Ontology: we have a reason for every term we use

why is the Web so successful?

Ontology: we have a reason for every term we use

- many reasons..., but one important reason:
- **anyone can publish anything, at any time**

Ontology: we have a reason for every term we use

- many reasons..., but one important reason:
- **anyone can publish anything, at any time**

for the Semantic Web idea to be successful, this still must be true:

anyone can publish anything, at any time

Ontology: we have a reason for every term we use

- many reasons..., but one important reason:
- **anyone can publish anything, at any time**

for the Semantic Web idea to be successful, this still has to be true:

anyone can publish anything, at any time

- on the current Web, you publish *HTML* blocks
- on the Semantic Web, you publish *RDF* blocks
- **because RDF is for the Semantic Web what HTML has been for the Web**

Ontology: we have a reason for every term we use

- and there is another major difference -
- **HTML is for human eyes, and RDF is for machine to read**

Ontology: we have a reason for every term we use

- and there is another major difference -
- **HTML is for human eyes, and RDF is for machine to read**

  therefore, RDF needs some **common terms** so that machine can share the **same understanding**

Ontology: we have a reason for every term we use

```
<?xml version="1.0"?>
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:myCamera="http://www.example.com/camera#">

   <rdf:Description
       rdf:about="http://www.example.com/camera#Nikon_D300">
     <rdf:type
        rdf:resource="http://www.example.com/camera#DSLR"/>
     <myCamera:manufactured_by
       rdf:resource="http://www.dbpedia.org/resource/Nikon"/>
     <myCamera:performance rdf:resource=
            "http://www.example.com/camera#PictureQuality"/>
   </rdf:Description>

   <rdf:Description
      rdf:about="http://www.example.com/camera#PictureQuality">
     <myCamera:evaluate>5 stars</myCamera:evaluate>
   </rdf:Description>

</rdf:RDF>
```

Ontology: we have a reason for every term we use

- why are we using the term `myCamera:manufactured_by`, `myCamera:performance`?
- somewhere, in some document, someone has defined that these are indeed the predicates one can use when describing a camera
- there are possibly more terms defined, and it is our choice which predicates to use when publishing our own descriptions
- if someone else is describing another camera, the same set of terms should be used
- **this way, machine can process these RDF documents easily**

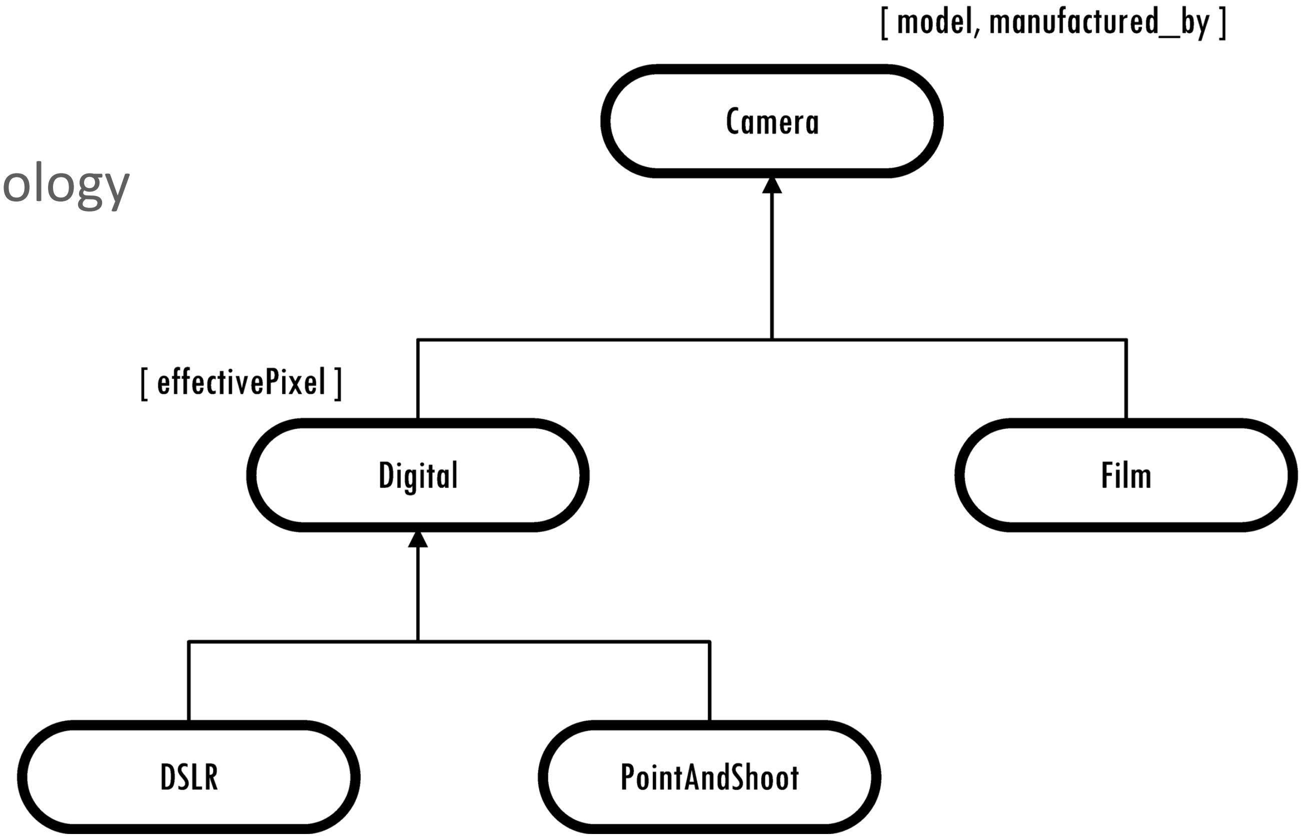
Ontology: we have a reason for every term we use

- a collection of these terms is called an **ontology**
- ontology contains *class* terms (classes, aka concept), and *relationships* among these terms
- ontology contains *property* terms, which describe various features and attributes of the classes/concepts
- ontology is often domain specific

Ontology: we have a reason for every term we use

example:

a simple ontology



RDFS: a language one can use to create ontologies

- RDFS = RDF Schema
- RDFS is a language one can use to create an ontology
- So, when distributed RDF documents are created, terms from this ontology can be used
- therefore, everything we say, we have a reason to say it

RDFS: a language one can use to create ontologies

- RDFS is a collection of terms one can use to define classes and properties for a specific domain
- just like RDF terms, all these RDFS terms are identified by pre-defined URIs and all these URIs share the following leading string:

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

remember

by convention:

- this URI prefix string is often associated with namespace prefix `rdfs`:

RDFS: a language one can use to create ontologies

often used *terms* in **rdfs: vocabulary** are listed here:

terms used for defining classes:
`rdfs:Resource`, `rdfs:Class`, `rdfs:Literal`, `rdfs:Datatype`

terms used for defining properties:
`rdfs:range`, `rdfs:domain`, `rdfs:subClassOf`,
`rdfs:subPropertyOf`, `rdfs:label`, `rdfs:comment`

utilities:
`rdfs:seeAlso`, `rdfs:isDefinedBy`

so, `rdfs:`*`name`* will be used to indicate a term from the RDFS vocabulary

RDFS: a language one can use to create ontologies

to define a class:

```
<?xml version="1.0"?>
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:myCamera="http://www.example.com/camera#">

<rdf:Description
     rdf:about="http://www.example.com/camera#Camera">
   <rdf:type
    rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

</rdf:RDF>
```

- here we declare: this resource, `http://www.example.com/camera#Camera`, is a class
- this is an ontology that contains only one class and nothing else

RDFS: a language one can use to create ontologies

a short form we can use to define a class:

```
<?xml version="1.0"?>
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:myCamera="http://www.example.com/camera#">

 <rdfs:Class rdf:about="http://www.example.com/camera#Camera" />

</rdf:RDF>
```

this is the form that is often used

RDFS: a language one can use to create ontologies

to define more classes, simply add them:

```
<?xml version="1.0"?>
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:myCamera="http://www.example.com/camera#">

 <rdfs:Class rdf:about="http://www.example.com/camera#Camera" />

 <rdfs:Class rdf:about="http://www.example.com/camera#Lens" />

 <rdfs:Class rdf:about="http://www.example.com/camera#Body" />

 <rdfs:Class rdf:about="http://www.example.com/camera#ValueRange" />

</rdf:RDF>
```

RDFS: a language one can use to create ontologies

sub-classes can also be defined:

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:myCamera="http://www.example.com/camera#"
    xml:base="http://www.example.com/camera#">

<rdfs:Class rdf:about="http://www.example.com/camera#Digital">
    <rdfs:subClassOf rdf:resource="#Camera"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.example.com/camera#Film">
   <rdfs:subClassOf rdf:resource="#Camera"/>
</rdfs:Class>
```

`xml:base` together with `rdf:resource`, specifies the full URI:

`http://www.example.com/camera#Camera`

RDFS: a language one can use to create ontologies

similarly, we can define more sub-classes:

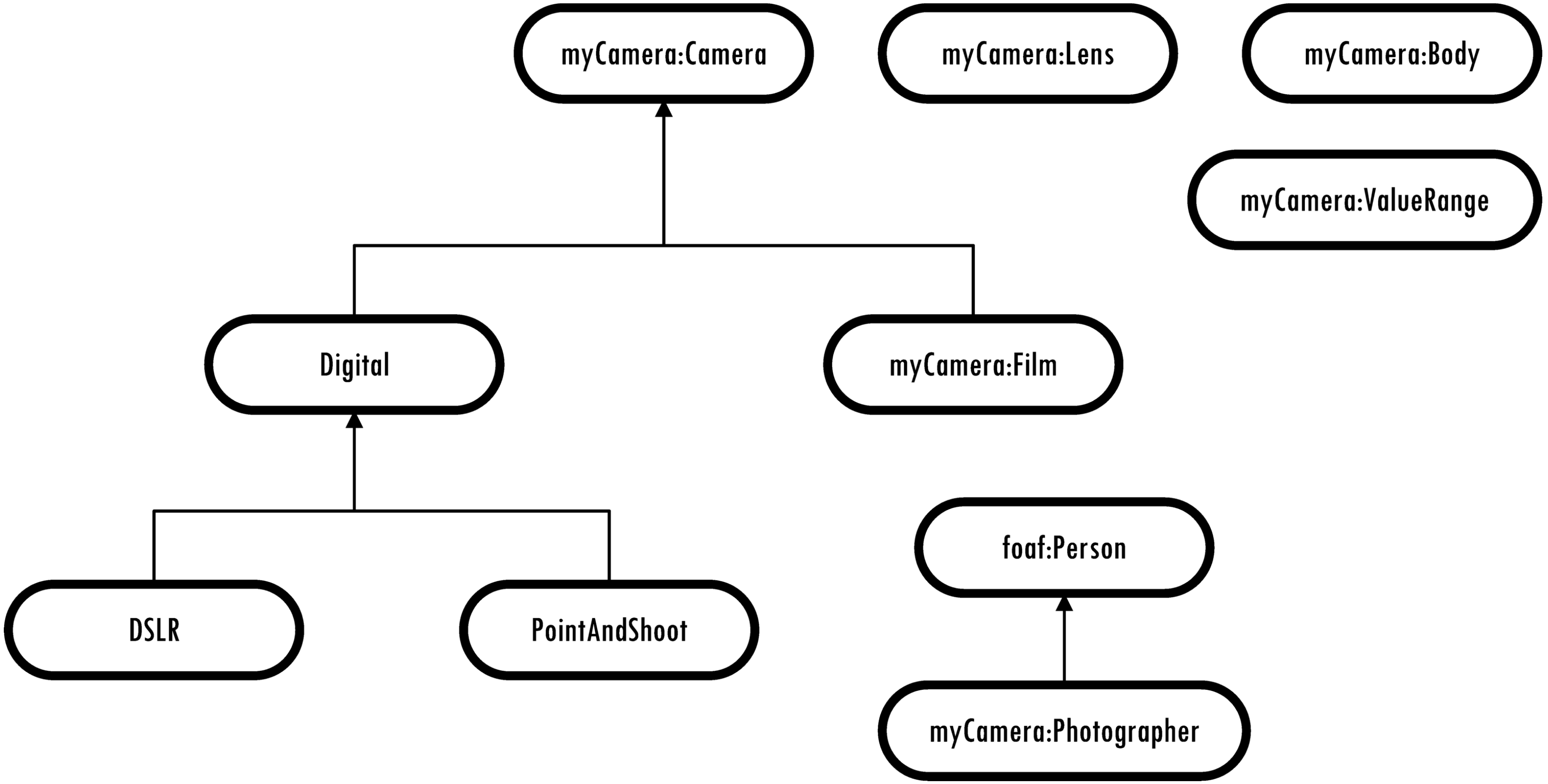
```
<rdfs:Class rdf:about="http://www.liyangyu.com/camera#DSLR">
   <rdfs:subClassOf rdf:resource="#Digital"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.liyangyu.com/camera#PointAndShoot">
   <rdfs:subClassOf rdf:resource="#Digital"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://www.liyangyu.com/camera#Photographer">
   <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
</rdfs:Class>
```

RDFS: a language one can use to create ontologies

the class definitions so far have defined the following class structure:



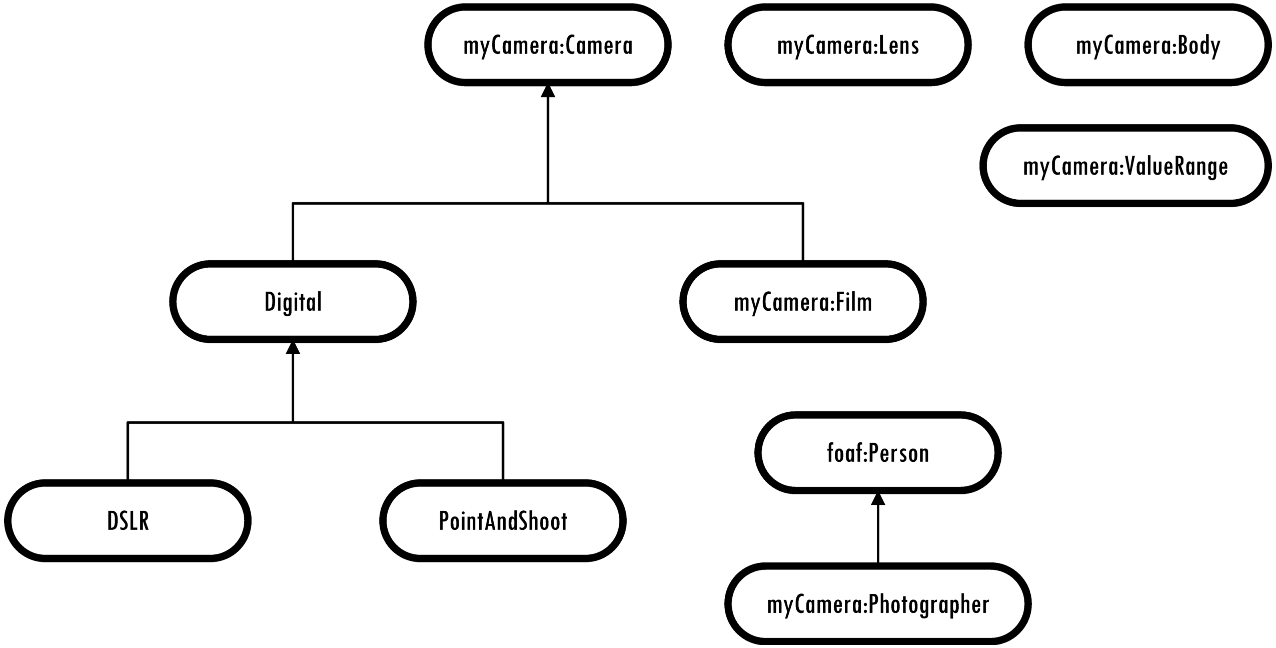
RDFS: a language one can use to create ontologies

- one can use `rdfs:subClassOf` property multiple times when defining a class
- when doing so, all the base classes introduced by `rdfs:subClassOf` will be `AND`ed together to create the new class

```
<rdfs:Class
  rdf:about="http://www.example.com/camera#Photojournalist">
   <rdfs:subClassOf rdf:resource="#Photographer"/>
   <rdfs:subClassOf rdf:resource="#Journalist"/>
</rdfs:Class>
```

this says: class `Photojournalist` is a sub-class of *both* `Photographer` class and `Journalist` class, thus, any instance of `Photojournalist` is an instance of `Photographer` and `Journalist` at the same time

RDFS: a language one can use to create ontologies



- except for the base-class and sub-class relationship, there seems to be no other bounds among these classes
- in real life, the relationships among these classes do exist
- the bounds, or the relationships among these classes, will be expressed by properties

RDFS: a language one can use to create ontologies

express relationship "a DSLR can be owned by a photographer" by defining property `owned_by`:

```
<rdf:Property
      rdf:about="http://www.example.com/camera#owned_by">
   <rdfs:domain rdf:resource="#DSLR"/>
   <rdfs:range rdf:resource="#Photographer"/>
</rdf:Property>
```

we define a property called `owned_by`. It can only be used to describe the characteristics of class **`DSLR`**, and its possible values can only be instances of class **`Photographer`**.

RDFS: a language one can use to create ontologies

- `rdfs:domain` and `rdfs:range` are used to express the relationship between two classes
- but they are not a must when defining properties

```
<rdf:Property
  rdf:about="http://www.example.com/camera#owned_by">
  <rdfs:domain rdf:resource="#DSLR"/>
</rdf:Property>
```

RDFS: a language one can use to create ontologies

- notice the separation of class definitions and property definitions in an ontology
- those who are used to the object-oriented world might find this fact uncomfortably strange
- in OO world, we may define a class called `DigitalCamera`, and we will then encapsulate several properties to describe a digital camera
- these properties will be defined at the same time when we define the class, and they are defined in the class scope as its member variables
- normally, these properties are not directly visible to the outside world

RDFS: a language one can use to create ontologies

- for ontology, it is quite a different story: we define a class, and very often we also indicate its relationships to other classes
- however, we never declare its member variables, i.e., the properties it may have
- so, a class is just an entity who may have relationships to other entities, what are inside this entity, i.e., its member variables/properties, are simply unknown
- the truth is: we declare properties separately and associate the properties with classes if we wish to do so, properties are never owned by any class, they are never local to any class either
- if we do not associate a given property to any class, this property is simply independent, and it can be used to describe any class

RDFS: a language one can use to create ontologies

Rule #3:
I can talk about any resource at my will, and if I chose to use an existing URI to identify the resource I am talking about, then the following is true:

- *the resource I am talking about, and the resource already identified by this existing URI are the same thing or concept;*
- *everything I have said about this resource is additional knowledge about that resource.*

- the separation of class-property definition in ontology is an implementation of the above rule
- it makes large-scale processing of distributed information easy and manageable

RDFS: a language one can use to create ontologies

- RDFS utility terms: `rdfs:seeAlso`, `rdfs:isDefinedBy`, `rdfs:label` and `rdfs:comment`
- the most important one is `rdfs:seeAlso`

```
<rdf:Description
            rdf:about="http://www.liyangyu.com/camera#Nikon_D300">
  <rdf:type rdf:resource="http://www.liyangyu.com/camera#DSLR"/>
  <rdfs:seeAlso rdf:resource="http://dbpedia.org/resource/Nikon_D300"/>
</rdf:Description>
```

- compare `<href... >` to `rdfs:seeAlso`
- `rdfs:seeAlso` is the link among RDF documents on the Semantic Web

RDFS: a language one can use to create ontologies

so, what is the benefit of having an ontology?

- it provides a common and shared understanding/definition about certain key concepts in the domain
- it offers the terms one can use when creating RDF documents in the domain
- it provides a way to re-use domain knowledge
- it makes the domain assumptions explicit
- it provides a way to encode knowledge and semantics such that machine can understand, and
- it makes automatic large-scale machine processing become possible

RDFS: a language one can use to create ontologies

another benefit is ontology make inferencing/reasoning become possible

- understand a resource’s class type by reading the property’s `rdfs:domain` tag
- understand a resource’s class type by reading the property’s `rdfs:range` tag

```
<rdf:Property rdf:about="http://www.example.com/camera#hasLens">
   <rdfs:domain rdf:resource="#Camera"/>
   <rdfs:range rdf:resource="#Lens"/>
</rdf:Property>
```

this says: when we describe a camera, we can use `hasLens` to describe it – this property can **only** be used on a **`Camera`** instance, and its value **has** to be a **`Len`** instance.

RDFS: a language one can use to create ontologies

another benefit is ontology make inferencing/reasoning become possible

- understand a resource's class type by reading the property's `rdfs:domain` tag
- understand a resource's class type by reading the property's `rdfs:range` tag

```
<rdf:Description
     rdf:about="http://www.example.com/camera#Nikon_D300">
    <myCamera:hasLens rdf:resource=
      "http://dbpedia.org/resource/Nikon_17-35mm_f/2.8D_ED-
      IF_AF-S_Zoom-Nikkor"/>
</rdf:Description>
```

what can be inferred from this?

RDFS: a language one can use to create ontologies

- understand a resource’s class type by reading the property’s `rdfs:domain` tag
- understand a resource’s class type by reading the property’s `rdfs:range` tag

```
<rdf:Description
     rdf:about="http://www.example.com/camera#Nikon_D300">
    <myCamera:hasLens rdf:resource=
      "http://dbpedia.org/resource/Nikon_17-35mm_f/2.8D_ED-
      IF_AF-S_Zoom-Nikkor"/>
</rdf:Description>
```

- `http://www.example.com/camera#Nikon_D300` is an instance of class `myCamera:Camera`
- `http://dbpedia.org/resource/Nikon_17-35mm_f/2.8D_ED-IF_AF-S_Zoom-Nikkor` is an instance of class `myCamera:Lens`

RDFS: a language one can use to create ontologies

understand a resource's super class type by following the class hierarchy described in the ontology

- imagine `myCamera:Camera` and `myCamera:Lens` both have a super class called `myCamera:OpticalInstrument`
- then `http://www.example.com/camera#Nikon_D300` is also an instance of `myCamera:OpticalInstrument`, and
- `http://dbpedia.org/resource/Nikon_17-35mm_f/2.8D_ED-IF_AF-S_Zoom-Nikkor` is also an instance of class `myCamera:OpticalInstrument`

RDFS: a language one can use to create ontologies

- understand more about the resource by using `rdfs:subPropertyOf`

```
<rdf:Property rdf:ID="parent">
   <rdfs:domain rdf:resource="#Person"/>
   <rdfs:range rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property rdf:ID="mother">
   <rdfs:subPropertyOf rdf:resouce="#parent"/>
</rdf:Property>

<Person rdf:ID="Tim">
   <mother>
      <Person rdf:resource="#Mary"/>
   </mother>
</Person>
```

RDFS: a language one can use to create ontologies

- understand more about the resource by using `rdfs:subPropertyOf`

```
<Person rdf:ID="Tim">
   <mother>
      <Person rdf:resource="#Mary"/>
   </mother>
</Person>
```

since `mother` is a sub-property of `parent`, machine can add the following statement automatically:

```
<Person rdf:ID="Tim">
   <parent>
      <Person rdf:resource="#Mary"/>
   </parent>
</Person>
```

RDFS: so, where is the semantics?

so, where is the semantics? the *meaning* of a term is defined by specifying

- what properties can be used to describe it, and
- what kinds of objects can be the values of these properties

RDFS: what is missing?

RDFS can be used to create light-weighted ontologies, it is not rich enough for many real-life situations:

- a person can have at most one SS number
- a person can have exactly 2 arms and 2 legs
- one class is the union of the other two classes
- two classes can be equivalent (DSLR vs. DigitalSLR)
- two classes can be totally disjoint
- many more...