RDF Schema (RDFS)

What is RDF Schema?

We can use RDF triples to express facts like:

```
ex:AlbertEinstein ex:discovered ex:TheoryOfRelativity .
```

But how we can refine such a fact?

- How we can define that the predicate `ex:discovered` has a person as subject and a theory as object?
- How we can express that Albert Einstein was a researcher and that every researcher is a human?

Such knowledge is called *schema* knowledge or *terminological* knowledge

✓ ***RDF Schema gives us the possibility to model such knowledge***

RDF Schema (short: RDFS)

- ➔ is part of the W3C RDF recommendation family
- ➔ used for schema/terminological knowledge
- ➔ itself is an RDF vocabulary (thus every RDF Schema graph is an RDF graph)
- ➔ its vocabulary is generic (not bound to a specific application area)
- ➔ allows to specify semantics of user-defined RDF vocabularies

The Namespace of RDF Schema is http://www.w3.org/2000/01/rdf-schema#

(common prefix: rdfs)

See Here: https://dvcs.w3.org/hg/rdf/raw-file/default/rdf-schema/index.html

Classes

- ➔ A *Class* is a set of things (or entities). In RDF these things are identified by URIs.
- ➔ The membership of an entity to a class is defined using the **rdf:type** property.

The fact that ex:MyBlueVWGolf is a member/instance of the class ex:Car can be expressed:

```
ex:MyBlueVWGolf   rdf:type   ex:Car .
```

A resource can belong to several classes:

```
ex:MyBlueVWGolf   rdf:type   ex:Car .
ex:MyBlueVWGolf   rdf:type   ex:GermanProduct .
```

Hierarchies of Classes

Classes can be arranged in hierarchies using the **rdfs:subClassOf** property.

Every ***ex:Car*** is an *ex:MotorVehicle*:

```
ex:Car   rdfs:subClassOf   ex:MotorVehicle .
```

Implicit Knowledge

Using the schema definition, we are able to identify implicit knowledge.

```
ex:MyBlueVWGolf  rdf:type          ex:Car .
ex:Car           rdfs:subClassOf   ex:MotorVehicle
```

implicitly contains the following statement as a logical consequence

```
ex:MyBlueVWGolf   rdf:type   ex:MotorVehicle .
```

Implicit Knowledge

The statements

```
ex:Car           rdfs:subClassOf   ex:MotorVehicle .
ex:MotorVehicle  rdfs:subClassOf   ex:Vehicle .
```

implicitly contains the following statement as a logical consequence

```
ex:Car   rdfs:subClassOf   ex:Vehicle .
```

We can see that rdfs:subClassOf is transitive.

Defining a Class

Every URI denoting a class is an instance of **rdfs:Class**. For defining an own class we have to write:

```
ex:Car   rdf:type   rdfs:Class .
```

Which makes rdfs:Class itself an instance of rdfs:Class.

```
rdfs:Class   rdf:type   rdfs:Class .
```

Equivalence of Classes

To express the equivalence of two classes we can use

```
ex:Car        rdfs:subClassOf   ex:Automobile .
ex:Automobile rdfs:subClassOf   ex:Car .
```

Which leads to the statement

```
ex:Car   rdfs:subClassOf   ex:Car .
```

We can see that rdfs:subClassOf is **reflexive**.

Predefined RDF Classes

There are several other predefined classes than rdfs:Class:

- **rdfs:Resource** is the class of all things. It is the superclass of all classes.
- **rdf:Property** is the class of all properties.
- **rdfs:Datatype** is the class of all datatypes.
- (Every instance of this class is a subclass of *rdfs:Literal*.)
- **rdfs:Literal** is the class of literal values such as Strings or Integers.
- (If such a literal is typed, it is an instance of *rdfs:Datatype*.)
- **rdf:langString** is the class of language-tagged string literals. It is an instance of *rdfs:Datatype* and a subclass of *rdfs:Literal*.

Predefined RDF Classes

- **rdf:XMLLiteral** is the class of XML literal values. Its a subclass of *rdfs:Literals* and an instance of *rdfs:Datatype*.
-
- **rdf:Statement** is the class of the RDF statements. So every RDF triple can be seen as an instance of this class with a *rdf:subject*, *rdf:predicate* and *rdf:object* property.
- **rdfs:Container** is a super-class of the RDF Container classes.
- (i.e. *rdf:Bag*, *rdf:Seq*, *rdf:Alt*)

Yes, *rdf:Property*, *rdf:XMLLiteral*, *rdf:Statement*, etc., are in the RDF vocabulary already, but only RDFS declares them to be classes.

Defining a Property

As we can define Classes we can define new Properties.

For expressing that there is a new Property we define it as an instance of the property class.

```
ex:drives   rdf:type   rdf:Property .
```

With this new Property we can express that Max drives a VW Golf (not just any one, but a specific one).

```
ex:Max   ex:drives   ex:MyBlueVWGolf .
```

Hierarchies of Properties

Using the **rdfs:subPropertyOf** property we can define a hierarchy of properties.

```
ex:drives   rdfs:subPropertyOf   ex:controls .
```

(You see that a vocabulary is often an idealized model of the real world!)

With the former statement

```
ex:Max   ex:drives   ex:MyBlueVWGolf .
```

We can infer that

```
ex:Max   ex:controls   ex:MyBlueVWGolf .
```

Range and Domain of Properties

Every property has a *Domain* and a *Range* that specify which class the subject or the object must have.

```
ex:Max   ex:drives   ex:MyBlueVWGolf .
  Domain                    Range
```

Using the Properties rdfs:domain and rdfs:range we can define the Domain and Range of a Property.

```
ex:drives   rdfs:domain   ex:Person .
ex:drives   rdfs:range    ex:Vehicle .
```

The same can be done for datatypes

```
ex:hasAge   rdfs:range   xsd:nonNegativeInteger .
```

Important to Understand

1. “must have” above is not a constraint in the sense of “if ex:MyBlueVWGolf is not an ex:Vehicle, then the RDF statement above is illegal”.
2. It means “given that ex:MyBlueVWGolf is used with ex:drives, we know that it is an ex:Vehicle (in addition to whatever else it may be)”.
3. Possibility (1) wouldn't make sense, as in RDF Schema there is no way of expressing that something is *not* an instance of some class.

Multiple Range Statements

The statements

```
ex:drives   rdfs:range   ex:Car .
ex:drives   rdfs:range   ex:Ship .
```

mean that the Range of `ex:drives` has to be both – an `ex:Car` and an `ex:Ship`!

If you want to express that the object of the Property has to be a car or a ship, a better expression would be

```
ex:Car      rdfs:subClassOf   ex:Vehicle .
ex:Ship     rdfs:subClassOf   ex:Vehicle .
ex:drives   rdfs:range        ex:Vehicle .
```

Implicit Knowledge

Once we define the Domain and Range of properties, we have to take care that using such properties could lead to unintended consequences. The schema

```
ex:isMarriedTo    rdfs:domain  ex:Person .
ex:isMarriedTo    rdfs:range   ex:Person .
ex:instituteAKSW  rdf:type     ex:Institution .
```

and the additional statement

```
ex:Max   ex:isMarriedTo   ex:instituteAKSW .
```

leads to the logical consequence

```
ex:instituteAKSW   rdf:type   ex:Person .
```

False Conclusion

Some people might be confused about the combination of specifying domain and range of a property and the hierarchy of the classes. So we want to look at an example:

```
ex:drives  rdfs:range       ex:Car .     # (1)
ex:Car     rdfs:subClassOf  ex:Vehicle . # (2)
```

These two triples do not entail the following relation

```
ex:drives   rdfs:range   ex:Vehicle .    # (3)
```

I.e. we do not gain new terminological knowledge. Still, of any concrete triple having the predicate **ex:drives**, we know that its object is of type **ex:Vehicle** – which is the same effect as if we had statement (3) in our schema.

Container Class

RDF Schema defines the **rdfs:Container** class which is a superclass for the Containers defined by RDF:

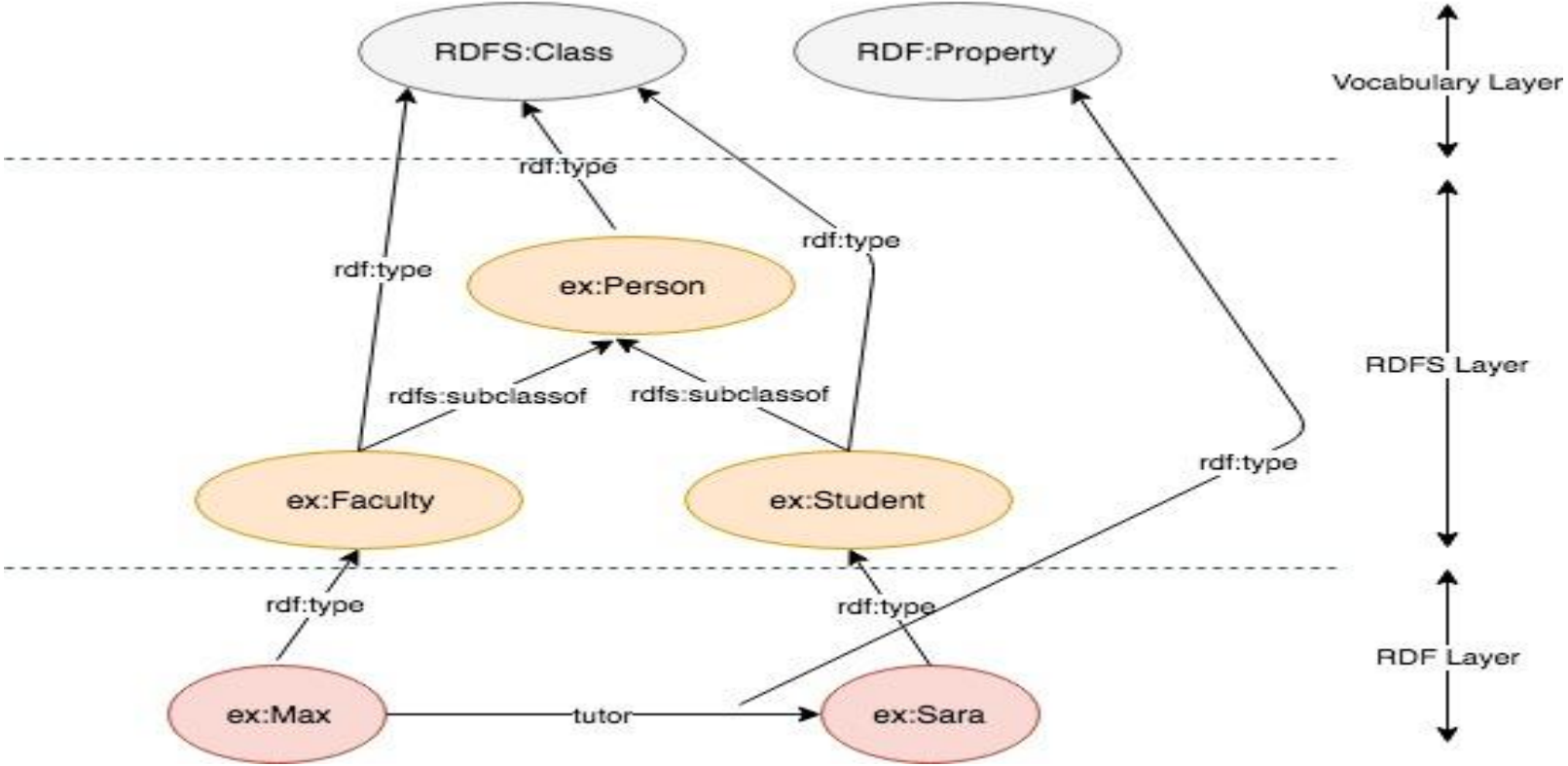
- **rdf:Seq**
- **rdf:Bag**
- **rdf:Alt**

Container Membership

RDF Schema defines new classes for working with Containers:

- The **rdfs:ContainerMembershipProperty** class which contains all properties that are used to state that a resource is a member of a container (e.g. *rdf:_1*, *rdf:_2*, …).
-
- The **rdfs:member** Property is a superproperty for all Properties of all Container membership Properties.
- (So every instance of *rdfs:ContainerMembershipProperty* is a *rdfs:subPropertyOf* the *rdfs:member* Property)

Overview


RDFS:Class
RDF:Property
Vocabulary Layer
rdf:type
rdf:type
rdf:type
ex:Person
RDFS Layer
rdfs:subclassof
rdfs:subclassof
rdf:type
ex:Faculty
ex:Student
rdf:type
rdf:type
RDF Layer
ex:Max
tutor
ex:Sara

Reference:

https://www.w3.org/TR/rdf-schema/

https://dvcs.w3.org/hg/rdf/raw-file/default/rdf-schema/index.html