v$  in a graph  $G$  that does not satisfy a shape  $\sigma$ , we want an *explanation* of this violation, so that we may repair it in the data. Since  $v$  satisfies the negation  $\neg\sigma$ , the neighborhood  $B^G(v, \neg\sigma)$  might serve as such an explanation.

Here, however, we have a problem, since so far we have been thinking of neighborhoods as subsets of the data, consisting of facts (edges) that are present

in the data (graph). When dealing with negative conditions, missing facts are equally important for explanations as present facts. For example, let  $\sigma$  be the shape “ $v$  has a colleague that is also a friend” (expressible in SHACL, using `sh:not` and `sh:disjoint`). When  $v$  does not satisfy  $\sigma$  in graph  $G$ , this can conceivably be repaired by adding a friend-edge to one of  $v$ ’s colleagues (or vice versa). However, if  $B^G(v, \neg\sigma)$  is a subgraph of  $G$ , it cannot include such a missing edge. As a matter of fact,  $B_{\text{prov}}$  is even defined to be just empty for disjointness constraints.

Indeed, a proper notion of repair in databases should contain missing facts as well as present facts [4]. For SHACL, this was pursued by Ahmetaj et al. [2]. They define an explanation<sup>1</sup> of a node  $v$  in a graph  $G$  for a shape  $\sigma$  to be a minimal pair  $(A, D)$ , where  $A$  is a set of edges missing in  $G$ , and  $D$  is a set of edges present in  $G$ , such that inserting  $A$  into  $G$  and deleting  $D$  from  $G$  would cause  $v$  to no longer satisfy  $\sigma$ .

This notion of explanation may be considered as an alternative definition for neighborhoods. It differs however in two important aspects from the previous proposals. Not only does it involve negative information, as already discussed; it is also no longer *deterministic*, due to the minimality requirement. For example, continuing the above example, suppose  $v$  has exactly two colleagues  $c_1$  and  $c_2$ , but no friends. Then each of the two missing friend-edges  $v \nrightarrow c_1$  and  $v \nrightarrow c_2$  are equally valid minimal explanations for  $v$  not having any colleagues that are also friends.

### 3 Provenance polynomials

At this point we have seen that several approaches exist to defining exactly what should be in a neighborhood, and there may be others. Is there a unique principled approach? Given the connection to data provenance, we should look at *provenance polynomials*, which are at the heart of most approaches in data provenance [9, 15].

In the context of databases, the provenance polynomial of a result  $\nu$  of a query  $Q$  to a database  $D$  is a compact representation of all the proofs why  $\nu$  indeed belongs to  $Q(D)$ . Provenance polynomials were first considered for queries expressed in positive-existential first-order logic [17]; later they were extended to full first-order logic [16, 28].

Since SHACL (without recursion) can be translated into first-order logic, in principle, we could use this to obtain provenance polynomials for SHACL. However, given the affinity of SHACL to description logic, here we adopt the provenance polynomials developed by Dannert and Grädel for modal logics, guarded logics, and description logics [13, 12].

To see how this works, we must recall the formalisation of SHACL as a logic, already mentioned in the Introduction. We make a few simplifications for

<sup>1</sup> They actually define explanations for non-satisfaction, but we present the same idea for satisfaction, so as to fit our story better. For logics closed under negation, as indeed SHACL is, this makes no difference.

what follows. We omit inverse properties and property paths. We also omit the constraints `sh:lessThan`, `sh:lessThanEq`, and `sh:uniqueLang`. We omit recursion (which has no standardized meaning) and therefore can also omit shape names and the `sh:hasShape` constraint.

Under the above simplifications, the syntax of *shapes*  $\phi$  is then given by the following grammar:

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

Here,  $c$  stands for node constants;  $t$  for tests;  $p$  and  $r$  for property names;  $P$  for finite sets of property names; and  $k$  for natural numbers. By *tests*, we mean any of the tests on single node values that are provided in SHACL, such as typechecking of a literal, regex matching of an IRI, etc.

Shapes are evaluated in RDF graphs; we formalize an RDF graph as a finite set of *edges*, where an edge is a triple of the form  $(s, p, o)$ , where  $s$  and  $o$  are nodes and  $p$  is a property name. We often abbreviate “RDF graph” to “graph”.

Let  $\phi$  be a shape, let  $G$  be a graph, and let  $a$  be a node in  $G$ . Table 1 now gives the definition of when  $a$  *satisfies*  $\phi$  in  $G$ , denoted by  $G, a \models \phi$ . The table omits the obvious logical meanings of  $\top$  (true),  $\perp$  (false), and the boolean connectives (and, or, not). In the table we use the notation  $\llbracket p \rrbracket^G(a)$  to denote the set  $\{b \mid (a, p, b) \in G\}$ .

**Table 1.** Semantics of shapes.

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

We introduce provenance polynomials as multivariate polynomials over the boolean semiring,<sup>2</sup> with edges playing the role of indeterminates. If  $e$  is an edge, we will write  $[e]$  when it is used as an indeterminate. We will refer to these indeterminates as *provenance tokens* (or simply tokens). To deal with negation, we assume (without loss of generality) that shapes are in negation normal form.

Now, let  $\phi$  be a shape in negation normal form, let  $G$  be a graph, and let  $a$  be a node in  $G$ . Table 2 defines the provenance polynomial  $\text{pol}(G, a, \phi)$  for

<sup>2</sup> The boolean semiring has two elements 0 and 1 with logical or as addition and logical and as multiplication. Note that, since  $1 + 1 = 1$ , any polynomial  $p$  is equal to  $p + p$ , also  $p + p + p$ , etc.

the SHACL logical operators. Since these operators are standard in description logics, our definition is in line with the definition given by Dannert and Grädel [12] for the description logic  $\mathcal{ALC}$ . Their definition is generalized here to deal with counting quantifiers.

**Table 2.** Definition of provenance polynomials for SHACL logical operators.

| $\phi$                 | $pol(G, a, \phi)$                                                                                                                 |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| $\top$                 | 1                                                                                                                                 |
| $\perp$                | 0                                                                                                                                 |
| $hasValue(c)$          | 1 if $a = c$ and 0 otherwise                                                                                                      |
| $\neg hasValue(c)$     | 0 if $a = c$ and 1 otherwise                                                                                                      |
| $test(t)$              | 1 if $a$ satisfies $t$ and 0 otherwise                                                                                            |
| $\neg test(t)$         | 0 if $a$ satisfies $t$ and 1 otherwise                                                                                            |
| $\phi_1 \wedge \phi_2$ | $pol(G, a, \phi_1) \cdot pol(G, a, \phi_2)$                                                                                       |
| $\phi_1 \vee \phi_2$   | $pol(G, a, \phi_1) + pol(G, a, \phi_2)$                                                                                           |
| $\forall p.\psi$       | $\prod_{b \in \llbracket p \rrbracket^G} [a, p, b] \cdot pol(G, b, \psi)$                                                         |
| $\geq_k p.\psi$        | $\sum \{ \prod_{b \in X} [a, p, b] \cdot pol(G, b, \psi) \mid X \subseteq \llbracket p \rrbracket^G(a) \ \& \ \#X = k \}$         |
| $\leq_k p.\psi$        | $\prod \{ \sum_{b \in X} [a, p, b] \cdot pol(G, b, \neg\psi) \mid X \subseteq \llbracket p \rrbracket^G(a) \ \& \ \#X = k + 1 \}$ |

*Example 1.* Consider the shape  $\phi$  expressing “ $v$  has at most one author who is not a student”:

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

Consider graph  $G$ :

$$G = \begin{array}{|l|l|l|} \hline c & \text{auth} & a_1 \\ a_1 & \text{rdf:type} & \text{prof} \\ c & \text{auth} & a_2 \\ a_2 & \text{rdf:type} & \text{stud} \\ \hline \end{array}$$

Node  $c$  satisfies  $\phi$  in  $G$ , since only  $a_1$  is not a student.

To compute  $pol(G, c, \phi)$ , let  $\sigma = hasValue(\text{stud})$  and let  $\psi = \leq_0 \text{rdf:type.}\sigma$ , so that  $\phi = \leq_1 \text{auth.}\psi$ . Note that  $\neg\psi$  in negation normal form equals  $\geq_1 \text{rdf:type.}\sigma$ . We calculate:

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

Recall that the triples in square brackets are provenance tokens.

Now suppose we would add to  $G$  a triple  $(c, \text{auth}, a_3)$ , so that  $c$  no longer would satisfy  $\phi$ . Then, due to  $X = \{a_1, a_3\}$ , the above result would be multiplied by 0 and the entire polynomial would become 0.  $\square$

The above example illustrates the following fundamental property, which is straightforward to prove formally.

**Theorem 1.**  $G, a \models \phi$  if and only if  $\text{pol}(G, a, \phi) \neq 0$ .

### 3.1 Neighborhoods from polynomials

Theorem 1 suggests at least two principled approaches to defining a neighborhood  $B^G(a, \phi)$ :

**All tokens:** Simply collect the edges from all tokens in  $\text{pol}(G, a, \phi)$ . We denote this by  $B_{\text{tok}}^G(a, \phi)$ .

**One monomial:** Every polynomial is a sum of monomials. Now collect the edges from the tokens in just one of the monomials. We denote this by  $B_{\text{mon}}^G(a, \phi)$ . This is a non-deterministic approach, since there may be many monomials in a polynomial.

Note that, if  $G, a \not\models \phi$ , the polynomial is 0, so by both approaches above, the neighborhoods would be empty, which is reasonable since the node does not satisfy the shape describing the neighborhood.

*Example 2.* In Example 1, there is only one monomial, and we have

$$B_{\text{tok}}^G(a, \phi) = B_{\text{mon}}^G(a, \phi) = \{(c, \text{auth}, a_2), (a_2, \text{rdf:type}, \text{stud})\}.$$

For a simple example involving multiple monomials, let  $\phi$  now denote the shape  $\geq_1 \text{auth}.\top$ , simply expressing that  $v$  has at least one author. For the same graph  $G$ , we have  $\text{pol}(G, c, \phi) = [c, \text{auth}, a_1] + [c, \text{auth}, a_2]$ , so

$$B_{\text{tok}}^G(c, \phi) = \{(c, \text{auth}, a_1), (c, \text{auth}, a_2)\}.$$

For  $B_{\text{mon}}^G(c, \phi)$ , there are two possible outcomes, namely the two singletons  $\{(c, \text{auth}, a_1)\}$  and  $\{(c, \text{auth}, a_2)\}$ .  $\square$

What with the constraints *eq* and *disj* and their negations? These go beyond description logic-like constraints, in that they independently test the presence or absence of edges in the graph. To deal with them, we must introduce explicit atomic edge formulas of the form  $E(e)$ , where  $e$  is an edge. These formulas do not belong to SHACL but help defining the polynomials. Essentially, the polynomial of  $E(e)$  in  $G$  equals the token  $[e]$  if  $e$  is present in  $G$ , and equals 0 otherwise.

Moreover, to deal with *disj* and  $\neg \text{eq}$  we must also introduce *negated* edge formulas and allow negative tokens, of the form  $[\bar{e}]$ . The polynomial of  $\neg E(e)$  equals  $[\bar{e}]$  if  $e$  is absent from  $G$ , and 0 otherwise. This then leads to the definitions in Table 3. One can verify that Theorem 1 continues to hold.

**Table 3.** Provenance polynomials for  $eq$ ,  $disj$ , and their negations.

| $\phi$            | $pol(G, a, \phi)$                                                                                                                                                                            |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $E(e)$            | $[e]$ if $e \in G$ and 0 otherwise                                                                                                                                                           |
| $\neg E(e)$       | $[\bar{e}]$ if $e \notin G$ and 0 otherwise                                                                                                                                                  |
| $eq(p, r)$        | $\prod_{b \in \llbracket p \rrbracket^{G(a)}} [a, p, b] \cdot pol(G, a, E(a, r, b))$<br>$\cdot \prod_{b \in \llbracket r \rrbracket^{G(a)}} [a, r, b] \cdot pol(G, a, E(a, p, b))$           |
| $disj(p, r)$      | $\prod_{b \in \llbracket p \rrbracket^{G(a)}} [a, p, b] \cdot pol(G, a, \neg E(a, r, b))$<br>$\cdot \prod_{b \in \llbracket r \rrbracket^{G(a)}} [a, r, b] \cdot pol(G, a, \neg E(a, p, b))$ |
| $\neg disj(p, r)$ | $\sum_{b \in \llbracket p \rrbracket^{G(a)}} [a, p, b] \cdot pol(G, a, E(a, r, b))$<br>$+ \sum_{b \in \llbracket r \rrbracket^{G(a)}} [a, r, b] \cdot pol(G, a, E(a, p, b))$                 |
| $\neg eq(p, r)$   | $\sum_{b \in \llbracket p \rrbracket^{G(a)}} [a, p, b] \cdot pol(G, a, \neg E(a, r, b))$<br>$+ \sum_{b \in \llbracket r \rrbracket^{G(a)}} [a, r, b] \cdot pol(G, a, \neg E(a, p, b))$       |

*Example 3.* To illustrate the four new constraints, consider the three graphs in Figure 1. We have:

$$\begin{aligned}
pol(G_1, a, eq(p, r)) &= [a, p, b]^2 [a, r, b]^2 \\
pol(G_2, a, disj(p, r)) &= [a, p, b_1] [\overline{a, r, b_1}] [a, r, b_2] [\overline{a, p, b_2}] \\
pol(G_3, a, \neg disj(p, r)) &= [a, p, b_2] [a, r, b_2] \\
pol(G_3, a, \neg eq(p, r)) &= [a, p, b_1] [\overline{a, r, b_1}] + [a, r, b_3] [\overline{a, p, b_3}]
\end{aligned}$$

□

$$G_1 = \begin{bmatrix} a & p & b \\ a & r & b \end{bmatrix} \quad G_2 = \begin{bmatrix} a & p & b_1 \\ a & r & b_2 \end{bmatrix} \quad G_3 = \begin{bmatrix} a & p & b_1 \\ a & p & b_2 \\ a & r & b_2 \\ a & r & b_3 \end{bmatrix}$$

**Fig. 1.** Graphs used in Example 3.

We may choose to extend the notion of a neighborhood by also allowing *negated edges*  $\bar{e}$  to be present, indicating that their absence is important for the satisfaction of the shape. In this case, we can generalize the definitions of  $B_{\text{tok}}$  and  $B_{\text{mon}}$ , taking edges from positive tokens as well as negated edges from negated tokens. We refer to such neighborhoods as *3-valued neighborhoods*.

*Example 4.* In Example 3, for  $B_{\text{tok}}^{G_3}(a, \neg eq(p, r))$ , we would obtain the 3-valued neighborhood

$$\{(a, p, b_1), (\overline{a, r, b_1}), (a, r, b_3), (\overline{a, p, b_3})\}.$$

For 3-valued  $B_{\text{mon}}^{G_3}(a, \neg eq(p, r))$  there would be the two possibilities

$$\{(a, p, b_1), (\overline{a, r, b_1})\} \quad \text{and} \quad \{(a, r, b_3), (\overline{a, p, b_3})\}.$$

□

For other applications, one may want to insist that a neighborhood really is a subgraph, so consists only of edges present in the graph. In that case we can define variants  $B_{\text{postok}}$  and  $B_{\text{posmon}}$  which take only the edges from positive tokens and disregard negative tokens in the polynomial. Interestingly, with one exception, one can verify that the neighborhood definition proposed for provenance for SHACL [14], which we denote here by  $B_{\text{prov}}$ , coincides with  $B_{\text{postok}}$ . The only exception is for  $disj(p, r)$ , for which  $B_{\text{prov}}$  defines the neighborhood to be empty, while  $B_{\text{postok}}^G(a, disj(p, r))$  consists of all  $p$ - and  $r$ -edges emanating from  $a$  in  $G$ .

## 4 Causality

A quite different, purely semantic approach to neighborhoods is through causality. In this realm, the notion of *actual cause* proposed by Halpern and Pearl [19] has been very influential. Adapting their more recent definition [18] to our setting, this amounts to the following. Let  $G, a \models \sigma$  as before. A *supercause* for  $G, a \models \sigma$  is a set  $S$  of positive and negated edges such that all positive edges belong to  $G$ ; all negated edges are missing from  $G$ ; and in the graph resulting from  $G$  by deleting the positive edges and inserting the negated edges,  $a$  no longer satisfies  $\phi$ . Now a *cause* is a minimal supercause.

*Example 5.* Take  $G_2, a \models disj(p, r)$  from Example 3. The set  $\{(\overline{a, p, b_2}), (a, r, b_2)\}$  is a supercause but not a cause. Indeed,  $\{(\overline{a, p, b_2})\}$  alone is already a cause, since inserting this missing edge would make  $p$  and  $r$  no longer disjoint at  $a$ . Similarly,  $\{(\overline{a, r, b_1})\}$  is a cause. The two causes for  $G_1, a \models eq(p, r)$  are  $\{(a, p, b)\}$  and  $\{(a, r, b)\}$  since deleting either of them causes equality to be no longer satisfied.

□

Of course, causes are exactly the explanations introduced by Ahmetaj et al. [2] in their work on repairs, already discussed in Section 2. It is tempting to use causes as neighborhoods. However, we will see soon that they may violate a fundamental property that we likely want from neighborhoods.

## 5 Desiderata for neighborhoods

At this point we understand that there are a variety of possible definitions of precisely what should be in a neighborhood. There is no single best definition, since there are different desiderata one may have about neighborhoods, and these desiderata are not all compatible with each other. We discuss this next, taking some of our work with Bogaerts [6] done in the setting of first-order logic, and adapting it slightly to the SHACL context.

**Determinism** has been mentioned a number of times already.  $B_{\text{tok}}$  is deterministic;  $B_{\text{mon}}$ , and also causes, are not. It depends on the application whether nondeterminism is acceptable. When using neighborhoods as a retrieval mechanism, as discussed in Section 2, we probably want determinism.

**Sufficiency** One desideratum seems so fundamental that it probably should be a hard requirement: given that  $v$  satisfies shape  $\sigma$  in graph  $G$ , then  $B(G, v, \sigma)$  should be such that  $v$  still satisfies  $\sigma$  in  $B(G, v, \sigma)$ .<sup>3</sup> When a neighborhood definition has this property for all  $B, v, \sigma$ , we call it *sufficient*. This terminology was coined by Glavic [15] in the context of data provenance.

*Example 6.* Consider  $G, c$  and  $\phi$  from Example 1 where we calculated  $\text{pol}(G, c, \phi)$  showing that

$$B_{\text{tok}}^G(c, \phi) = \{(c, \text{auth}, a_2), (a_2, \text{rdf:type}, \text{stud})\}.$$

We see that  $c$  also satisfies  $\phi$  in this neighborhood.

Next let us continue Example 3. From the polynomials given there we obtain:

$$\begin{aligned} B_{\text{postok}}^{G_1}(a, \text{eq}(p, r)) &= \{(a, p, b), (a, r, b)\} \\ B_{\text{postok}}^{G_2}(a, \text{disj}(p, r)) &= \{(a, p, b_1), (a, r, b_2)\} \\ B_{\text{postok}}^{G_3}(a, \neg \text{disj}(p, r)) &= \{(a, p, b_2), (a, r, b_2)\} \\ B_{\text{postok}}^{G_3}(a, \neg \text{eq}(p, r)) &= \{(a, p, b_1), (a, r, b_3)\} \end{aligned}$$

We again see that  $a$  still satisfies the respective constraint in each of the neighborhoods.  $\square$

The above example is no coincidence: it can be proven [14] that  $B_{\text{prov}}$ , which we have seen earlier is almost the same as  $B_{\text{postok}}$ , is indeed a sufficient provenance definition. Also  $B_{\text{posmon}}$  can be shown to be sufficient, adapting proof techniques used in the context of first-order logic [6].

Causes (Section 4) are typically not sufficient. For a simple example, let  $\sigma$  be the shape  $\geq_1 p. \top \wedge \geq_1 r. \top$ . Node  $a$  satisfies  $\sigma$  in graph  $G = \{(a, p, b), (a, r, c)\}$ . There are two causes, namely the which node  $a$  satisfies in the two singleton subsets of  $G$ . In neither of them,  $a$  satisfies  $\sigma$ .

**Causal relevance** While a single cause may not provide a sufficient notion of neighborhood, interesting notions can still be obtained based on causality. Indeed, a natural desideratum on neighborhoods can be that it exclusively consists of edges that come from causes (where different edges may come from different causes). We call such edges *causally relevant*. For example, it can be shown [6] that restricting  $B_{\text{tok}}$  to only causally relevant edges still yields a sufficient notion of neighborhood.

<sup>3</sup> When using 3-valued neighborhoods, we should be careful about what we mean by satisfaction [6].

**Minimality** Given some set  $X$  of desiderata, we may naturally require neighborhoods to be *minimal for  $X$* . Minimality almost always leads to nondeterminism, as there will be typically multiple minimal candidates. For example, just take sufficiency, and the shape  $\geq_1 p.\top \vee \geq_1 r.\top$  requiring that  $v$  has a  $p$ -edge or an  $r$ -edge. When  $v$  has both, we can keep only one of them and have a minimally sufficient neighborhood.

## 6 Conclusion

Coming back to our initial motivation for studying neighborhoods in Section 2, what, if any, should the standard semantics of DESCRIBE USING a shape be in SPARQL? As there is no “best” definition, it is probably best left unstandardized. Yet,  $B_{\text{postok}}$  seems to be a very reasonable semantics. When we want something smaller, we can further restrict to causally relevant edges, but, the computational complexity may be high. The computational complexity of finding causes for SHACL shapes is already known to be high [2]. Perhaps, the complexity of finding the causally relevant edges in the provenance polynomial may be lower; this may be a direction for further research.

Computing neighborhoods, especially in the presence of property paths in shapes (which we have omitted here for simplicity) can be a challenge [14]. That also seems to be a good direction for further research.

In this paper, we have worked with SHACL, but neighborhoods for ShEx have already been investigated [22, 20]. A comparison between the approaches, in particular comparing nonrecursive ShEx to provenance polynomials for non-recursive SHACL, would be interesting.

Finally, we mention that RDF is more than just labeled graphs; edge labels can be nodes by themselves. Designing an extension of SHACL that treats properties on equal footing as subjects and objects is another interesting direction for further research.

## Acknowledgments

We thank Bart Bogaerts, Thomas Delva, and Anastasia Dimou for fruitful collaborations on the topics discussed in this paper.

## References

1. Abiteboul, S., Manolescu, I., Rigaux, P., Rousset, M.C., Senellart, P.: Web Data Management. Cambridge University Press (2012)
2. Ahmetaj, S., David, R., Ortiz, M., Polleres, A., Shehu, B., Simkus, M.: Reasoning about explanations for non-validation in SHACL. In: Bienvenu, M., Lakemeyer, G., et al. (eds.) Proceedings 18th International Conference on Principles of Knowledge Representation and Reasoning. pp. 12–21. IJCAI Organization (2021)
3. Allemang, D., Hendler, J., Gandon, F.: Semantic Web for the Working Ontologist: Effective Modeling for Linked Data, RDFS, and OWL. ACM (2020)

4. Bertossi, L.: Database Repairing and Consistent Query Answering. Synthesis Lectures on Data Management, Springer, Cham (2011)
5. Bogaerts, B., Jakubowski, M., Van den Bussche, J.: SHACL: A description logic in disguise. In: Gottlob, G., Inclezan, D., Maratea, M. (eds.) Proceedings 16th International Conference on Logic Programming and Nonmonotonic Reasoning. Lecture Notes in Computer Science, vol. 13416, pp. 75–88. Springer (2022)
6. Bogaerts, B., Jakubowski, M., Van den Bussche, J.: Postulates for provenance: Instance-based provenance for first-order logic. Proceedings of the ACM on Management of Data **2**(2), 95:1–95:16 (2024)
7. Boneva, I., Labra Gayo, J., Prud’hommeaux, E.: Semantics and validation of shape schemas for RDF. In: d’Amato, C., Fernandez, M., Tamma, V., et al. (eds.) Proceedings 16th International Semantic Web Conference. Lecture Notes in Computer Science, vol. 10587, pp. 104–120. Springer (2017)
8. Buil-Aranda, C., Ugarte, M., et al.: A preliminary investigation into SPARQL query complexity and federation in Bio2RDF. In: Cali, A., Vidal, M.E. (eds.) Proceedings 9th Alberto Mendelzon International Workshop on Foundations of Data Management. CEUR Workshop Proceedings, vol. 1378 (2015)
9. Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in databases: why, how and where. Foundations and Trends in Databases **1**(4), 379–474 (2009)
10. Corman, J., Reutter, J., Savkovic, O.: Semantics and validation of recursive SHACL. In: Vrandecic, D., et al. (eds.) Proceedings 17th International Semantic Web Conference. Lecture Notes in Computer Science, vol. 11136, pp. 318–336. Springer (2018), extended version, technical report KRDB18-01, <https://www.inf.unibz.it/kldb/tech-reports/>
11. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 concepts and abstract syntax. W3C Recommendation (Feb 2014)
12. Dannert, K., Grädel, E.: Provenance analysis: A perspective for description logics? In: Lutz, C., Sattler, U., Tinelli, C., Turhan, A.Y., Wolter, F. (eds.) Description Logic, Theory Combination, and All That, Lecture Notes in Computer Science, vol. 11560, pp. 266–285. Springer (2019)
13. Dannert, K., Grädel, E.: Semiring provenance for guarded logics. In: Madarász, J., Székely, G. (eds.) Hajnal Andréka and István Németi on the Unity of Science, Outstanding Contributions to Logic, vol. 19, pp. 55–79. Springer (2021)
14. Delva, T., Dimou, A., Jakubowski, M., Van den Bussche, J.: Data provenance for SHACL. In: Stoyanovich, J., Teubner, J., et al. (eds.) Proceedings 26th International Conference on Extending Database Technology. pp. 285–297. OpenProceedings.org (2023)
15. Glavic, B.: Data provenance: Origins, applications, algorithms, and models. Foundations and Trends in Databases **9**(3–4), 209–441 (2021)
16. Grädel, E., Tannen, V.: Semiring provenance for first-order model checking. arXiv:1712.01980 (2017)
17. Green, T., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proceedings 26th ACM Symposium on Principles of Database Systems. pp. 31–40. ACM (2007)
18. Halpern, J.: A modification of the Halpern-Pearl definition of causality. In: Yang, Q., Wooldridge, M. (eds.) Proceedings 24th International Joint Conference on Artificial Intelligence. pp. 3022–3033. AAAI Press (2015)
19. Halpern, J., Pearl, J.: Causes and explanations: A structural-model approach. part i: Causes. British Journal for the Philosophy of Science **56**, 843–887 (2005)
20. Iglesias Préstamo, A., Labra Gayo, J.: Using Pregel to create knowledge graphs subsets described by non-recursive shape expressions. In: Ortiz-Rodríguez, F., Vil-lazón-Terrazas, B., et al. (eds.) Proceedings 5th Iberoamerican Conference and

- 4th Indo-American Conference on Knowledge Graphs and Semantic Web. Lecture Notes in Computer Science, vol. 14382, pp. 120–134. Springer (2023)
21. Knublauch, H., Kontokostas, D.: Shapes constraint language (SHACL). W3C Recommendation (Jul 2017)
  22. Labra Gayo, J.: Creating knowledge graph subsets using shape expressions. arXiv:2110.11709 (Oct 2021)
  23. Labra Gayo, J., Prud'hommeaux, E., Boneva, I., Kontokostas, D.: Validating RDF Data. Synthesis Lectures on Data, Semantics, and Knowledge, Springer, Cham (2018)
  24. Ortiz, M.: A short introduction to SHACL for logicians. In: Hansen, H., Scedrov, A., et al. (eds.) Logic, Language, Information and Computation, Proceedings 29th WoLLIC. Lecture Notes in Computer Science, vol. 13923, pp. 19–32. Springer (2023)
  25. Schreiber, G., Raimond, Y.: RDF 1.1 primer. W3C Working Group Note (Jun 2014)
  26. ShEx—shape expressions. <https://shex.io>
  27. SPARQL 1.2 community group: DESCRIBE using shapes, <https://github.com/w3c/sparql-12/issues/39>
  28. Tannen, V.: Provenance analysis for FOL model checking. ACM SIGLOG News 4(1), 24–36 (2017)
  29. Verborgh, R., Vander Sande, M., Hartig, O., et al.: Triple Pattern Fragments: A low-cost knowledge graph interface for the Web. Journal of Web Semantics 37–38, 184–206 (2016)