Università di Roma Tor Vergata

University of Rome “Tor Vergata”

RDF

Resource Description Framework

Manuel Fiorelli

fiorelli@info.uniroma2.it

Important dates for RDF

1999 – RDF is adopted as a W3C Recommendation

The Word Wide Web Consortium (W3C) is an international community developing open standards for the Web (called Recommendations)

2004 – RDF 1.0

six documents (Primer, Concepts, Syntax, Semantics, Vocabulary, and Test Cases) jointly replace the original specification of RDF and the RDF Schema (2000 Candidate Recommendation)

2014 – RDF 1.1 is the latest update to RDF

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Characteristics

- Independence
- Shareability
- Scalability

- Everything is a resource
  - properties are resources
  - values can be resources
  - statements can be resources

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

RDF Data Model

Order and multiplicity of the triples are not relevant

An *RDF Graph* is a set of *triples*:

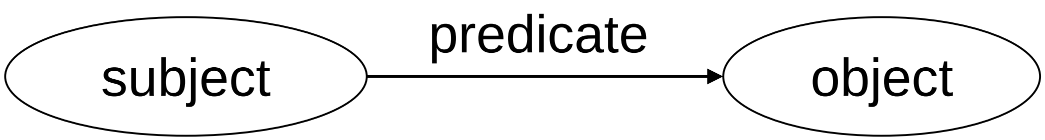
The order of the triple components matters

Triple := (*subject*, *predicate*, *object*)

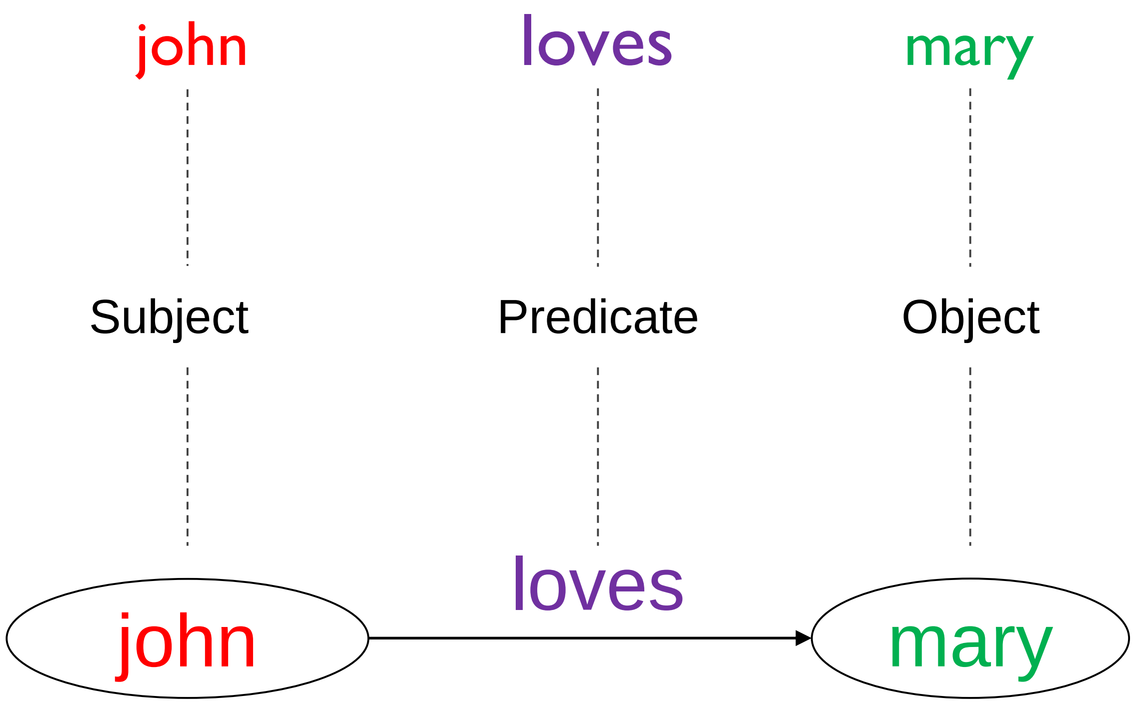
- **Subject** is a resource
- **Predicate** is a property of the subject resource
- **Object** is the value of the property (it can be a resource described through RDF triples)

RDF Data Model

A set of triple can be seen as a direct labeled graph, in which every triple assumes the following shape:



RDF Data Model



RDF Data Model

The components of a triple can be:

- An **IRI**[1] (Internationalized Resource Identifier) is a UNICODE string conforming to RFC 3987.
  - Examples: http://art.uniroma2.it/fiorelli · mailto:fiorelli@info.uniroma2.it · urn:lex:it:stato:legge:2003-09-21;456
  - Extends URI (Uniform Resource Identifier) with support for other scripts in addition to the latin one
- A **literal**[2] is composed of:
  - *lexical form* (a UNICODE string),
  - a *datatype IRI*,
  - and if only if the datatype is http://www.w3.org/1999/02/22-rdf-syntax-ns#langString a non empty *language tag* (conforming to BCP 47)
- A **blank node** (bnode)

[1] in RDF 1.0 there were RDF URI References
[2] in RDF 1.0 there were plain literals without datatype and, optionally, with language tag

RDF Data Model

There are *some constraints* on the use of IRIs, blank nodes and literals in a triple:

- The **subject** can be an IRI or blank node
- The **predicate** can be an IRI
- The **object** can be an IRI, blank node or literal

RDF Data Model

RDF Data Model: Why IRIs?

- *Extensible vocabulary*
- *Global scope* (any appearance of an IRI must denote the same resource; otherwise, we say there is an IRI collision)
  - Anyone can therefore mention resources defined elsewhere
- Ownership rules determine an *IRI owner*, who can establish its referent:
  - By defining a specification for the IRI: it can be an RDF document!
  - If the IRI is dereferenceable (eg: http:// or https://), it is possible to communicate its specification by making the IRI dereference to that specification (à la linked data)

RDF: Dataset

- Version 1.1 introduced the notion of *dataset* into the RDF specification (this notion already existed in the SPARQL specification).
- A dataset consists of:
  - One default graph (it does not have a name)
  - Zero or more named graphs. Each named graph is a pair formed by an IRI or blank node[1] (the graph name) and an RDF graph

[1] actually SPARQL disallows the use of blank nodes as graph names.

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

RDF Data Model

Any IRI or literal *denotes* something (a resource):

- The *referent* of an IRI
- The *literal value* of a literal

Blank nodes are discussed later…

A triple represents a *statement*

- A given relation (denoted by the predicate) holds between the resources denoted by the subject and object

The meaning of an RDF graph is thus the logical conjunction of the statements associated with each triple

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

RDF: expressive power

- RDF corresponds to the existential-conjunctive (EC) subset of first order logics
  - it does not include negation (NOT)
  - it does not include disjunction (OR)

- Unusual for a resctrition on first-order logics, RDF allows statements about relations:

  es:

```
type(loves, social_relationship)
loves(Tom, Mary)
```

RDF: blank nodes

- A triple can include a blank node (bnode), which behaves as an existentially quantified variable:

```
love(?x, Mary)
```

  tells us that "somebody loves Mary", or, more precisely, something loves Mary

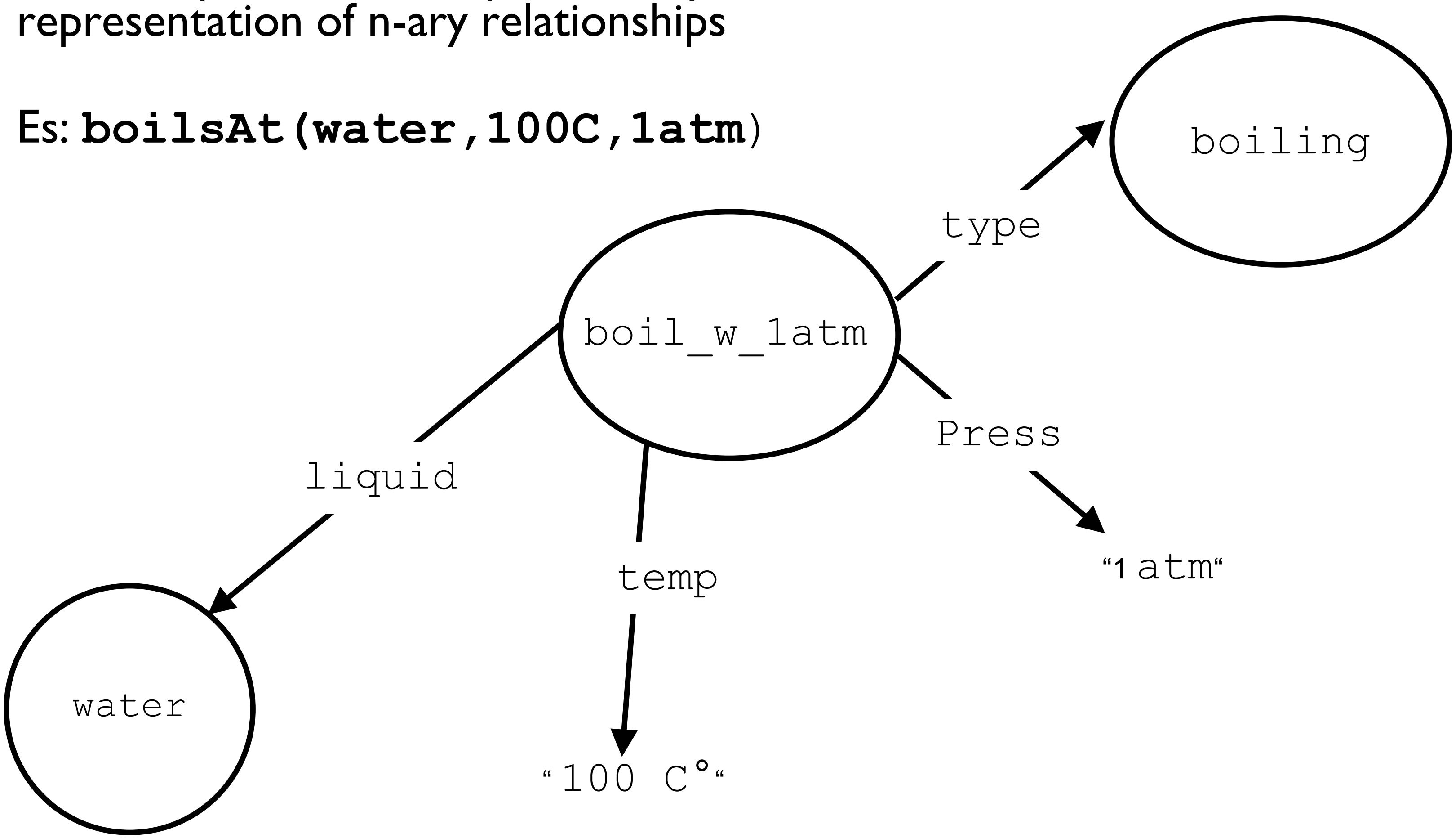
- The combination of different statements, through variable unification, allows us to express complex knowledge, still not fully grounded:

```
gender(?x, male) AND loves(?x, Mary)
```

  tells us that "Mary is loved by a male"

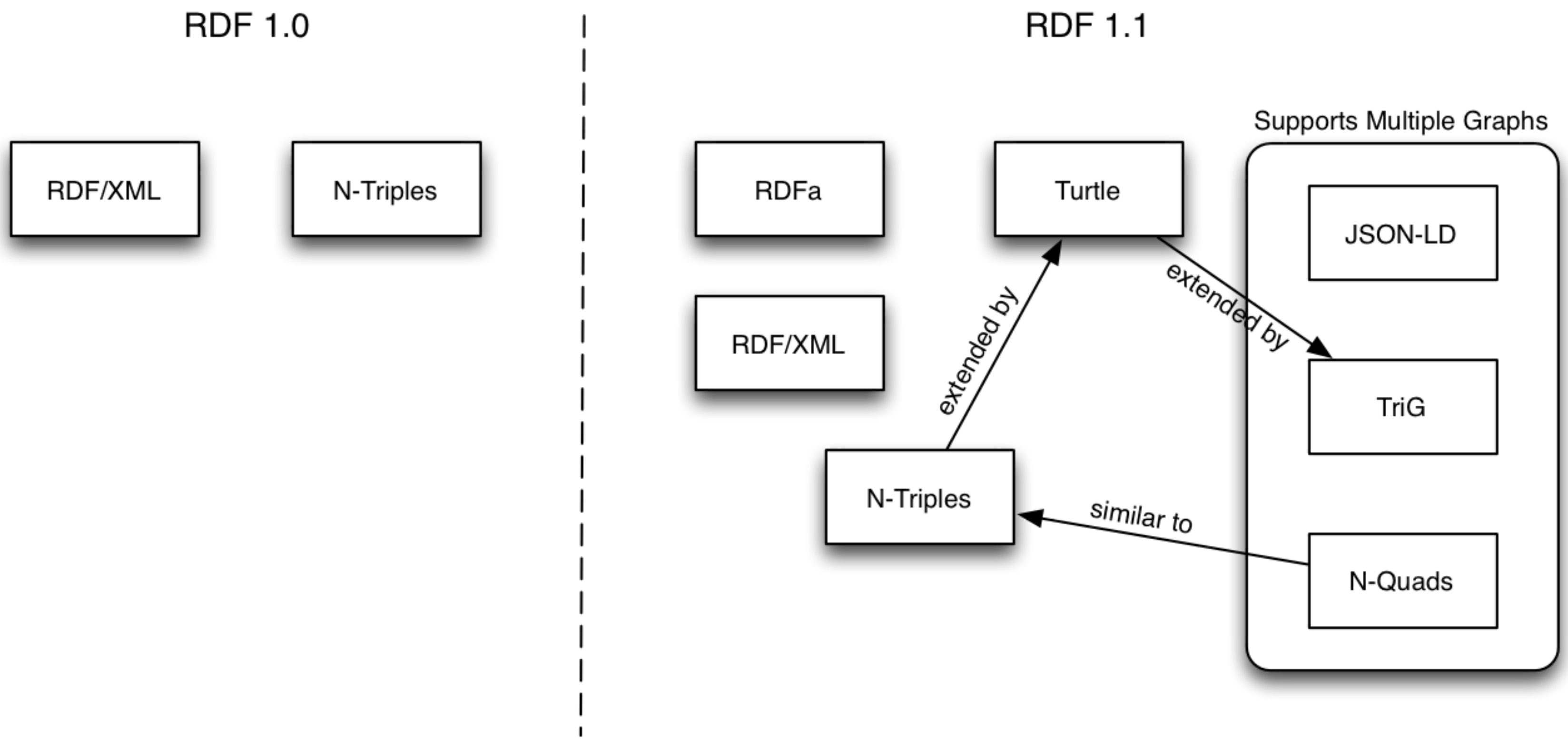
n-ary relationships

- The sole presence of explicit binary relations does not forbid the representation of n-ary relationships
- Es: **`boilsAt(water,100C,1atm`**)



RDF: Concrete Syntaxes

RDF/XML is no longer the only recommended format. A host of new serialization formats have been standardized.

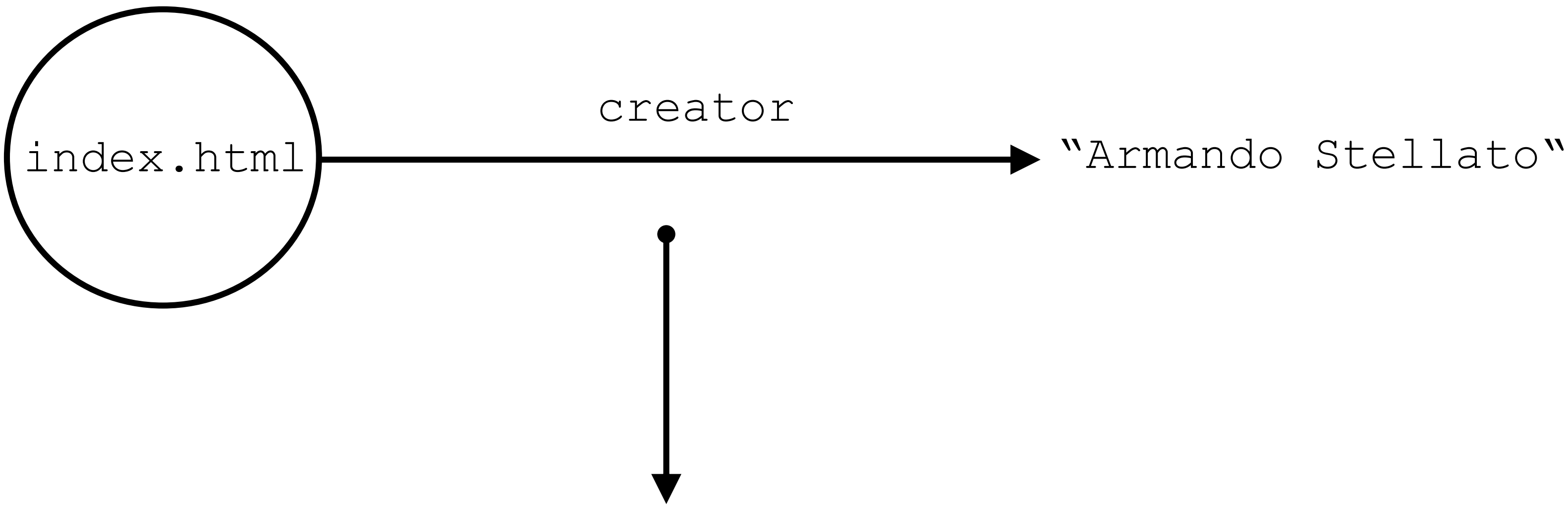


Source: https://www.w3.org/TR/rdf11-new/#section-serializations

RDF: Concrete Syntaxes

- There are diverse syntaxes for representing RDF. Here a few of them:
  - RDF/XML
    - It is a fully XML compliant (and based on) notation for RDF. As for N3, the triple structure can be hidden behind more complex syntactic structures. It has been originally one of the most diffused serializations for RDF
  - N-Triples
    - The closer notation to RDF abstract form, it consists in a series of triples: subject-predicate-object
      notation
  - Notation 3 (N3)
    - it includes several syntactic abbreviations that facilitate reading, hiding the strict triple structure characterizing RDF. This characterisitc, together with simple syntactic structures (differently from the heavy XML syntax) make N3 the most compact serialization. As for XML, N3 uses namespaces. Notation 3 has an expressive power which goes beyond RDF.
  - Turtle
    - thought to be a subset of N3 (and a superset of N-Triples) which can only serialize valid RDF graphs. It is today considered a much better alternative to RDF/XML

RDF/XML : a small example

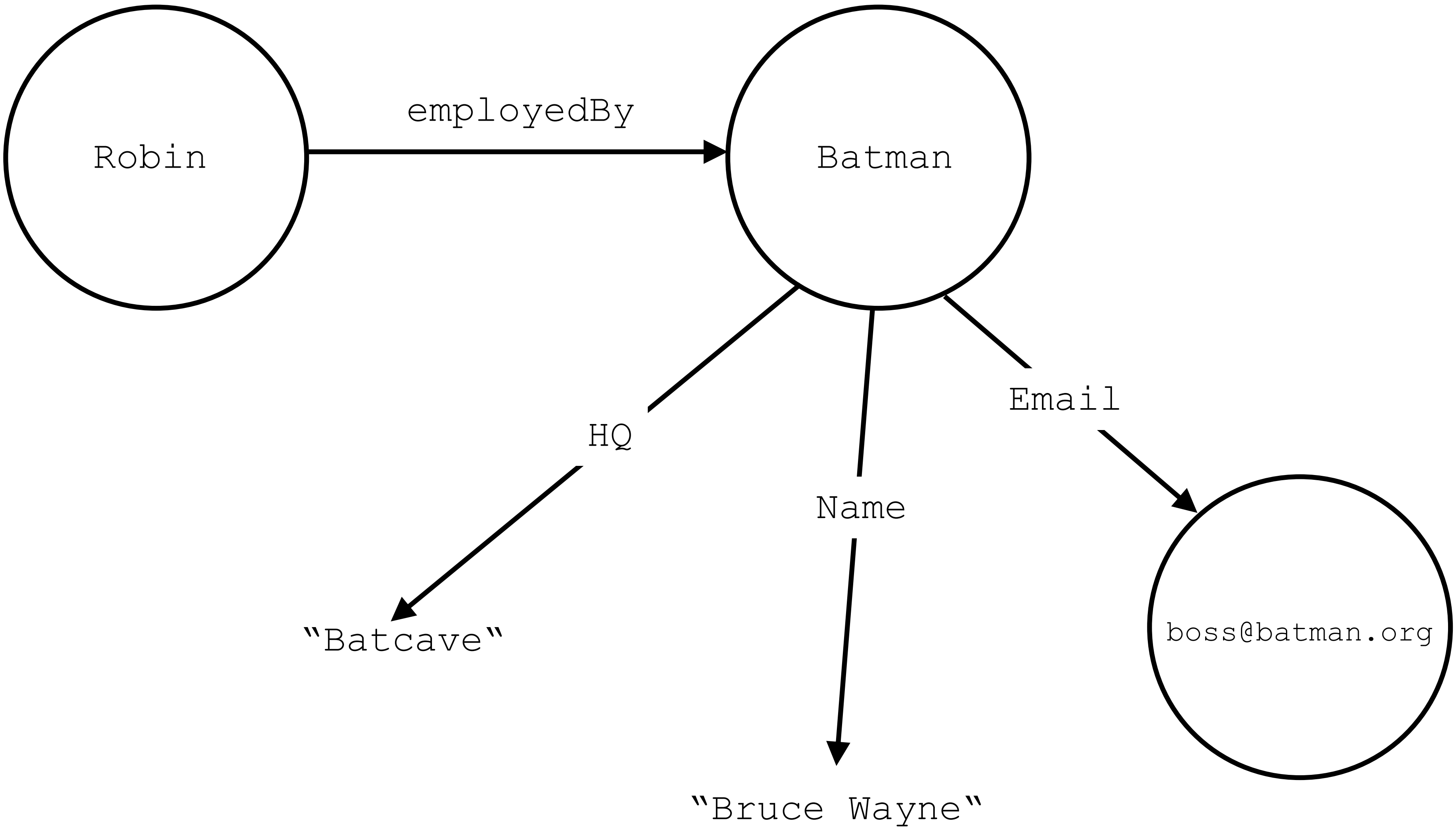



```
<rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc = "http://purl.org/dc/elements/1.1/">

  <rdf:Description rdf:about
                       = "http://art.uniroma2.it/stellato/index.html">
    <dc:creator>Armando Stellato</dc:creator>
  </rdf:Description>

</rdf:RDF>
```

RDF : a slightly more complex example…



RDF : a slightly more complex example…

and the related RDF/XML serialization…

```
<rdf:RDF
       xml:base = “http://www.Batman.org”
       xmlns:rdf = “http://www.w3.org/1999/02/22-rdf-syntax-ns#”
       xmlns:mySchema = “http://www.Batman.org/mySchema/”>

  <rdf:Description rdf:about = “http://www.Batman.org#Robin/”>
    <mySchema:employedBy rdf:resource = “#Batman”/>
  </rdf:Description>

  <rdf:Description rdf:ID = “Batman”>
    <mySchema:HQ>Batcave</mySchema:HQ>
    <mySchema:Name>Bruce Wayne</mySchema:Name>
    <mySchema:Email rdf:resource = “mailto:boss@batman.org“ />
  </rdf:Description>

</rdf:RDF>
```

Turtle

…and its corresponding Turtle serialization

```
@base <http://www.Batman.org> .
@prefix mySchema: <http://www.Batman.org/mySchema/> .

<http://www.Batman.org#Robin> mySchema:employedBy <#Batman> .

<#Batman> mySchema:HQ "Batcave" ;
          mySchema:Name "Bruce Wayne" ;
          mySchema:Email <mailto:boss@batman> .
```

Trig

The following dataset has only one named graph

Curly braces can be omitted for the default graph

```
{
    _:a foaf:name "Bob" .
    _:a foaf:mbox <mailto:bob@oldcorp.example.org> .
    _:a foaf:knows _:b .
}
GRAPH <http://example.org/alice>
{
    _:b foaf:name "Alice" .
    _:b foaf:mbox <mailto:alice@work.example.org> .
}
```

Identical blank node labels in different graph identify the same node

The GRAPH keyword is optional for named graphs

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

XML vs RDF

- RDF criticists and detractors have often sustained how RDF is useless with respect to what XML and XML Schema (and other, most recent, schema languages for XML) already provide

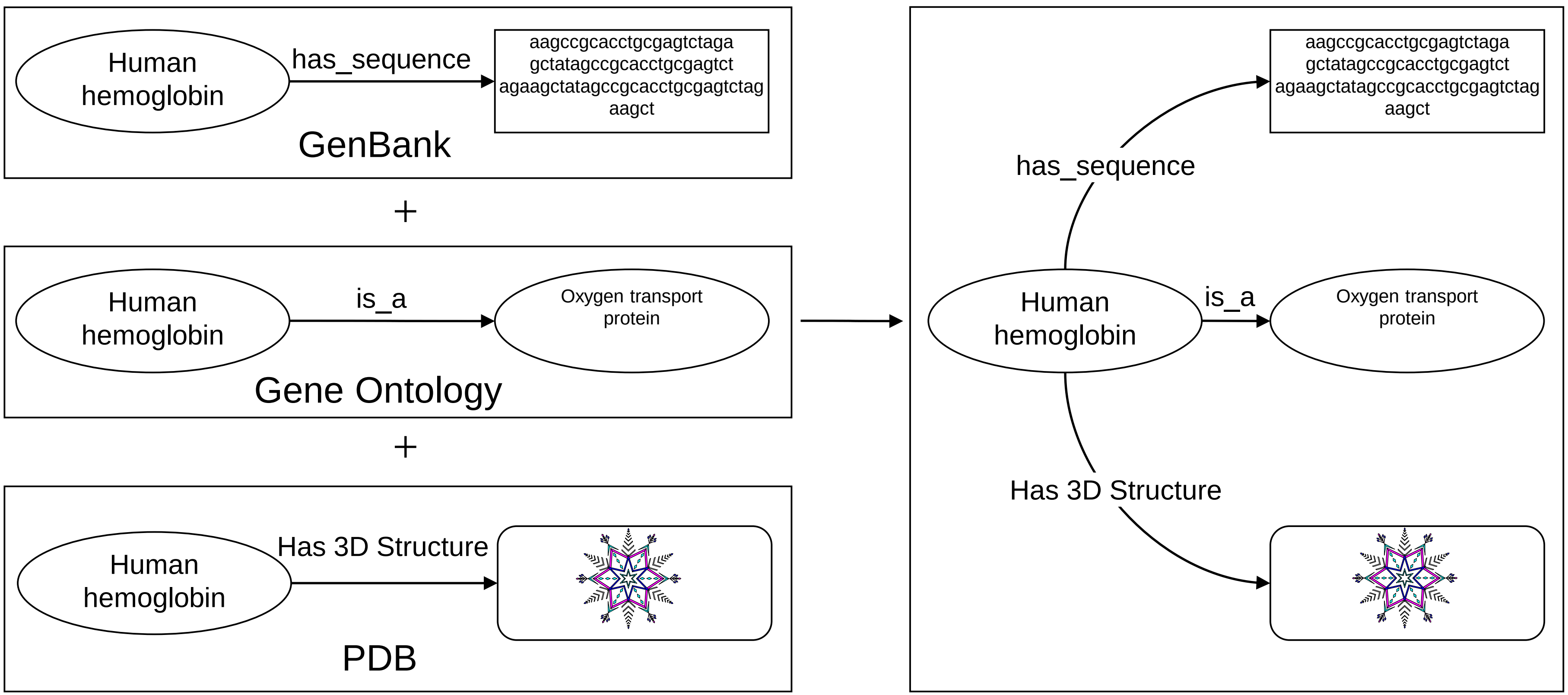
  “RDF is seen by some as an overly complex technology, trying to solve a problem XML and HTTP already solve” (Van Dijck, 2003)

- but RDF adds, <u>through a very simply model</u>, shared semantics for interpreting data

- Pros:
  - no syntactic variance (e.g. attribute or nested element?): the XML serialization of a RDF graph is non-unique, however…who cares about different serializations if all of them can univocally be traced back to the same RDF graph which originated them?
  - Explicit semantics allow for better and more immediate integration of different distributed resources

XML vs RDF (2)

- Example: merge of two RDF fragments



XML vs RDF (3)

- Example: merge of two XML fragments

<sequence id=“abc123”>atagccgtacctgcgagtct </sequence>

Generic Sequence XML

\+

<is-a><object>human-hemoglobin</object><type>oxygen-transfer-protein</type></is-a>

Generic Gene Ontology XML

\+

<structure><protein name=“hh434”/><atom x=“-30” y=“40” elem=“H”/>…</structure>

Generic protein structure xml

→

La integrazione richiede una profonda conoscenza degli schemi sorgenti, al fine di produrre delle complesse trasformazioni XSLT verso uno schema comune.

RDFS: RDF Schema

- RDF (as-is) lacks the possibility to:
  - specify different levels of abstraction
  - organize resources into explicit categories
  - define restrictions on properties
- RDFS extends RDF with a vocabulary for defining schemas, e.g.:
  - Class, Property
  - type, subClassOf, subPropertyOf
  - range, domain
- with such an extension, RDF(S) can be considered to all effects a knowledge representation language

RDFS : Core vocabulary

Core Classes

- rdfs:Resource
- rdfs:Literal
- rdfs:XMLLiteral
- rdfs:Class
- *rdf:Property*
- rdfs:DataType

Core Properties

- *rdf:type*
- rdfs:SubClassOf
- rdfs:SubPropertyOf
- rdfs:Domain
- rdfs:Range
- rdfs:Label
- rdfs:Comment

RDFS : Core vocabulary

- ***rdfs:Resource*** – All the things described through RDF expressions are resources and are considered as *instances* of class *rdfs:Resource*

- ***rdfs:Class*** – represents the generic concept of type or category. Can be used to define any kind of thing, e.g. web pages, persons, documents, document types…

- ***rdf:type*** – This element already exists in the RDF vocabulary but, in RDFS, it binds resources to the categories (classes) they belong to. It is a construct analogous to the *instance-of* of *OO design*

RDFS : Core vocabulary

- ***rdf:Property*** – this resource comes from the RDF vocabulary as well, and represents the subset of all RDF resources that are properties

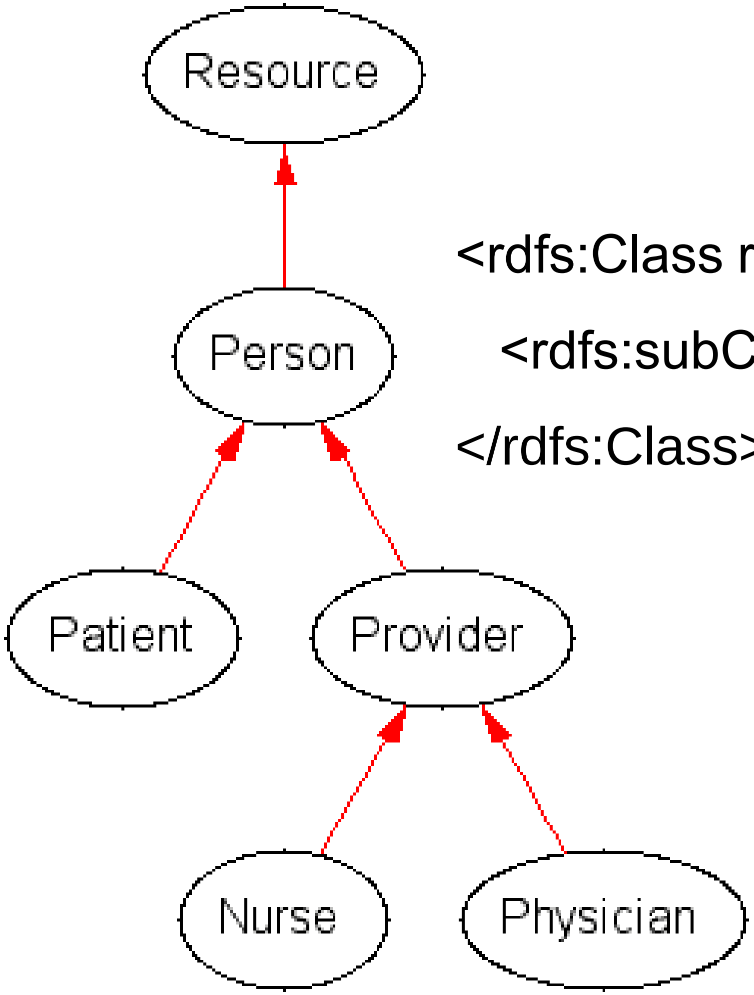
- ***rdfs:subClassOf*** – this property defines a super/subset relation between classes. It is a transitive property (note that the characteristic of being transitive comes from the semantics of the RDFS language and is hard-wired into the subClassOf property; it cannot be assigned to any property)

- ***rdfs:subPropertyOf*** – This property is used to specify that a property is a specialization of another property

RDFS : Core vocabulary

- ***rdfs:range*** – states that all values for a given property belong to a given class
- ***rdfs:domain*** – states that all resources characterized by a given property belong to a given class
- ***Annotation properties*** – they do not play any role in the semantics of the language but provide a useful way for commenting/documenting a repository and its data
  - *rdfs:comment*: the most general way to comment something. It usually provides a natural language definition of the resource it describes
  - *rdfs:label*: provides terms for describing a resource.
  - *rdfs:seeAlso*: it holds a pointer to another resource containing further information about the resource described by this property
  - *rdfs:isDefinedBy*: is a subproperty of *rdfs:seeAlso and* points to the resource that describe the subject resource,

RDFS: Schema Example

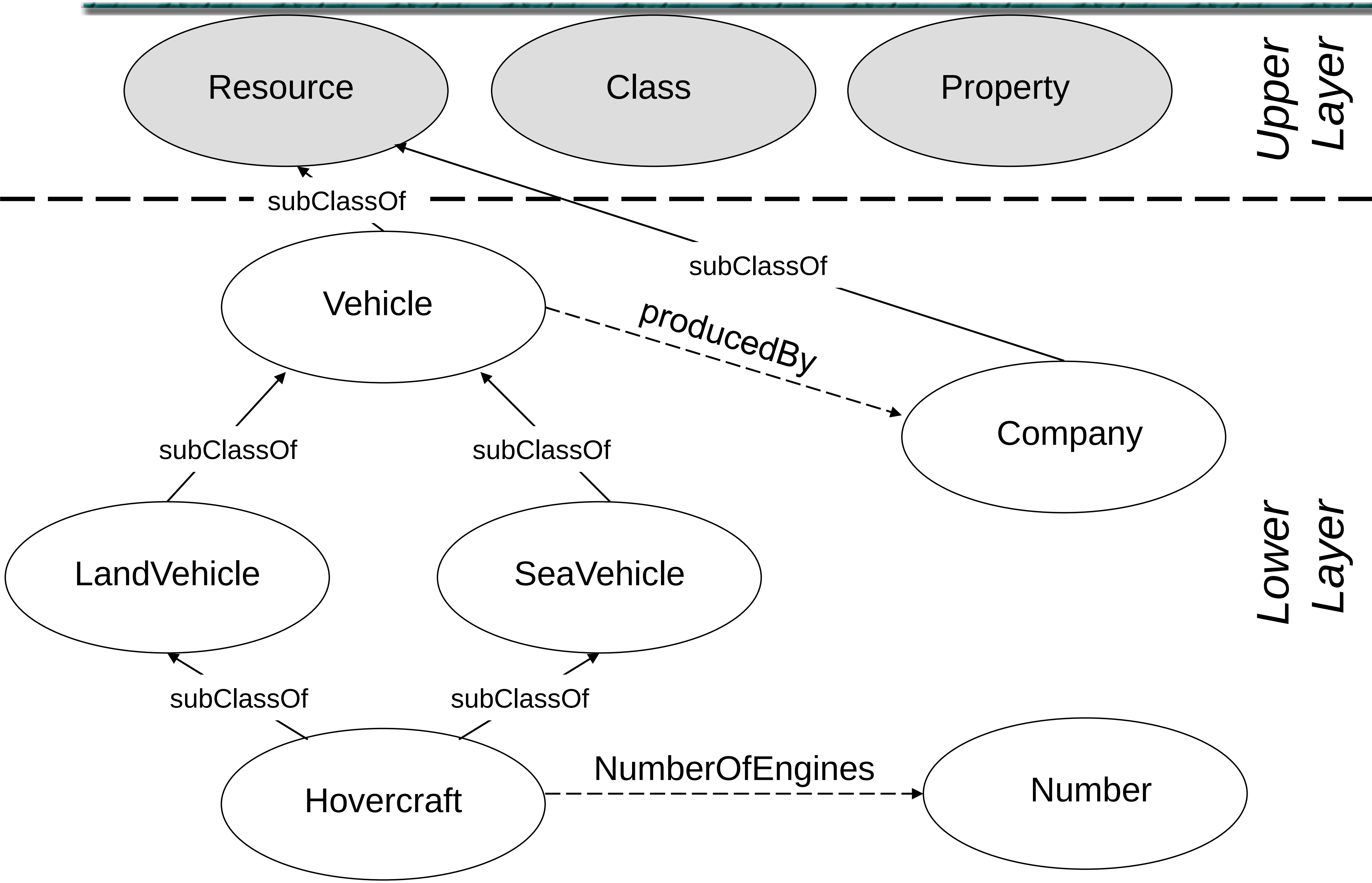


<rdfs:Class rdf:ID="Provider">

<rdfs:subClassOf rdf:resource="#Person"/>

</rdfs:Class>

RDFS : Schema Example



RDFS : Schema Example

```
<rdf:RDF
        xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"

<rdf:Description rdf:ID="Vehicle">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>

<rdf:Description rdf:ID="LandVehicle">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Vehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="SeaVehicle">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Vehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Hovercraft">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#LandVehicle"/>
 <rdfs:subClassOf rdf:resource="#SeaVehicle"/>
</rdf:Description>
```

RDFS : Schema Example

…continues from the previous page…

```
<rdf:Description rdf:ID="Company">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>

<rdf:Description rdf:ID="producedBy">
  <rdf:type resource=""http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#Vehicle"/>
  <rdfs:range rdf:resource="#Company"/>
  <rdfs:label xml:lang="en">Vehicle Producer</rdfs:label>
</rdf:Description>

<rdf:Description rdf:ID="NumberOfEngines">
  <rdf:type resource=""http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#Hovercraft"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-nx#Number"/>
  <rdfs:comment xml:lang="en">This property states how many engines the
                              hovercraft has</rdfs:comment>
</rdf:Description>

</rdf:RDF>
```

RDFS limitations

- RDFS is too weak for describing resources with enough level of detail
  - No contextualized range/domain contraints
    - It is not possible to constraint the range of `hasChild` to `Person` when it is applied to persons and to `Elephant` when applied to elephants
  - No existentiality/cardinality constraints
    - it is not possible to state that all *instances* of `Person` have a mother that is also a `person`, or that all people have exactly 2 parents
  - No transitive, inverse or symmetric property
    - It is not possible to state that `isPartOf` is a transitive property, that `hasPart` is the inverse of `isPartOf` or that `touches` is symmetric
  - …

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

From RDF to OWL

- Two languages developed by extending (part of) RDFS
  - OIL: developed by group of (largely) European researchers (several from EU OntoKnowledge project)
  - DAML-ONT: developed by group of (largely) US researchers (in DARPA DAML programme)
- Efforts merged to produce DAML+OIL
  - Development was carried out by "Joint EU/US Committee on Agent Markup Languages"
  - Extends ("DL subset" of) RDF
- DAML+OIL submitted to W3C as basis for standardisation
  - Web-Ontology (WebOnt) Working Group formed
  - WebOnt group developed OWL (Web Ontology Language) language based on DAML+OIL
  - OWL language now a W3C Recommendation (latest version is OWL 2)

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

RDFS/OWL: Semantics

- RDFS/OWL semantics, differently from previous approaches in knowledge representation, as frame, based on *constraint checking*, is **inferential**.
- Given a *world theory*, instead of verifying that only objects of our knowledge base (*instance data*) satisfy constraints imposed by that theory, it is possible to add (*infer*) new information (rigorously **monotonically**, that is without retracting previous information) to the theory and/or to the description of the objects so that these are still a model for the *theory*
- RDFS/OWL Semantics: two important characteristics
  - **Open World Assumption** (OWA)
  - **No Unique Name Assumption**

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

CWA and NF

- The CWA (**Closed World Assumption**), characterizing traditional databases, and the NF (**negation-as-failure**), characterizing logic programming languages as Prolog, are deeply connected
- Given the atomic ground formula A, the CWA says that:
  - if a knowledge base KB does not entail *A*, then *A* is false
- Given the atomic ground formula A, the NF says that:
  - if it is not possible to prove A in a knowledge base KB, then *A* is false in that KB

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Non Monotonicity of CWA and NF

- CWA and NF are naturally bound to a non-monotonic perspective on the world
- E.g.: a DB has only one fact: woman(MarilynMonroe)
  and we query the following
  - Query: ?- woman(MarilynMonroe).
    - Answer:yes
  - Query: ?- man(MarilynManson)
    - Answer:no (by using CWA/NF).
  - we then update the DB with man(MarlynManson).
  - Query: ?- man(MarlynManson)
    - Answer:yes

OWL Reasoning

- The OWL interpretation, differently from previous approaches to knowledge representation, such as frame models, based on *constraint checking*, is **inferential**.
- A few examples:

```
eg:Document rdf:type owl:Class;
        rdfs:subClassOf [ a  owl:Restriction;
                             owl:onProperty dc:author;
                             owl:minCardinality 1^^xsd:integer].

eg:myDoc rdf:type eg:Document .
```

The description of myDoc is not incomplete eve if the mincard for author is 1, because the author could be defined "somewhere else in the world" (Open World Assumption)

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

OWL Reasoning

- The OWL interpretation, differently from previous approaches to knowledge representation, such as frame models, based on *constraint checking*, is **inferential**.
- A few examples:

```
eg:Document rdf:type owl:Class;
            rdfs:subClassOf [ a  owl:Restriction;
                                 owl:onProperty eg:copyrightHolder;
                                 owl:maxCardinality 1^^xsd:integer].
eg:myDoc rdf:type eg:Document ; eg:copyrightHolder eg:institute1 ;
eg:copyrightHolder eg:institute2 .
```

The two valus for *eg:copyrightHoder* raise any problem? No, there could exist two names to address the same thing, so we assume that *institute1* and *institute2* denote the same object

OWL Reasoning

- The OWL interpretation, differently from previous approaches to knowledge representation, such as frame models, based on *constraint checking*, is **inferential**.
- A few examples:

```
eg:Document rdf:type owl:Class;
        owl:equivalentClass [a owl:Restriction;
                    owl:onProperty eg:author ;
                    owl:allValuesFrom eg:Person ].

eg:myDoc rdf:type eg:Document ;
eg:author eg:Daffy. eg:Daffy rdf:type eg:Duck.

eg:myDoc2 eg:author eg:Dave . eg:Dave rdf:type eg:Person.
```

Duffy is inferred to be ALSO a Person (due to the assertion of equivalentClass on the restriction allValuesFrom Person)

is myDoc2 a Document? we cannot know, because in the world there could be other authors of this book that are not of type Person

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Useful References

- RDF 1.1 Concepts and Abstract Syntax (http://www.w3.org/TR/rdf11-concepts/)
- RDF 1.1 N-Triples (https://www.w3.org/TR/n-triples/)
- RDF 1.1 Turtle (https://www.w3.org/TR/turtle/)
- RDF 1.1 N-Quads (https://www.w3.org/TR/n-quads/)
- RDF 1.1 TriG (http://www.w3.org/TR/trig/)
- RDF 1.1 XML Syntax (https://www.w3.org/TR/rdf-syntax-grammar/)
- RDF Schema 1.1 (https://www.w3.org/TR/rdf-schema/)
- What's New in RDF 1.1(http://www.w3.org/TR/rdf11-new/)
- OWL 2 Web Ontology Language. Document Overview (Second Edition) (https://www.w3.org/TR/owl2-overview/)

Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli