RDFS

INFO216 – Lab 7

RDFS subClassOf

```
g.add((ex.Cade, RDF.type, ex.Student))
```

CADE

← **A (rdf type) Student**

```
g.add((ex.Student, RDFS.subClassOf, FOAF.Person))
```

Every Student is also a Person

⇒ (implies)

CADE

← **A (rdf type) Person**

RDFS subPropertyOf

```
g.add((ex.Cade, ex.studentAt, ex.UIB))
```

CADE

← **Student At UIB**

```
g.add((ex.studentAt, RDFS.subPropertyOf, ex.attends))
```

Someone who is a student somewhere also attends that place

⇒ (implies)

CADE

← **Attends UIB**

RDFS range & domain

```
g.add((ex.Emma, ex.flyTo, ex.Bergen))
```

In a “flyTo” triple, the subject is always a person, and the object is always a city

```
g.add((ex.flyTo, RDFS.domain, FOAF.Person))
g.add((ex.flyTo, RDFS.range, ex.City))
```

```
g.add((ex.Emma, RDF.type, FOAF.Person))
g.add((ex.Bergen, RDF.type, ex.City))
```

THESE ARE ADDED!

⇒ (implies)

A (rdf type) Person →  EMMA

 BERGEN ← **A (rdf type) City**

Like this

TIP: In a domain triple, imagine the subject is replaced with the “original” subject, and the property replaced with rdf:type; and the subject is replaced with the “original” object in a range triple.

RDFS Closure

RDFSClosure / DeductiveClosure

The entailments (and axioms) of your graph can be shown with closure.

```
engine = owlrl.RDFSClosure.RDFS_Semantics(graph, False, False, False)
engine.closure()
engine.flush_stored_triples()

OR

owlrl.DeductiveClosure(owlrl.RDFS_Semantics).expand(g)
```

Axioms (non-datatype)

Daxioms (datatype)

RDFS

Closure Showcase in VSCode

https://drive.google.com/file/d/1ZirgVmG35LnQAIvWsKwZF6DYMb4nKYUB/

INFO216_Coding > 2023 > Labs > lab7 > lab7_example.py > ...

```python
from rdflib import Graph, RDFS, Namespace, RDF, FOAF, Literal, XSD
import owlrl

g = Graph()
ex = Namespace('http://example.org/')

g.bind("ex", ex)
g.bind("foaf", FOAF)

#populate the graph
g.add((ex.Cade, RDF.type, ex.Student))
g.add((ex.Cade, ex.studentAt, ex.UIB))

g.add((ex.Emma, ex.flyTo, ex.Bergen))

#rules
g.add((ex.Student, RDFS.subClassOf, FOAF.Person))
g.add((ex.studentAt, RDFS.subPropertyOf, ex.attends))

g.add((ex.flyTo, RDFS.domain, FOAF.Person))
g.add((ex.flyTo, RDFS.range, ex.City))


print(g.serialize())

```