Extending Shape Expressions for different types of Knowledge Graphs

Jose Emilio Labra Gayo
WESO Research group
University of Oviedo, Spain

Motivation

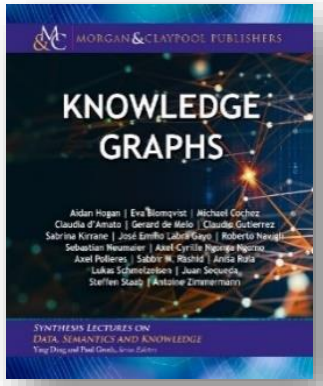
Improve data quality of kowledge graphs

- We need tools to describe and validate their content

Knowledge Graphs = flexible, graph-like data

Several types of Knowledge Graphs

- RDF-based Knowledge Graphs
- Labeled Property Graphs
- Wikibase Graphs (based on Wikidata)
- RDF-Star



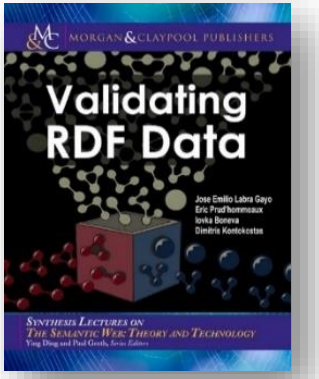
https://kgbook.org/

ShEx

Concise & human-readable language

Goal: describe & validate RDF data

- Developed in 2014
- Compact syntax similar to Turtle, SPARQL
- Semantics inspired by regular expressions



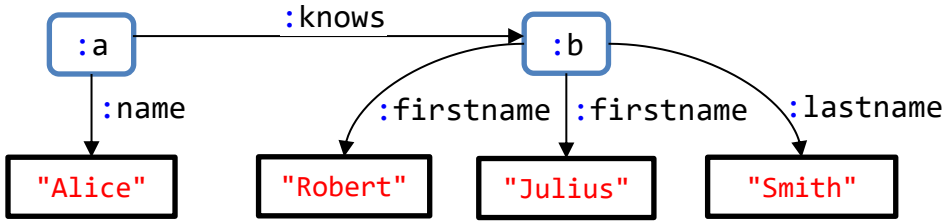
http://book.validatingrdf.com

Adopted by Wikidata in 2019 (Entity Schemas Namespace)

2024 IEEE Shape Expressions Working Group

Simple RDF

- Graph = Set of triples
  - Triple = (Subject, Predicate, Object)
  - where
    - Subject = IRI or BNode
    - Predicate = IRI
    - Object = IRI or BNode or Literal
- Several syntaxes:
  - Turtle, N-Triples, …
- RDF Ecosystem:
  - SPARQL, RDFS, OWL,…



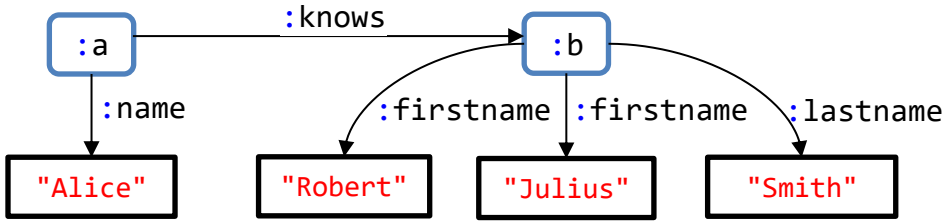

```
prefix : <http://example.org/>

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

ShEx example

A Shape describes the neighbourhood of a node

- Triple Expressions
- Node constraints
- Cardinality




```
prefix : <http://example.org/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>

<Person> {
 :firstname xsd:string * ;
 :lastname  xsd:string
 :knows     @<Person>  *
}
```

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

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

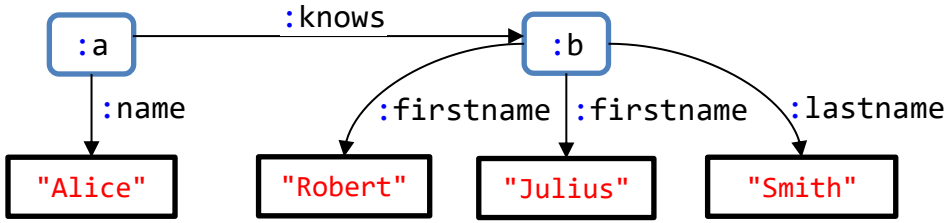
ShEx example

ShEx accepts regular expresión operators on triple expressions

EachOf (;), OneOf (|), grouping

```
prefix :    <http://example.org/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>

<Person> {
    ( :name      xsd:string
    | :firstName xsd:string * ;
      :lastname  xsd:string
    ) ;
    :knows       @<Person> *
}
```




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

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

```
:c :name      "Carol" ;
   :firstname "Carol" ;
   :knows     :b      .
```

Abstract syntax of ShEx

ShEx schema is a tuple $<L, \delta>$

where L = set of labels and δ: $L \rightarrow se$

se	::=	cond	Basic boolean condition on nodes (node constraint)
	|	se_1 AND se_2	Conjunction of se_1 and se_2
	|	$@l$	Shape label reference for $l \in L$
	|	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

Example with ShEx abstract syntax

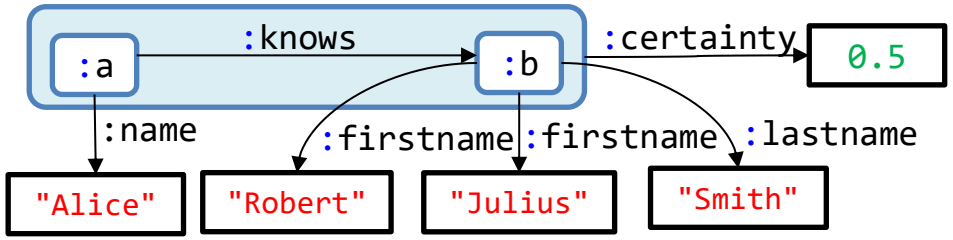
```
prefix :    <http://example.org/>
prefix xsd: <http://.../XMLSchema#>

<Person> {
    ( :name   xsd:string
    | :firstName xsd:string *;
      :lastname xsd:string
    ) ;
    :knwows @<Person> *
}
```

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

RDF-Star

RDF-Star (RDF 1.2) extends RDF allowing subjects and objects to be triples




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

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

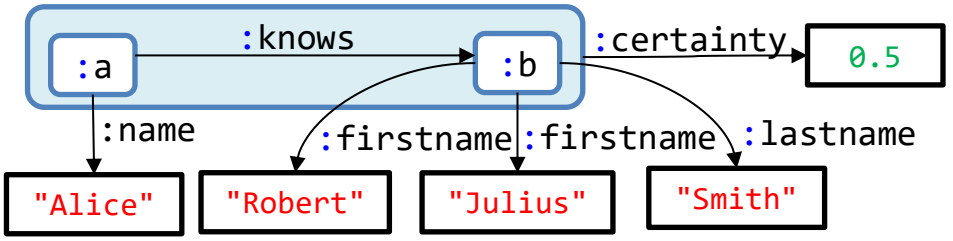
ShEx-Star

Example

```
<Person> {
    ( :name    xsd:string
    | :firstName xsd:string + ;
      :lastname xsd:string
    ) ;
   << :knwows @<Person> >> {| :certainty xsd:decimal |} *
}
```

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

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



ShEx-Star abstract syntax

ShEx-Star adds two new rules for triple expressions te

$se \quad ::= \quad \ldots$ same as in ShEx

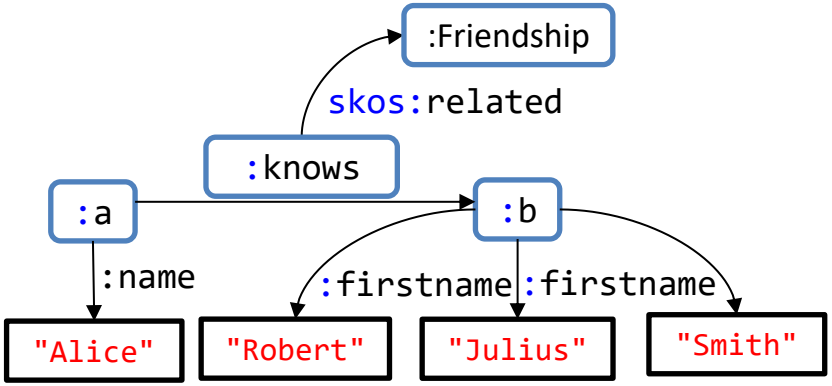
te	$::=$	$te_1; te_2$	Each of te_1 and te_2
	$\mid$	$te_1 \mid te_2$	Either te_1 or te_2
	$\mid$	$te*$	Zero or more te
	$\mid$	$\llcorner \xrightarrow{p} se$	Outgoing Triple with predicate p and object conforming to se
	$\mid$	$se \xrightarrow{p} \llcorner$	Incoming triple with predicate p and subject conforming to se
	$\mid$	ϵ	Empty triple expression
	$\mid$	$\ll \llcorner \xrightarrow{p} se \gg \{\mid te \mid\}$	Outgoing Triple term constraint with predicate p
	$\mid$	$\ll se \xrightarrow{p} \llcorner \gg \{\mid te \mid\}$	Incoming triple term constraint with predicate p

$$\delta(Person) = \{ \; (\llcorner \xrightarrow{name} String \mid \llcorner \xrightarrow{fistname} String*; \llcorner \xrightarrow{lastname} String); \ll \llcorner \xrightarrow{knows} @Person \gg \{\mid \llcorner \xrightarrow{certainty} Float \mid\}* \; \}$$

```
<Person> {
    ( :name   xsd:string
    | :firstName xsd:string + ;
      :lastname xsd:string
    ) ;
   << :knwows @<Person> >> {| :certainty xsd:decimal |} *
}
```

RDF with nodes as properties

In RDF Graphs, nodes can also be properties




```
prefix : <http://example.org/>
prefix skos: <http://.../skos/core#>

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

ShEx-N

Example:

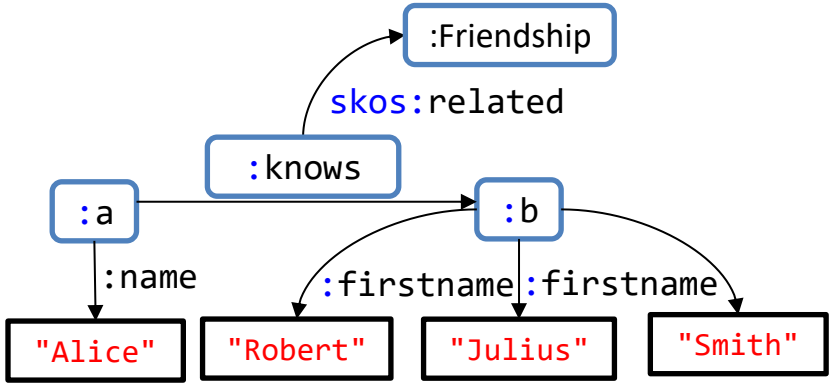
```
<FriendshipProperty> {
  skos:related [ :Friendship ] ;
  @<Person> >-< @<Person>
}
<Person> {
    ( :name   xsd:string
    | :firstName xsd:string + ;
      :lastname xsd:string
    ) ;
    :knows @<Person> *
}
```

$$\delta(FriendShipProperty) = \{ \; \_ \xrightarrow{skos:related} [:Friendship] \; ; \; @Person \rightleftharpoons @Person \; \}$$

$$\delta(Person) = \{ \; \_ \xrightarrow{:name} String \; | \; \_ \xrightarrow{:firstname} String* \; ; \; \_ \xrightarrow{:lastname} String \; ; \; \_ \xrightarrow{knows} @Person \; * \; \}$$

```
prefix : <http://example.org/>
prefix skos: <http://.../skos/core#>

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



ShEx-N abstract syntax

We add a new rule for triple expressions *te*

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 $l \in L$
	|	`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
	|	$se_1 \xrightarrow{\_} se_2$	Triple constraint with focus node acting as predicate and subject conforming to se_1 and object conforming to se_2

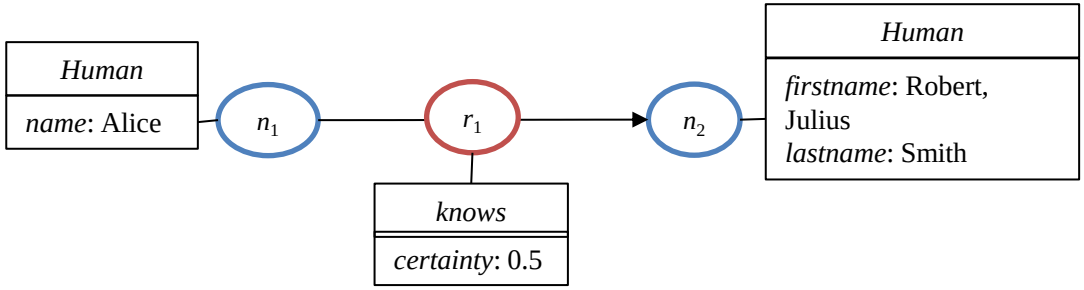
Property graphs

Nodes and edges can have a set of property – value pairs

Nodes can have labels and edges can have a label

Quite popular in industry: Neo4J, TinkerPop, Neptune, etc.

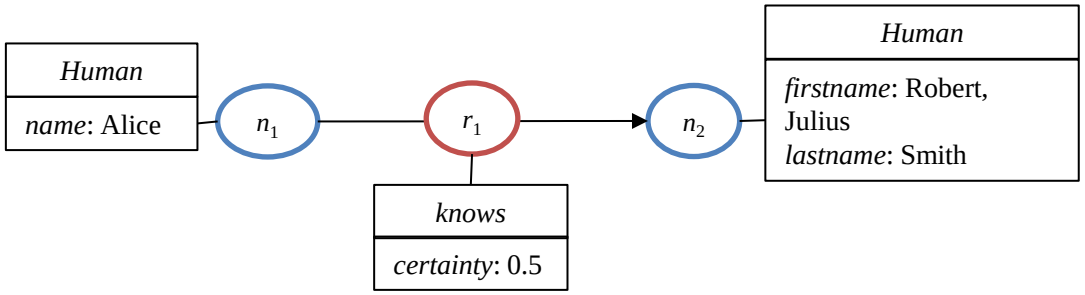
Example



PShEx: ShEx for property graphs

Example

```
<Person> [ Human ] AND
  [| name: String |
     firstname: String *,
     lastname: String
  |] AND {
     knows @<Person> [|
       certainty: Decimal
     |] *
  }
```



PShEx abstract syntax

PShEx adds property value specifiers (*pvs*) to nodes and edges

se	::=	$cond_{t_s}$	Basic boolean condition on set of types $t_s \subseteq T$
	|	se_1 AND se_2	Conjunction
	|	$@l$	Shape label reference for $l \in L$
	|	pvs	Property-value specifiers of a node
	|	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
	|	$\llcorner \xrightarrow{p} @ pvs$	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 $v_s \subseteq V$

```
<Person> [ Human ] AND
  [| name: String |
     firstname: String *,
     lastname: String
  |] AND {
     knows @<Person> [|
       certainty: Decimal
     |] *
  }
```

$$
\begin{array}{rcl}
L & = & \{ Person \} \\
\delta(Person) & = & hasType_{Human}\ \text{AND} \\
 & & \lfloor name : String \mid firstname : String*, lastname : String \rfloor\ \text{AND} \\
 & & \{ \llcorner \xrightarrow{knows} @Person\ \lfloor certainty : Decimal \rfloor * \}
\end{array}
$$

Semantics

We define the semantics using 2 conformance relationships and several inference rules

$$G, n, \tau \vDash se = \text{node } n \text{ in graph } G \text{ conforms to } se \text{ with assignment } \tau$$

$$G, ts, \tau \Vdash te = \text{neighborhood } ts \text{ of graph } G \text{ conform to } te \text{ with assignment } \tau$$

More details in the paper

ShEx semantics

Shape expressions *se*

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

$$ShapeRef\frac{\delta(l) = se \quad G, n, \tau \vDash se}{G, n, \tau \vDash @l} \qquad ClosedShape\frac{neighs(n, G) = ts \quad G, ts, \tau \Vdash te}{G, n, \tau \vDash \texttt{CLOSED } \{te\}}$$

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

Triple expressions *te*

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

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

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

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

ShEx-Star semantics

Same rules as for ShEx plus:

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

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

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

ShEx-N semantics

Same rules as in ShEx plus:

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

$$\begin{array}{lcl}
\delta(FriendShipProperty) & = & \{ \quad \sqcup \xrightarrow{skos{:}related} [:Friendship] \ ; \\
 & & \quad @Person \overset{\sqcup}{\rightarrow} @Person \\
 & & \} \\
\delta(Person) & = & \{ \quad \sqcup \xrightarrow{:name} String \\
 & & \quad | \quad \sqcup \xrightarrow{:firstname} String* \ ; \sqcup \xrightarrow{:lastname} String \ ; \\
 & & \quad \sqcup \xrightarrow{knows} @Person \ * \\
 & & \}
\end{array}$$

PShEx semantics

Semantics of shape expressions *se* (similar to ShEx)

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

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

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

PShEx semantics

Semantics of property value specifiers *ps*

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

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

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

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

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

PShEx semantics

Semantics of triple expressions *te* (similar to ShEx)

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

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

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

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

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

Conclusions

ShEx = similar to a Grammar for Knowledge graphs

Can be extended for other kinds of Knowledge Graphs

- RDF-Star: ShEx-Star
- General RDF: ShEx-N
- Property graphs: P-ShEx
- Wikibase graphs: WShEx*

* *WShEx: A language to describe and validate Wikibase entities*, Jose E. Labra G., In Wikidata Workshop, International Semantic Web Conference. CEUR Proceedings, vol 3265 -2022

Future work

Prioritize which of those lines to follow
Use cases and usability of tools are important

- ShEx-Star
  - Align with current work on RDF 1.2
  - Implement prototype
- ShEx-N
  - Define compact syntax and implement it
  - Identify use cases and expressiveness
- PShEx
  - Define compact syntax and implement prototype
  - We have already implemented WShEx (Wikibase data model)
    - WShEx was very useful to describe and create Wikidata subsets

End