

June 4st, 2025

Graphwise

Vienna University of Economics and Business

OWL strict

A Constrained OWL Fragment for Knowledge Graph Practicioners



Robert David, Albin Ahmeti, Shqiponja Ahmetaj, Axel Polleres



Motivation

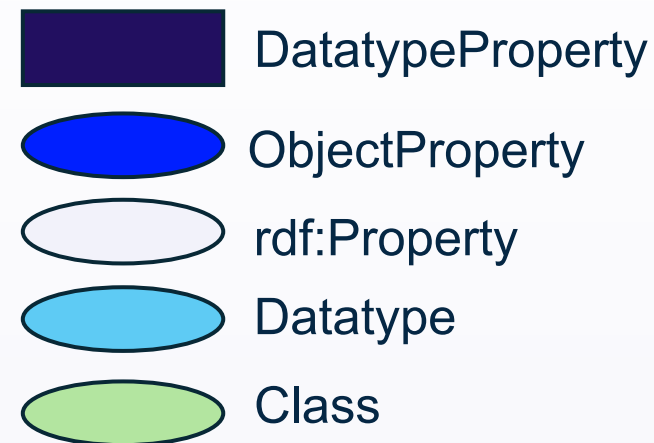
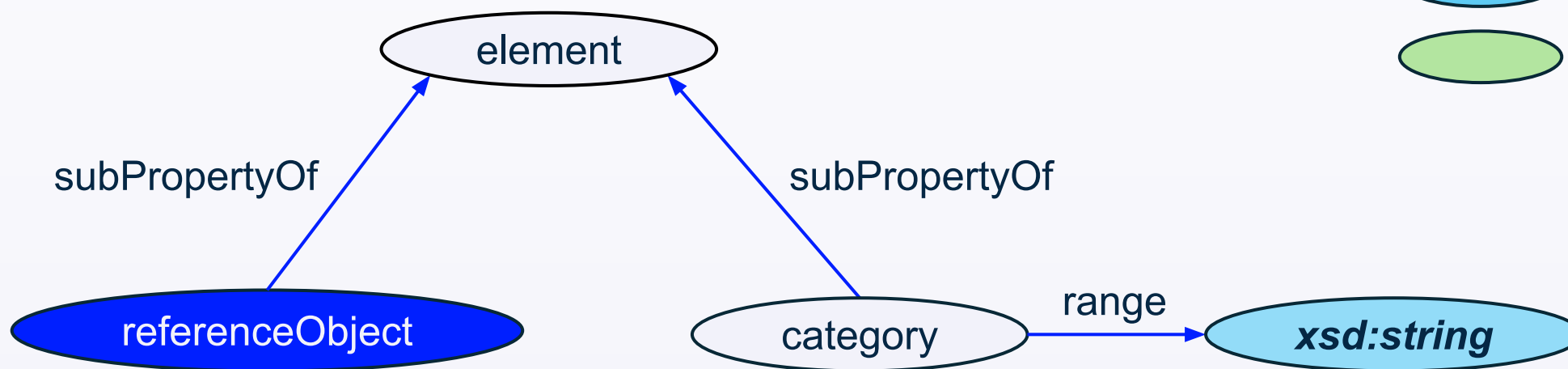


- RDF is a very flexible, does not restrict use
- OWL ontologies are represented using RDF
- **BUT:** Flexibility causes ambigUlties!
- Our Scenario: Enterprise Knowledge Graphs - what matters?
 - Quality & stability of schemas
 - Editors to manage ontologies
 - Applications using (instance) data



Ambiguity Example

Part of the DITA ontology





AmbigUity Example – Protégé

Active ontology x Entities x Individuals by class x DL Query x

Annotation properties | Datatypes | Individuals

Classes | Object properties | Data properties

Object property hierarchy: category ? ? ? ? ?

Annotations Usage

Annotations: category

Annotations +

owl:topObjectProperty

- element
 - category
 - referenceObject

Characteristics ? ? ? ? ?

Description: category

☐ Functional

☐ Inverse functional

☐ Transitive

☐ Symmetric

☐ Asymmetric

☐ Reflexive

☐ Irreflexive

Equivalent To +

SubProperty Of +

Inverse Of +

Domains (intersection) +

Ranges (intersection) +

Disjoint With +

SuperProperty Of (Chain) +

element

Active ontology x Entities x Individuals by class x DL Query x

Annotation properties | Datatypes | Individuals

Classes | Object properties | Data properties

Data property hierarchy: category ? ? ? ? ?

Annotations Usage

Annotations: category

Annotations +

owl:topDataProperty

- category
- element

Characteristic ? ? ? ? ?

Description: category

☐ Functional

Equivalent To +

SubProperty Of +

Domains (intersection) +

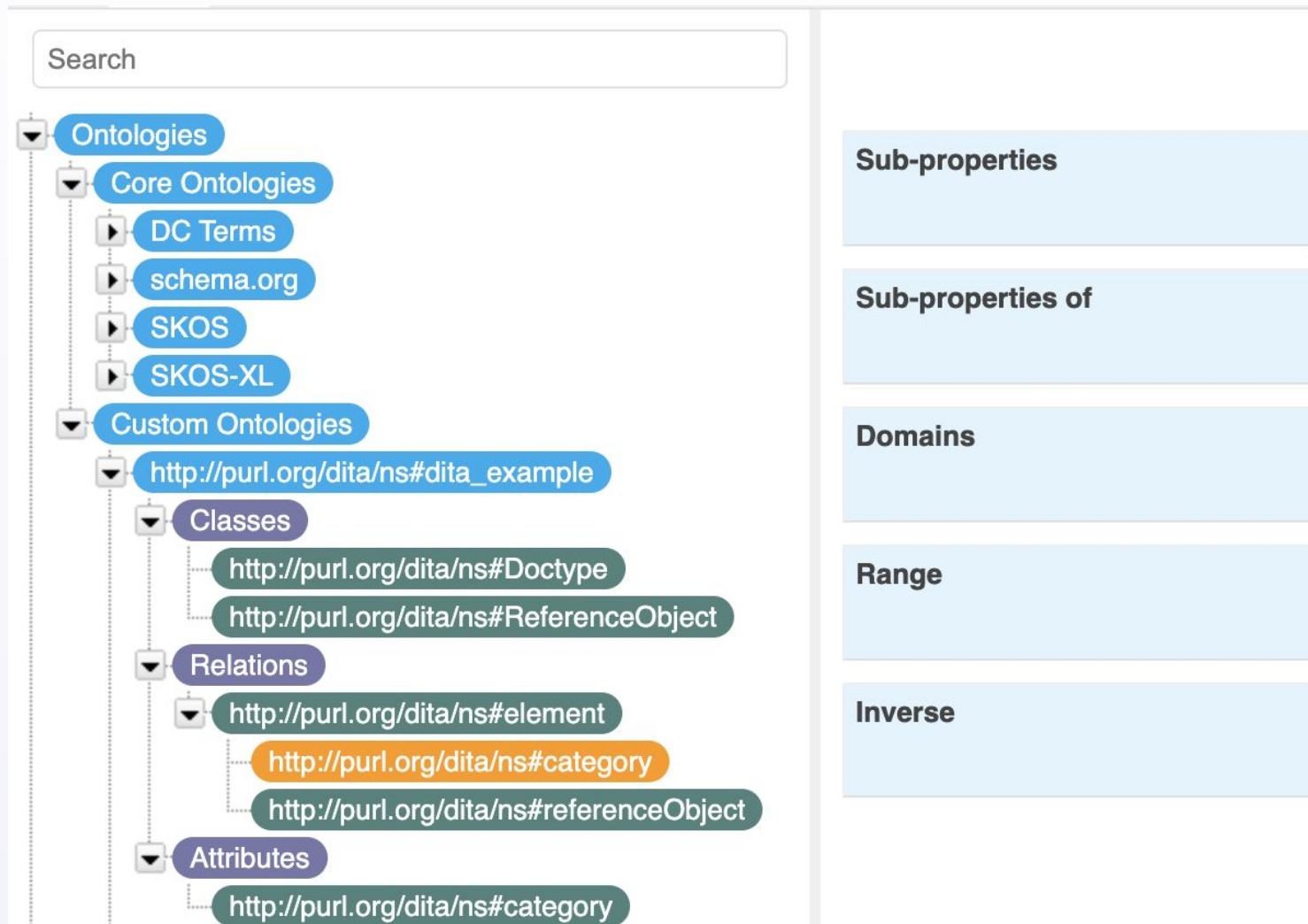
Ranges +

xsd:string

Disjoint With +

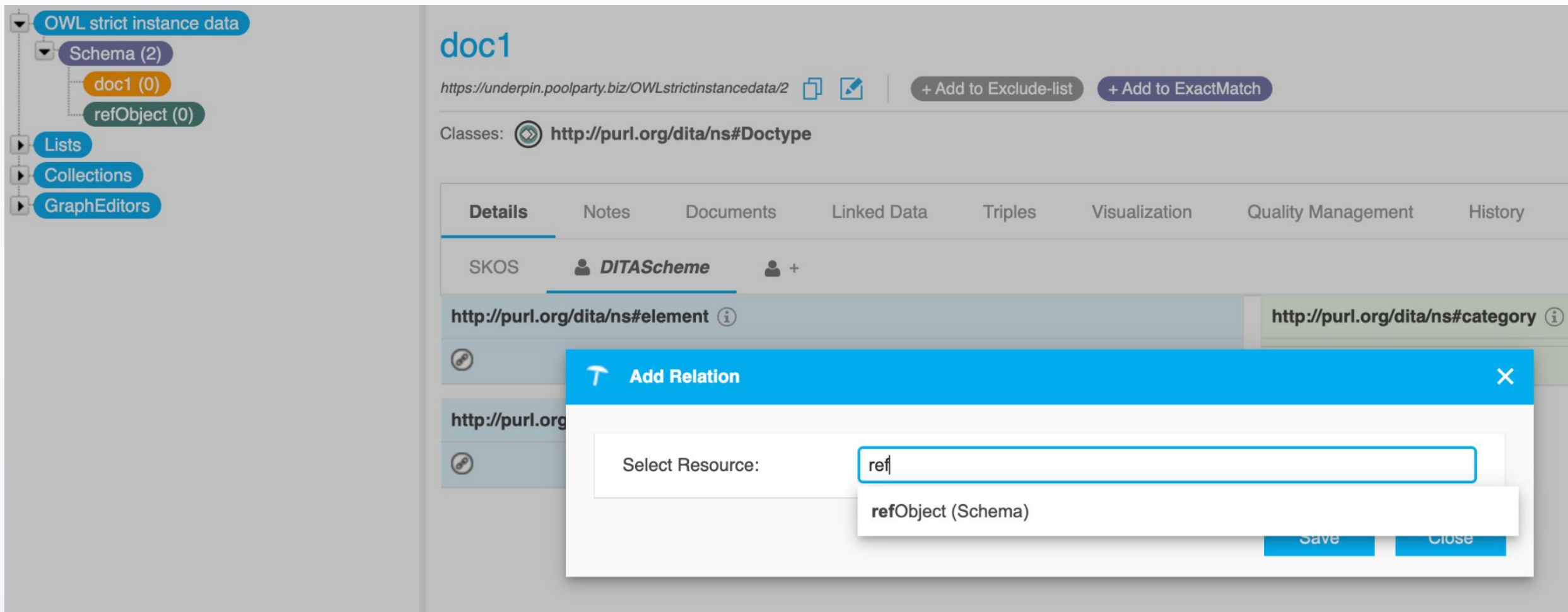


AmbigUity Example – PoolParty





Concrete Range Type – Relation



The screenshot displays the GRAPHWISE interface. On the left, a sidebar shows a tree structure: 'OWL strict instance data' (expanded) contains 'Schema (2)' (expanded) with 'doc1 (0)' and 'refObject (0)', and 'Lists', 'Collections', and 'GraphEditors'. The main area shows 'doc1' with the URL 'https://underpin.poolparty.biz/OWLstrictinstancedata/2'. Below this, 'Classes:' lists 'http://purl.org/dita/ns#Doctype'. A tabbed interface at the bottom includes 'Details' (selected), 'Notes', 'Documents', 'Linked Data', 'Triples', 'Visualization', 'Quality Management', and 'History'. The 'Details' tab shows 'SKOS' and 'DITAScheme'. A table lists resources: 'http://purl.org/dita/ns#element' and 'http://purl.org/dita/ns#category'. An 'Add Relation' dialog box is open, showing 'Select Resource:' with a search input containing 'ref' and a dropdown list with 'refObject (Schema)'. 'Save' and 'Close' buttons are at the bottom right of the dialog.



Concrete Range Type - Datatype

OWL strict instance data

Schema (2)

doc1 (0)

refObject (0)

Lists

Collections

GraphEditors

doc1

<https://underpin.poolparty.biz/OWLstrictinstancedata/2>

+ Add to Exclude-list

+ Add to ExactMatch

Classes: <http://purl.org/dita/ns#Doctype>

Details

Notes

Documents

Linked Data

Triples

Visualization

Quality Management

History

SKOS

DITAScheme

+

<http://purl.org/dita/ns#element>

<http://purl.org/dita/ns#referenceObject>

<http://purl.org/dita/ns#category>

Add Literal

Literal:

Save

Close



OWL strict



- Goal: to resolve ambiguities
 - by rules and constraints...
 - ... enforcing ***syntactic restrictions*** on the RDF graph→ “well-defined/behaving fragment”
- i.e., syntax constraints only (RDF graph); no OWL reasoner needed



How about existing OWL fragments?



GRAPHWISE
AI THRIVES ON WHOLE DATA

Name	W3C Rec.	Purpose
RDF 1999	yes	first RDF recommendation version
RDF 1.0	yes	February 2004
RDF 1.1	yes	June 2014
RDFS 1.0	yes	schema description
RDFS 1.1	yes	schema description
Minimal RDF(S) [26]	no	small and practical
OWL 1 Lite	yes	minimum complexity
OWL 1 DL	yes	Description logics
OWL 1 Full	yes	maximum expressivity
OWL 2 EL	yes	many classes & properties
OWL 2 QL	yes	query rewriting
OWL 2 RL	yes	rules & reasoning
RDFS++	no	AllegroGraph's minimal OWL profile
OWL Horst [30]	no	scalable reasoning
OWLSIF	no	Oracle's OWL with if-semantics
OWLPrime	no	Oracle's application-oriented OWL profile
OWL DBP [3]	no	DBPedia schema
OWL Wikidata	no	Wikidata schema
OWL LD [15]	no	Linked data practical use
OWL IC [29]	no	integrity constraints
OWL Flight [11]	no	logic programming & constraints

- None of those “does the job”... Reasoning ***does not fix*** ambiguities



OWL strict has an orthogonal goal:



Restricting the ***vocabulary use*** (*tailored for editors*):

$C = \{ \textit{RDFSClass}, \textit{OWLClass}, \textit{Datatype}, \textit{Property},$
 $\textit{ObjectProperty}, \textit{DatatypeProperty}, \textit{AnnotationProperty},$
 $\textit{OntologyProperty}, \textit{SymmetricProperty},$
 $\textit{FunctionalProperty} \}$

$P = \{ \textit{subClassOf}, \textit{subPropertyOf}, \textit{domain}, \textit{range},$
 $\textit{inverseOf}, \textit{disjointWith} \}$



Requirements to avoid Ambiguities



- **R-0** “standard use” of the RDF(S) and OWL vocab, i.e.

~~`rdf:type rdfs:subClassOf my:Class.`~~

- **R-1** ontology terms in exactly one of the OWL built-in classes
OWLClass, Datatype, ObjectProperty, DatatypeProperty
- **R-2** “consistent” *subpropertyOf* hierarchy, i.e., subproperties preserve the same class membership
- **R-3** *domain* and *range* restricted to *one* (union of) class, i.e. choices to satisfy the domain/range
- **R-4** “consistent” *domain* and *range* in *subpropertyOf* hierarchies, i.e. same (sub)classes on super-properties



Restricting the syntax:

- 26 syntactic restrictions on the ontology RDF graph
- Implement the requirements **R-0** to **R-4**
- Examples
 - **SR-4** If $\text{subPropertyOf}(x, y) \in G$, then x and y are both members of exactly one class of *ObjectProperty* or *DatatypeProperty*.
 - **SR-13** If $\text{range}(x, y), \text{DatatypeProperty}(x) \in G$, then $\text{Datatype}(y) \in G$.
 - **SR-18** If $\text{subPropertyOf}(x, y), \text{ObjectProperty}(x) \in G$, then $\text{ObjectProperty}(y) \in G$.



Expressing OWL strict in SHACL



Set of 10 SHACL shapes to implement syntactic restrictions.

Disjoint class membership constraints (for all ontology terms):

ClassShape XOR DatatypeShape XOR PropertyShape

PropertyShape -> ObjectPropertyShape XOR DatatypePropertyShape

Additional property restrictions:

FunctionalPropertyShape, SymmetricPropertyShape, InversePropertyShape

Consistency in the subproperty hierarchy:

DomainConsistencyShape, RangeConsistencyShape



Expressing OWL strict in SHACL

SHACL implementation example:

:ObjectPropertyShape a sh:NodeShape;

```
sh:class owl:ObjectProperty;
```

SR-10

```
sh:property [ sh:path rdfs:range; sh:node :ClassShape; sh:maxCount 1 ];
```

SR-12

```
sh:property [ sh:path ( rdfs:range owl:unionOf [ sh:zeroOrMorePath rdf:rest ] rdf:first );  
  sh:node :ClassShape; ];
```

SR-12

```
sh:property [ sh:path rdfs:subPropertyOf; sh:class owl:ObjectProperty; ];
```

SR-17

```
sh:not [ sh:class owl:DatatypeProperty; ].
```

SR-0



Transforming ontologies to OWL strict



Using **SHACL repairs** (see our ISWC 2022 paper):

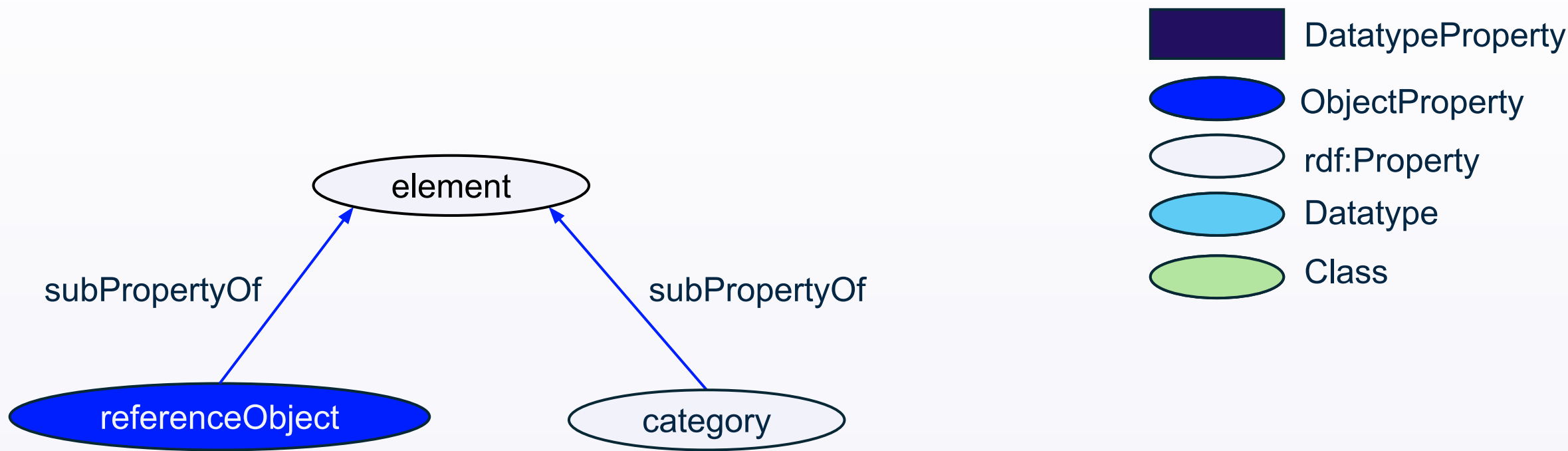
- Algorithm to modify SHACL data graph ontology RDF to conform to the SHACL constraints:
 - produces **minimal** sets of additions and deletions of triples...
 - ... guaranteed to conform to OWL strict



Repairing Ambiguities Example 1/3



GRAPHWISE
AI THRIVES ON WHOLE DATA



SR-4 If $subPropertyOf(x, y) \in G$, then x and y are both members of exactly one class of ObjectProperty or DatatypeProperty.



SR-18 If $subPropertyOf(x, y), ObjectProperty(x) \in G$, then $ObjectProperty(y) \in G$.

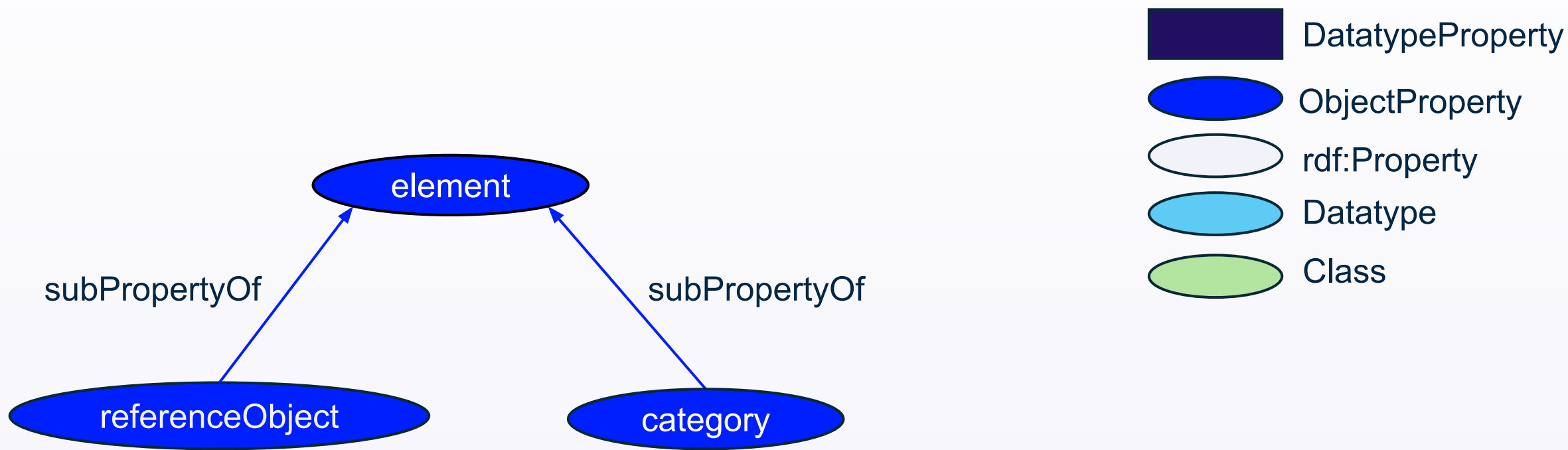
SR-19 If $subPropertyOf(x, y), DatatypeProperty(x) \in G$, then $DatatypeProperty(y) \in G$.



Repairing Ambiguities Example 1/3



GRAPHWISE
AI THRIVES ON WHOLE DATA



SR-4 If $subPropertyOf(x, y) \in G$, then x and y are both members of exactly one class of ObjectProperty or DatatypeProperty.



SR-18 If $subPropertyOf(x, y), ObjectProperty(x) \in G$, then $ObjectProperty(y) \in G$.

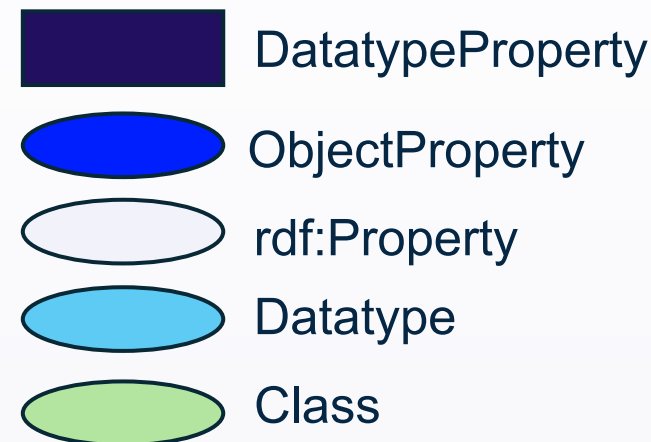
SR-19 If $subPropertyOf(x, y), DatatypeProperty(x) \in G$, then $DatatypeProperty(y) \in G$.



Repairing Ambiguities Example 2/3



GRAPHWISE
AI THRIVES ON WHOLE DATA



SR-6 If $range(x, y) \in G$, then x is a member of exactly one class of ObjectProperty or DatatypeProperty.

SR-12 If $range(x, y), ObjectProperty(x) \in G$, then $OWLClass(y) \in G$.

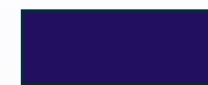
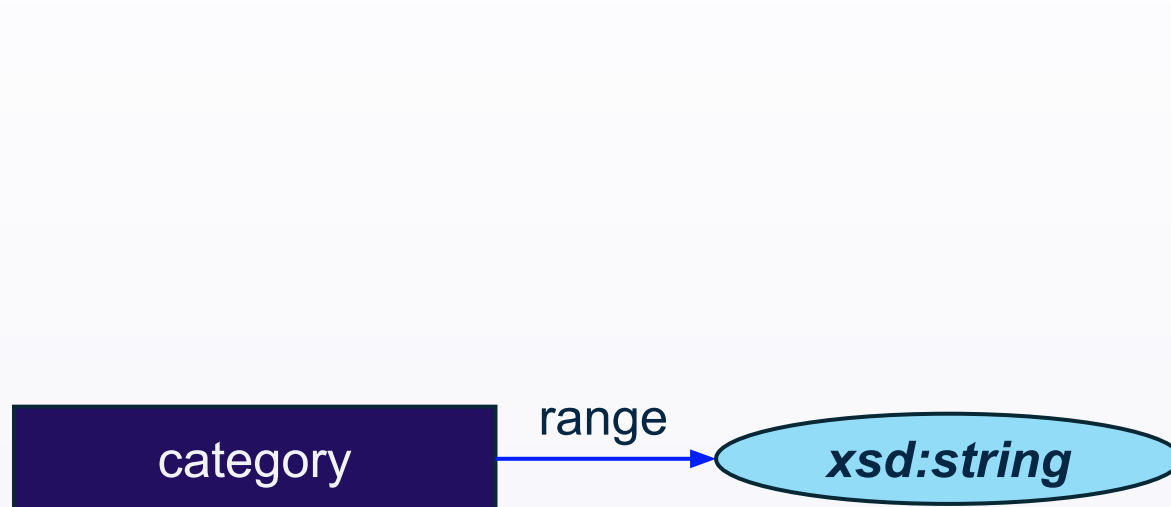
SR-13 If $range(x, y), DatatypeProperty(x) \in G$, then $Datatype(y) \in G$.



Repairing Ambiguities Example 2/3



GRAPHWISE
AI THRIVES ON WHOLE DATA



DatatypeProperty



ObjectProperty



rdf:Property



Datatype



Class



SR-6 If $range(x, y) \in G$, then x is a member of exactly one class of ObjectProperty or DatatypeProperty.

SR-12 If $range(x, y), ObjectProperty(x) \in G$, then $OWLClass(y) \in G$.

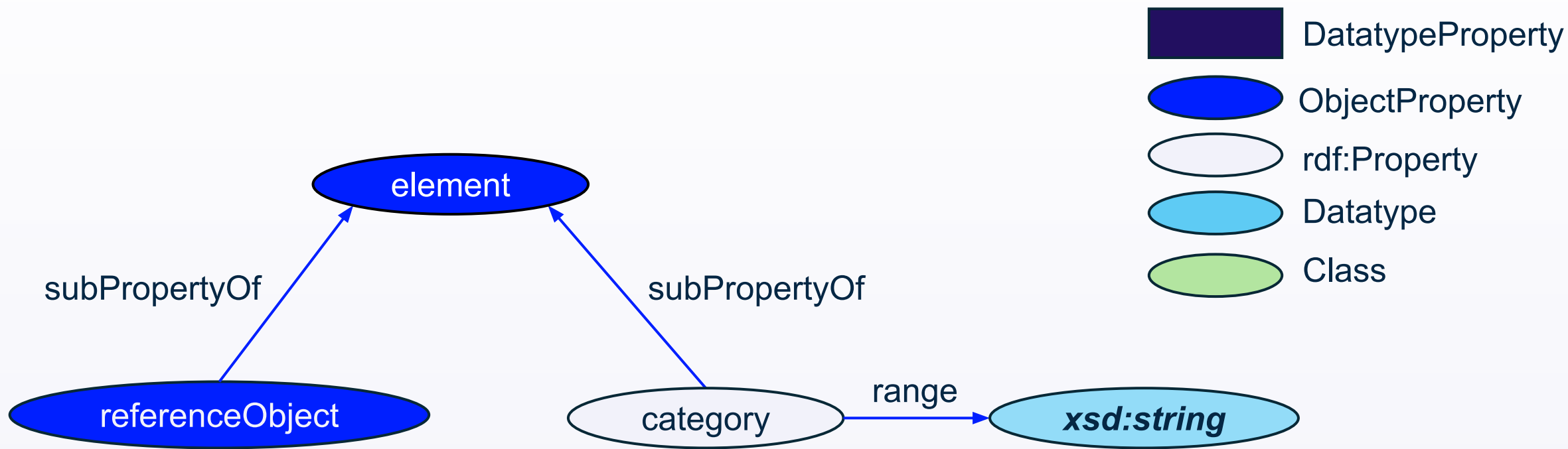
SR-13 If $range(x, y), DatatypeProperty(x) \in G$, then $Datatype(y) \in G$.



Repairing Ambiguities Example 3/3



GRAPHWISE
AI THRIVES ON WHOLE DATA



SR-0 There is no node in $V(G)$ that belongs to more than 1 of the classes **OWLClass**, **Datatype**, **ObjectProperty** and **DatatypeProperty**.

SR-18 If $subPropertyOf(x, y)$, $ObjectProperty(x) \in G$, then $ObjectProperty(y) \in G$.

SR-13 If $range(x, y)$, $DatatypeProperty(x) \in G$, then $Datatype(y) \in G$.

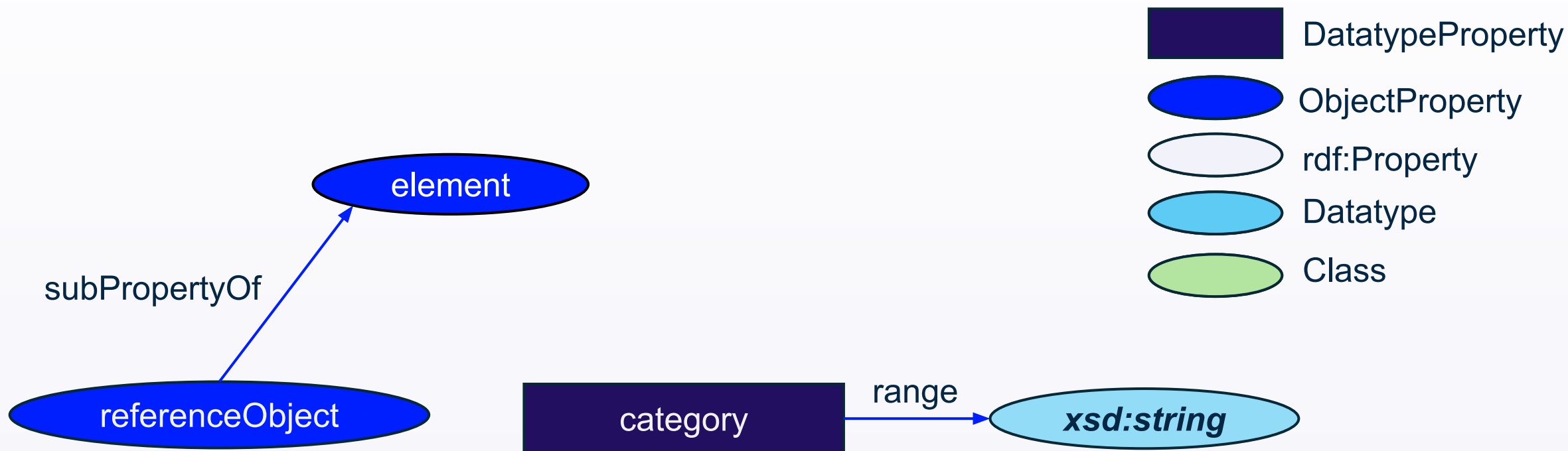




Repairing Ambiguities Example 3/3



GRAPHWISE
AI THRIVES ON WHOLE DATA



SR-0 There is no node in $V(G)$ that belongs to more than 1 of the classes `OWLClass`, `Datatype`, `ObjectProperty` and `DatatypeProperty`.

SR-18 If $subPropertyOf(x, y)$, $ObjectProperty(x) \in G$, then $ObjectProperty(y) \in G$.

SR-13 If $range(x, y)$, $DatatypeProperty(x) \in G$, then $Datatype(y) \in G$.





Evaluation with published ontologies



GRAPHWISE
AI THRIVES ON WHOLE DATA

We “repaired” the following ontologies:

- Darwin Information Typing Architecture (DITA)
- gist
- European Union Agency for Railways (ERA)
- SemOpenAlex
- EBUCorePlus
- Data Product Model (DPROD)



Evaluation with published ontologies

Ontology	Initial size	Additions	Deletions	OWL _{strict} size	Conforms
DITA	490	104	9	585	yes
gist core	3060	50	9	3101	yes
ERA	7163	69	54	7178	yes
SemOpenAlex	848	0	0	848	yes
EBUCorePlus	11930	2044	27	13947	yes
DPROD	147	11	7	151	yes

Table 2. Statistics for OWL_{strict} ontologies.



Open questions/Future works:



Conformance for ontologies was successfully achieved, BUT:

- Preferences - define repair choices, e.g. keep `rdfs:subPropertyOf`
 - which ones? how to specify?
- Additional quality requirements, e.g. labels
- Performance optimizations for repairs (large ontologies)

THANK YOU