RDF Schema (RDFS) Summary

RDF itself has no way to define application-specific classes and properties. Rather, they are described as an RDF vocabulary, using extensions to RDF provided by RDF Schema (or RDFS). The resources in the RDFS vocabulary have URIrefs beginning with

`http://www.w3.org/2000/01/rdf-schema#`

This is conventionally associated with QName prefix `rdfs:`. Vocabulary descriptions (schemas) written in RDFS are legal RDF graphs.

Describing Classes

A class corresponds to the generic concept of a Type or Category. Classes are described using the RDFS resources `rdfs:Class` and `rdfs:Resource` and the properties `rdf:type` and `rdfs:subClassOf`. A class is any resource having an `rdf:type` property whose value is the resource `rdfs:Class`. The `rdfs:subClassOf` property is transitive. The following is an N3 example of class structure with multiple subclasses.

```
ex:MotorVehicle       rdf:type          rdfs:Class .
ex:PassengerVehicle   rdf:type          rdfs:Class .
ex:Van                rdf:type          rdfs:Class .
ex:Truck              rdf:type          rdfs:Class .
ex:MiniVan            rdf:type          rdfs:Class .
ex:PassengerVehicle   rdfs:subClassOf   ex:MotorVehicle .
ex:Van                rdfs:subClassOf   ex:MotorVehicle .
ex:Truck              rdfs:subClassOf   ex:MotorVehicle .
ex:MiniVan            rdfs:subClassOf   ex:Van .
ex:MiniVan            rdfs:subClassOf   ex:PassengerVehicle .
```

The following is the RDF/XML version of this (using typed nodes).

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
"http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/vehicles">
<rdfs:Class rdf:ID="MotorVehicle"/>
<rdfs:Class rdf:ID="PassengerVehicle">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Truck">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Van">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>
<rdfs:Class rdf:ID="MiniVan">
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdfs:Class>
</rdf:RDF>
```

Describing Properties

In RDFS, properties are described using the RDF class `rdf:Property`, and the RDFS properties `rdfs:domain`, `rdfs:range`, and `rdfs:subPropertyOf`. All properties in RDF are instances of class `rdf:Property`. A new property is described by assigning the property a URIref, and describing that resource with an `rdf:type` property whose value is the resource `rdf:Property`, as in

```
exterms:weightInKg  rdf:type  rdf:Property .
```

RDF Schema provides vocabulary for describing how properties and classes are intended to be used together. The most important are RDFS properties `rdfs:range` and `rdfs:domain`. `rdfs:range` indicates that the values of a given property are instances of a designated class and can also be used to indicate that the value of a property is given by a typed literal. If a property is given more than one range class, then any resource that is a value of that property must be an instance of all those classes (i.e., the effective range is the intersection of the defined ranges). Note that there is no way to define datatypes in RDFS; datatypes are defined externally to RDF (and to RDFS), and referred to in RDF statements by their URIrefs. `rdfs:domain` indicates that a given property applies to a designated class. As with ranges, multiple domains for a given property are given an intersection interpretation.

Predefined property `rdfs:subPropertyOf` denotes the specialization relationship between two properties. The meaning of this relationship is that, if *P* is a subproperty of *Q* and *x P y*, then *x Q y*. A property may be a subproperty of zero, one, or more properties. All `rdfs:range` and `rdfs:domain` properties that apply to a property also apply to its subproperties.

The following is the full example of a vehicle schema (in RDF/XML), with multiple examples of subproperty, domain, and range relations.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/vehicles">
<rdfs:Class rdf:ID="MotorVehicle"/>
<rdfs:Class rdf:ID="PassengerVehicle">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Truck">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Van">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>
<rdfs:Class rdf:ID="MiniVan">
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Person"/>
<rdfs:Datatype rdf:about="&xsd;integer"/>
<rdf:Property rdf:ID="registeredTo">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
```

```
    <rdfs:range rdf:resource="#Person"/>
  </rdf:Property>
  <rdf:Property rdf:ID="rearSeatLegRoom">
    <rdfs:domain rdf:resource="#PassengerVehicle"/>
    <rdfs:range rdf:resource="&xsd;integer"/>
  </rdf:Property>
  <rdf:Property rdf:ID="driver">
    <rdfs:domain rdf:resource="#MotorVehicle"/>
  </rdf:Property>
  <rdf:Property rdf:ID="primaryDriver">
    <rdfs:subPropertyOf rdf:resource="#driver"/>
  </rdf:Property>
  </rdf:RDF>
```

Interpreting RDF Schema Declarations

RDF differs from most programming language type systems in several ways. One important difference is that a class in an object-oriented programming language has a collection of specific properties, whereas an RDF schema describes properties as applying to specific classes of resources, using domain and range properties. In the programming-language class description, attribute (say) `author` is part of the description of class (say) `Book` and applies only to instances of class `Book`. An attribute called `author` in another class is considered a different attribute. That is, the *scope* of an attribute description in programming languages is restricted to the class or type in which it is defined. In RDF, however, property descriptions are, by default, independent of class definitions. They have, by default, global scope (but may be restricted using domain specifications). A benefit of the RDF property-based approach is that it is easier to extend the use of property definitions to originally unanticipated cases.

Another result of the global scope of RDF property descriptions is that we cannot define a given property as having locally-different ranges depending on the class of the resource it's applied to.

Another important difference between programming-language type systems and RDF is that RDFS descriptions are not necessarily prescriptive like PL type declarations. For example, suppose a programming language declares a class `Book` with an `author` attribute having values of type `Person`. This gives a group of constraints. But RDFS provides schema information as additional descriptions of resources and does not prescribe how these descriptions should be used by an application. For example, suppose an RDF schema states that an `ex:author` property has an `rdfs:range` of class `ex:Person`. This schema-supplied information might be used in various ways.

- One application might interpret this statement as specifying part of a template for RDF data it is creating. It must then ensure that any `ex:author` property has a `ex:Person` value. That is, it interprets the schema description as a constraint (as a programming language does).
- Another application might interpret this statement as providing additional info about data it is receiving—information not explicitly provided in the original data.
- A third application might use the statement for consistency checking.

Depending on how the application interprets the property descriptions, a description of an instance might be considered valid either without some of the schema-specified

properties or with additional properties. RDF schema statements are always descriptions. They may also be prescriptive, but only if the application interpreting them wants to treat them so.

Other Schema Information

We here simply list several resources that provide supplementary information, along with a short description and identification of their domains and ranges when they are properties.

`rdfs:label` (domain `rdfs:Resource` and range `rdfs:Literal`) is a property used to provide a human-readable version of a resource's name.

`rdfs:comment` (domain `rdfs:Resource` and range `rdfs:Literal`) is a property used to provide a human-readable description of a resource.

`rdfs:Container` is a super-class of the RDF Container classes.

`rdfs:ContainerMembershipProperty` class has as instances the properties `rdf:_1`, `rdf:_2`, `rdf:_3` , ...

`rdfs:member` is a property that is a super-property of all the container membership properties.

`rdfs:seeAlso` (domain and range both `rdfs:Resource`) is a property used to indicate a resource that might provide additional information about the subject resource.

`rdfs:isDefinedBy` (domain and range both `rdfs:Resource`) is a property used to indicate a resource defining the subject resource.