

RDF

Resource Description Framework

Andrea Turbati

turbati@info.uniroma2.it

- Indipendenza
- Condivisibilità
- Scalabilità
- Ogni cosa è una risorsa
 - Le proprietà sono risorse
 - I valori possono essere risorse
 - Le asserzioni possono essere risorse

- Un modello è un insieme di asserzioni (statements, in inglese)
- **Asserzione: = (soggetto, predicato, oggetto)**
- Il predicato è una *proprietà* del soggetto
- Il soggetto è una risorsa
- L'oggetto è un valore (o una risorsa a sua volta)

- RDF può essere visto come un grafo diretto etichettato, in cui ogni asserzione assume la seguente forma:



- Sia i nodi/entità che gli archi/proprietà possono essere identificate tramite degli URI
- Il significato di un grafo RDF è la *coniunzione* di tutte le sue asserzioni

- RDF corrisponde al sottoinsieme esistenziale-congiuntivo (EC) della logica del primo ordine
 - Non ammette la negazione (NOT)
 - Non ammette la disgiunzione (OR)
- Cosa inusuale per un linguaggio che rappresenti una restrizione della logica del primo ordine, RDF permette asserzioni riguardanti relazioni:

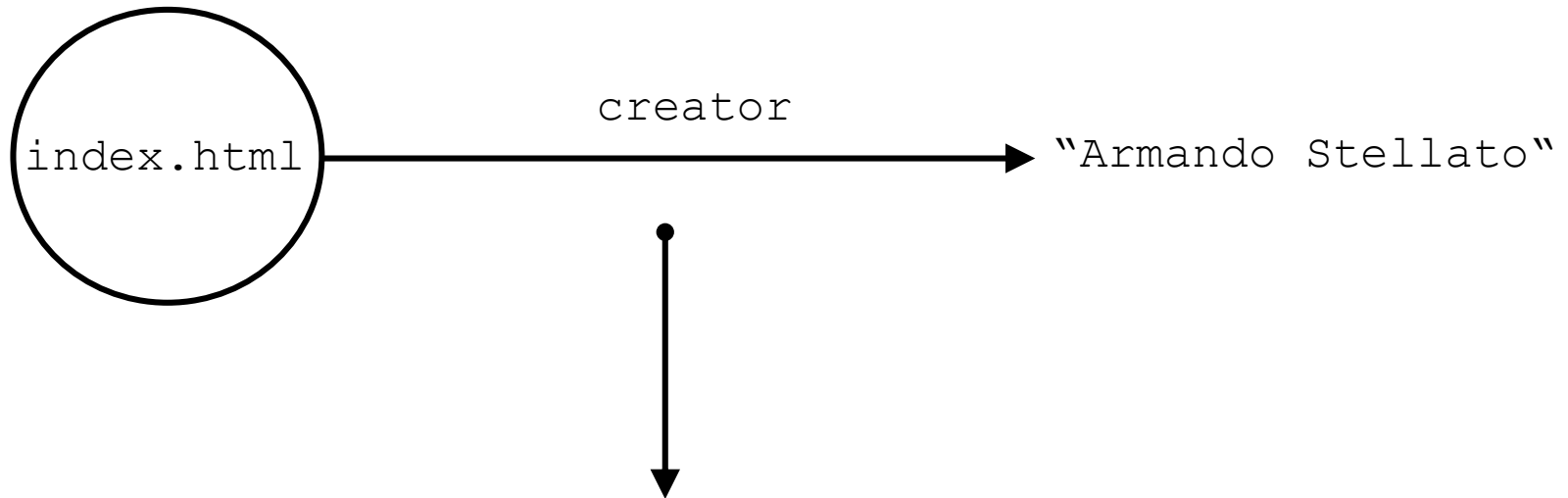
es:

```
type(loves, social_relationship)
```

```
loves(Tom, Mary)
```

- Esistono diverse sintassi per rappresentare dei grafi RDF:
 - N-Triples
 - La notazione più prossima alla forma astratta, una serie di triple del tipo soggetto-predicato-oggetto, identificate da URI
 - Notation 3 (N3)
 - Contiene numerose abbreviazioni sintattiche che agevolano la lettura mascherando la rigorosa struttura a triple che contraddistingue ogni documento RDF. Questa caratteristica unita a forme sintattiche elementari molto semplici (diverse dal pesante XML) ne fa il tipo di serializzazione più compatta tra quelli esistenti. Come XML, fa utilizzo dei namespace per aumentare la modularità.
 - RDF/XML
 - È un tipo di notazione completamente XML compliant. Come per N3, la struttura a triple può essere mascherata attraverso costrutti sintattici più complessi.

RDF/XML : Un piccolo esempio

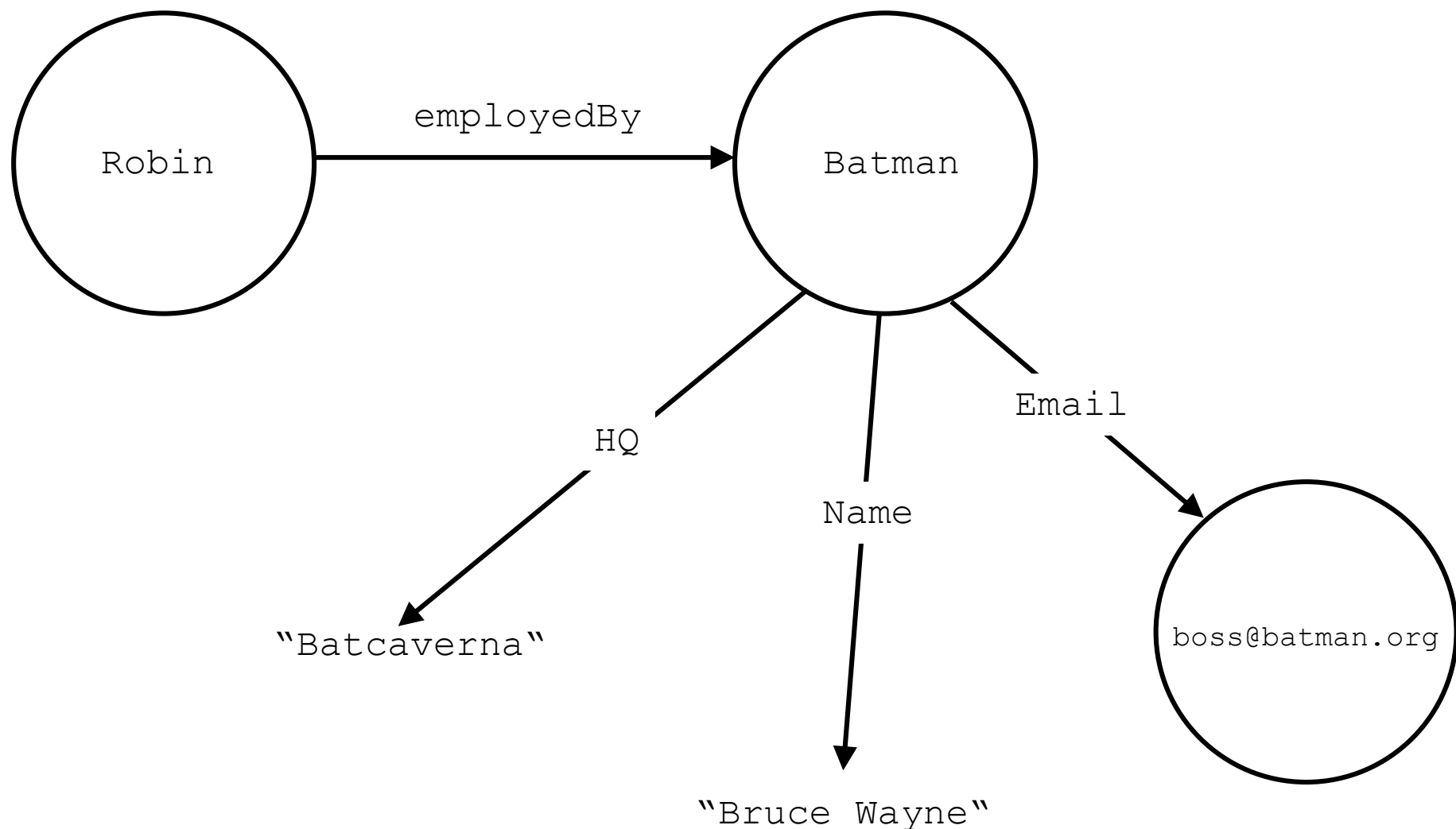


```
<rdf:rdf xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc = "http://purl.oclc.org/DC">

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

</rdf:rdf>
```

RDF : Un esempio più complesso...



E la relativa serializzazione RDF/XML...

```
<rdf:rdf
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:mySchema = "http://www.Batman.org/mySchema/">

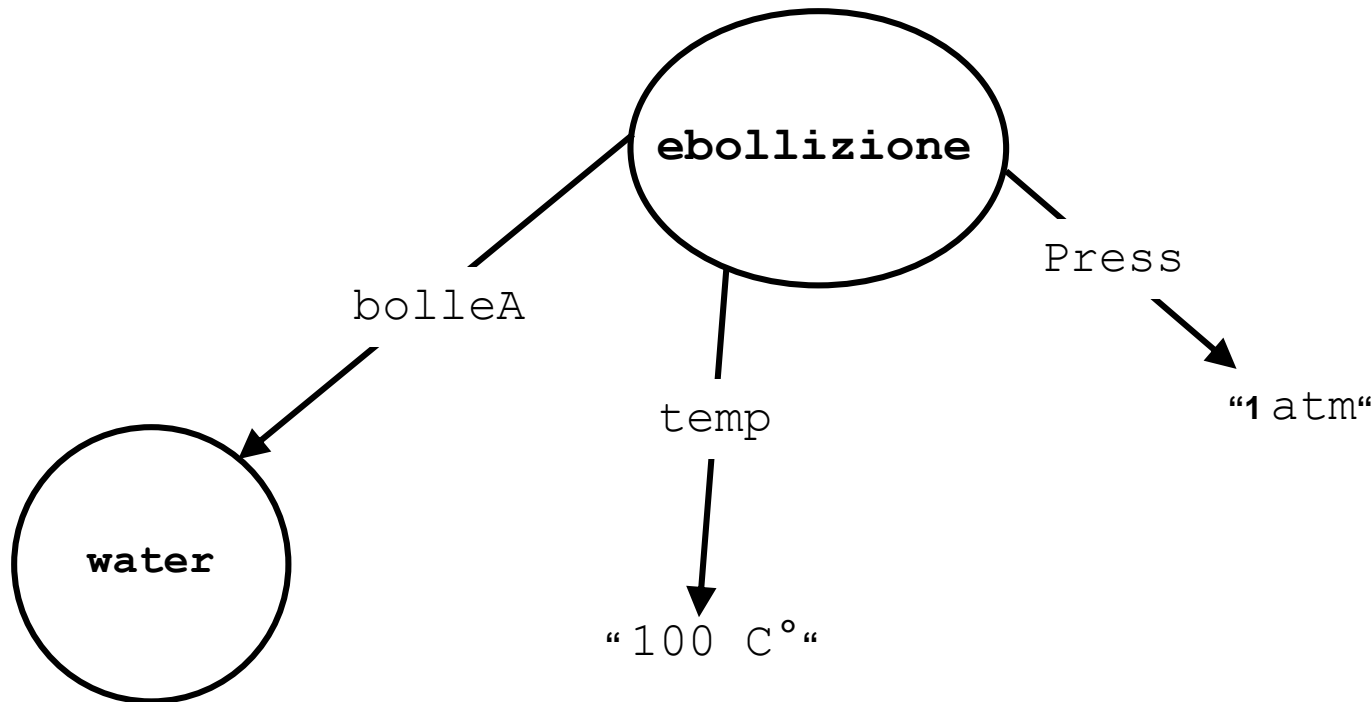
  <rdf:description rdf:about = "http://www.Batman.org/Robin/">
    <mySchema:employedBy rdf:resource = "#Batman"/>
  </rdf:description>

  <rdf:description 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>
```

Relazioni n-arie

- La sola presenza di sole relazioni binarie esplicite non impedisce la rappresentazione di relazioni di arità arbitraria.
- Es: `bolleA(water, 100C, 1atm)`



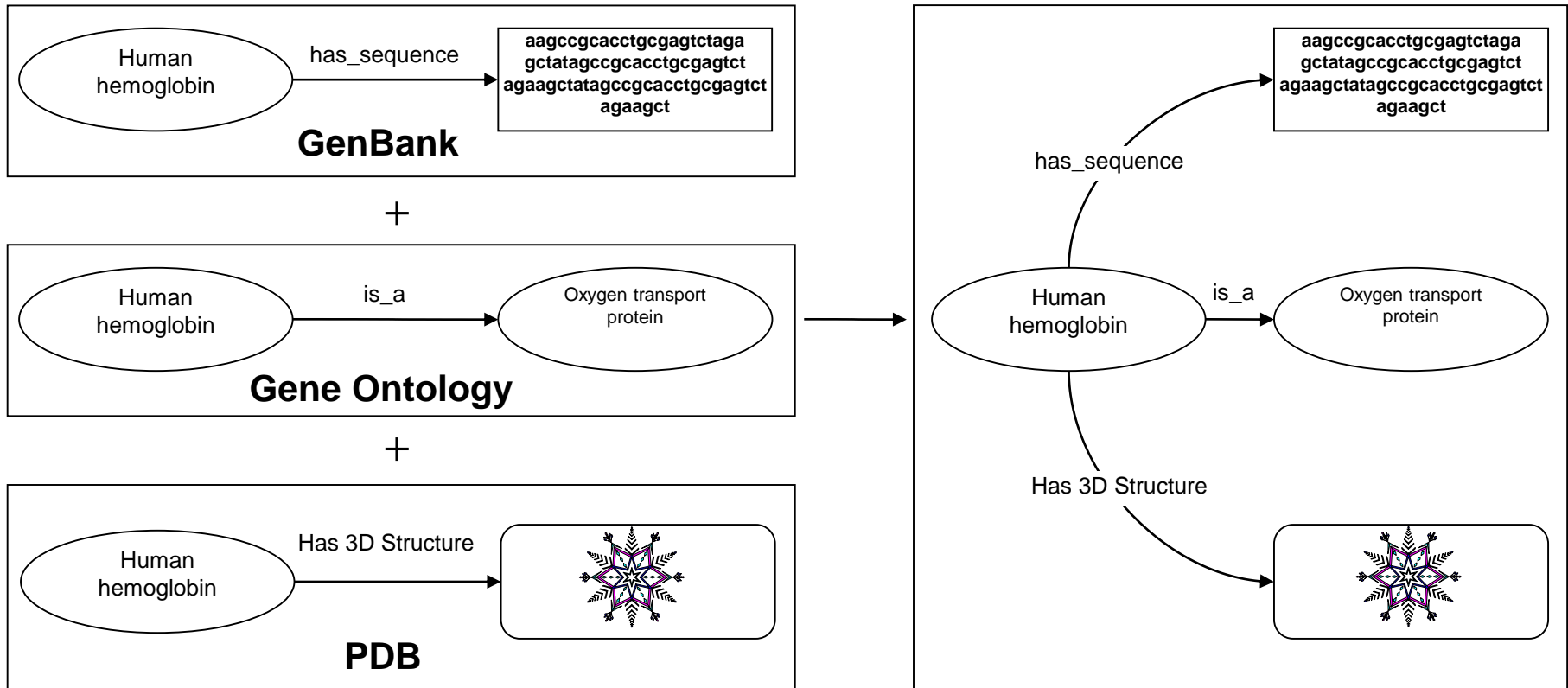
- I detrattori di RDF hanno spesso sostenuto la sua inutilità rispetto a quanto già fornito da XML e XML Schema

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

- RDF aggiunge però, pur attraverso un modello estremamente semplice, una semantica condivisa per interpretare i dati
- Vantaggi:
 - Minore ambiguità sintattica (e.g. attributo o elemento annidato?): una volta definito un grafo RDF, la sua serializzazione XML è univoca
 - La presenza di semantica esplicita implica un più immediato processo di integrazione di diverse risorse

XML vs RDF (2)

- Esempio: unione di due frammenti RDF



XML vs RDF (3)

- Esempio: unione di due frammenti XML

```
<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

+

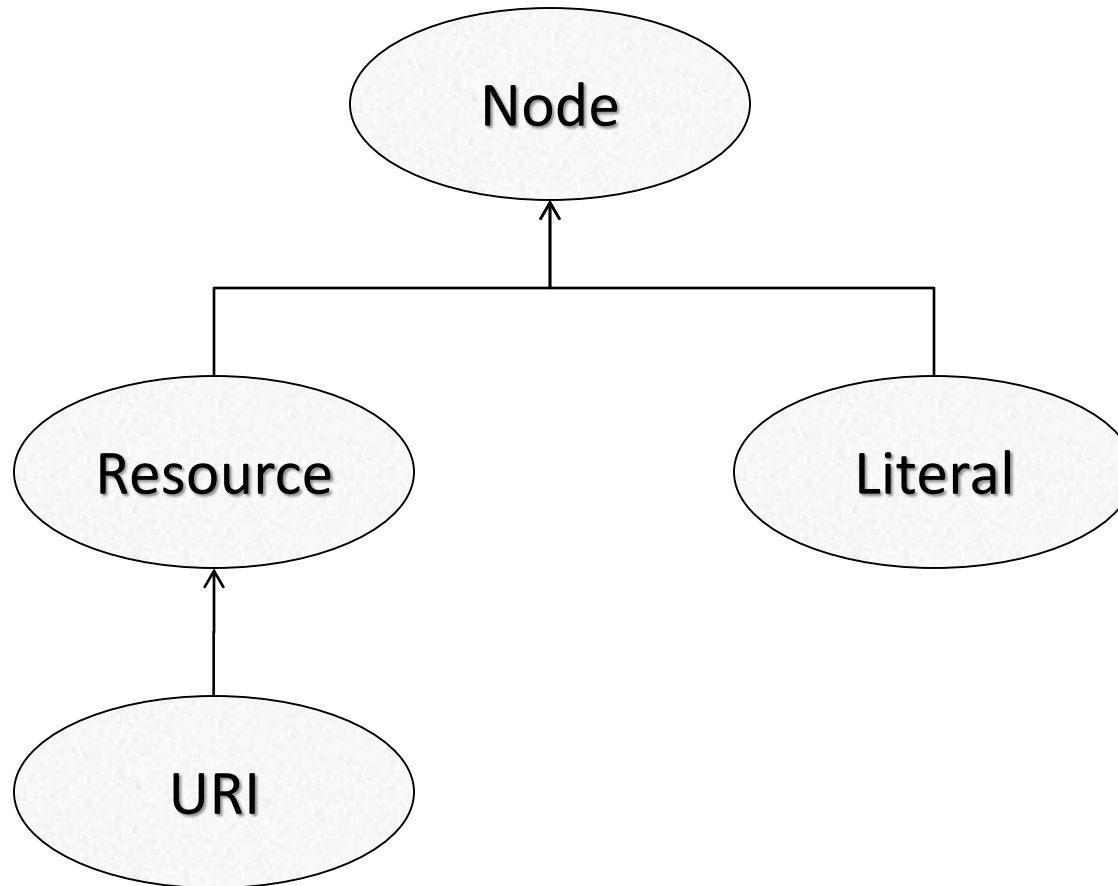
```
<structure><protein name="hh434"/><atom x="-30" y="40"  
elem="H"/>...</structure>
```

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.

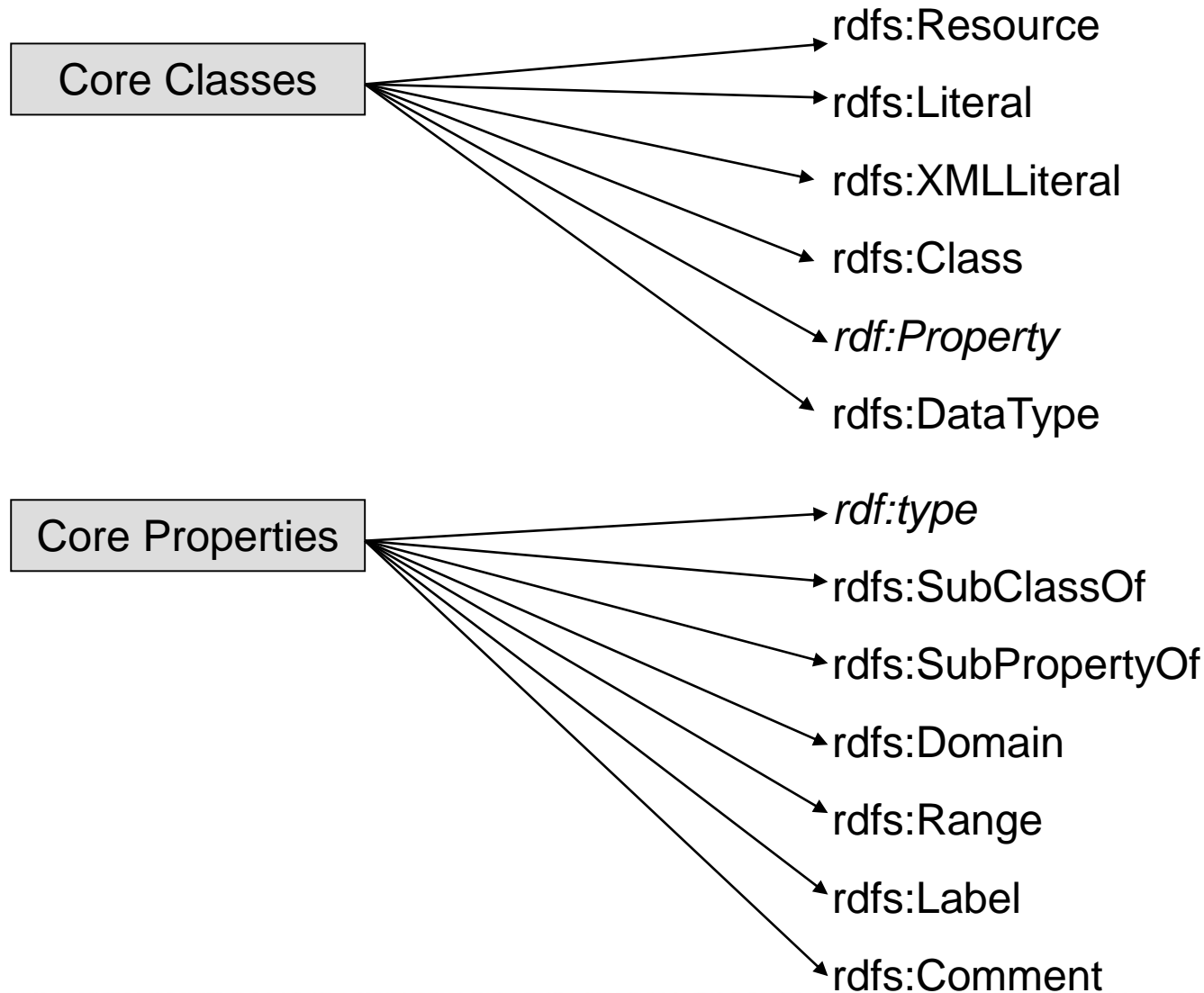
- Al di là dei costrutti complessi che possiamo definire per descrivere dei pattern di rappresentazione evoluti, il modello RDF è completamente basato su **triple**.
- Esistono dei vincoli sulla struttura di una tripla RDF
 - `rdf_triple(Resource, URI, Node)`
 - Il ***subject*** di una tripla è sempre una *risorsa*, può essere o meno identificata da un URI, ma non può essere un tipo di dati primitivo
 - Il ***predicate*** di una tripla è una risorsa sempre caratterizzata da un URI
 - L'***object*** non ha particolari restrizioni, e può contenere sia risorse che tipi primitivi (literals)

Gerarchia dei tipi di nodo in un grafo RDF



- RDF manca di:
 - Possibilità di specificare differenti livelli di astrazione
 - Organizzazione delle risorse in categorie esplicite
 - Restrizioni sulle proprietà
- RDFS estende RDF con un vocabolario per definire schemi, e.g.:
 - Class, Property
 - type, subclassOf, subPropertyOf
 - range, domain
- Con tale estensione, RDF(S) può essere considerato a tutti gli effetti un linguaggio per la rappresentazione della conoscenza

RDFS : Vocabolario principale

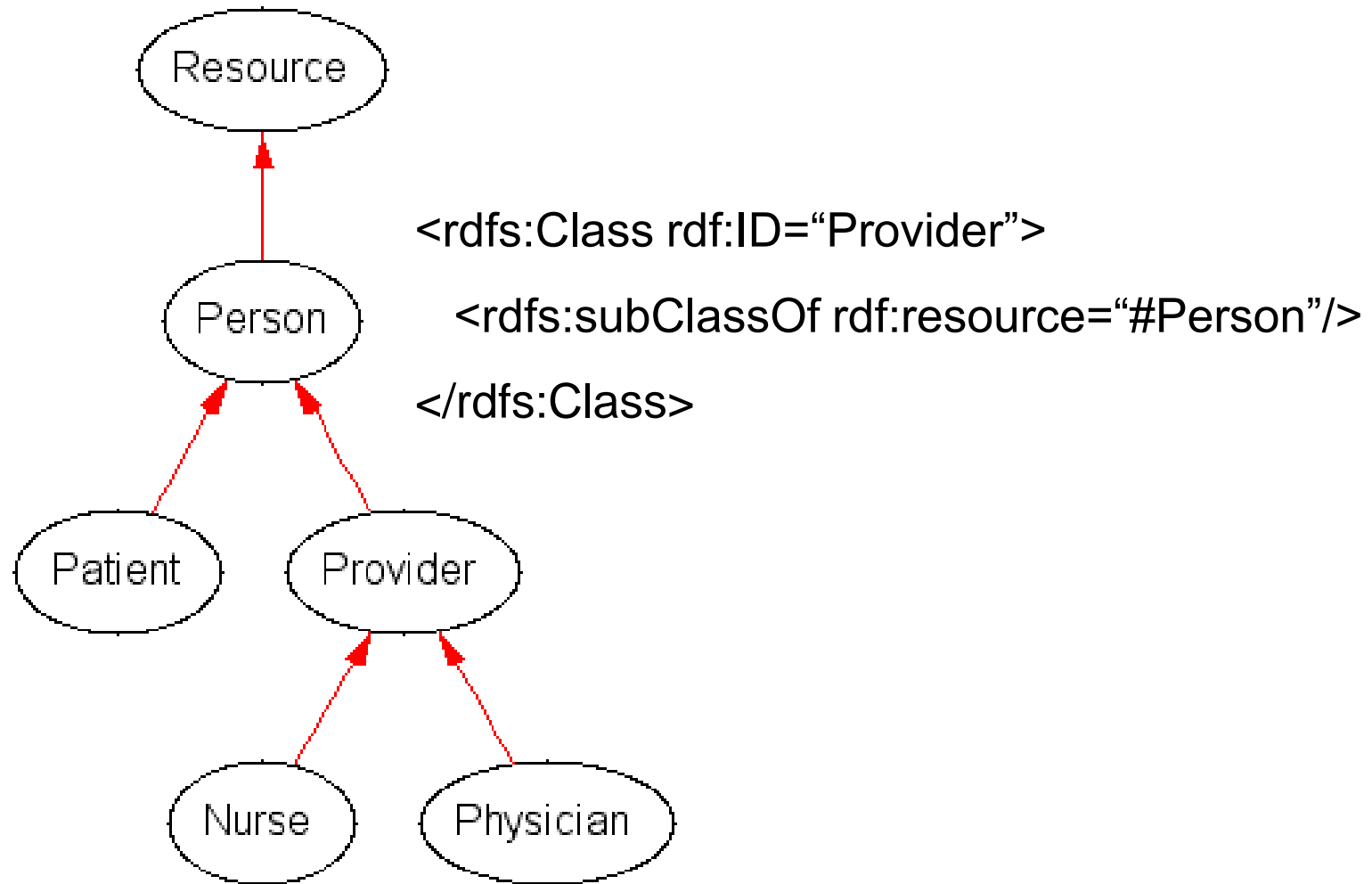


- ***rdfs:Resource*** – Tutte le cose descritte da espressioni RDF sono risorse e sono considerate *istanze* della classe *rdfs:Resource*
- ***rdfs:Class*** – rappresenta il generico concetto di tipo o categoria. Può quindi essere definito per rappresentare qualsiasi cosa, e.g. pagine Web, persone, tipi di documento...
- ***rdf:type*** – Questo elemento esiste già nel vocabolario RDF, ma in RDFS lega delle risorse alle categorie (Classi) cui appartengono. È analogo al costrutto instance-of dell'OO *design*

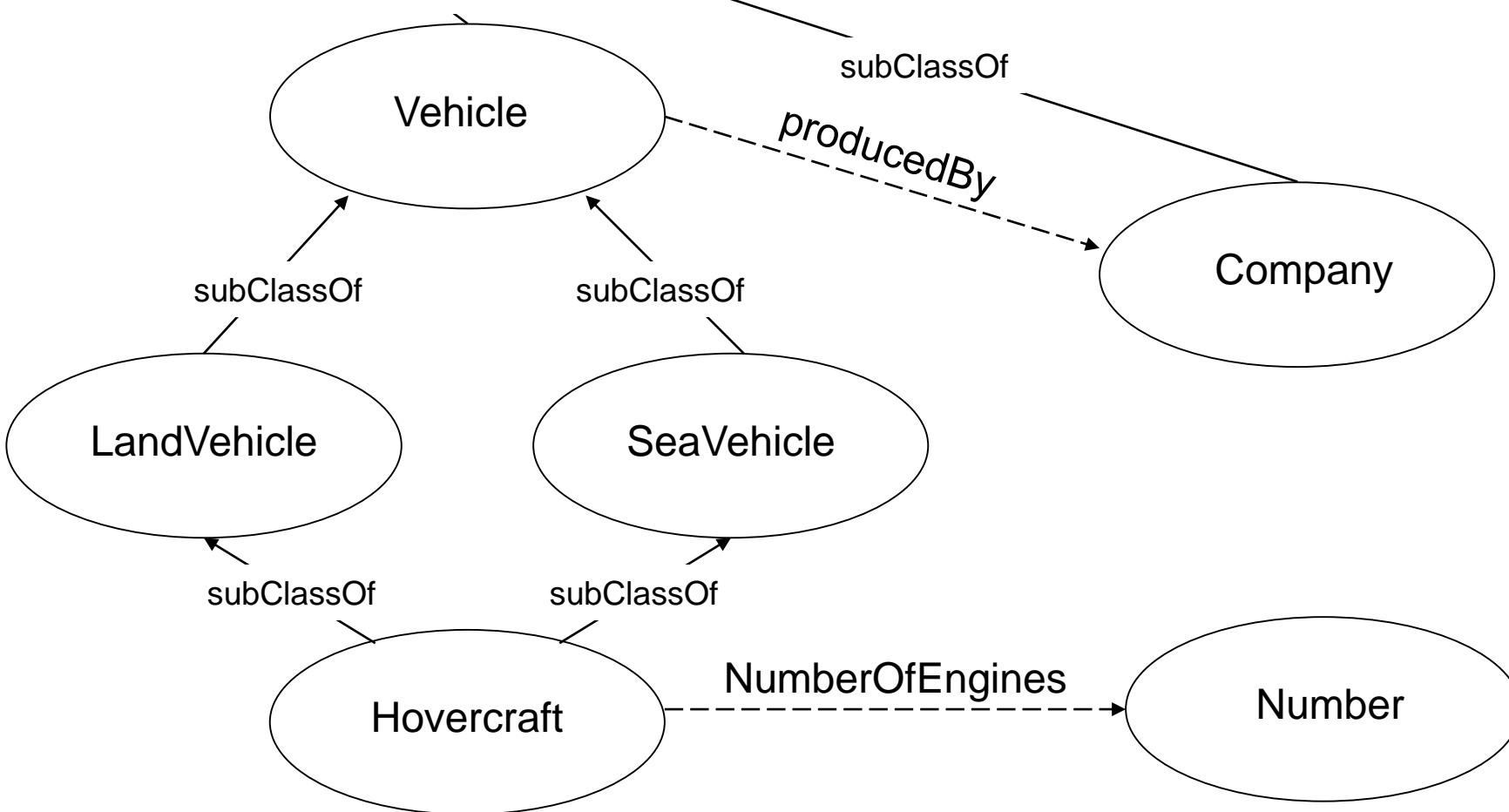
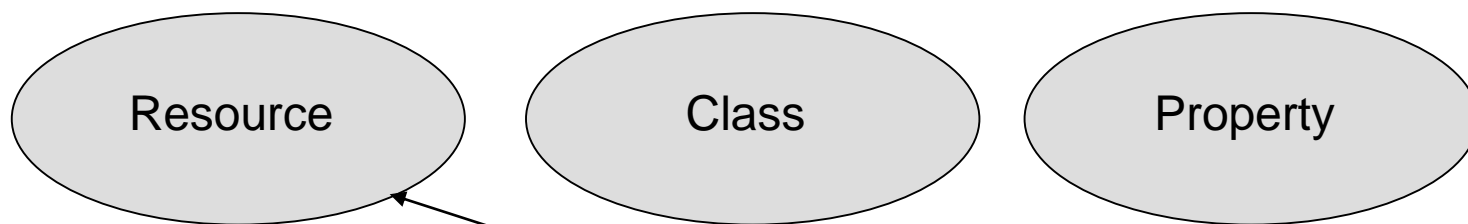
- ***rdf:Property*** – anche questo proviene dal vocabolario RDF, e rappresenta il sottoinsieme di tutte le risorse RDF che sono proprietà
- ***rdfs:subClassOf*** – questa proprietà definisce una relazione di super/sottoinsieme tra classi. Questa proprietà è transitiva
- ***rdfs:SubPropertyOf*** – Questa proprietà è usata per indicare che una proprietà è una specializzazione di un'altra proprietà

- ***rdfs:Range*** – definisce da a quale classe appartengono i valori che una determinata proprietà può assumere
- ***rdfs:Domain*** – specifica che ogni risorsa che possiede una certa proprietà è istanza di una o più classi
- ***Annotation properties*** – non giocano alcun ruolo nella semantica del linguaggio ma forniscono un utile mezzo per commentare un repository di dati
 - *rdfs:comment*: la proprietà di comment più generale. In genere fornisce una descrizione in linguaggio naturale della risorsa che la contiene
 - *rdfs:label*: fornisce nomi alternativi per indicare una risorsa. Con l'attributo `xml:lang` è possibile specificare il linguaggio in cui tale commento è inserito
 - *rdfs:seeAlso*: contiene un puntatore ad un'altra risorsa che contiene ulteriori informazioni circa il soggetto di tale proprietà
 - *rdfs:isDefinedBy*: è una sottoproprietà di *rdfs:seeAlso* e indica la risorsa che definisce la risorsa soggetto,

Esempio di Schema RDF



RDF-Schema : Esempio di Schema



RDF-Schema : Esempio di Schema

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

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

<rdf:description 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>
```

...Continua dalla pagine precedente...

```
<rdf:description 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 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 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>
```


Limiti di RDFS

- RDFS **troppo debole** per descrivere risorse con sufficiente dettaglio
 - Nessun **vincolo di range e/o domain contestualizzato**
 - Non si può limitare il range di `hasChild` a `person` quando è applicato a persone e a `elephant` quando applicato a elefanti
 - Nessun vincolo di **esistenza/cardinalità**
 - Non si può dire che tutte le *istanze* di `person` hanno una madre che è anche una `person`, o che le persone hanno esattamente 2 genitori
 - Nessuna proprietà **transitiva, inversa or simmetrica**
 - Non si può affermare che `isPartOf` è una proprietà transitiva, che `hasPart` è l'inversa di `isPartOf` o che `touches` è simmetrica
 - ...

- Two languages developed by extending (part of) RDF
 - **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** language based on DAML+OIL
 - OWL language now a W3C **Proposed Recommendation**

- La semantica di RDFS/OWL, contrariamente a precedenti approcci nella rappresentazione della conoscenza, come i frame, basati su controllo di vincoli (*constraint checking*), è di tipo **inferenziale**.
- Data quindi una *teoria del mondo*, invece di verificare esclusivamente che gli oggetti della nostra base di conoscenza (*instance data*) soddisfino i vincoli imposti da tale teoria, sarà possibile aggiungere (*inferire*) nuova informazione (in modo rigorosamente **monotono**, cioè senza contraddire asserzioni precedenti) alla teoria e/o alla descrizione degli oggetti per far sì che questi siano ancora un modello per la *teoria*
- Caratteristiche della semantica RDFS/OWL:
 - **Open World Assumption** (OWA)
 - **No Unique Name Assumption**

- La CWA (**Closed World Assumption**), tipica delle basi di dati tradizionali e la NF (**negation-as-failure**) che caratterizza linguaggi di programmazione logica come il prolog, sono intimamente legate
- Data la formula atomica ground A , la CWA ci dice che:
 - Se una base di conoscenza KB non ha come conseguenza logica A , allora A è falsa
- Data la formula atomica ground A , la NF ci dice che:
 - Se non è possibile dimostrare A in una base di conoscenza KB, allora A è falsa in quella KB

- CWA e NF sono naturalmente connesse ad una visione non-monotonica del mondo
- Es: abbiamo un DB con il solo fatto: donna(MarilynMonroe) ed eseguo delle query in prolog
 - Query: ?- donna(MarilynMonroe).
 - Answer:yes
 - Query: ?- uomo(MarilynManson)
 - Answer:no (usando la CWA/NF).
 - Aggiorniamo quindi il DB con uomo(MarilynManson).
 - Query: ?- uomo(MarilynManson)
 - Answer:yes

- L'interpretazione di OWL, contrariamente a precedenti approcci nella rappresentazione della conoscenza, come i frame, basati su controllo di vincoli (*constraint checking*), è di tipo **inferenziale**.
- Qualche esempio:

```
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 .
```

La descrizione di myDoc non è incompleta anche se la mincard per author è 1, perchè l'autore potrebbe essere definito "somewhere else in the world" (Open World Assumption)

- L'interpretazione di OWL, contrariamente a precedenti approcci nella rappresentazione della conoscenza, come i frame, basati su controllo di vincoli (*constraint checking*), è di tipo **inferenziale**.
- Qualche esempio:

```
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 .
```

I due valori su for eg:copyrightHoder destano problemi? No, potrebbero esistere due nomi per indicare la stessa cosa, quindi si suppone che institute1 e institute2 denotino la stessa cosa

- L'interpretazione di OWL, contrariamente a precedenti approcci nella rappresentazione della conoscenza, come i frame, basati su controllo di vincoli (*constraint checking*), è di tipo **inferenziale**.
- Qualche esempio:

```
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 è inferita essere
ANCHE una Person (a causa
dell'asserzione di
equivalentClass sulla
restriction allValuesFrom
Person)

myDoc2 è un Document?
Non possiamo saperlo,
perché nel mondo potrebbero
esserci altri autori di questi
libro che non di tipo Person