Producing, Consuming and Preserving Linked Data

Carlo Meghini

Istituto di Scienza e Tecnologie della Informazione
Consiglio Nazionale delle Ricerche – Pisa

ADA 2014
Split, July 3rd, 2014

Part I

The Web & the Semantic Web

The World Wide Web

The web consists of two main ingredients:

- a **knowledge base,** where knowledge is expressed informally (text) or pictorially (images, videos, graphics) and is embedded in structures such as hypertexts (HTML documents)
- a mechanism to access knowledge by GETting the structure that contains it

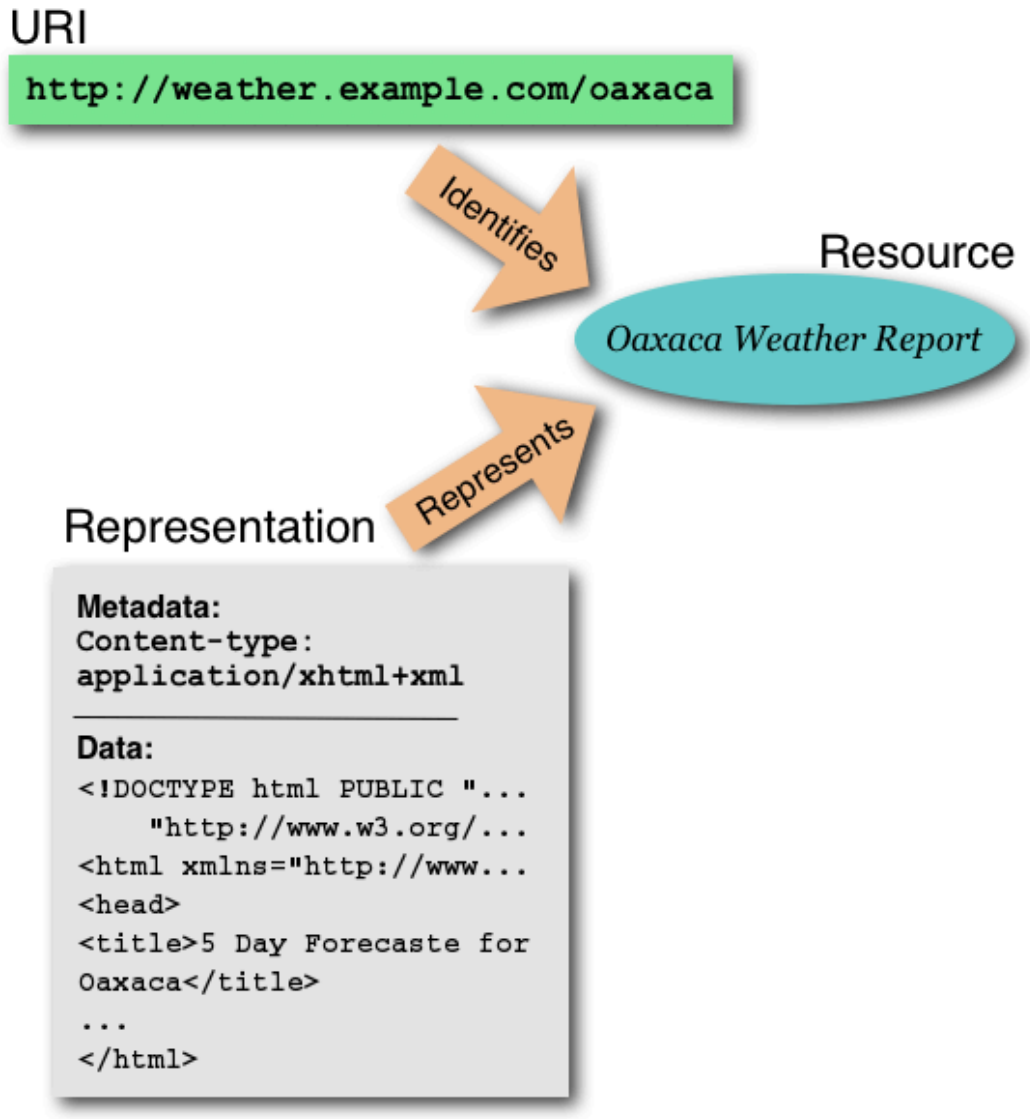
Conceptually, the web is based on a few, simple notions:

- *resource*: everything that has an identity
  - In particular, a web resource is a structure accessible on the web
- *URI*: a string of characters that univocally identifies a resource
- *state*: the way a resource is at a certain time
- *representation*: data that encode the state of a resource
  - a state can be represented by many different representations.

A human can access knowledge using a web browser in few steps:

1. the user gives a URI to the browser
2. the browser asks its server to retrieve a representation of the state of the resource identified by the given URI
3. the web server complies and delivers the representation to the client
4. the client displays the obtained representation to the user

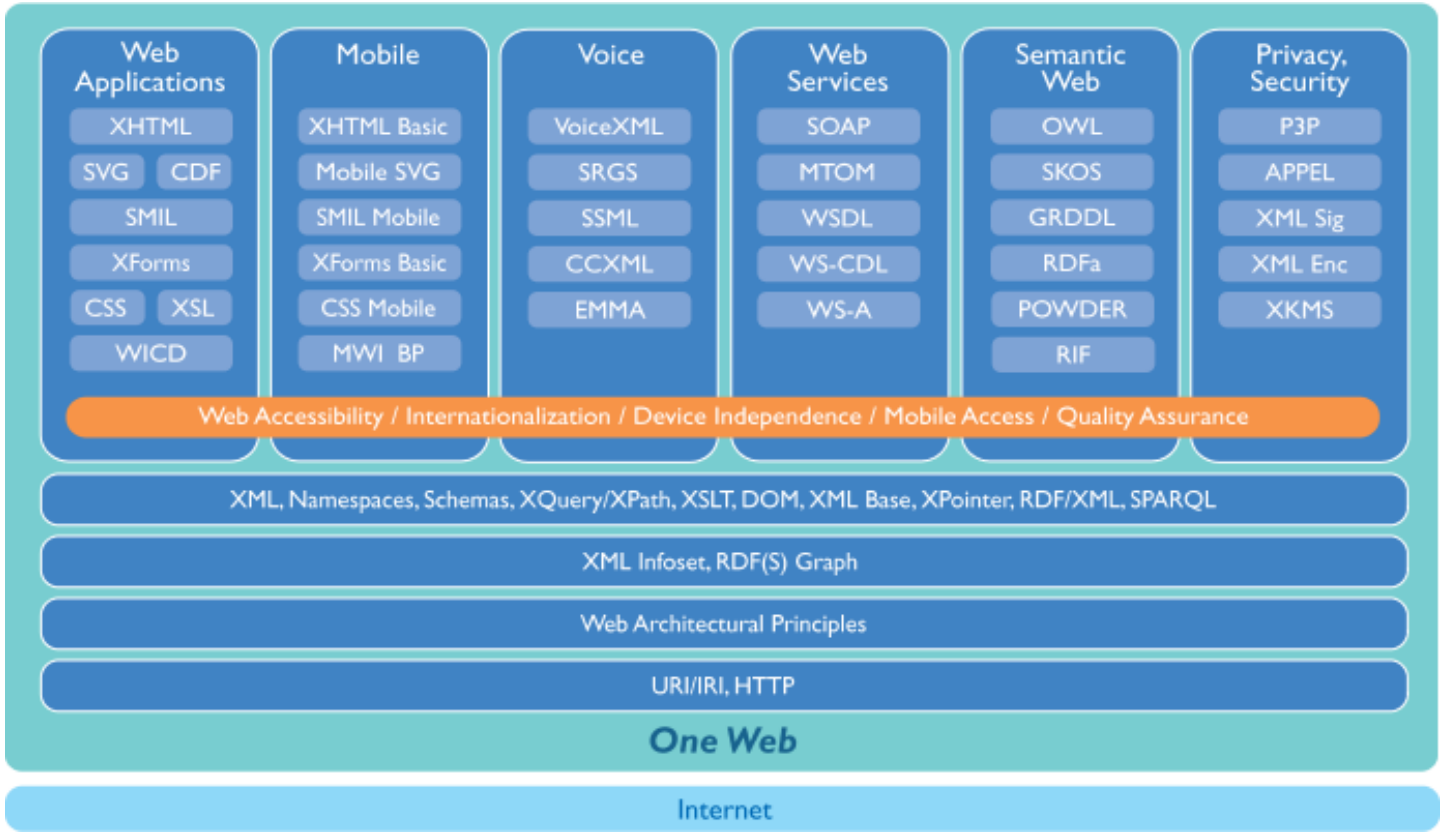
In practice:


URI
http://weather.example.com/oaxaca
Identifies
Resource
Oaxaca Weather Report
Represents
Representation
Metadata:
Content-type:
application/xhtml+xml
Data:
<!DOCTYPE html PUBLIC "...
"http://www.w3.org/...
<html xmlns="http://www...
<head>
<title>5 Day Forecaste for
Oaxaca</title>
...
</html>

The web stack

Based on this simple mechanism, the web has developed considerably in the last 20 years.

Today it is a platform for exchanging services that can be accessed by a variety of devices:



The semantic web

The semantic web is a parallel web, that differs from the original web in the kind of knowledge that is stored and served to the users.

The knowledge found on the semantic web is *formal* knowledge, that is knowledge expressed in a formal language having:

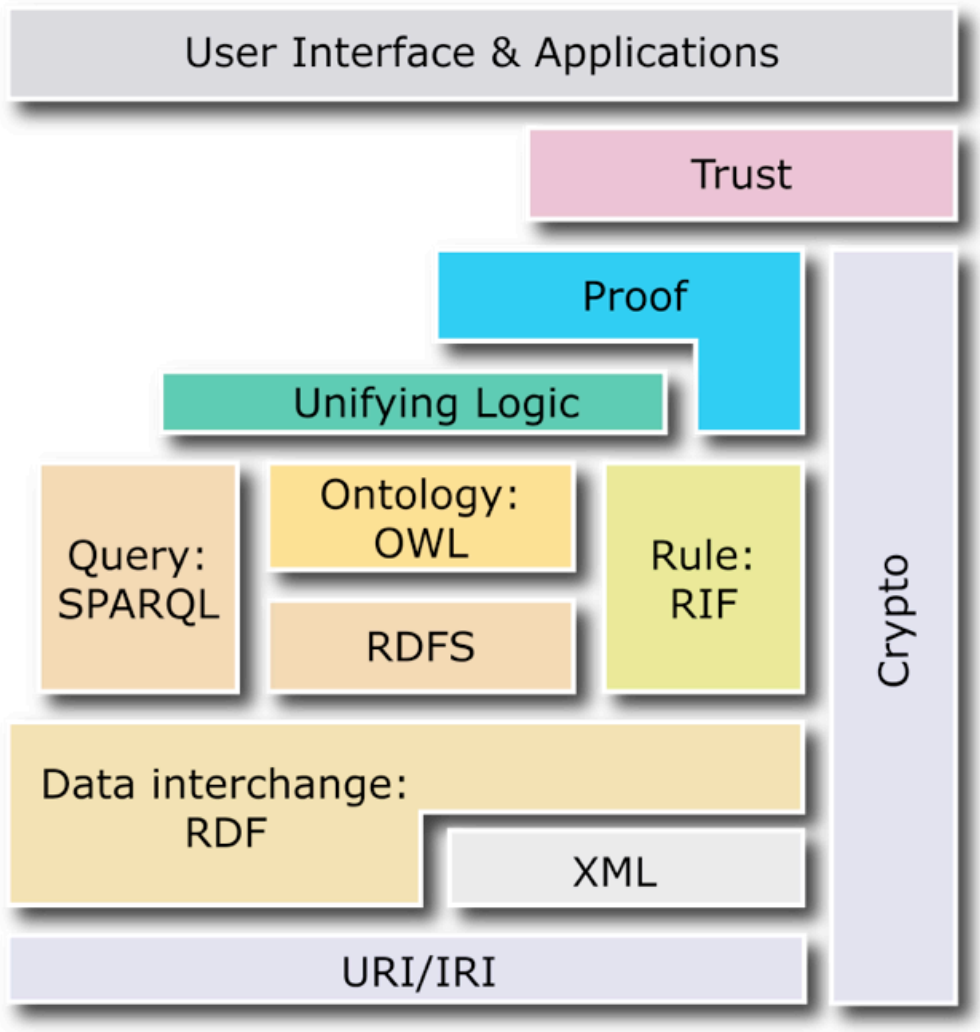
- a machine-readable notation
- a formal syntax that is strongly coupled with the web architecture
- a formal semantics that provides an access mechanism.

The semantic web started as a *vision* by the inventor of the web:

Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. Scientific American Magazine, 2001.

The vision is becoming true via Linked Data.

Semantic web architecture



Part II

URIs

Names on the web

A fundamental problem in building a knowledge base:

- syntax of *singular terms*, that is of the expressions that denote the individuals of the domain of discourse.

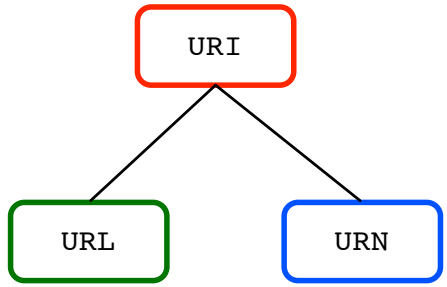
On the web, names are **Uniform Resource Identifiers** (URIs).

Types of URIs

Locators *vs.* names

A URI can be classified as:

- a locator
- a name



1. A Uniform Resource Locator (URL) is a URI that identifies resources by their primary access mechanism (*e.g.*, their network *location*).
2. A Uniform Resource Name (URN) is a URI that is required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable.

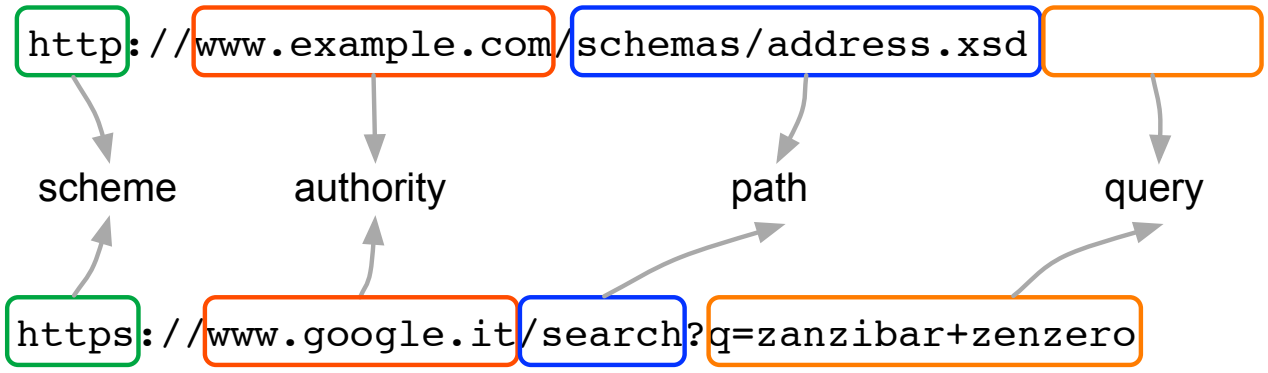
A URN's primary purpose is persistent labeling of a resource with an identifier.

URI Syntactic Components

Absolute URIs

The generic URI syntax consists of a sequence of four main components:

```
scheme ':' ('//' authority)? path? ('?' query)?
```



```
scheme ':' ('//' authority)? path? ('?' query)?
```

scheme Just as there are many different methods of access to resources, there are a variety of schemes for identifying such resources. The schema component defines the semantics for the remainder of the URI string.

authority Many URI schemes include a top hierarchical element for a naming an authority that **governs the namespace of the remainder of the URI**, typically defined by an Internet-based server.
The authority component is optional and, if present, it is preceded by a double slash "//".

Path The path component contains data, specific to the authority (or the scheme if there is no authority), identifying the resource **within the scope of that scheme or authority**.
The path may consist of a sequence of path segments separated by a single slash "/" character.

Query The query component is a string of information to be interpreted by the resource. It can be recognized by its preceding "?"
The query component is optional and in URLs it is typically used to provide parameters, *e.g.* to a web service.

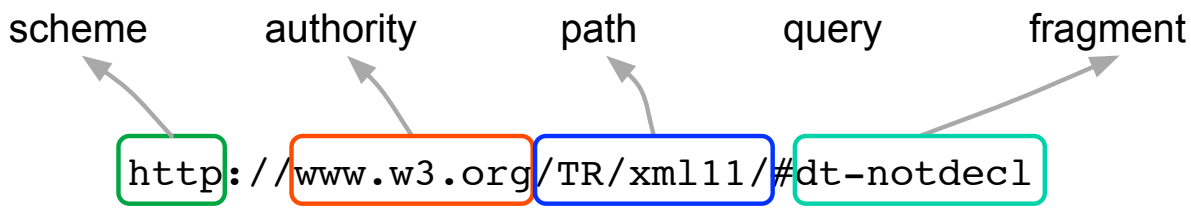
URI References

A URI reference is a URI (absolute or relative) with additional information attached, in the form of a fragment identifier.

```
scheme ':' ('//' authority)? path? ('?' query)? ('#' fragment)?
```

Fragments are often used to address a part of a resource:

- a section of an HTML file (in this case it is a retrievable part), for example `http://www.w3.org/TR/xml11/#dt-notdecl`
- or a term in a namespace, for example `http://www.w3.org/2001/XMLSchema#integer`.



IRIs

International Resource Identifiers (IRIs) are URIs whose syntax has been extended with the many symbols in non-Latin alphabets.

Part III

Resource Description Framework

RDF Syntax and semantics

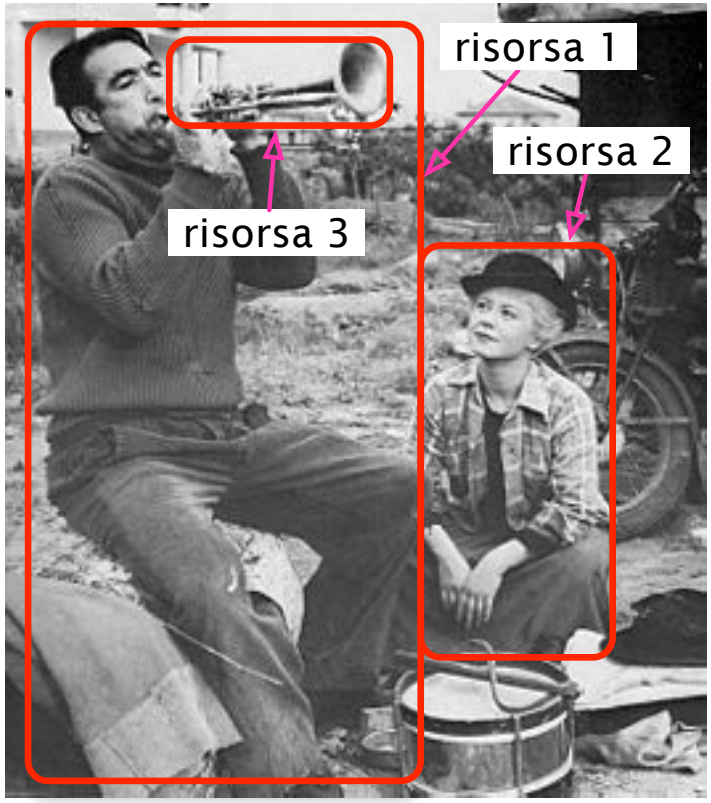
The Resource Description Framework (RDF) is a formal language for representing knowledge recommended by the W3C in 2004.

RDF views the world as consisting of two kinds of resources:

- individuals
- relations

For instance:

Let's see how to represent this world in RDF.
First, we must decide on a conceptualization, that is which indivduals and which relations we are interested in.
Our conceptualization is about people and musical facts.



Based on our conceptualization, we have three individuals:

- a man
- a woman
- a trumpet

and four relations:

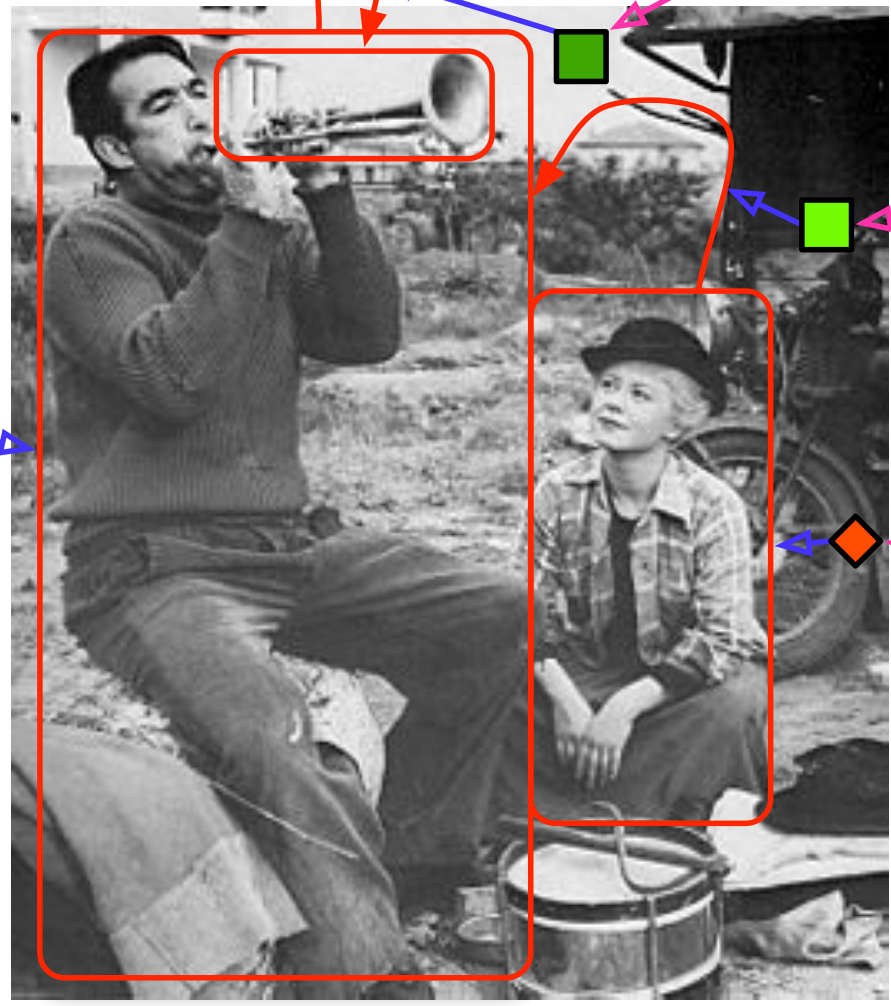
- who plays what, binary
- who listens to whom, binary
- who is a man, unary
- who is a woman, unary

risorsa 6: relazione binaria di suonare

risorsa 7: relazione binaria di ascoltare

risorsa 4: relazione unaria di essere un uomo

risorsa 5: relazione unaria di essere una donna



The next step is ontological commitment: we need to decide the names for denoting the resources in our discourse.

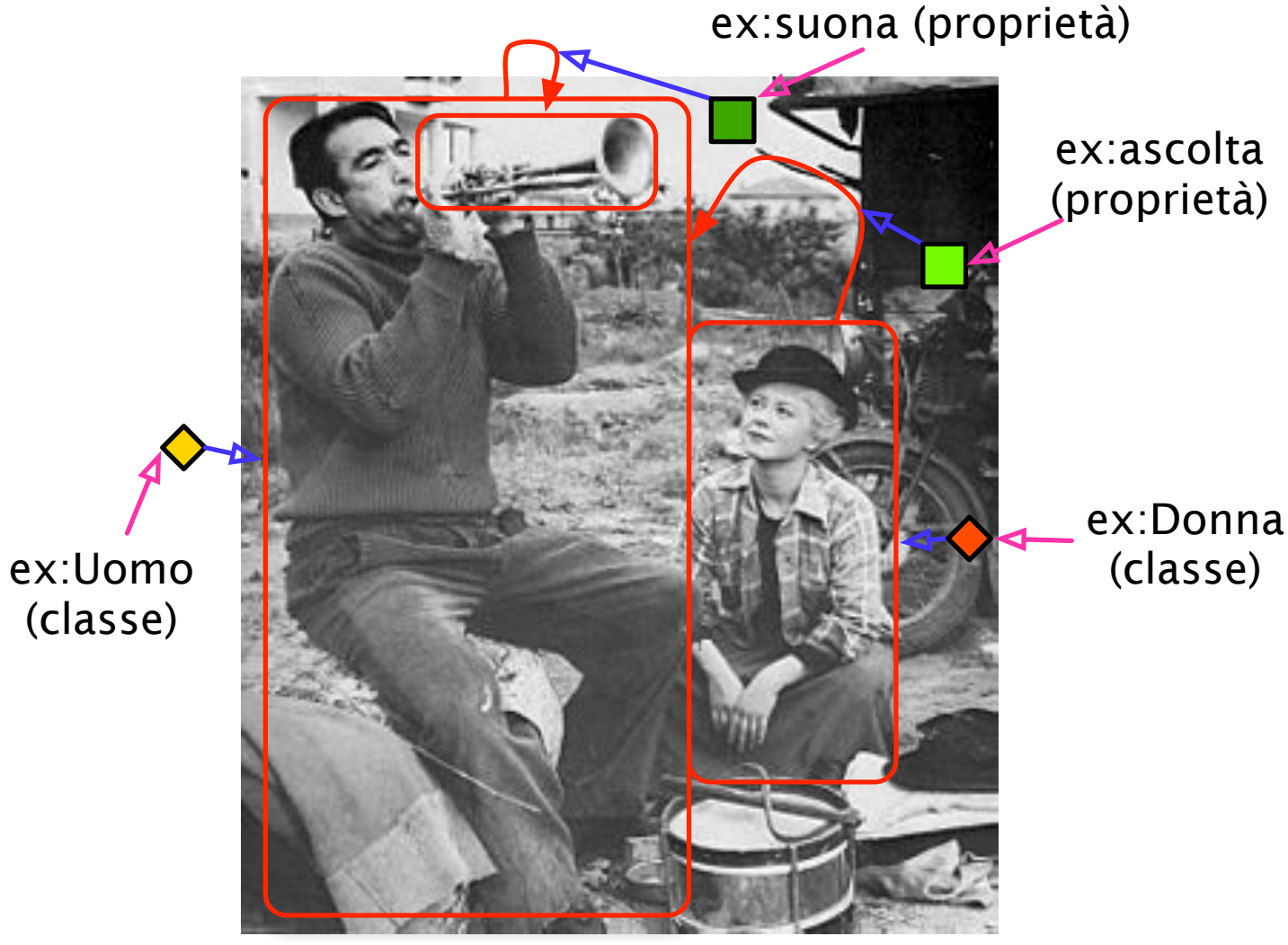
RDF is built on top of the web architecture, so it uses URIs as names.
In order to construct URIs, we need to decide:

- a schema, we choose HTTP
- an authority, we choose `http://www.example.org`, abbreviated as `ex:`
- a path for each individual.

We choose:

- `ex:zampanò`
- `ex:gelsomina`
- `ex:trumpet`

In RDF, unary relations are represented by *classes* and binary relations are represented by *properties.* Classes and properties are themselves resources.



Now we have a name for each resource, but still need to connect individuals and the relations that link them.

In order to establish this connection, RDF offers **statements**, which are triples.

A triple is the minimal unit of discourse in RDF and is composed by:

1. a subject, identifying the resource the statement is about
2. a predicate, a URI that identifies a property of the subject
3. an object, identifying the resource linked to the subject by the property

in this order

We need to make two types of connections:

- to connect resources to the unary relation they belong in
  - the resource `ex:zampanò` belongs to the unary relation `ex:Man`
- pairs of resorces to the binary relation they belong in
  - the pair `ex:zampanò ex:trumpet` belongs in the relation `ex:play`

For the former kind of connection:

resource-id `rdf:type` class

In our example:

```
ex:zampanò rdf:type ex:Man .
ex:gelsomina rdf:type ex:Woman .
```

For the latter kind of connection:

first-resource-id binary-relation second-resource-id

In our example:

```
ex:zampanò ex:play ex:trumpet .
ex:gelsomina ex:listen ex:zampanò .
```

A set of RDF triples is an *RDF graph.*

Our RDF reporesentation of this world is given by the following graph:

```
ex:zampanò rdf:type ex:Man .
ex:gelsomina rdf:type ex:Woman .
ex:zampanò ex:play ex:trumpet .
ex:gelsomina ex:listen ex:zampanò .
```

The functions that associates a URI to the resource denoted by the URI is called *RDF interpretation,* or just interpretation.

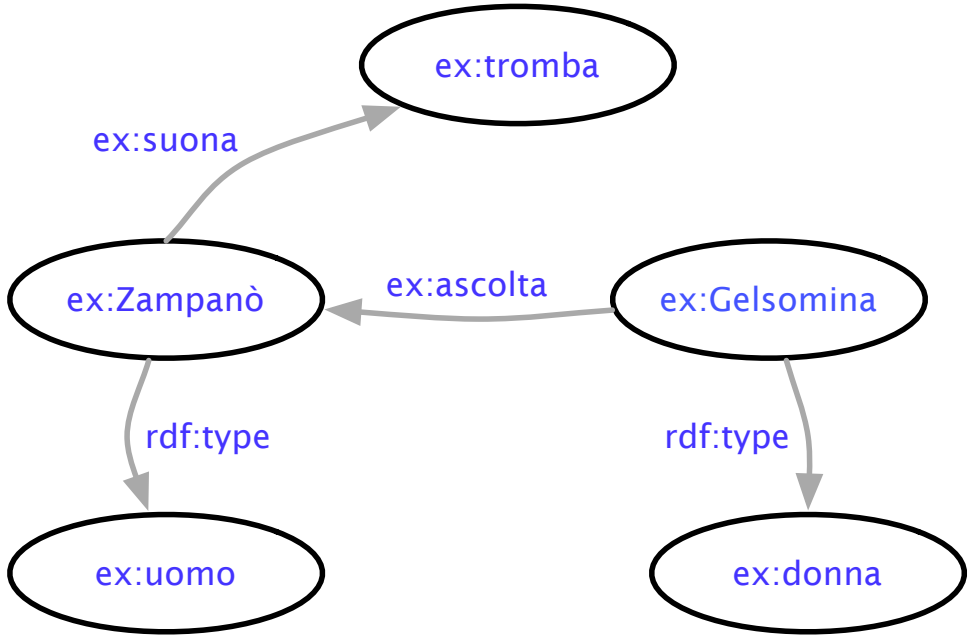
The meaning of a triple is the fact that it describes, based on the interpretation of URIs.

The semantics of a graph is the set of facts that the triples in the graph describe, that is the view of the world according to the producer of the graph.

An RDF triple can be represented graphically as two labelled nodes connected by a labelled arc:

- the node where the arc starts is labelled by the triple subject;
- the node where the arc ends is labelled by the triple object;
- the arc is labelled by the triple predicate.

The graphical version of an RDF graph is a directed, labelled graph, where different arcs can have different labels.




```
ex:zampanò rdf:type ex:Man .
ex:gelsomina rdf:type ex:Woman .
ex:zampanò ex:play ex:trumpet .
ex:gelsomina ex:listen ex:zampanò .
```

Literals

Let us assume we need to express that the name of Gelsomina is the text "Gelsomina".

We already have an identifier for Gelsomina (`ex:gelsomina`). We then need two more names:

- for the property to have a name
- for the text "Gelsomina"

To make knowledge understandable, it is advisable to use names that are as widely known as possible.

One of the most popular ontologies of social relations is FOAF (Friend Of A Friend), http://xmlns.com/foaf/spec/.

- From FOAF we then re-use property `foaf:name` for our naming property

For the text “Gelsomina” we can use a URI that clearly denotes strings, such as for instance:

`http://www.example.org/string/Gelsomina`

so that anyone who can understand the latin alphabet and our naming convention, can understand that “Gelsomina” is the name of our resource.

RDF, instead, uses a syntactic shortcut: *literals,* that is character strings between quotes, who represent themselves.

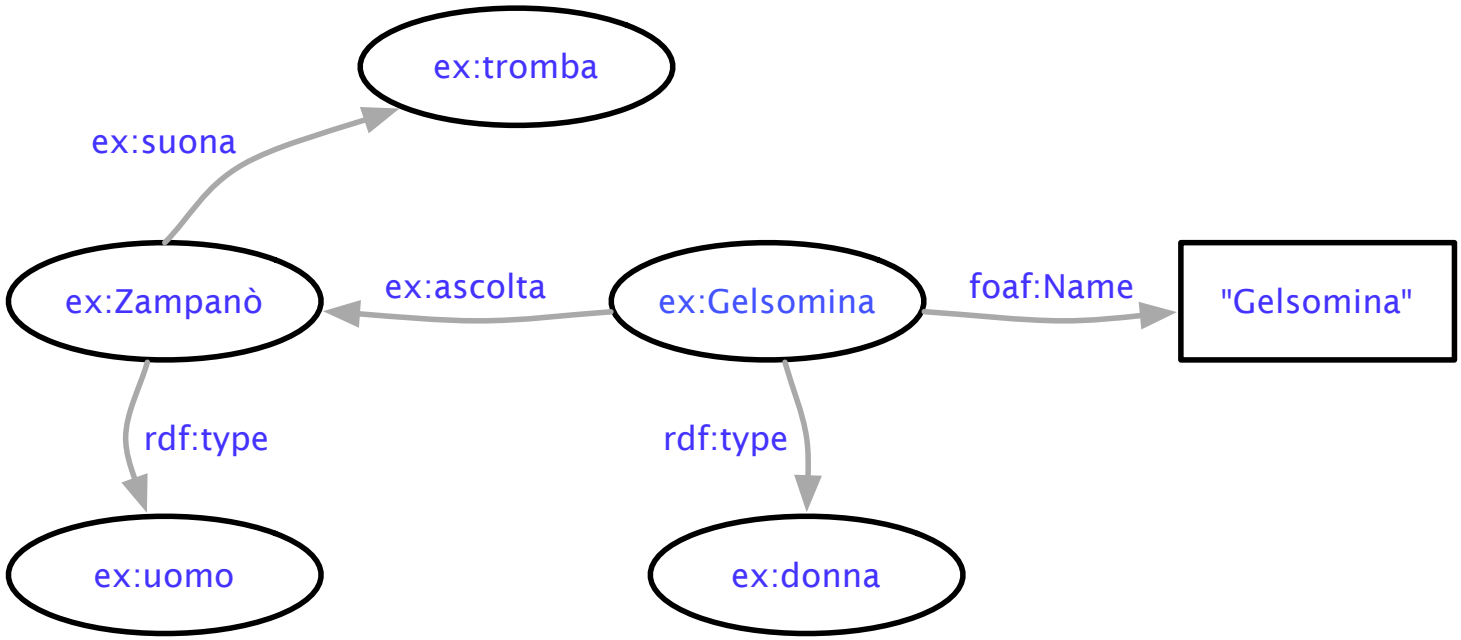
So:

```
ex:gelsomina foaf:name "Gelsomina" .
```

“Gelsomina” is an *untyped* literal, it has no type.

Literals can only be used in triples as objects. Why?

Literals are natural language fragments that live in RDF triples and are very useful to interpret triples.



RDF concrete syntax

So far we have used an abstract or a graphical notation for triples. To make RDF graph readable by machines, we need to encode them in texts.

The encoding of a graph in text is called *serialization.*

Serialization considers the arcs of a graph:

- Every arc is a triple
- The order in which the arcs are considered is not important.

Historical notations:

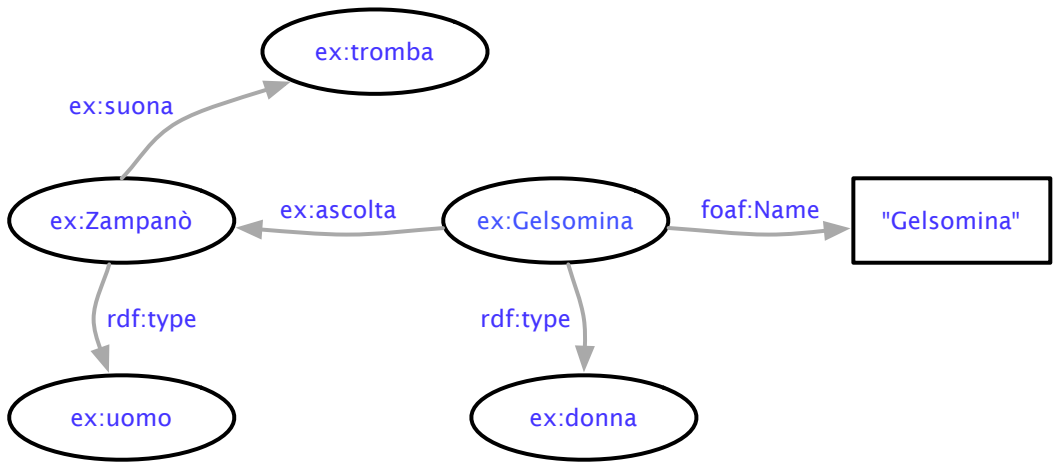
1. Notation 3 (N3) da Berners-Lee, 1998
2. N-Triples da W3C, 2004 (sottoinsieme di N3)
3. Turtle, N-Triples with convenient abbreviations

Turtle

Previous example in Turtle:

```
@prefix ex: <http://www.example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/spec/> .

ex:zampano    rdf:type     ex:Man .
ex:zampano    ex:play     ex:trumpet .
ex:gelsomina  rdf:type     ex:Woman .
ex:gelsomina  ex:listen   ex:zampano .
ex:gelsomina  foaf:name   "Gelsomina" .
```



RDF/XML

Turtle is very easy to read but it is not supported by many software libraries.

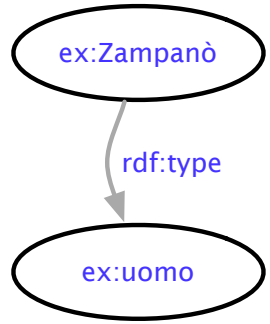
Insrtead, almost all programming languages include software libraries for processing XML documents.

RDF/XML is the XML schema to serialize RDF graph.

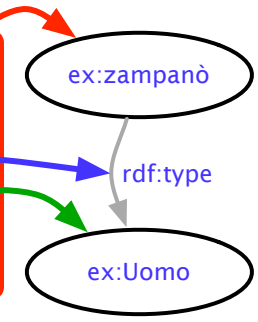
Example:

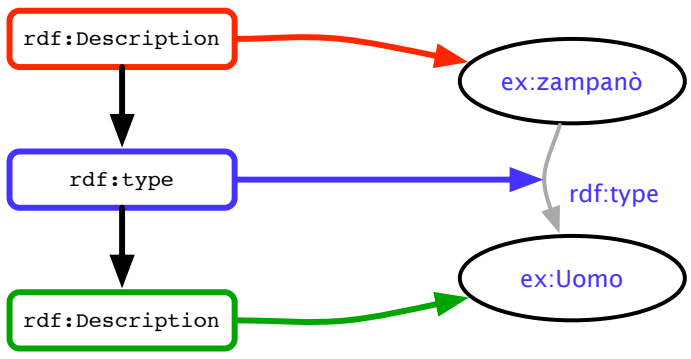
```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:ex="http://www.example.org/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

  <rdf:Description rdf:about="http://www.example.org/zampano">
    <rdf:type>
      <rdf:Description rdf:about="http://www.example.org/Uomo">
      </rdf:Description>
    </rdf:type>
  </rdf:Description>
</rdf:RDF>
```



```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:ex="http://www.example.org/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://www.example.org/zampano">
    <rdf:type>
      <rdf:Description rdf:about="http://www.example.org/Uomo">
      </rdf:Description>
    </rdf:type>
  </rdf:Description>
</rdf:RDF>
```





In the RDF/XML document, the root is always an element of type `rdf:RDF`.

And there are two types of elements:

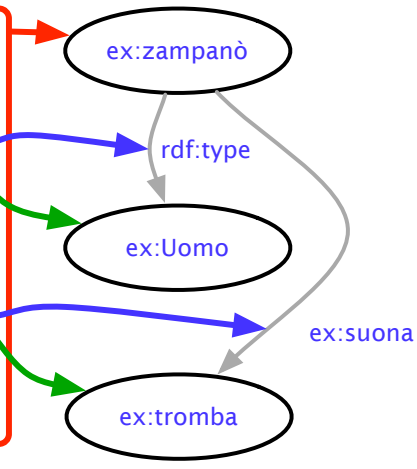
- node elements, encoding nodes of the RDF graph
  - node elements are of type `rdf:Description`
  - the URI of the graph node is the value of the attribute `rdf:about`
- property elements, encoding the arcs of the RDF graph
  - the type of property elements is the label of the arc (*i.e.*, the property of the corresponding triple)

Every triple of the graph can be written in this way, and the whole graph is then transformed into an XML document.

There are several abbreviations that can be used.

Arcs outgoing of the same node can be represented by sibling property elements:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:ex="http://www.example.org/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://www.example.org/zampano">
    <rdf:type>
      <rdf:Description rdf:about="http://www.example.org/Uomo">
      </rdf:Description>
    </rdf:type>
    <ex:suona>
      <rdf:Description rdf:about="http://www.example.org/tromba">
      </rdf:Description>
    </ex:suona>
  </rdf:Description>
</rdf:RDF>
```



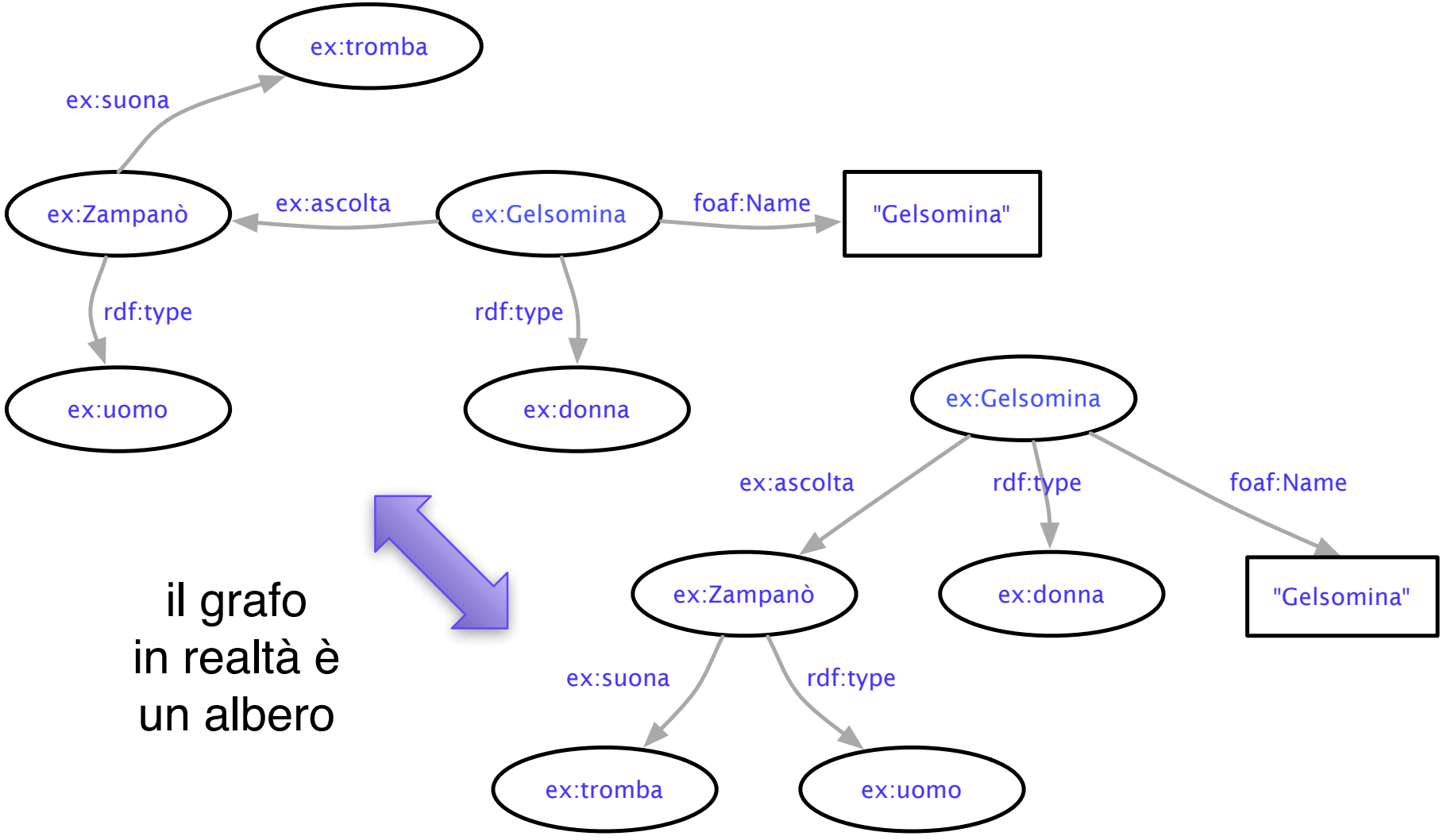
Here, `rdf:type` and `ex:play` are sibling elements, that is sub-element of the same element.

In addition, elements may be nested in order to serialize paths in graphs.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:ex="http://www.example.org/"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns=foaf:"http://xmlns.com/foaf/spec/>

  <rdf:Description rdf:about="http://www.example.org/gelsomina">      Una risorsa
    <foaf:Name>"Gelsomina"</foaf:Name>                                di nome "Gelsomina"
    <rdf:type>
      <rdf:Description rdf:about="http://www.example.org/Donna">      e donna
      </rdf:Description>
    </rdf:type>
    <ex:ascolta>                                                      e ascolta
      <rdf:Description rdf:about="http://www.example.org/zampano">    una risorsa che
        <rdf:type>
          <rdf:Description rdf:about="http://www.example.org/Uomo">   è un uomo
          </rdf:Description>
        </rdf:type>
        <ex:suona>                                                    e suona
          <rdf:Description rdf:about="http://www.example.org/tromba"> una tromba
          </rdf:Description>
        </ex:suona>
      </rdf:Description>
    </ex:ascolta>
  </rdf:Description>
</rdf:RDF>
```

This is easy because . . .



Datatypes

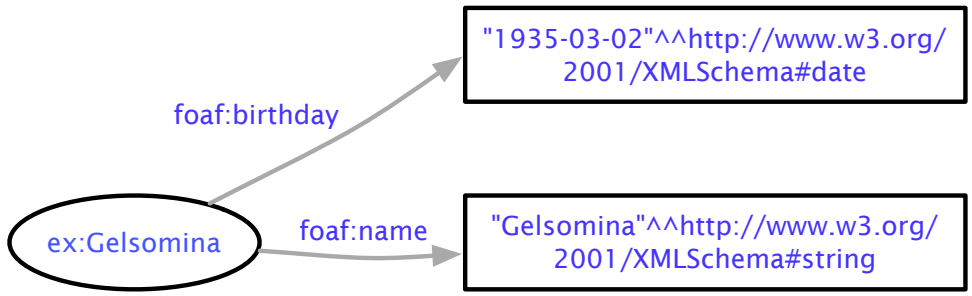
Literals can be typed to represent in a structured way:

- dates
- measures

and other common values.

The type is identified by a URI and the W3C recommends using types defined in XML Schema.

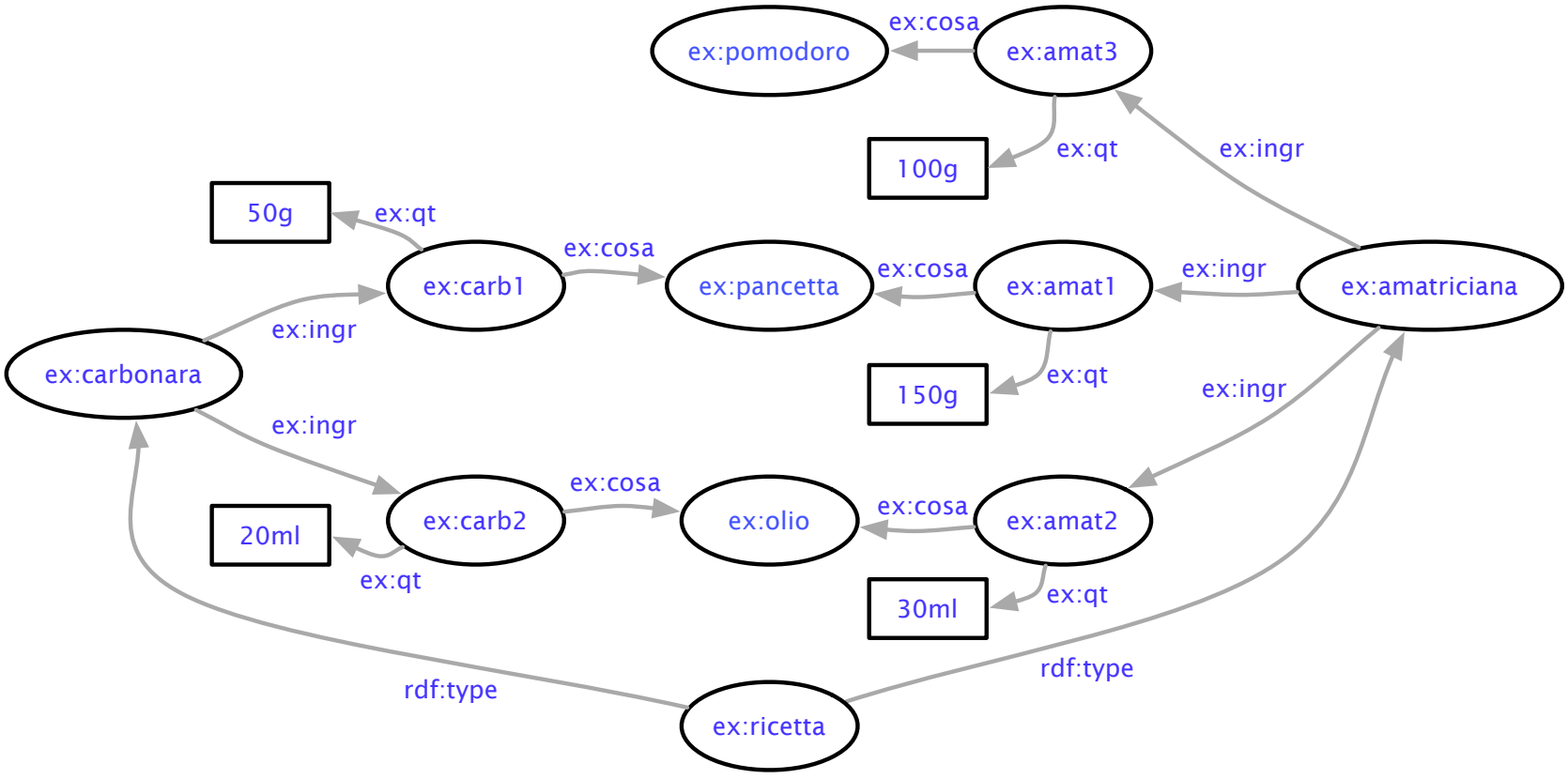
A typed literal is a literal followed by two special “hat” characters and then by the URI of the type.



Many-valued relationships

In order to represent knowledge in RDF, we need to express any relation as a binary relation.

If we have a ternary relation: (Recipy, Ingredient, Quantity), we can express the knowledge in it by means of three binary relations:



There are general methods to deal with *n*-ary relations in RDF, with *n* greater than 2.

These methods are called *ontology patterns*, because each of them responds to a large set of use cases.

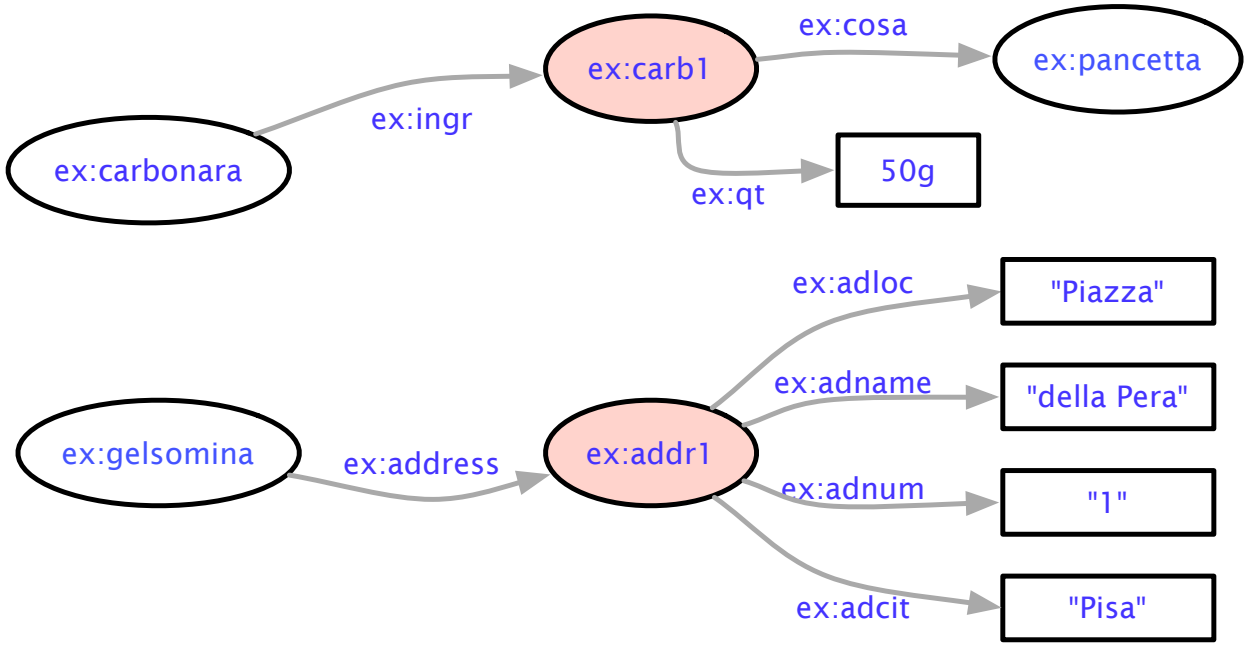
Reference:

Defining N-ary Relations on the Semantic Web. W3C Working Group Note. 12 April 2006. http://www.w3.org/TR/swbp-n-aryRelations/

Blank nodes

Sometimes it is useful to create nodes in RDF graphs without giving them an explicit URI.

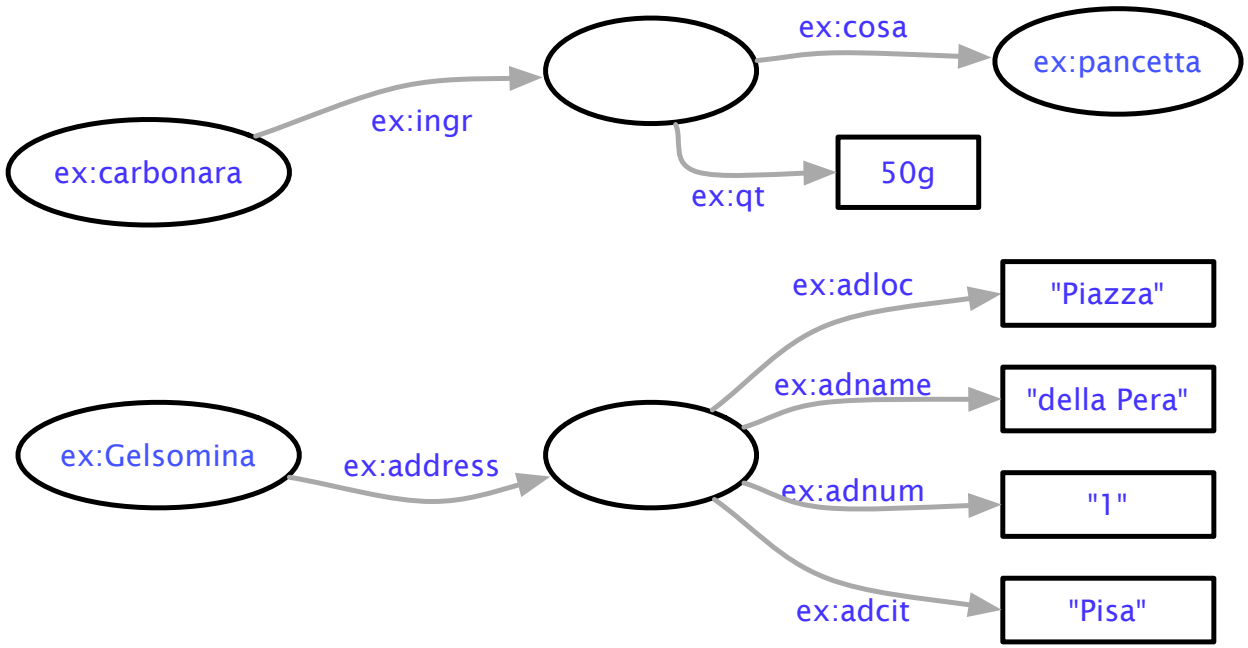
- dummy nodes created, *e.g.*, for reducing a ternary relation
- resources that are not important



In these cases, a *blank node* can be used, that is a node with only a local identifier, to identify it within the graph in which it occurs.

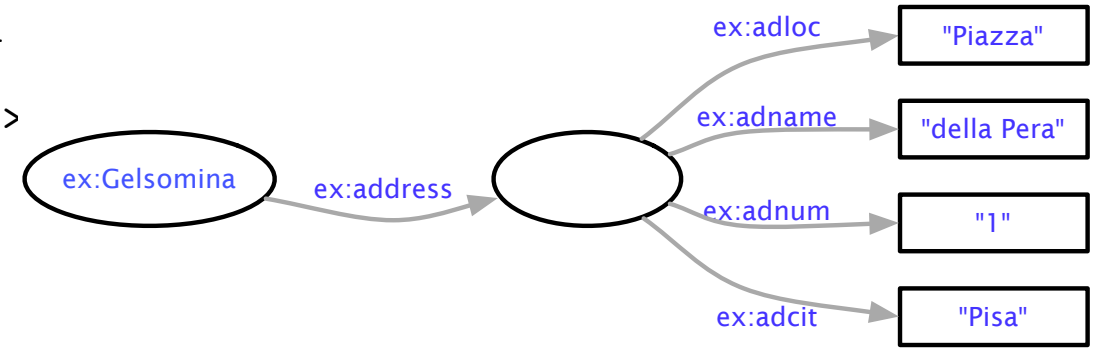
The meaning of a blank node is "there exists a resource", of which nothing is known except the knowledge declared in the graph where the blank node occurs.

Blank nodes can be used only as subjects or objects of triples.



Blank node are serialized in RDF/XML by using the attribute `rdfnodeID`, whose value is a literal that uniquely identifies the node in the present graph.

```
<rdf:Description rdf:about="http://www.example.org/gelsomina">
  <ex:address rdf:nodeID="ga"/>
</rdf:Description>
<rdf:Description rdf:nodeID="ga"
                 ex:adloc="Piazza"
                 ex:adname="della
                 ex:adnum="1"
                 ex:adcit="Pisa"/>
```



In a different graph, `ga` may refer to a different blank node.

The semantics of an RDF graph does not change if all id's of blank nodes are consistently replaced by other id's.

In Turtle, blank nodes are encoded by using a very special prefix: the underscore _

```
@prefix ex: <http://www.example.org/> .
@prefix foaf: <http://xmlns.com/foaf/spec/> .

ex:gelsomina  ex:address   _:ga .
_:ga          ex:adloc     "Piazza" ;
              ex:adname    "della Pera" ;
              ex:adnum     "1" ;
              ex:adcit     "Pisa" .
```

and this can be abbreviated:

```
@prefix ex: <http://www.example.org/> .
@prefix foaf: <http://xmlns.com/foaf/spec/> .

ex:gelsomina  ex:address
            [ ex:adloc     "Piazza" ; ex:adname    "della Pera" ;
              ex:adnum     "1" ;      ex:adcit     "Pisa" ] .
```

Summary

The RDF vocabulary seen so far includes three types of entities:

- individuals, representing resources
- properties, representing binary relations
- classes, representing unary relations

These entities are expressed by URIs defined by the user, or by literals (but only for objects in triples).

We have also encountered some URIs defined by RDF:

- `rdf:type` to link an individual to the class where it belongs
  - `ex:zampanò rdf:type ex:Man .`
- `rdf:XMLLiteral` the class of all XML values

These URIs are part of the RDF Vocabulary, historically the first RDF vocabulary.

Conclusions

RDF allows to represent elementary facts as triples.

The objects in triples can be literals or snippet XML.

Subjects or objects can be blank nodes.

Thanks to XML Schema we can represents typed literals.

Limitations:

- In RDF it is not possible to represent negative knowledge (Gelsomina is not a man) or disjunctive knowledge (Either Zampanò or Gelsomina is playing).
- We can't express generic knowledge, such as:
  - every man is a person
  - if someone plays a trumpet, it is a person
  - if something is played, it is a trumpet

To overcome the former limitation, we need to look at OWL.

To overcome the latter limitation, we need to look at richer vocabularies, namely RDF Schema, which will allow us to define simple ontologies.

Part IV

RDF Schema

Classes and instances

In RDF it is possible to represent knowledge about binary and unary relationships.

```
ex:zampanò rdf:type ex:Man .
```

`ex:Man` is a class, and membership of `ex:zampanò` in the class is said as "`ex:zampanò` is an instance of `ex:Man`".

In RDF Schema (RDFS) it is possible to state that a resource is a class:

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

Prefix `rdfs:` is typically used for RDF Schema:

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

it identifies the namespace that contains the URIs of RDFS.

`Class` has classes as instances, and therefore is a metaclass.

`Class` is a very special metaclass, the metaclass of all classes.

In fact, that `ex:Man` is a class is a logical consequence of the fact that `ex:zampanò` is an instance of `ex:Man`.

- But for better readabilty, it is stated explicitly.

The triples that are logical consequences of an RDFS graph are defined in a very precise way, and algorithms for the derivation of them are now very common.

`rdfs:Class` is an instance of itself:

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

This ia typically implicit in every graph, so it is a *valid* triple.

The RDF vocabulary includes several classes (`rdf:Property`, `rdf:XMLLiteral`, ecc.) and all these are instances of `rdfs:Class`.

The RDFS vocabulary adds more classes:

- `rdfs:Resource` all resources
- `rdfs:Literal` all literals, included all datatypes
- `ContainerMembershipPropery` all properties

Subclasses

Suppose that in our application we need to know what are the known persons, regardless of their gender.

We have two person classes: `ex:Man` e `ex:Woman`.

So we could ask:

- which men are known
- which women are known

and take the union of these queries to find our answer.

This is not only boring, it is also partial: if in the future we introduce the classe `ex:Boy` we need to re-write our query to include also

- which boys are known

It is much simpler to introdu a classe `ex:Person`

But then we need to make sure that every man and every woman are person, by establishing the appropriate relation between the corresponding clasees `ex:Person` `ex:Man` and `ex:Woman`.

In order to state that every man and every woman is a person, in RDFS we state that class `ex:Man` and class `ex:Woman` are *sub-classes* of `ex:Person`:

```
ex:Man rdfs:subClassOf ex:Person .
ex:Woman rdfs:subClassOf ex:Person .
```

We also say that `ex:Person` is a superclass of `ex:Man` and of `ex:Woman`.

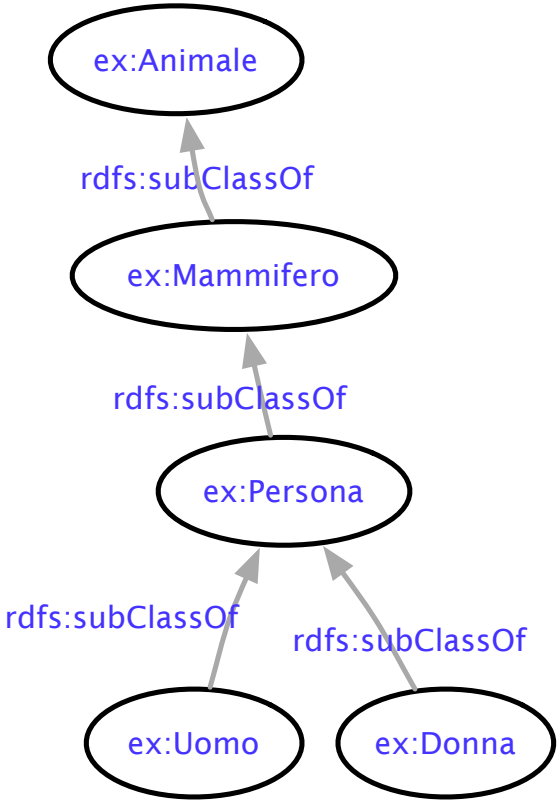
`rdfs:subClassOf` is an RDFS property, therefore the triple

```
rdfs:subClassOf rdf:type rdf:Property .
```

is valid.

The `rdfs:subClassOf` property can be used to build class *hierarchies*

Class hierarchies



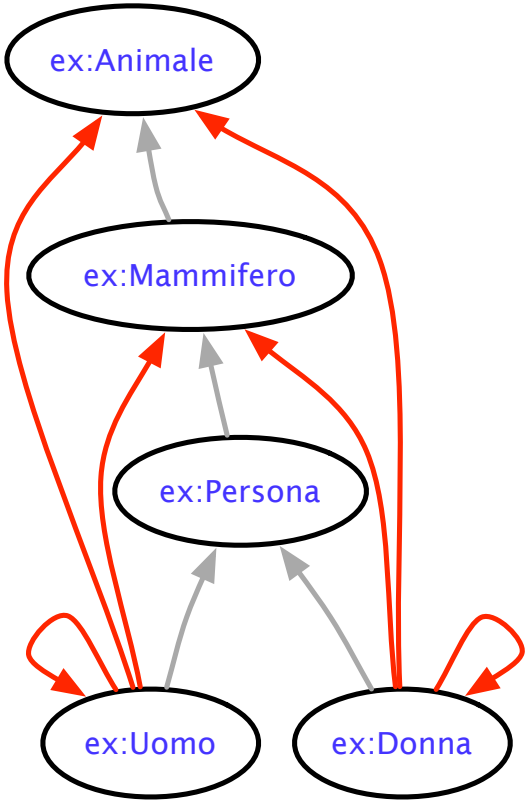

```
ex:Man rdfs:subClassOf ex:Person

ex:Woman rdfs:subClassOf ex:Person

ex:Person rdfs:subClassOf ex:Mammifero

ex:Mammifero rdfs:subClassOf ex:Animal
```

Class hierarchies



The red arcs are logical consequences, due to the fact that in mathematics the subset relation is reflexive and transitive

(Note: not all consequences are depicted)

Other consequences:

```
ex:gelsomina rdf:type ex:Person
ex:zampano rdf:type ex:Mammifero
```

An RDFS graph that consists exclusively of `rdfs:subClassOf` triples is called a *taxonomy.*

In the real world, we have at least two types of taxonomies:

- scientific taxonomies (regno, philum, classe, ordine, famiglia, genus e specie). These are modelled by `rdfs:subClassOf`
- linguistic taxonomies, such as those found in thesauri. These have a different epistemic value than scientific taxonomies and are therefore modelled by a different vocabulary: SKOS.

Class equivalence:

```
ex:Animal rdfs:subClassOf ex:Animal
ex:Animal rdfs:subClassOf ex:Animal
```

Taxonomic cycles.

Properties

In RDF it is possible to express some ontological knowledge

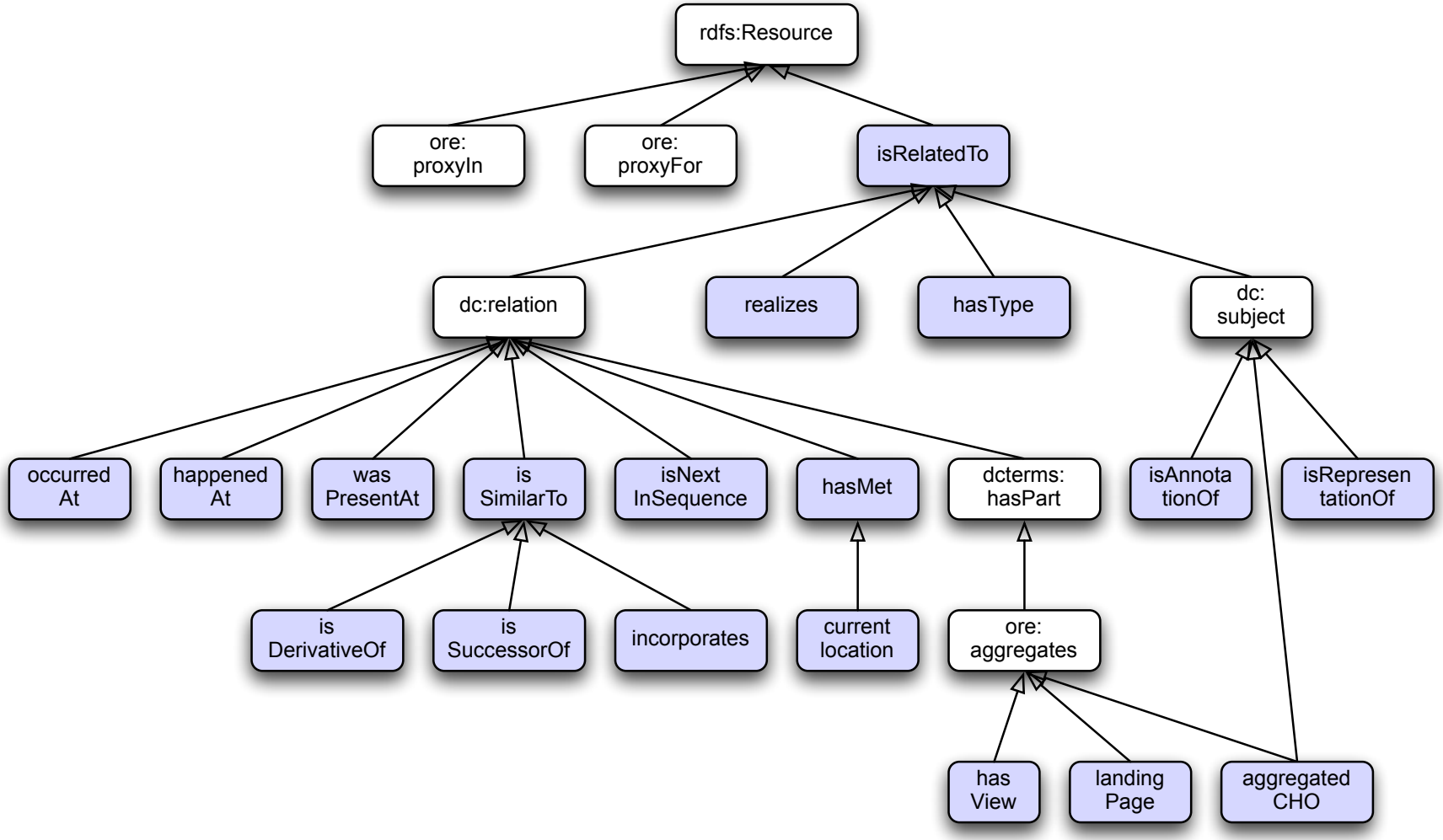
```
ex:play rdf:type rdf:Property
```

But there is a lot more to say about binary relations.

A binary relation is a set, therefore it can have subsets.

- In maths, to be a divisor is a subset of to be smaller or equal
- in geometry, to be a point on a radius is to be an internal point of a circle
- in sociaty, to be happily married implies to know each other

Taxonomy of EDM Properties



Subset relation between binary relations is expressed in RDFS via `rdfs:subPropertyOf` statements:

```
ex:playWell rdfs:subPropertyOf ex:play
```

To be read as:

- `ex:playWell` is a sub-property of `ex:play`
- `ex:play` is a super-property `ex:playWell`

Consequences:

- `rdfs:subPropertyOf rdf:type rdf:Property .` is valid
- `rdfs:subPropertyOf` is reflexive and transitive.
- If `ex:zampano ex:playWell ex:trumpet` then `ex:zampano ex:play ex:trumpet`

Restrictions on Properties

Another imprtant characteristic of a relation concerns:

- the *domain* of the relation
- the *range* of the relation

For instance, `ex:play` has as domain persons and as range musical instruments

- if someone plays something, then this someone is a person and this something is a musical instrument

One can implement this manually:

```
ex:zampano ex:play ex:trumpet
ex:zampano rdf:type ex:Person
ex:trumpet rdf:type ex:MusicalInstrument
```

But this is time-consuming and error-prone.

In RDFS, it is possible to obtain this knowledge as implicit knowledge, which is a logical consequence of the explicit knowledge.

To this end, we must declare domain and range of the involved property:

```
ex:play rdfs:domain ex:Person
ex:play rdfs:range ex:MusicalInstrument
```

The effect of the former declaration is that any resource that is the subject of a triple whose predicate is `ex:play` it is automatically understood to be an instance of class `Persona`.

The same for range.

`rdfs:domain` and `rdfs:range` are Properties that specify the domain and range of a Property, respectively.

```
rdfs:domain rdfs:domain rdf:Property
rdfs:domain rdfs:range rdfs:Class
rdfs:range rdfs:domain rdf:Property
rdfs:range rdfs:range rdfs:Class
```

We have seen that the triples ia a graph are assumed to be all simultaneously true.

Therefore:

```
ex:play rdfs:domain ex:Person
ex:play rdfs:domain ex:Artist
```

mean that whoever `ex:play` something is **both** a `ex:Person` **and** an `ex:Artist`.

The same for range.

Now, suppose we add the triples:

```
ex:cita ex:play ex:trumpet
ex:cita rdf:type ex:Monkey
```

Based on the previous rules, we now have as implicit knowledge:

```
ex:cita ex:type ex:Person
```

Intuitively, we have created an inconsistency, because **we** know that monkeys are **not** persons.

But we did not specify that, so there is no inconsistency.

The RDFS Vocabulary

The RDFS Vocabulary is a set of URIs that includes all the URIs in the RDF Vocabulary, plus the following classes:

- `rdfs:Resource` the class of all resources
- `rdfs:Class` the class of all classes
- `rdfs:Literal` the class of all literals
- `rdfs:Datatype` the class of all datatypes
- and more

and the following classes properties:

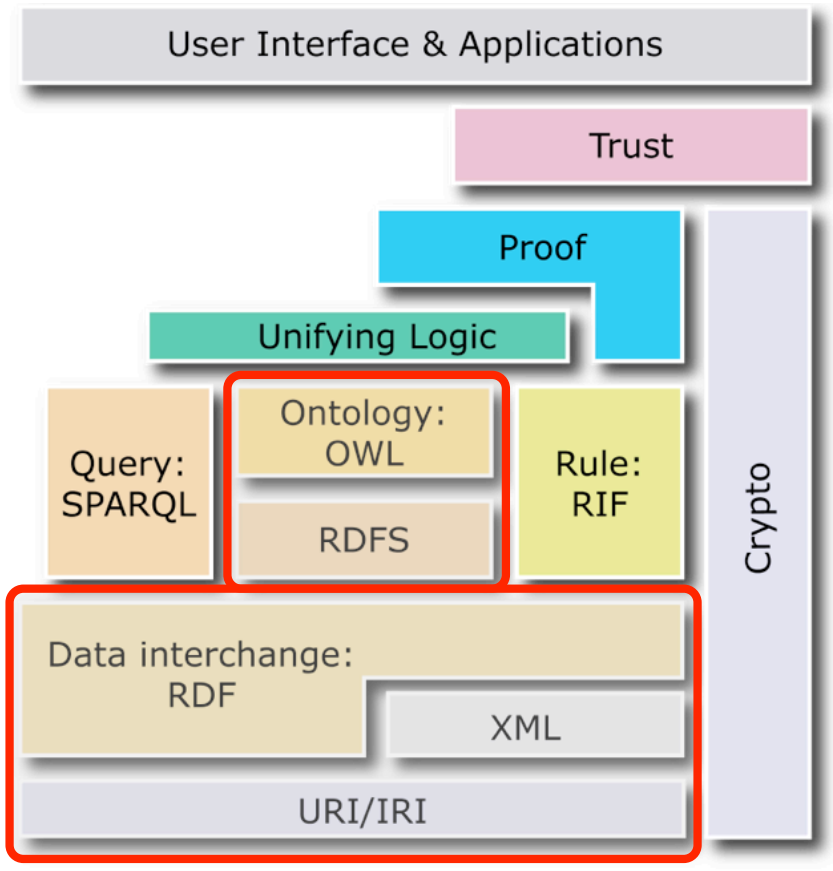
- `rdfs:domain`
- `rdfs:range`
- `rdfs:subClassOf`
- `rdfs:subPropetyOf`
- `rdfs:label` for labelling a resource
- `rdfs:comment` for commenting about a resource
- `rdfs:seeAlso` for associating to a resource another resource giving further info
- and more

Part V

Ontology Web Language

In order to represent useful and realistic ontologies, a more expressive language than RDF is needed.

Web Ontology Language (OWL).



The OWL family

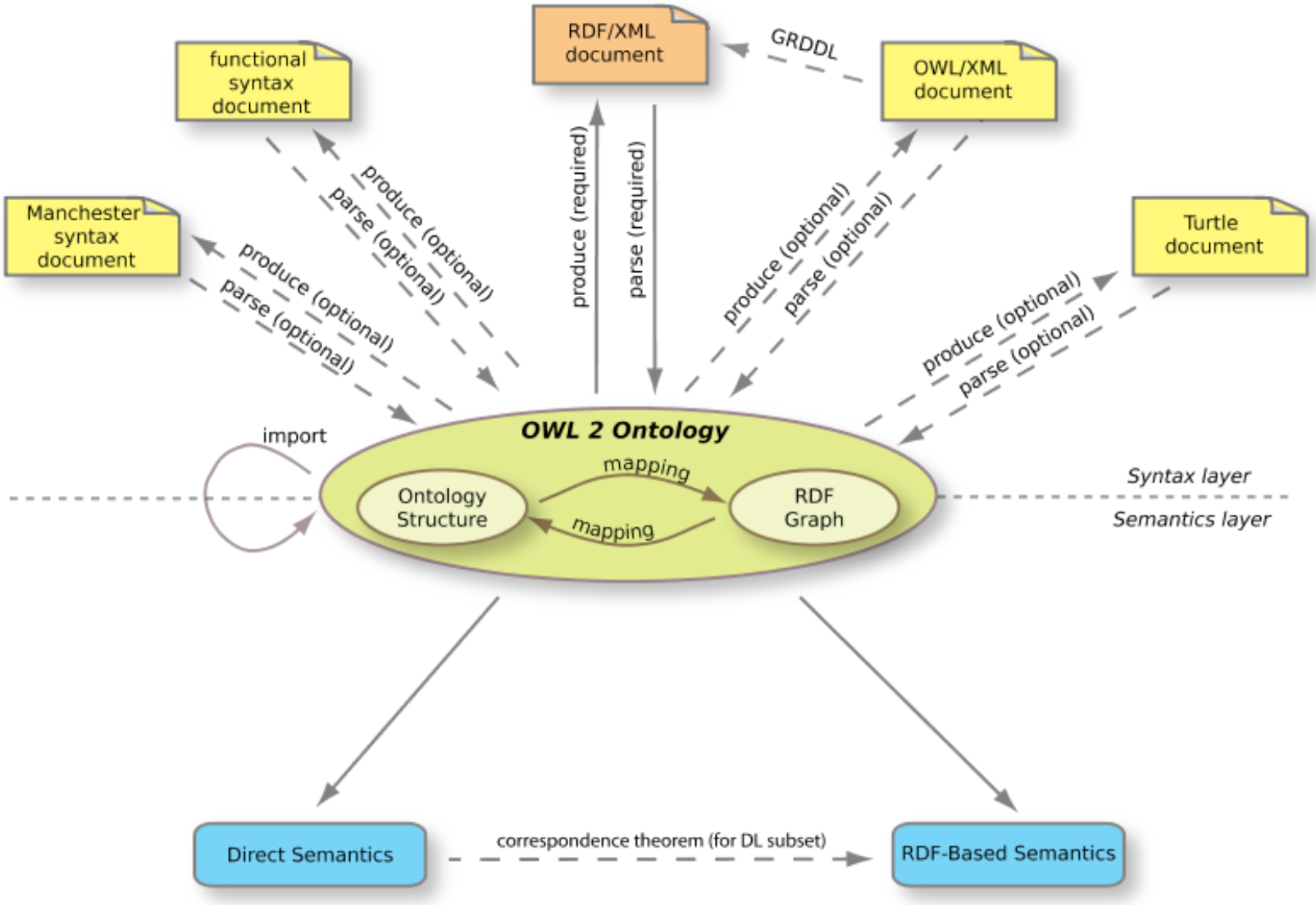
OWL 2 is a W3C Recommendation of October 2009.

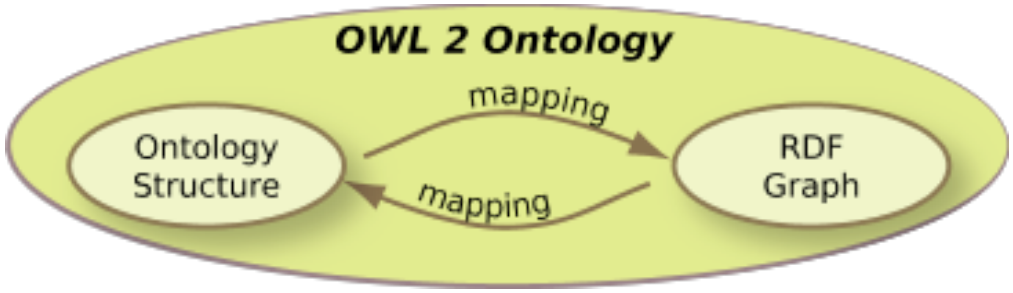
Predecessor: OWL 1, 2004.

Back compatibility: all OWL 1 ontologies are also OWL 2 ontologies.

OWL 2 extends OWL 1 con:

- a richer language, fully compatible with OWL 1, but with more constructs adding expressivity
- three new profiles addressing applications, all efficiently implementable.


RDF/XML document
GRDDL
OWL/XML document
functional syntax document
Manchester syntax document
Turtle document
produce (optional)
parse (optional)
produce (required)
parse (required)
produce (optional)
parse (optional)
produce (optional)
parse (optional)
produce (optional)
parse (optional)
import
OWL 2 Ontology
mapping
mapping
Ontology Structure
RDF Graph
Syntax layer
Semantics layer
Direct Semantics
correspondence theorem (for DL subset)
RDF-Based Semantics

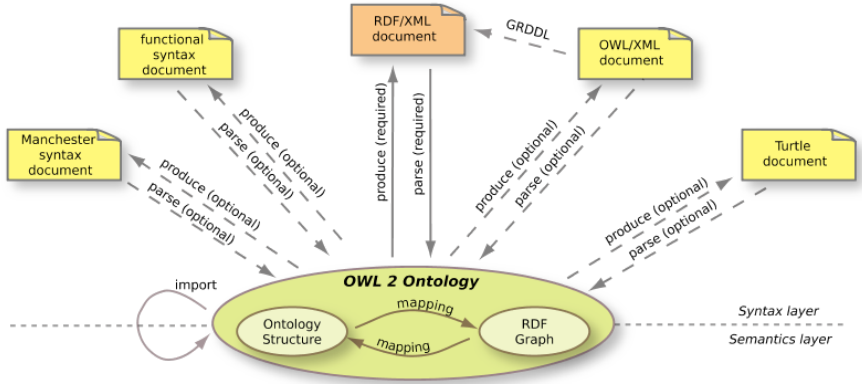


An OWL 2 ontology can be an RDF graph (of a very rich vocabulary) or an equivalent conceptual, specified in UML.

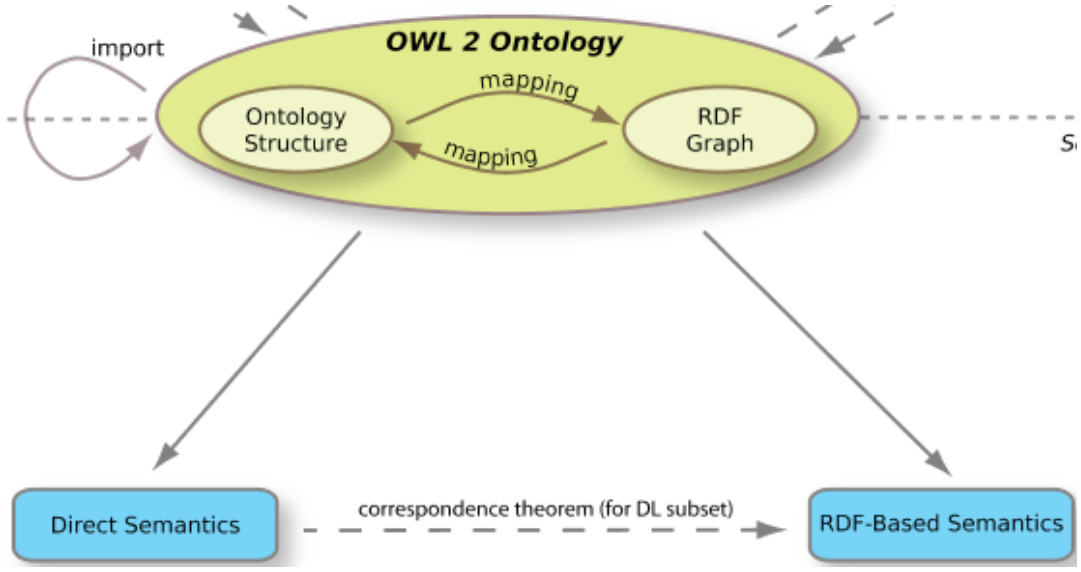
There is a translation between the two, so they are really a syntactic variant of one another.

But both are at an abstract level, they are not serialized, that is expressed in one of the accepted notations.

There are two families of notations:



- Notations based on RDF: the ontology is an RDF graph on the OWL vocabulary.
  - RDF/XML, the only "official" notation, which must be supported by all tools that want to qualify as OWL tools.
  - Turtle, easier to read, supported only by some tools
- Notations based on OWL:
  - Manchester, very easy to read and write
  - Functional, directly reflecting the UML conceptual structure
  - OWL/XML, which can be managed with any XML tool



As a consequence, there are two semantics:

- RDF-based: assigns meaning to ontologies specified as RDF graphs
- Direct: assigns meaning to ontologies specified in the functional syntax

The two semantics are mathematically equivalent.

Part VI

Linked Data

Linked Data are data that follow 4 recommendations:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs so that they can discover more things.

Ingredients:

- language: URIs, RDF, SPARQL
- mechanics: HTTP look up

Linked Data is a way of publishing data on the Web that:

- encourages reuse
- reduces redundancy
- maximises its (real and potential) inter-connectedness
- enables network effects to add value to data

The Web of Linked Data:

- Analogy: a global database
- Designed for machines first, humans later
- Handles descriptions of things
- Links between URIs and not between pages
- Semantics is made explicit by having one language for everything

How to Publish Linked Data on the Web

A 15 steps recipe.

The steps form the basis for different workflows that can be used to publish Linked Data, depending on purpose, data and context.

Data of interest:

- knowledge organization systems (classification schemes, thesauri)
- authority files
- digital contents and their descriptions
- catalogues
- catalogue data including circulation data sets

All these datasets should have links within themselves and should establish outgoing links to many other web resources, in order to attract many incoming links.

1. Motivation
2. Management approval
3. Sorting out the legal and financial issues
4. Assessment of skills & data available
5. Tools assessment and evaluation
6. Dataset analysis
7. URI assignment
8. Vocabulary Modeling
9. Generation of RDF Data
10. Enriching the data
11. Describing the data set
12. Evaluating the Dataset
13. Publishing
14. Incoming links
15. Curation

URI assignment

Each resource in the dataset has to be identified by a unique URI, created according to the following guidelines:

- Use HTTP URIs so that they are dereferenceable.
- Ensure that the URIs are from a namespace that you control.
- Make sure your URIs do not carry implementation details which can change over time.
- It is advisable to use meaningful natural keys in URIs as unique identifiers of resources
  - for example, books can be identified by using the ISBN number instead of primary keys in the local database.

One resource in a dataset usually leads to the creation of at least three URIs:

- the one that represents the real world object
- the one that represents its HTML representation
- the one that represents its RDF/XML representation.

One way:

- http://dbpedia.org/resource/New_York_City
- http://dbpedia.org/data/New_York_City
- http://dbpedia.org/page/New_York_City

Another way:

- http://mydomain.com/thing
- http://mydomain.com/thing.rdf
- http://mydomain.com/thing.html

Vocabulary Modeling

Vocabulary modelling has to do with creating controlled terminology giving explicit meaning to the concepts in your dataset, and this is a key process in linked data.

If the original data is in some complex form such as relational tables, the semantics of the tables and the attributes has to be understood and encoded in RDFS or in OWL.

The emphasis is on the use of already existing vocabularies for the sake of interoperability.

A vocabulary of choice should:

- be widely used to ensure widespread use of your dataset
- be actively maintained according to a clear governance process
- cover enough of your dataset to justify its terms
- be expressive enough to suit your particular requirements.

Creating your own vocabulary should be the very last option in vocabulary modelling.

Whenever you create a new vocabulary in the linked data world you have created a data island and have decreased the level of understanding in that domain.

Unless you are a very well known authority in that domain, it is likely that your vocabulary will not be used by someone else.

Creating a vocabulary is a very complex process which needs linguistic skills and domain expertise.

In the event that you create your own vocabularies, consider relating your new vocabulary to known vocabularies, making your concepts (classes and properties) sub-concepts of those in known vocabularies.

Enriching the data

This process involves defining news triples that define relationships internally within the dataset (internal links) or relationships with outside resources (outgoing links).

- For internal links it must be ensured that every part of the dataset is reachable by a crawler when it is following links and therefore each file has to be connected to related files in the same dataset.
- For outgoing links, it is advisable to start by linking to such datasets as dbpedia, Geonames, Europeana, VIAF and others, which are already well established and stable in the linked data world.

This ensures that your dataset is easily discoverable since these are widely linked to by many other datasets.

Describing the data set

Before publishing, there is the need to provide a description of the dataset, which includes:

- provenance metadata–the history of that dataset, how it was generated and the technical processes that have been undergone to establish the dataset
- license and waiver metadata–how that dataset maybe used by third parties.
- the topic of a data set
- URI of the dataset
- location of SPARQL end-point
- data dumps
- last-modified date of the dataset
- change frequency

These data may be described using the Vocabulary of Interlinked datasets (voiD), which is an RDF vocabulary.

Publishing the data set

The most obvious way to publish Linked Data on the Web is to make the URIs that identify data items dereferenceable into RDF descriptions.

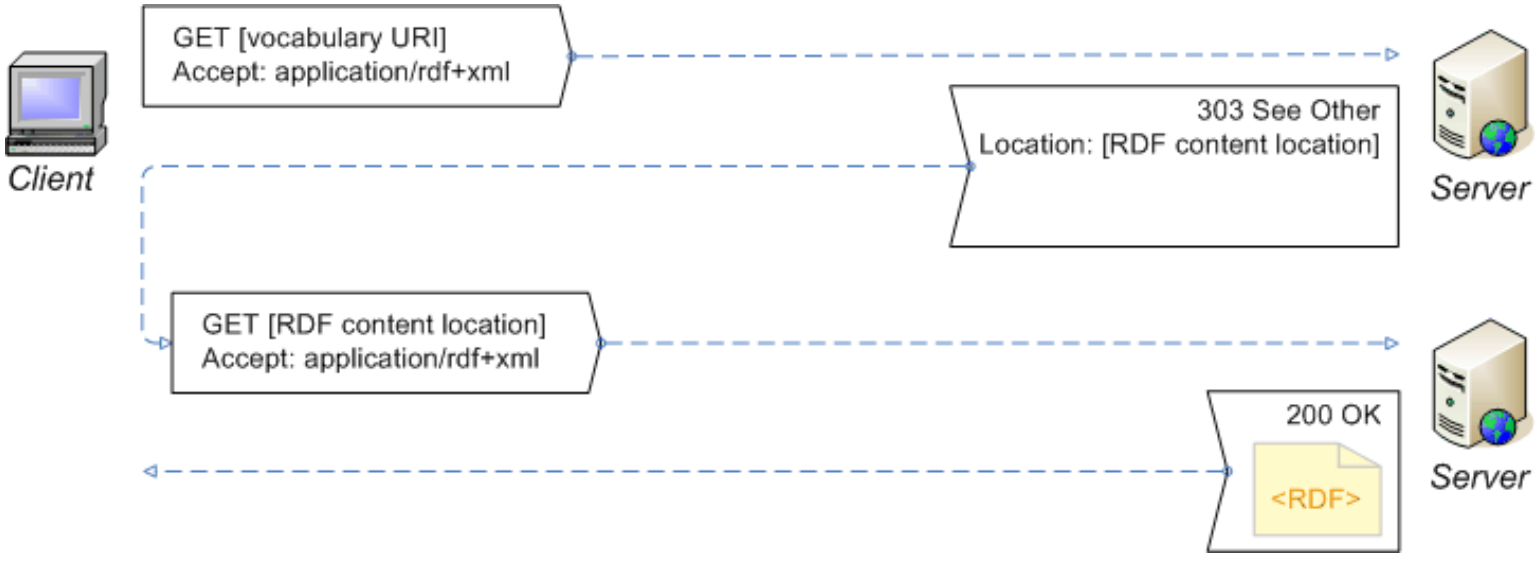
In addition, various Linked Open Data providers, including libraries, provide two alternative means of accessing the data:

- via SPARQL endpoints
- by providing RDF dumps of the complete data set

In general the system should provide access to both the RDF and HTML representations of the data.

This is usually done by configuring 303 redirects in triple stores in response to a client request to access either the HTML representation of an object or its RDF representation.

Architecture



Compare with the web mechanism for accessing informal knowledge

Incoming links

Incoming links originate from other datasets, linking into your dataset.

Third parties need to be convinced that your dataset is valuable to them so that they can link to it. However it is usually difficult for them to know your value unless you do some sort of marketing and promotion actions.

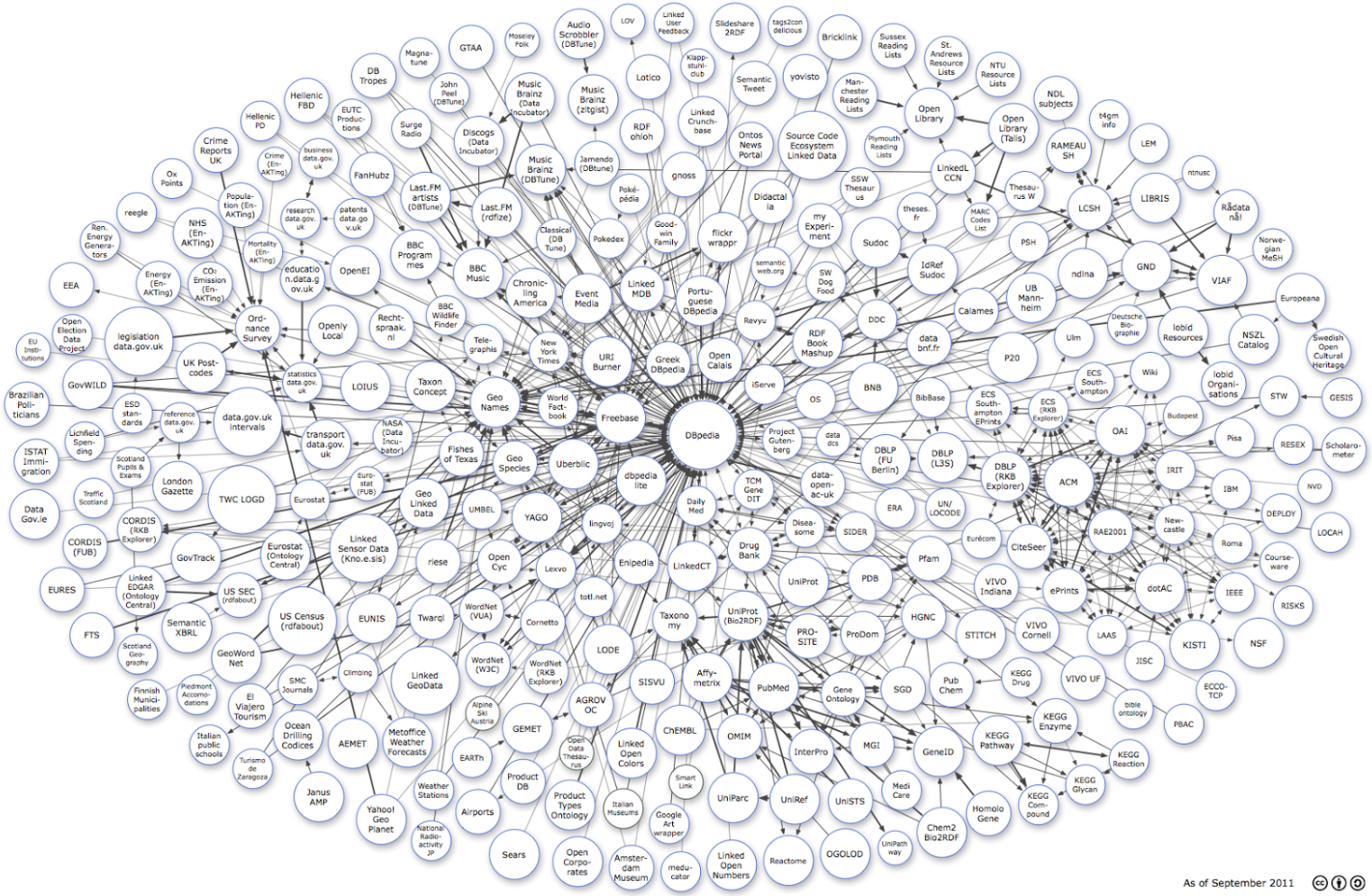
As a starting point a publisher can create triples that link to their own dataset and ask third parties like dbpedia to add those triples to their own dataset.

To continue attracting new links, there might be the need to employ marketing techniques so that the dataset is known by new users and be linked to.

Conclusions

Linked Data are the way the semantic web is coming true.

Check out: linkeddata.org



Publishing Linked Data is an expensive process but it pays back in terms of making one's own data "linked" into the global database.

Sharing is the way to reduce costs:

- URIs, for inceasing the linkedness
- vocabularies, for
  - increasing interoperability
  - cutting costs of vocabulary development and maintenance

Part VII

Preserving Linked Data

Issues

The easy part:

- LD are formal knowledge: they consist of statements expressed in a formal language, complying to strict syntactic and serialization rules
- The additional statements that are added to data for preservation purposes (Representation Information, Preservation Description Information) are also formal knowledge
- The vocabularies used in all these statements are also formal knowledge

So, once we learn how to deal with the preservation of formal knowledge, we can preserve everything regarding LD.

The difficult part:

- LD are distributed in nature
  - URIs are the links
  - URIs in a LD dataset may (and actually *should*) come from many different vocabularies
  - URIs of resources owned by other authorities are re-used from other datasets
- LD require the web infrastructure to be accessed
  - If one insists on forever accessibility, one may need to preserve the whole web infrastructure: a clearly impossible task

So, we need to make some assumptions, such as:

- The most important authorities will make sure their LD datasets will live a long time
  - Amongst them, the W3C
- the web will stay for a while
  - Backward compatible evolution

Programme

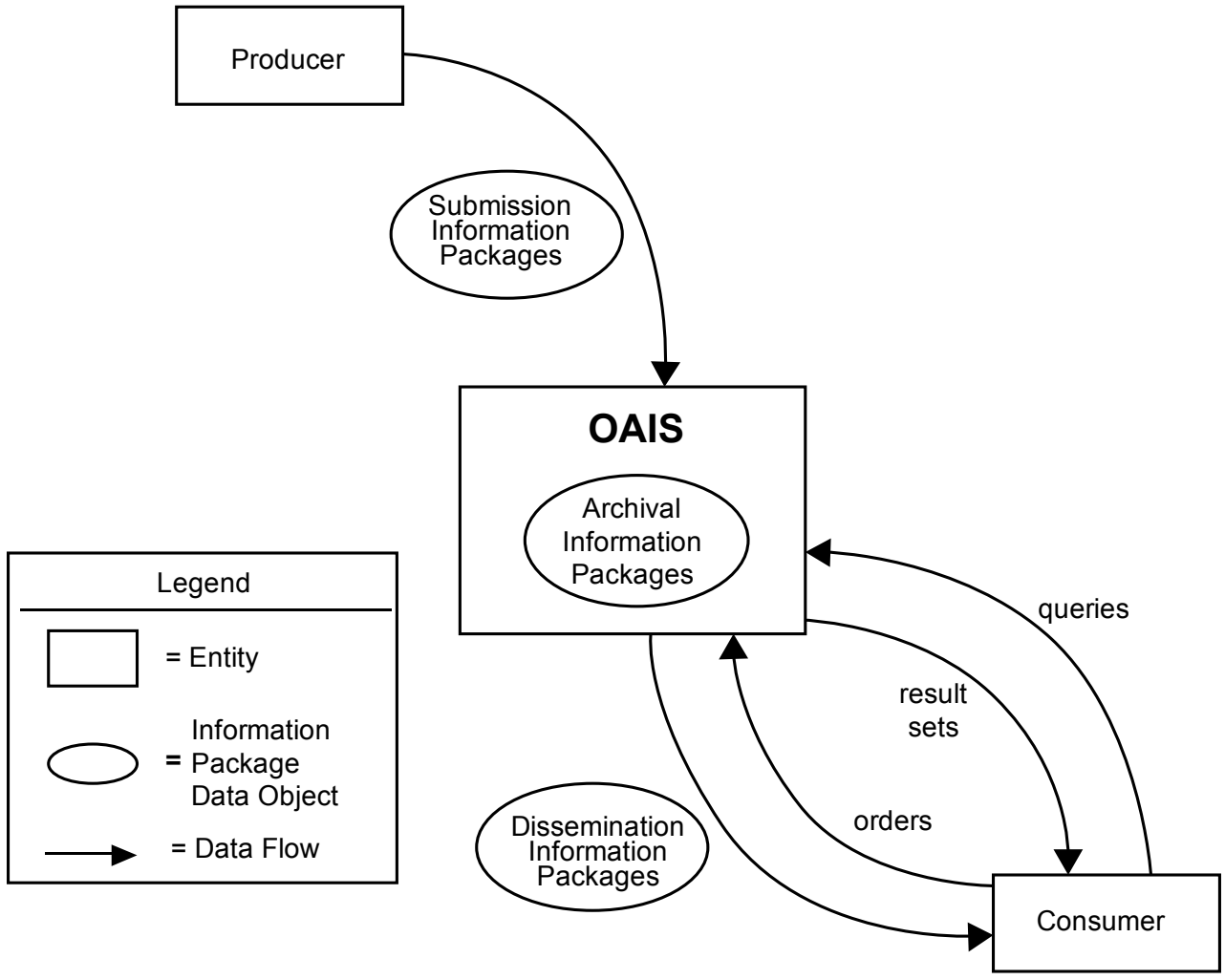
Ingesting Linked Data

- what should an archive require of the LD producer in order to accept the responsibility of making a LD dataset accessible and usable over the long term?

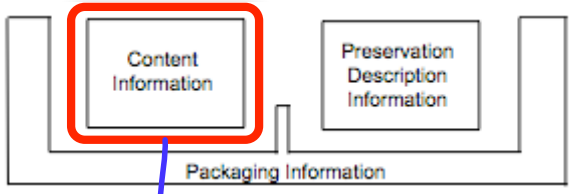
Managing change in Linked Data

- what are the changes that can compromise accessibility and usability of a LD dataset?
  - besides the typical ones . . .

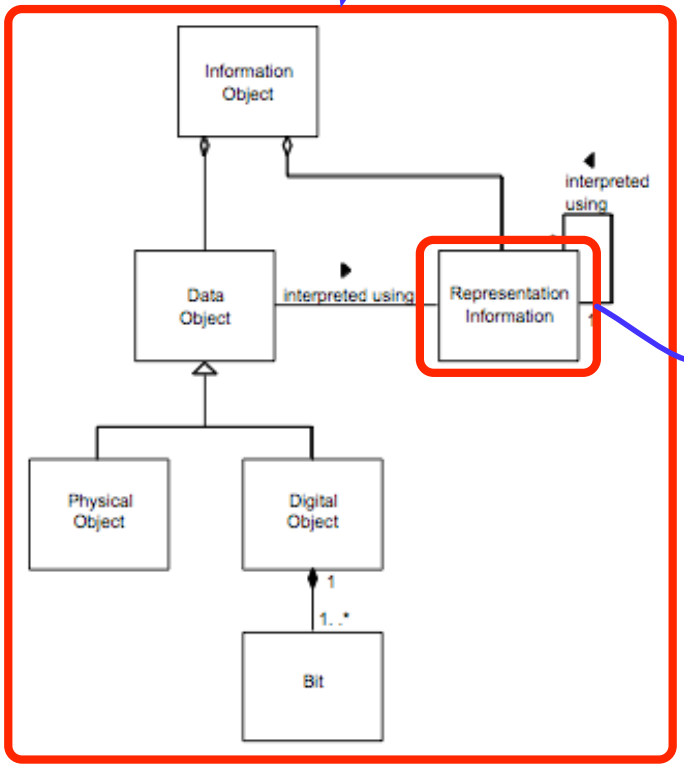
The OAIS Picture of an archive

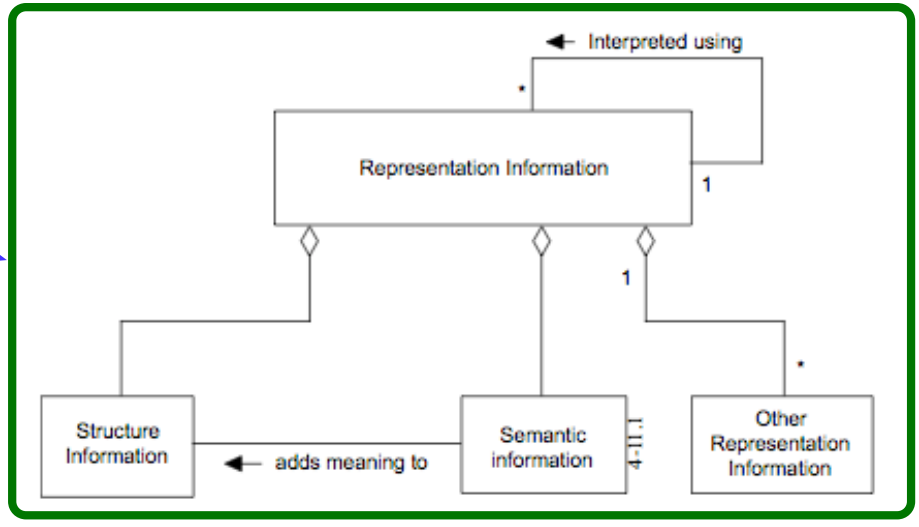


Ingesting Linked Data, *i.e.*, Making an SIP out of an LDD



An Information Package according to the OAIS Information Model





Ingesting Linked Data, *i.e.*, Making an SIP out of an LDD

One can establish the following mappings between Representation Information and Linked Data:

- Structure Information is given by the definition of the serialization format.
  - RDF is a data model, there are many serializations, all Unicode based (RDF/XML, RDFa in HTML, Turtle, etc). Which one do we go for? All of them?
- Semantic Information consists of three parts:
  - the semantics of RDF (as defined in the W3C docs)
  - the semantics of the RDF vocabularies the graph is built on
  - implicit triples, both those coming from the semantics of RDF and those coming from the semantics of the vocabularies. Two options:
    - Preserve the documents where the calculi for computing implicit triples are described
    - Preserve one (which one? maybe all?) inference engine, which may fit in the OAIS picture as "Other Representation Information".

Assuming W3C can successfully preserve their recommendations, we are left with two problems:

- preserving the vocabularies used in an LD dataset, other than those defined by the W3C
- preserving inference engines

The latter problem is a "classical" preservation problem, it is not specific of LD, so we can re-use any method and tool devised in the digital preservation area for solving it.

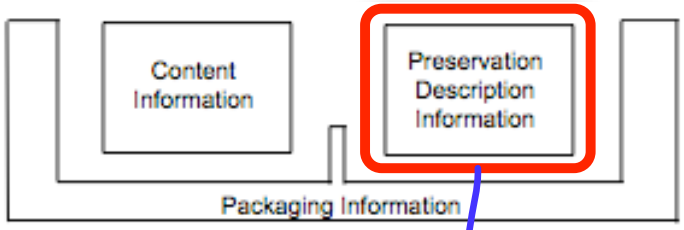
The former problem is LD specific, and currently an open question.

Archivists propose to ingest all vocabularies. Technically this is a viable option because vocabularies are not "big". However:

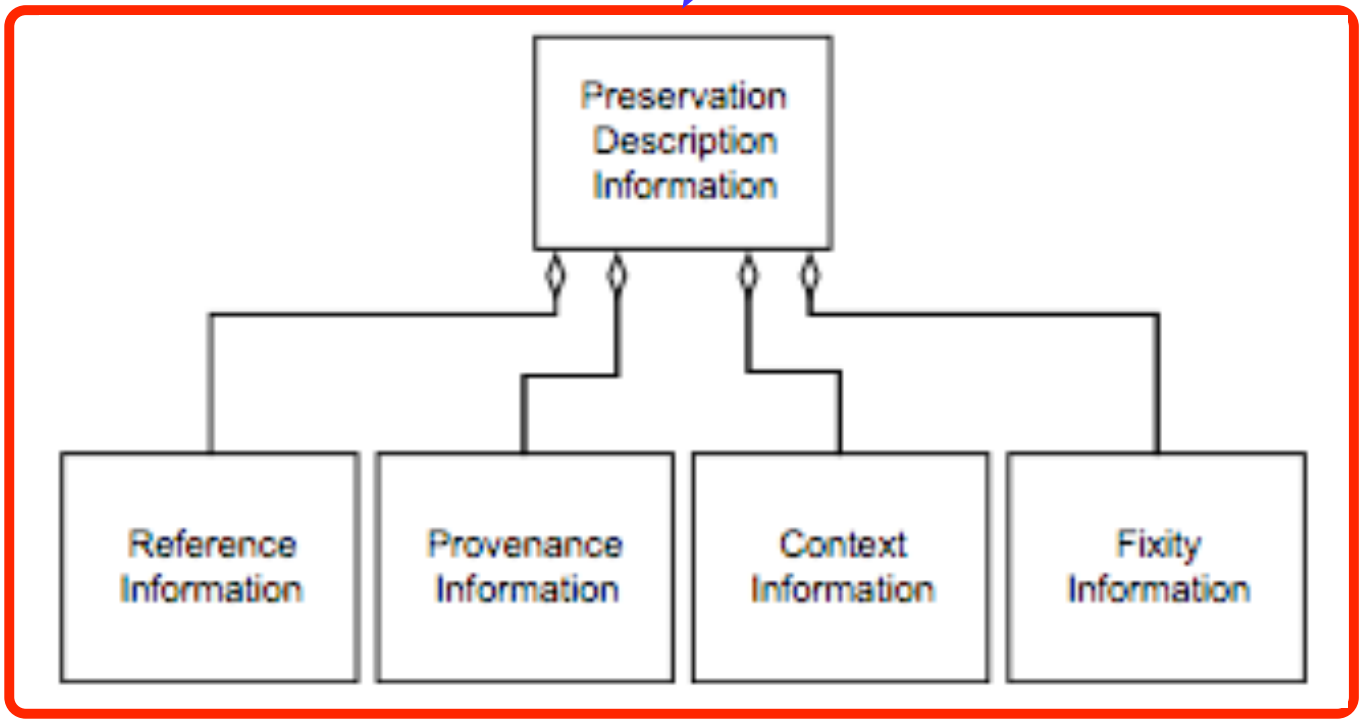
- vocabularies refer other vocabularies, ingesting everyhting may take a while (unless you believe the "designated community" concept)
- vocabularies evolve, ingesting them may make your SIP very perishable

- vocabularies refers

Ingesting Linked Data, *i.e.*, Making an SIP out of an LDD



An Information Package according to the OAIS Information Model



In the case of LDD

- reference is typically done via HTTP IRIs
- fixity information (“the information that documents the authentication mechanisms and provides authentication keys to ensure that the Content Information object has not been altered in an undocumented manner–the Cyclical Redundancy Check (CRC) code for a file”) pertains to a lower level of abstraction. Re-use existing techniques.
- context information (“the information that documents the relationships of the Content Information to its environment. This includes why the Content Information was created and how it relates to other Content Information objects") is what RDF can do very well:
  - by recommending IRIs for identifying both resources and their relationships
  - by providing triples for establishing the connections between resources via relationships
  - by also providing N-Quads and VoID

Provenance Information ("the information that documents the history of the Content Information. This information tells the origin or source of the Content Information, any changes that may have taken place since it was originated, and who has had custody of it since it was originated")

- The PROV vocabulary is recommended by the W3C for expressing Provenance Information. PROV is designed precisely to represent how the RDF was made, what is the history of this dataset before and after ingest.

Open question: is PROV adequate to archival requirements?

Managing change in Linked Data

Changes in the technology used by the archive to preserve the data

The OAIS has the responsibility of monitoring and reacting to these changes.

Business as usual.

Managing change in Linked Data

Changes in the Content Data being preserved

Example The DBPedia LDD is continuously updated by the addition of new triples.

This type of change is rather uncontroversial from the preservation point of view:

- when the owner of the data decides that the changes are significant enough, a new snapshot of the data is taken by re-ingesting the Content Data to the archive.
- The archive sole responsibility is to possibly keep track of the versioning relationships that exist between the different snapshots taken in time from the same Dataset.

Managing change in Linked Data

Changes in the Representation Information or in the Preservation Description Information

Examples

- Migration to a new format of the preserved LDD requires the update of the Representation Information with the new format.
- The organization producing the data goes out of business, and the responsibility of the data is transferred to a different organization. This change needs to be reflected in the Context Information.

A new Submission Information Package is created which must be ingested and properly related to the one it is a new version of.

If the change generates from inside the archive, the producer of the data needs to be consulted.

Managing change in Linked Data

Changes in the vocabularies used in the Content Data or in PDI

Examples

- East Germany and West Germany are (re)united into Germany. The gazetteer that was in use in the archive is updated: a new term for Germany is introduced, whereas the old terms for East and West Germany are deprecated. The statement that the content data was generated in East Germany is part of context in the Preservation Description Information, and must be updated because the term "East Germany" used in it will soon be obsolete and the Designated Community will no longer understand it.
- The definition of "planet" has changed. The ontology of astrophysics that was in use in the preserved data is updated: a new term for planet is introduced and properly axiomatized, whereas the old term is deprecated. The statement that the Content Data is about a planet is part of the Representation Information and needs to be retracted because, according the new meaning of "planet", it is no longer true.

The archive should have in place a mechanism to monitor the vocabularies of the preserved LD and, whenever a change occurs to one

Managing change in Linked Data

Changes in web resources other than those discussed here

Examples

- The Content data contain the URL of an image, and the image goes off-line after a few years. As a consequence, the preserved LDD has a dangling reference.
- The Representation Information of the same astrophysical LDD refers to a PDF document describing some important characteristics of the preserved data. The PDF document was on-line at ingestion time, but after a few years the organization maintaining it changes their access right policy and the document is put behind a billing service. As a consequence, it is no longer accessible.

Web archiving solutions would work most of the times.

Conclusions

Preservation of Linked Data is becoming a fundamental problem, as more and more information is born in linked data format.

It is as hard as the preservation of non-linked data . . .

. . . with some additional problems due to the distributed nature of linked data, and to its dependence on the web infrastructure.

Technical solutions are not hard to achieve.

Shared effort and some more vocabularies are needed.