Extending Shape Expressions for different types of knowledge graphs

Jose Emilio Labra-Gayo[1]

[1]*WESO lab - University of Oviedo*

Abstract

Shape Expressions (ShEx) has been proposed as a concise and human-readable language to describe and validate RDF. Inspired by regular expressions, it offers an expressive formalism to describe graph structures based on regular bag expressions. Although plain RDF is one of the common data formats employed to represent knowledge graphs, there have been several proposals to either extend RDF with the so-called RDF-Star or RDF 1.2, or to employ other formalisms like property graphs. In this paper we present an overview and comparison of those approaches and propose three possible extensions of ShEx: ShEx-Star which can be used to validate RDF-Star, ShEx-N: that can be used when nodes also act as properties in RDF and PShEx, which can be used to describe property graphs. We present some examples and a semantics of each extension.

1. Introduction

Although Knowledge graphs have been successfully adopted by the industry, an important aspect of their practical application is the quality of the data that they contain. In order to increase their quality, it is necessary to have some mechanisms that can check the conformance of the data to some kind of schema. Knowledge graphs are usually considered schema-less, because there is no mandatory schema, however, in most cases the data curators have an implicit schema in mind. Having the possibility to materialize that implicit schema into a machine processable form that can be automatically verified can mitigate the risk of non-conformant data.

In practice, there are several types of technologies that can be used for knowledge graphs [1]: Directed edge-labeled graphs, whose main representative are RDF graphs, and Property graphs, which allow property–value pairs and labels to be associated with nodes and edges.

In the case of RDF, two main technologies have been proposed for validation: ShEx [2] and SHACL [3], which are based on the notion of a shape as a description of the topology of some specific kind of nodes. In this way, it is possible to define a schema as a set of shapes which describe the expected properties of some nodes, their expected cardinalities and the kind of nodes. These schemas, can be used to validate RDF data and check if it conforms to those shapes. We employ ShEx in the paper because it can be seen as a description language for RDF acting as grammar where a ShEx schema represents the set of all the RDF graphs that conform to it.

labra@uniovi.es (J. E. Labra-Gayo)
http://labra.weso.es (J. E. Labra-Gayo)
0000-0001-8907-5348 (J. E. Labra-Gayo)

CEUR Workshop Proceedings (CEUR-WS.org)

There has been a proposal for an extension of RDF called RDF-Star [4] and a Working Group is currently working taking that proposal as input to define what is currently called RDF 1.2 [5]. The proposal extends RDF with the possibility of having triples as subjects or objects in statements. The relationship between that extension and property graphs has already been studied [6].

Although ShEx was defined to describe and validate RDF, we consider that it can be extended to describe and validate RDF-Star (or RDF 1.2), so in this paper we explore a possible extension of the language in that direction. Apart from that, the shapes in ShEx are usually centered on describing nodes that act as either subjects or objects, but in RDF, it is also possible to add statements whose subjects or objects are also the predicates of other statements. We also explore a possible extension of ShEx where it is possible to define shapes about nodes that act as properties. Finally, although there have been several proposals to define schemas for property graphs, we consider that the grammar-based approach of ShEx can also be helpful, so we also explore what it would look like to extend ShEx for describing and validating property graphs.

The main contributions of this paper are to present three extensions of ShEx: ShEx-Star for RDF-Star (section 3), ShEx-N for describing nodes as properties (section 4) and PShEx for property graphs (section 5), with their abstract syntax and semantic definitions.

2. RDF and ShEx

Definition 1 (RDF triple and RDF Graph)**.** Given a set of IRIs I, a set of blank nodes B and a set of literals Lit, an *RDF triple* is a tuple (s, p, o) where $s \in I \cup B$ is called the subject, $p \in I$ is called the predicate and $o \in I \cup B \cup Lit$ is called the object. An RDF graph G is a set of RDF triples.

There are several syntaxes for RDF graphs like Turtle, N3, RDF/XML, etc. In this document, we will use Turtle.

Example 1 (Example of an RDF graph in Turtle)**.** The following snippet contains a simple RDF graph with two nodes `:a` and `:b`.

```
prefix : <http://example.org/>

:a :name "Alice" ;
   :knows :b .
:b :firstname "Robert", "Julius" ;
   :lastname "Smith" .
```

The neighbors of a node $n \in V$ in an RDF graph G are defined as $neighs(n, G) = \{(n, p, y) \mid (n, p, y) \in G\} \cup \{(x, p, n) \mid (x, p, n) \in G\} \cup \{(x, n, y) \mid (x, n, y) \in G\}$.

Shape Expressions (ShEx) were proposed as such a language in 2014 [2]. It was designed as a high-level and concise domain-specific language to describe RDF. The syntax of ShEx is inspired by Turtle and SPARQL, while the semantics was inspired by RelaxNG and XML Schema. In this section we describe a simplified abstract syntax of ShEx following [7][1].

[1]The full specification of ShEx is available at https://shex.io/shex-semantics/

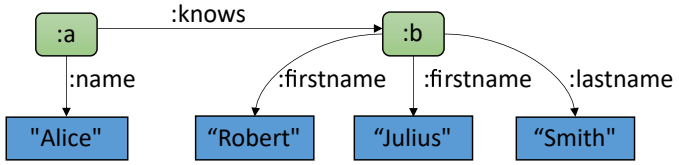

Figure 1: Basic RDF graph example

Definition 2 (ShEx schema)**.** A *ShEx Schema* is defined as a tuple $\langle L, \delta \rangle$ where L is the set of shape labels, and $\delta : L \to S$ is a total function from labels to shape expressions. The set of shape expressions $se \in S$ is defined using the following abstract syntax:

se	::=	cond	Basic boolean condition on nodes (node constraint)
	|	s	Shape
	|	se_1 AND se_2	Conjunction of se_1 and se_2
	|	$@l$	Shape label reference for $l \in L$
s	::=	CLOSED $\{te\}$	Closed shape
	|	$\{te\}$	Open shape
te	::=	$te_1; te_2$	Each of te_1 and te_2
	|	$te_1 \mid te_2$	Either te_1 or te_2
	|	$te*$	Zero or more te
	|	$\_ \xrightarrow{p} se$	Outgoing Triple with predicate p and object conforming to se
	|	$se \xrightarrow{p} \_$	Incoming triple with predicate p and subject conforming to se
	|	ϵ	Empty triple expression

Intuitively, shape expressions define conditions about nodes while triple expressions define conditions about the neighborhood of nodes, and shapes qualify those neighborhoods by disallowing triples with other predicates in the case of closed shapes or allowing them in the case of open shapes. We omit negation and disjunction operator to simplify the interactions between negation and recursion, which led to a stratified negation requirement in ShEx.

The restrictions imposed on shape expressions schemas in [8] also apply here. Namely, in a schema (L, δ, S)

- The shape label references used by the definition function δ are themselves defined, i.e. if $@l$ appears in some shape definition, then l belongs to L;
- No definition $\delta(l)$ uses a reference $@l$ to itself, neither directly nor transitively, except while traversing a shape. For instance, $\delta(l) = @l$ AND se is forbidden, but $\delta(l) = \{\_ \xrightarrow{p} @l\}$ is allowed.

Example 2 (Example of ShEx schema)**.** A ShEx schema that describes the RDF graph presented in example 1 can be defined as:

$$
\begin{array}{rcl}
L & = & \{Person\} \\
\delta(Person) & = & \{ \; (\_ \xrightarrow{name} String \mid \_ \xrightarrow{fistname} String*; \_ \xrightarrow{lastname} String); \\
 & & \quad \_ \xrightarrow{knows} @Person \\
 & & \}
\end{array}
$$

ShEx has several concrete syntaxes like a compact syntax (ShExC) and an RDF syntax defined based on JSON-LD (ShExJ) [2].

Example 3 (Example of ShEx schema in ShExC)**.** Example of a ShEx schema using ShEx compact syntax.

```
prefix : <http://example.org/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
<Person> {
 (:name xsd:string |
  :firstname xsd:string * ; :lastname xsd:string );
 :knows @<Person> *
}
```

The semantics of ShEx schemas is based on a conformance relation parameterized by a shape assignment: we say that node n in graph G conforms to shape expression se with shape assignment τ, and we write $G, n, \tau \vDash se$.

The following rules are defined similar to [9], where it is shown that there exists a unique maximal shape assignment $\tau_{\max}$ that allows us to define conformance independently of the shape assignment. The conformance relation is defined recursively on the structure of se by the set of inference rules presented in table 1 where $preds(te)$ is the set of predicates that appear in a triple expression te and can be defined as:

$$Cond\frac{cond(n) = true}{G, n, \tau \vDash cond} \qquad AND\frac{G, n, \tau \vDash se_1 \quad G, n, \tau \vDash se_2}{G, n, \tau \vDash se_1 \ \texttt{AND}\ se_2}$$

$$ClosedShape\frac{neighs(n, G) = ts \quad G, ts, \tau \Vdash te}{G, n, \tau \vDash \texttt{CLOSED}\ \{te\}}$$

$$OpenShape\frac{ts = \{\langle x, p, y\rangle \in neighs(n, G) \mid p \in preds(te)\} \quad G, ts, \tau \Vdash te}{G, n, \tau \vDash \{te\}}$$

Table 1
Inference rules for ShEx shape expressions

$$\begin{array}{lcl} preds(te_1; te_2) & = & preds(te_1) \cup preds(te_2) \\ preds(te_1 \mid te_2) & = & preds(te_1) \cup preds(te_2) \\ preds(\_ \xrightarrow{p} te) & = & \{p\} \\ preds(te*) & = & preds(te) \\ preds(\epsilon) & = & \emptyset \end{array}$$

The rules for node constraints (*Cond*) and conjunction are as expected. A node n conforms to an open shape with triple expression te if its neighborhood restricted to the triples with predicates from te conform, meaning that triples whose predicates are not mentioned in te are

[2] See ShEx specification [8] for details.

not constrained by the shape (rule *OpenShape*). Conformance to a closed shape requires to consider the whole neighborhood of the node (rule *ClosedShape*).

Conformance to a triple expression uses a second relation defined on sets on neighborhood triples *ts* instead of nodes *n*. The set of neighborhood nodes *ts* of a graph G conforms to a triple expression *te* with shape assignment τ, written as $G, ts, \tau \Vdash te$, as defined by the inference rules in table 2.

$$EachOf\frac{(ts_1, ts_2) \in part(ts) \quad G, ts_1, \tau \Vdash te_1 \quad G, ts_2, \tau \Vdash te_2}{G, ts, \tau \Vdash te_1; te_2}$$

$$OneOf_1\frac{G, ts, \tau \Vdash te_1}{G, ts, \tau \Vdash te_1 \mid te_2} \qquad OneOf_2\frac{G, ts, \tau \Vdash te_2}{G, ts, \tau \Vdash te_1 \mid te_2}$$

$$TC_1\frac{ts = \{\langle x, p, y\rangle\} \quad G, y, \tau \vDash @l}{G, ts, \tau \Vdash \_ \xrightarrow{p} @l} \qquad TC_2\frac{ts = \{\langle y, p, x\rangle\} \quad G, y, \tau \vDash @l}{G, ts, \tau \Vdash @l \xrightarrow{p} \_}$$

$$Star_2\frac{(ts_1, ts_2) \in part(ts) \quad G, ts_1, \tau \Vdash te \quad G, ts_2, \tau \Vdash te*}{G, ts, \tau \Vdash te*} \qquad Star_1\frac{}{G, \emptyset, \tau \Vdash te*}$$

Table 2
Inference rules for ShEx triple expressions

A shape assignment τ for graph G and S is called *valid* if for every node n in G and every shape expression label l defined in S, if $n@l \in \tau$, then $G, n, \tau \vDash @l$.

According to Boneva et al [7], for every graph G, there exists a unique maximal valid shape assignment $\tau_{\max}$ such that if τ is a valid shape assignment for G and S, then $\tau \subseteq \tau_{\max}$.

3. RDF-Star and ShEx-Star

RDF-Star has been proposed as an extension of RDF where the subjects and objects can be triples. We present a formal definition of RDF-Star based on [10]:

Definition 3 (RDF-Star)**.** An RDF-Star triple is a tuple t defined recursively as follows: Any RDF triple $t \in (I \cup B) \times I \times (I \cup B \cup Lit)$ is an RDF-Star triple; and given RDF-Star triples t and t' and RDF terms $s \in (I \cup B)$, $p \in I$ and $o \in (I \cup B \cup Lit)$, the tuples $(\ll t \gg, p, o)$, $(s, p, \ll t' \gg)$ and $(\ll t \gg, p, \ll t' \gg)$ are RDF-Star triples. An RDF-Star graph is a set of RDF-Star triples.

Example 4 (Example of an RDF-Star graph in Turtle-Star notation)**.** The following snippet contains a simple RDF-Star graph with two nodes `:a` and `:b`.

```
prefix : <http://example.org/>

:a :name "Alice" .
<< :a :knows :b >> :certainty 0.5 .
:b :firstname "Robert", "Julius" ;
```

```
  :lastname "Smith" .
```

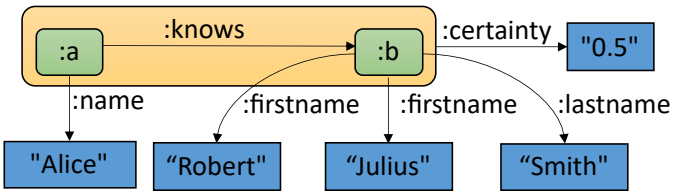

Figure 2: Basic RDF-Star example

We can extend ShEx to support ShEx-Star by adding the following declaration to the definition of triple expressions *te*:

te	::=	...	Same definitions as in 2
	|	$\ll \_ \xrightarrow{p} se \gg \{\|te\|\}$	Outgoing Triple term constraint with predicate p
	|	$\ll se \xrightarrow{p} \_ \gg \{\|te\|\}$	Incoming triple term constraint with predicate p

Example 5 (Example of a ShEx-Star schema)**.** A ShEx-Star schema that describes the RDF graph presented in example 4 can be defined as:

$$\begin{aligned}\delta(Person) = \{\ & (\_ \xrightarrow{name} String \mid \_ \xrightarrow{fistname} String^*; \_ \xrightarrow{lastname} String); \\ & \ll \_ \xrightarrow{knows} @Person \gg \{\|\_ \xrightarrow{certainty} Float\|\}^* \\ \}\end{aligned}$$

The expression $\ll \_ \xrightarrow{p} se \gg \{|te|\}$ describes a triple term whose predicate is p and whose object conforms to the shape expression *se* and that can be the subject of triples conforming to triple expression *te*. The formal semantics can be described as:

$$TTC_1 \frac{ts = \{\langle \ll t \gg, p, y\rangle\} \quad G, y, \tau \vDash se \quad neighs(\ll t \gg, G) = ts' \quad G, ts', \tau \Vdash te}{G, ts, \tau \Vdash \ll \_ \xrightarrow{p} se \gg \{|te|\}}$$

$$TTC_2 \frac{ts = \{\langle x, p, \ll t \gg\rangle\} \quad G, x, \tau \vDash se \quad neighs(\ll t \gg, G) = ts' \quad G, ts', \tau \Vdash te}{G, ts, \tau \Vdash \ll se \xrightarrow{p} \_ \gg \{|te|\}}$$

Table 3
Inference rules for ShEx-* new triple term expressions

4. ShEx-N: Describing nodes that act as properties

In the RDF data model, predicates can also act as subjects or objects or triples. This aspect is not taken into account in traditional ShEx, where the shapes describe the topology of nodes without considering their potential role as predicates.

Example 6 (Example of an RDF graph a node acting as a property)**.** The following snippet contains a simple RDF graph where :knows is both a node and a property.

```
prefix : <http://example.org/>
prefix skos: <http://www.w3.org/2004/02/skos/core#>

:a :name "Alice" ;
   :knows :b .
:b :firstname "Robert", "Julius" ;
   :lastname "Smith" .
:knows skos:related :Friendship .
```

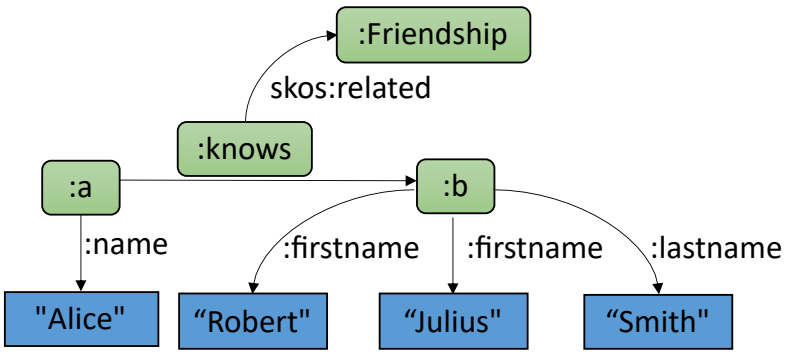

Figure 3: RDF example with a property as a node

In order to capture these appearances in a single shape, it is possible to add a new kind of triple expression:

te	::=	...	Same definitions as in 2
	|	$se_1 \xrightarrow{\sqcup} se_2$	Triple constraint with focus node acting as predicate and subject conforming to se_1 and object conforming to se_2

The semantics of triple term constraints can be defined as:

$$NP_1 \frac{ts = \{\langle s, x, o\rangle\} \quad G, s, \tau \vDash se_1 \quad G, o, \tau \vDash se_2}{G, ts, \tau \Vdash se_1 \xrightarrow{\sqcup} se_2}$$

Table 4
Inference rules for ShEx-N

Example 7 (Example of a ShEx-N schema)**.** The following ShEx-N schema defines the shape *FrienshipProperty* which validates the node :knows in example 6:

$$
\begin{array}{lcll}
\delta(FriendShipProperty) & = & \{ & \_ \xrightarrow{skos:related} [:\ Friendship]\ ; \\
 & & & @Person \xrightarrow{\_} @Person \\
 & & \} & \\
\delta(Person) & = & \{ & \_ \xrightarrow{:name} String \\
 & & | & \_ \xrightarrow{:firstname} String^*\ ;\ \_ \xrightarrow{:lastname} String \\
 & & \} &
\end{array}
$$

5. Property graphs and PShEx

Property graphs have become popular thanks to several commercial graph databases like Neo4j [3], JanusGraph [4] or Sparksee [5]. A property graph has unique identifiers for each node/edge and allows to add property-value annotations to each node/edge in the arc as well as type annotations. The following definition of a property graph follows [11].

Definition 4 (Property graph)**.** Given a set of types T, a set of properties P, and a set of values V, a *property graph* G is a tuple $\langle N, E, \rho, \lambda_n, \lambda_e, \sigma \rangle$ where $N \cap E = \emptyset$, $\rho : E \mapsto N \times N$ is a total function, $\lambda_n : N \mapsto FinSet(T)$, $\lambda_e : E \mapsto T$, and $\sigma : N \cup E \times P \mapsto FinSet(V)$.

A property graph is formed by a set of node identifiers N and a set of edges E where ρ associates a pair of nodes (n_1, n_2) to every $e \in E$ where n_1 is the subject and n_2 is the object, λ_n associates a set of types for node identifiers (notice that property graphs allow nodes to have more than one type), λ_e associates a types for each edge identifier, and σ associates a set of values to pairs (i, p) such that $i \in N \cup E$ is a node or edge and $p \in P$ is a property.

Example 8. As an example, we will represent information that Alice knows Robert with a certainty of 0.5

$$
\begin{array}{l}
T = \{\text{Person}, \text{knows}\} \quad P = \{\text{name}, \text{certainty}\} \quad V = \{\text{"Alice"}, \text{"Robert"}, \text{"Julius"}, \text{"Smith"}, 0.5\} \\
N = \{n_1, n_2\} \qquad E = \{r_1\} \qquad \rho = r_1 \mapsto (n_1, n_2) \\
\lambda_n = n_1 \mapsto \{\text{Person}\},\ n_2 \mapsto \{\text{Person}\} \quad \lambda_e = r_1 \mapsto \text{knows} \\
\sigma = (n_1, name) \mapsto \{\text{"Alice"}\}\ \ (n_2, firstname) \mapsto \{\text{"Robert"}, \text{"Julius"}\} \\
\quad (n_2, lastname) \mapsto \{\text{"Smith"}\}\ (r_1, certainty) \mapsto \{0.5\}
\end{array}
$$

Figure 4 presents a possible visualization of a property graph.

We define a ShEx extension called PShEx that can be used to describe and validate Property graphs. In property graphs, nodes and edges can have associated labels as well as a set of property/values. In this way, it is necessary to adapt the definition of ShEx to describe pairs or property/values. PShEx is composed of three main categories: shape expressions (*se*) that describe the shape of nodes, triple expressions (*te*) that describe the shape of edge relationships and property-value expressions (*pvs*) that describe sets of property/values associated with node/edge identifiers.

[3] https://neo4j.com/

[4] https://janusgraph.org/

[5] https://www.sparsity-technologies.com/#sparksee

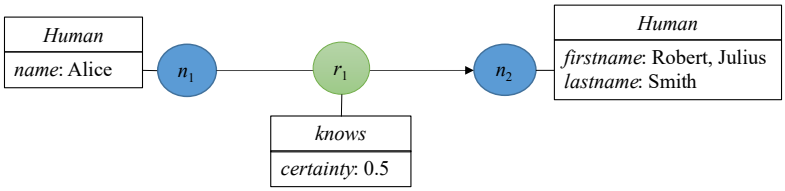

Figure 4: Example graph visualization of a property graph

Definition 5 (PShEx schema)**.** A *PShEx Schema* is a tuple $\langle L, \delta \rangle$ where L set of shape labels, and $\delta : L \to S$ is a total function from labels to shape expressions $se \in S$ defined using the abstract syntax:

se	::=	$cond_{t_s}$	Basic boolean condition on set of types $t_s \subseteq T$
	|	s	Shape
	|	se_1 **AND** se_2	Conjunction
	|	$@l$	Shape label reference for $l \in L$
	|	pvs	Property-value specifiers of a node
s	::=	**CLOSED** $\{te\}$	Closed shape
	|	$\{te\}$	Open shape
te	::=	$te_1; te_2$	Each of te_1 and te_2
	|	$te_1 \mid te_2$	Some of te_1 or te_2
	|	$te*$	Zero or more te
	|	$\_ \xrightarrow{p} @l\ pvs$	Triple constraint with property type p whose nodes satisfy the shape l and property-values pvs
pvs	::=	$\lfloor ps \rfloor$	Open property-value specifiers ps
	|	$\lceil ps \rceil$	Closed property-value specifiers ps
ps	::=	$ps_1\ ,\ ps_2$	Each of ps_1 and ps_2
	|	$ps_1 \mid ps_2$	OneOf of ps_1 or ps_2
	|	$ps*$	zero of more ps
	|	$p : cond_v$	Property p with value conforming to $cond_v$ $cond_{v_s}$ is a boolean condition on sets of values $v_s \subseteq V$

Example 9. As an example, we can define a PShEx schema that describes the property graph from example 8 where $hasType_t$ is a condition that is satisfied when the set of types of a node contains the type t, i.e. $hasType_t(vs) =$ `true` if $t \in vs$ and $String, Float$ are conditions on the values that are satisfied when the values have the corresponding type.

$$
\begin{aligned}
L &= \{ Person \} \\
\delta(Person) &= hasType_{Human}\ \textbf{AND} \\
&\quad \lfloor name : String \mid firstname : String*, lastname : String \rfloor\ \textbf{AND} \\
&\quad \{ \_ \xrightarrow{knows} @Person\ \lfloor certainty : Float \rfloor^* \}
\end{aligned}
$$

In order to define the semantic specification of PShEx we will need to define the neighborhood of a node in a property graph.

Definition 6 (Neighborhood of node in property graph)**.** The neighbors of a node $n \in N$ in a property graph $G = \langle N, E, \rho, \lambda_n, \lambda_e, \sigma \rangle$ are defined as $neighs(n) = \{(n, p, y, vs) \mid \exists v \in E$ such that $\rho(v) = (n, y) \wedge \lambda_e(v) = p \wedge vs = \{(k, v) \mid \sigma(k, v) = ws \wedge v \in ws\}\}$

The semantic specification of PShEx can be defined in a similar way to the ShEx one. Given a property graph G, and a shape assignment τ, a node identifier $n \in N$ conforms with a shape expression se, which is represented as $G, n, \tau \vDash se$ and follows the rules presented in 5 where $preds(te)$ is the set of edge labels (or predicates) that appear in a triple expression te and can be defined as:

$$Cond_{ts}\frac{\lambda_n(n) = vs \quad cond_{ts}(vs) = true}{G, n, \tau \vDash cond_{ts}} \qquad AND\frac{G, n, \tau \vDash se_1 \quad G, n, \tau \vDash se_2}{G, n, \tau \vDash se_1 \ \texttt{AND}\ se_2}$$

$$ClosedShape\frac{neighs(n, G) = ts \quad G, ts, \tau \Vdash s'}{G, n, \tau \vDash \texttt{CLOSED}\ \{te\}}$$

$$OpenShape\frac{ts = \{\langle x, p, y\rangle \in neighs(n, G) \mid p \in preds(te)\} \quad G, ts, \tau \Vdash te}{G, n, \tau \vDash \{te\}}$$

Table 5
Rules for PShEx shape expressions

$$\begin{array}{lcl} preds(te_1; te_2) & = & preds(te_1) \cup preds(te_2) \\ preds(te_1 \mid te_2) & = & preds(te_1) \cup preds(te_2) \\ preds(\_ \xrightarrow{p} te) & = & \{p\} \\ preds(te*) & = & preds(te) \\ preds(\epsilon) & = & \emptyset \end{array}$$

As in the case of ShEx, the previous definition uses a second conformance relation defined on sets of triples ts instead of nodes n. The set of neighborhood nodes ts from a property graph G conforms to a triple expression te with shape assignment τ, written $G, ts, \tau \Vdash s$, as defined by the inference rules represented in table 6.

$$EachOf\frac{(ts_1, ts_2) \in part(ts) \quad G, ts_1, \tau \Vdash te_1 \quad G, ts_2, \tau \Vdash te_2}{G, ts, \tau \Vdash te_1; te_2}$$

$$OneOf_1\frac{G, ts, \tau \Vdash te_1}{G, ts, \tau \Vdash te_1 \mid te_2} \qquad OneOf_2\frac{G, ts, \tau \Vdash te_2}{G, ts, \tau \Vdash te_1 \mid te_2}$$

$$TripleConstraint\frac{ts = \{\langle x, p, y, s\rangle\} \quad G, y, \tau \vDash @l \quad G, s, \tau \vdash qs}{G, ts, \tau \Vdash \_ \xrightarrow{p} @l\ qs}$$

$$Star_1\frac{}{G, \emptyset, \tau \Vdash te*}$$

$$Star_2\frac{(ts_1, ts_2) \in part(ts) \quad G, ts_1, \tau \Vdash te \quad G, ts_2, \tau \Vdash te*}{G, ts, \tau \Vdash te*}$$

Table 6
Rules for PShEx triple expressions

In the case of PShEx we declare a new conformance relationship $G, s, \tau \vdash qs$ between a graph

G a set $s \in P \times V$ of property-value elements, a shape assignment τ and a property-value specifier pvs whose rules are defined in table 7 where $props(pvs)$ is the set of properties that appear in a property-value specifier ps and can be defined as:

$$OpenPVs\frac{s' = \{(p,v) \in s | p \in props(ps)\} \quad G, s', \tau \vdash ps}{G, s, \tau \vdash \lfloor ps \rfloor} \qquad ClosePVs\frac{G, s, \tau \vdash ps}{G, s, \tau \vdash \lceil ps \rceil}$$

$$EachOfPs\frac{G, s, \tau \vdash ps_1 \quad G, s, \tau \vdash ps_2}{G, s, \tau \vdash ps_1\ ,\ ps_2}$$

$$OneOfPs_1\frac{G, s, \tau \vdash ps_1}{G, s, \tau \vdash ps_1 \mid ps_2} \qquad OneOfPs_2\frac{G, s, \tau \vdash ps_2}{G, s, \tau \vdash ps_1 \mid ps_2}$$

$$StarPs_1\frac{}{G, \emptyset, \tau \vdash ps*} \qquad StarPs_2\frac{(s_1, s_2) \in part(s) \quad G, s_1, \tau \vdash ps \quad G, s_2, \tau \vdash ps*}{G, s, \tau \vdash ps*}$$

$$PropertyValue\frac{s = \{(p,w)\} \quad conv_v(w) = \texttt{true}}{G, s, \tau \vdash p\ :\ cond_v}$$

Table 7
Rules for PShEx property-value specifiers

$$\begin{array}{lcl} props(ps_1\ ,\ ps_2) & = & props(ps_1) \cup props(ps_2) \\ props(ps_1 \mid ps_2) & = & props(ps_1) \cup props(ps_2) \\ props(ps*) & = & preds(ps) \\ props(p\ :\ cond_v) & = & \{p\} \end{array}$$

As in the case of ShEx, the semantics of ShEx schemas can be defined independently on shape assignments. A shape assignment τ for graph G and S is called *valid* if for every node n in G and every shape expression label l defined in S, if $n@l \in \tau$, then $G, n, \tau \vDash @l$.

6. Related work

ShEx was initially proposed in 2014 [2] as a concise and human readable language to describe and validate RDF. It was based on a variant of regular expressions called Regular Bag Expressions [12] which also supports recursive shapes. Combining negation with recursive shapes was later studied in [9] where a well founded formal semantics for ShEx for proposed based on stratification. Our definition of ShEx is based on that work, although we omitted negation, disjunction and EXTRA declarations of shape expressions in this paper. After ShEx was proposed, a W3C Data Shapes working group was chatered whose result was SHACL, proposed as a recommendation in 2017 [3]. The specification of SHACL didn't have an abstract syntax and left the semantics of recursive shapes as an implementation dependent feature enabling the appearance of several proposals that define an abstract syntax and add semantics for SHACL with negation and recursion [13, 14, 15, 16]. A comparison between both ShEx and SHACL was provided in [17] while in [18], a simple language was defined that can be used as a common subset of both. We consider that some of our extensions to ShEx could also be applied to SHACL.

The approach followed in this paper to extend ShEx was started in this paper [19] where we had already proposed an initial version of PShEx and WShEx [20], another extension of ShEx to support the Wikibase data model. WShEx is conceptually similar to PShEx although a distinct feature of the Wikidbase data model is that the values of properties can also be nodes in the graph, which can be considered as a generalized property graphs model. This model was called MARS (Multi-Attributed Relational Structures) in [21].

There are several proposals for property graphs schemas. GQL is an upcoming ISO standard (ISO39075) [6] which is currently being developed and addresses the property graph model. In order to provide support for GQL, PG-Schema [22] was proposed as a simple schema language for property graphs. PG-Schema does not support cardinality constraints on edges, which could be simulated using PG-Keys [23]. In [24], the authors propose a common framework for property graph schema languages based on first order logic rules which supports cardinality constraints. The closest proposal to PShEx would be the Property Graph Shapes Language (ProGS) [11] although that language is based on SHACL and some of the differences SHACL vs ShEx could also be applied to ProGS vs PShEx. For example, PShEx doesn't have property path expressions and ProGS doesn't have regular bag expressions. Another difference is that ProGS includes negation and recursion while in the version of PShEx included in this paper we omitted negation. Comparison between RDF-Star and property graphs at the data model have already been studied. In [25] proposes a common model for RDF, RDF-Star and property graphs that they call statement graph, which is inspired by the OneGraph [26] vision. The conversion between property graphs and RDF/RDF-Star is also studied in [6].

7. Conclusions and future work

We have presented three extensions to ShEx for different types of knowledge graphs: ShEx-* adds support for RDF-Star, ShEx-N adds a constraint to describe nodes that act as properties, and PShEx can be used to describe and validate property graphs. We presented an abstract syntax and formal semantics for each of the variants.

There are several lines of future work. On the theoretical level, it will be necessary to research the computational complexity implications of these features as well as the combination of these features with other features from ShEx like negation, disjunction, EXTRA declarations, and even the potential integration of all the features together in a single language. On the implementation level: it will be necessary to propose a concrete compact syntax for each variant that can be useful by data model engineers as well as devise algorithms that can be used to check the conformance of graphs to the corresponding schemas. Although we used ShEx in this paper, another line of future work would be to check which of the features proposed in this paper could also be applied to SHACL.

Acknowledgments

This work has been partially funded by the Project ANGLIRU: *ANGLIRU: Applying kNowledge Graphs to research data interoperabiLIty and ReUsability*, code: PID2020-117912RB. The discus-

[6] https://www.iso.org/standard/76120.html

sions held during our participation in Dagstuhl seminar 24102 Shapes in graph data: theory and implementation helped us to write some of the ideas included in this paper.

References

[1] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutiérrez, S. Kirrane, J. E. Labra Gayo, R. Navigli, S. Neumaier, A.-C. Ngonga Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. F. Sequeda, S. Staab, A. Zimmermann, Knowledge Graphs, number 22 in Synthesis Lectures on Data, Semantics, and Knowledge, Springer, 2021. URL: https://kgbook.org/. doi:10.2200/S01125ED1V01Y202109DSK022.

[2] E. Prud'hommeaux, J. E. Labra Gayo, H. Solbrig, Shape Expressions: An RDF Validation and Transformation Language, in: H. Sack, A. Filipowska, J. Lehmann, S. Hellmann (Eds.), Proceedings of the 10th International Conference on Semantic Systems, SEMANTICS 2014, Leipzig, Germany, September 4-5, 2014, ACM Press, 2014, pp. 32–40. doi:10.1145/2660517.2660523.

[3] H. Knublauch, D. Kontokostas, Shapes Constraint Language (SHACL), W3C Recommendation 20 July 2017, W3C Recommendation, World Wide Web Consortium, 2017. URL: https://www.w3.org/TR/2017/REC-shacl-20170720/.

[4] R. Cyganiak, D. Wood, M. Lanthaler, RDF-star and SPARQL-star, Draft Community Group Report, World Wide Web Consortium, 2021. URL: https://w3c.github.io/rdf-star/cg-spec.

[5] O. hartig, P.-A. Champin, G. Kellogg, A. Seaborne, RDF 1.2 Concepts and Abstract Syntax, W3C Working Draft, World Wide Web Consortium, 2024. URL: https://www.w3.org/TR/rdf12-concepts/.

[6] S. Khayatbashi, S. Ferrada, O. Hartig, Converting property graphs to rdf: a preliminary study of the practical impact of different mappings, in: Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences and Systems (GRADES) and Network Data Analytics (NDA), SIGMOD/PODS '22, ACM, 2022. doi:10.1145/3534540.3534695.

[7] I. Boneva, J. E. Labra Gayo, E. G. Prud'hommeaux, Semantics and Validation of Shapes Schemas for RDF, in: C. d'Amato, M. Fernández, V. A. M. Tamma, F. Lécué, P. Cudré-Mauroux, J. F. Sequeda, C. Lange, J. Heflin (Eds.), The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I, volume 10587 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 104–120.

[8] E. Prud'hommeaux, I. Boneva, J. E. Labra Gayo, G. Kellog, Shape Expressions Language 2.0, 2017. URL: https://shexspec.github.io/spec/.

[9] I. Boneva, J. E. Labra Gayo, E. Prud'hommeaux, Semantics and validation of shapes schemas for rdf, in: International Semantic Web Conference, 2017.

[10] O. Hartig, Foundations of RDF* and SPARQL* – An Alternative Approach to Statement-Level Metadata in RDF, in: J. L. Reutter, D. Srivastava (Eds.), Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017, volume 1912 of *CEUR Workshop Proceedings*, Sun SITE Central Europe (CEUR), 2017. URL: http://ceur-ws.org/Vol-1912/paper12.pdf.

[11] P. Seifer, R. Lämmel, S. Staab, ProGS: Property Graph Shapes Language, in: International

Semantic Web Conference, volume 12922, Springer, 2021, pp. 392–401. doi:https://doi.org/10.1007/978-3-030-88361-4_23. arXiv:2107.05566.

[12] S. Staworko, I. Boneva, J. E. Labra Gayo, S. Hym, E. G. Prud'hommeaux, H. R. Solbrig, Complexity and Expressiveness of ShEx for RDF, in: 18th International Conference on Database Theory, ICDT 2015, volume 31 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 195–211.

[13] J. Corman, J. L. Reutter, O. Savković, Semantics and Validation of Recursive SHACL, in: D. Vrandečić, K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L. Kaffee, E. Simperl (Eds.), The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I, volume 11136 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 318–336.

[14] M. Andresel, J. Corman, M. Ortiz, J. L. Reutter, O. Savkovic, M. Simkus, Stable model semantics for recursive shacl, in: Proceedings of The Web Conference 2020, WWW '20, ACM, 2020. doi:10.1145/3366423.3380229.

[15] B. Bogaerts, M. Jakubowski, Fixpoint semantics for recursive SHACL, in: A. Formisano, Y. A. Liu, B. Bogaerts, A. Brik, V. Dahl, C. Dodaro, P. Fodor, G. L. Pozzato, J. Vennekens, N. Zhou (Eds.), Proceedings 37th International Conference on Logic Programming (Technical Communications), ICLP Technical Communications 2021, Porto (virtual event), 20-27th September 2021, volume 345 of *EPTCS*, 2021, pp. 41–47. URL: https://doi.org/10.4204/EPTCS.345.14. doi:10.4204/EPTCS.345.14.

[16] A. Chmurovic, M. Šimkus, Well-founded semantics for recursive shacl, in: CEUR Workshop Proceedings :, number 3203 in CEUR Workshop proceedings, 2022, pp. 2–13. URL: http://ceur-ws.org/Vol-3203/.

[17] J. E. Labra Gayo, E. Prud'hommeaux, I. Boneva, D. Kontokostas, Validating RDF Data, volume 7 of *Synthesis Lectures on the Semantic Web: Theory and Technology*, Morgan & Claypool, 2017. URL: https://doi.org/10.2200/s00786ed1v01y201707wbe016. doi:10.2200/s00786ed1v01y201707wbe016.

[18] J. E. Labra Gayo, H. García-González, D. Fernández-Alvarez, E. Prud'hommeaux, Challenges in RDF Validation, in: G. Alor-Hernández, J. L. Sánchez-Cervantes, A. Rodríguez-González, R. Valencia-García (Eds.), Current Trends in Semantic Web Technologies: Theory and Practice, Studies in Computational Intelligence, Springer, 2019, pp. 121–151. doi:10.1007/978-3-030-06149-4_6.

[19] J. E. L. Gayo, Creating knowledge graphs subsets using shape expressions, 2021. arXiv:2110.11709.

[20] J.-E. Labra-Gayo, Wshex: A language to describe and validate wikibase entities, in: L. Kaffee, S. Razniewski, G. Amaral, K. S. Alghamdi (Eds.), Proceedings of the 3rd Wikidata Workshop 2022 co-located with the 21st International Semantic Web Conference (ISWC2022), Virtual Event, Hanghzou, China, October 2022, volume 3262 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3262/paper3.pdf.

[21] M. Marx, M. Krötzsch, V. Thost, Logic on MARS: ontologies for generalised property graphs, in: C. Sierra (Ed.), Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17), International Joint Conferences on Artificial Intelligence, 2017, pp. 1188–1194. doi:10.24963/ijcai.2017/165.

[22] R. Angles, A. Bonifati, S. Dumbrava, G. Fletcher, A. Green, J. Hidders, B. Li, L. Libkin,

V. Marsault, W. Martens, F. Murlak, S. Plantikow, O. Savkovic, M. Schmidt, J. Sequeda, S. Staworko, D. Tomaszuk, H. Voigt, D. Vrgoc, M. Wu, D. Zivkovic, Pg-schema: Schemas for property graphs, Proceedings of the ACM on Management of Data 1 (2023) 1–25. doi:10.1145/3589778.

[23] R. Angles, A. Bonifati, S. Dumbrava, G. Fletcher, K. W. Hare, J. Hidders, V. E. Lee, B. Li, L. Libkin, W. Martens, F. Murlak, J. Perryman, O. Savković, M. Schmidt, J. Sequeda, S. Staworko, D. Tomaszuk, Pg-keys: Keys for property graphs, in: Proceedings of the 2021 International Conference on Management of Data, SIGMOD/PODS '21, ACM, 2021. doi:10.1145/3448016.3457561.

[24] N. Beeren, G. Fletcher, A formal design framework for practical property graph schema languages, in: J. Stoyanovich, J. Teubner, N. Mamoulis, E. Pitoura, J. Mühlig, K. Hose, S. S. Bhowmick, M. Lissandrini (Eds.), Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023, OpenProceedings.org, 2023, pp. 478–484. URL: https://doi.org/10.48786/edbt.2023.40. doi:10.48786/EDBT.2023.40.

[25] E. Gelling, G. Fletcher, M. Schmidt, Bridging graph data models: Rdf, rdf-star, and property graphs as directed acyclic graphs, CoRR abs/2304.13097 (2023). URL: https://doi.org/10.48550/arXiv.2304.13097. doi:10.48550/ARXIV.2304.13097. arXiv:2304.13097.

[26] O. Lassila, M. Schmidt, O. Hartig, B. Bebee, D. Bechberger, W. Broekema, A. Khandelwal, K. Lawrence, C. M. Lopez Enriquez, R. Sharda, B. Thompson, The onegraph vision: Challenges of breaking the graph model lock-in1, Semantic Web 14 (2022) 125–134. doi:10.3233/sw-223273.