



Università degli Studi di Roma “Tor Vergata”

RDF

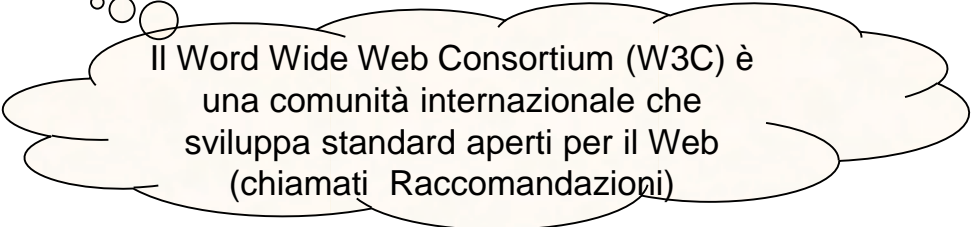
Resource Description Framework

Armando Stellato

stellato@uniroma2.it

Date importanti per RDF

1999 – RDF diviene una *raccomandazione del W3C*



Il Word Wide Web Consortium (W3C) è una comunità internazionale che sviluppa standard aperti per il Web (chiamati Raccomandazioni)

2004 – RDF 1.0

sei documenti (Primer, Concepts, Syntax, Semantics, Vocabulary, and Test Cases) rimpiazzano congiuntamente la specifica originale di RDF e RDF Schema (2000 Candidate Recommendation)

2014 – RDF 1.1 è l'aggiornamento più recente RDF

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

Il modello di dati RDF

Ordine e molteplicità delle triple
non sono rilevanti

Un *Grafo RDF* è un insieme di *triple*:

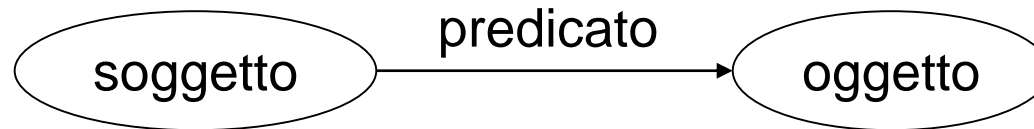
L'ordine dei
componenti di
una tripla è
significativo

Tripla := (soggetto, predicato, oggetto)

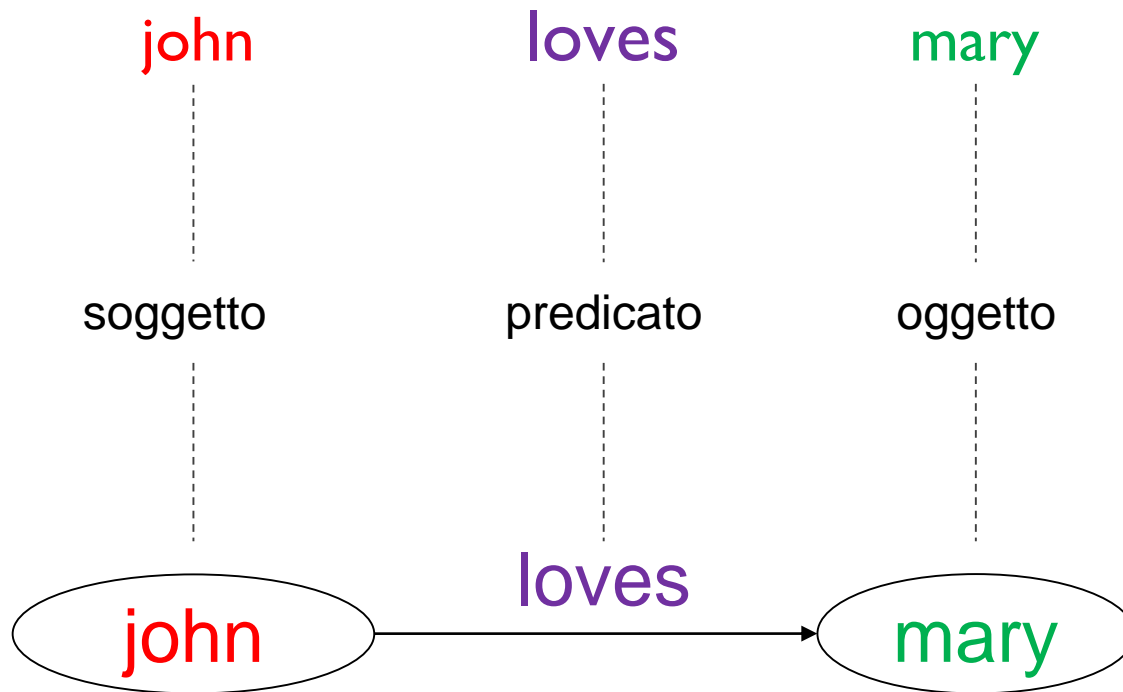
- **soggetto** è un risorsa
- **predicato** è una proprietà della risorsa soggetto
- **oggetto** è il valore della proprietà (può essere una risorsa descritta attraverso triple RDF)

Il modello di dati RDF

Un insieme di triple può essere visto come un *grafo diretto etichettato*, nel quale ogni tripla assume la seguente forma:



Il modello di dati RDF



Il modello di dati RDF

I *componenti di una tripla* possono essere:

- Un **IRI**¹ (Internationalized Resource Identifier) è una stringa UNICODE conforme a RFC 3987.
 - Esempi: `http://art.uniroma2.it/fiorelli` · `mailto:fiorelli@info.uniroma2.it` · `urn:lex:it:stato:legge:2003-09-21;456`
 - Estende URI (Uniform Resource Identifier) con il supporto ad altri alfabeti (*script*) oltre quello latino
- Un **literal**² è composto da:
 - *lexical form* (una stringa UNICODE),
 - l'IRI di un *datatype*,
 - se e solo se il datatype è `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString`, un *language tag* non vuoto (conforme a BCP 47)
- Un **blank node** (bnode)

¹ in RDF 1.0 c'erano invece gli RDF URI References
² in RDF 1.0 c'erano i plain literal senza datatype e, opzionalmente, con language tag

Ci sono alcuni vincoli sull'uso di IRI, blank node e literal in una tripla:

- Il **soggetto** può essere un IRI o un blank node
- Il **predicato** può essere un IRI
- L'**oggetto** può essere un IRI, blank node o literal

Il modello di dati RDF



Il modello di dati RDF: Perché gli IRI?

- *Vocabolario estendibile*
- *Ambito (scope) globale* (ogni occorrenza di un IRI deve denotare la stessa risorsa; altrimenti, si dice che c'è una collisione di IRI)
 - Ognuno può pertanto menzionare risorse definite da altre parti
- Le regole di proprietà definiscono un *proprietario di un IRI*, che può stabilirne il *referente*:
 - Definendo una specifica per l'IRI: può essere un documento RDF!
 - Se l'IRI è dereferenceable (es. `http://` or `https://`), è possibile comunicare la sua specifica facendo sì che l'IRI punti a quella specifica (à la linked data)

- La versione 1.1 ha introdotto la nozione di *dataset* nella specifica RDF (questa nozione esisteva già nella specifica SPARQL).
- Un **dataset** consiste di:
 - Un *default graph* (non ha un nome)
 - Zero o più *named graph*. Ogni *named graph* è una coppia formata da un IRI o *blank node*¹ (il nome del grafo) e da un grafo RDF

¹ in realtà, SPARQL vieta l'uso di *blank nome* come nomi di grafo

Ciascun IRI o literal *denota* qualcosa (una risorsa):

- Il *referente* di un IRI
- Il *literal value* di un literal

Blank nodes sono discussi successivamente...

Una tripla rappresenta un'*asserzione* (statement)

- Una certa relazione (denotata dal predicato) vale tra le risorse denotate dal soggetto e dall'oggetto

Il significato di un grafo RDF è quindi la congiunzione logica degli statement associati alle triple

RDF: potere espressivo

- 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)
```

RDF: blank node

- Una tripla può anche contenere blank node (bnode), che si comportano come variabili quantificate esistenzialmente:

`love (?x, Mary)`

ci dice che “qualcuno ama Mary”, o, più precisamente, che qualcosa ama Mary

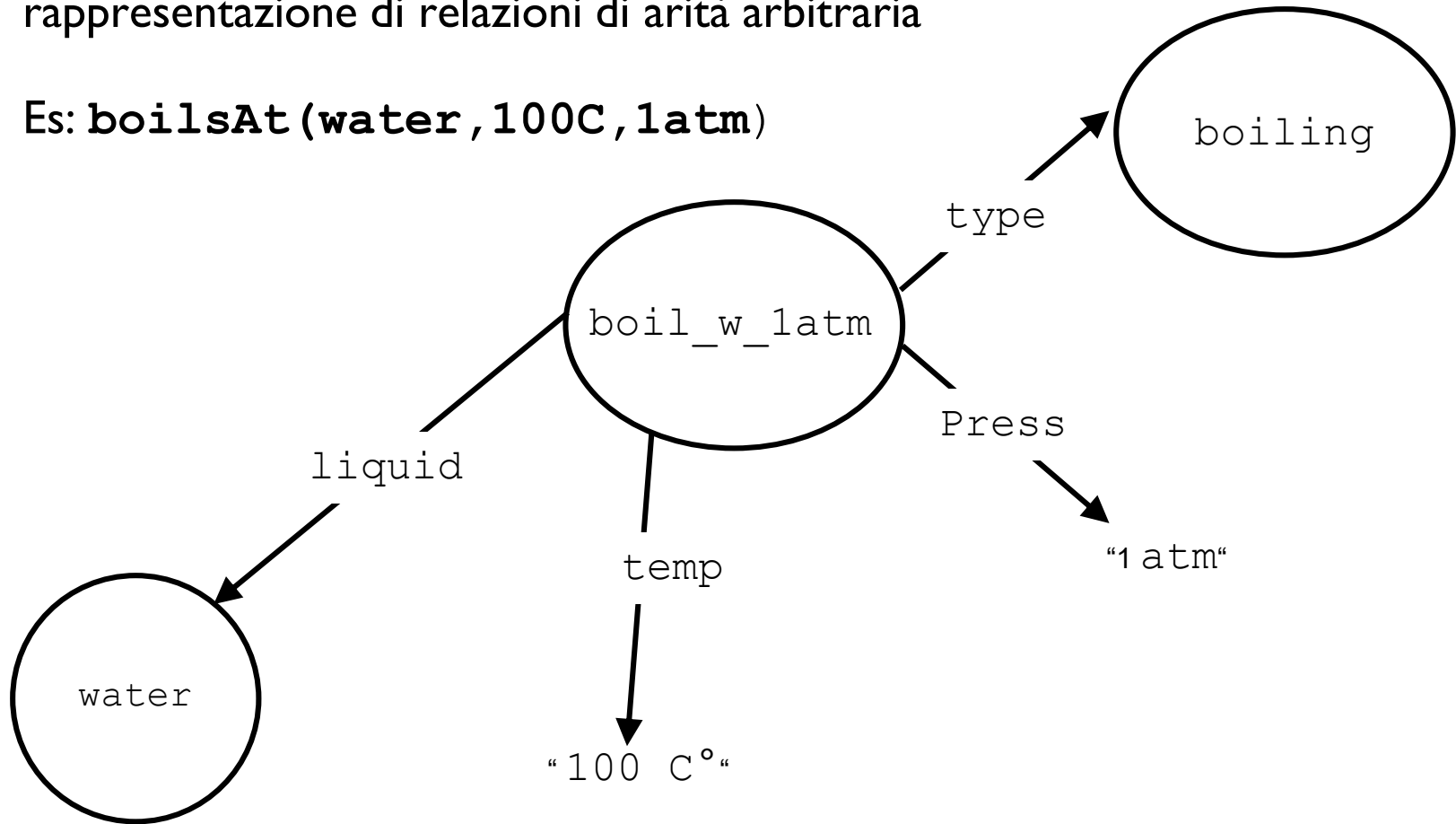
- La combinazione di diverse asserzioni, tramite l’unificazione delle variabili, ci permette di esprimere conoscenza complessa, ma non completamente istanziata:

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

ci dice che “Mary è amata da un maschio”

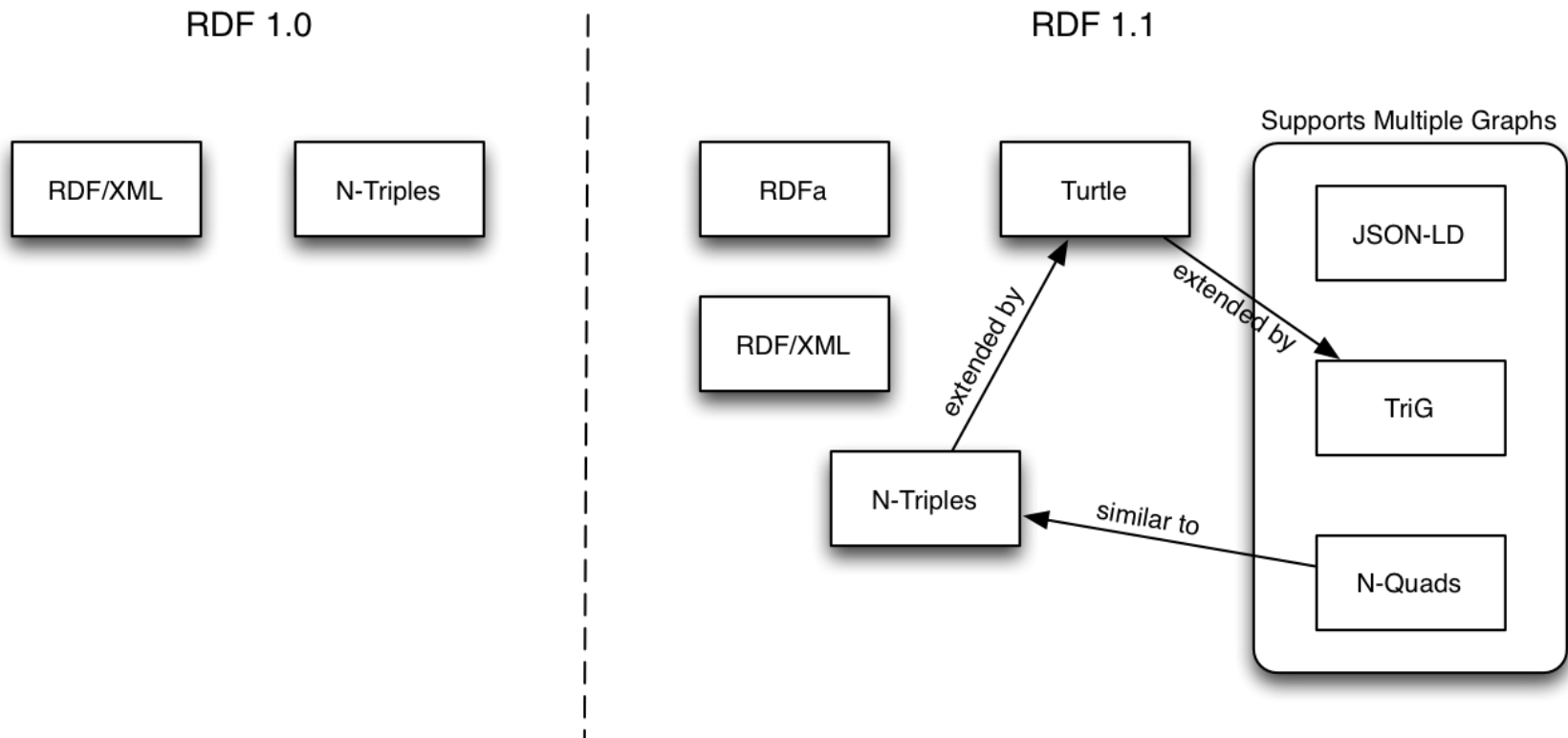
Relazione n-arie

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



RDF: Sintassi Concrete

RDF/XML non è più l'unico formato raccomandato: sono stati standardizzati una serie di nuovi formati di serializzazione



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

RDF: Sintassi Concrete

Ci sono diverse sintassi concrete per RDF. Ecco alcune di esse:

– RDF/XML

- È una notazione per RDF pienamente compatibile con (e basato su) XML. Al pari di N3, la struttura a triple può essere mascherata attraverso costrutti sintattici più complessi. È stata originariamente una delle serializzazioni RDF più diffuse

– N-Triples

- La notazione più prossima alla forma astratta di RDF, consistendo in una serie di triple soggetto-predicato-oggetto

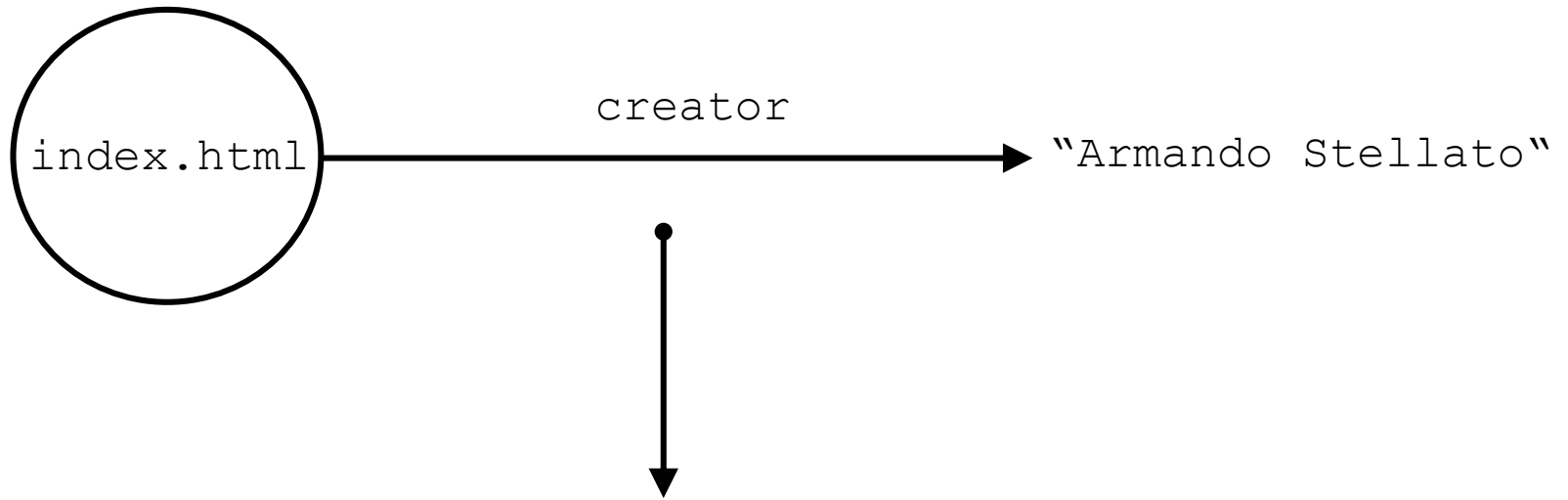
– Notation 3 (N3)

- Contiene numerose abbreviazioni sintattiche che agevolano la lettura, mascherando la rigorosa struttura a triple caratterizzante RDF. Questa caratteristica unita a forme sintattiche elementari molto semplici (diverse dal pesante XML) fa di N3 la serializzazione più compatta. Come XML, N3 usa namespaces per abbreviare gli IRI. Notation 3 ha un potere espressivo che va oltre RDF.

– Turtle

- Pensato come un sottoinsieme di N3 (e un soprainsieme di N-Triples) che può soltanto serializzare grafi RDF validi. Oggigiorno è considerato un'alternativa molto migliore a RDF/XML

RDF/XML : un piccolo esempio



```

<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>

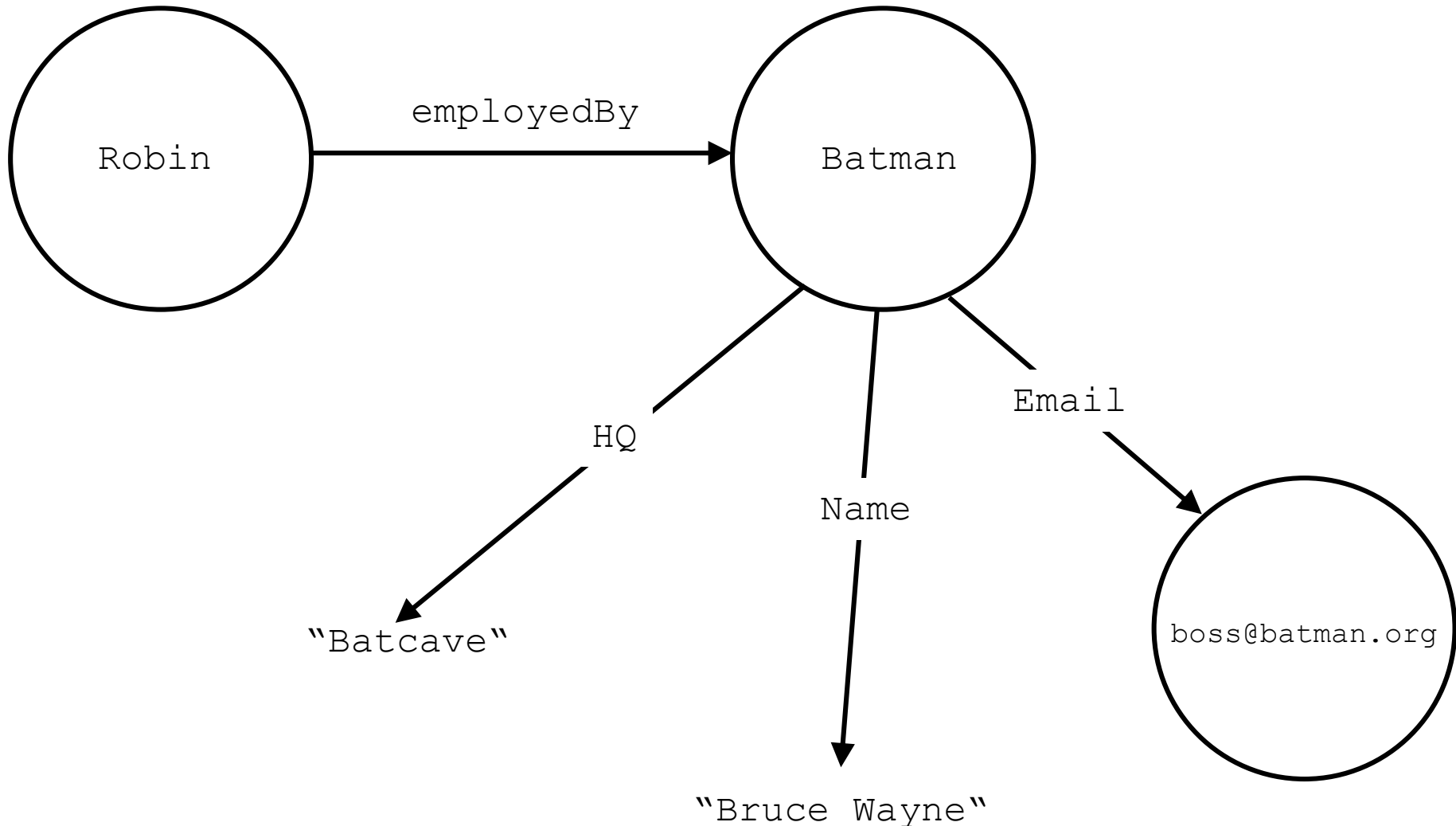
```

```

</rdf:RDF>

```

RDF : un esempio leggermente più complesso...



RDF : un esempio leggermente più complesso...

E la relativa serializzazione RDF/XML

```
<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>
```

...e la relativa serializzazione Turtle

```
@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> .
```

Il dataset seguente ha soltanto un named graph

Le parentesi graffe possono essere omesse nel caso del 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> .
}
```

La keyword GRAPH è opzionale per i named graph

Etichette uguali in diversi named graph identificano lo stesso bnode

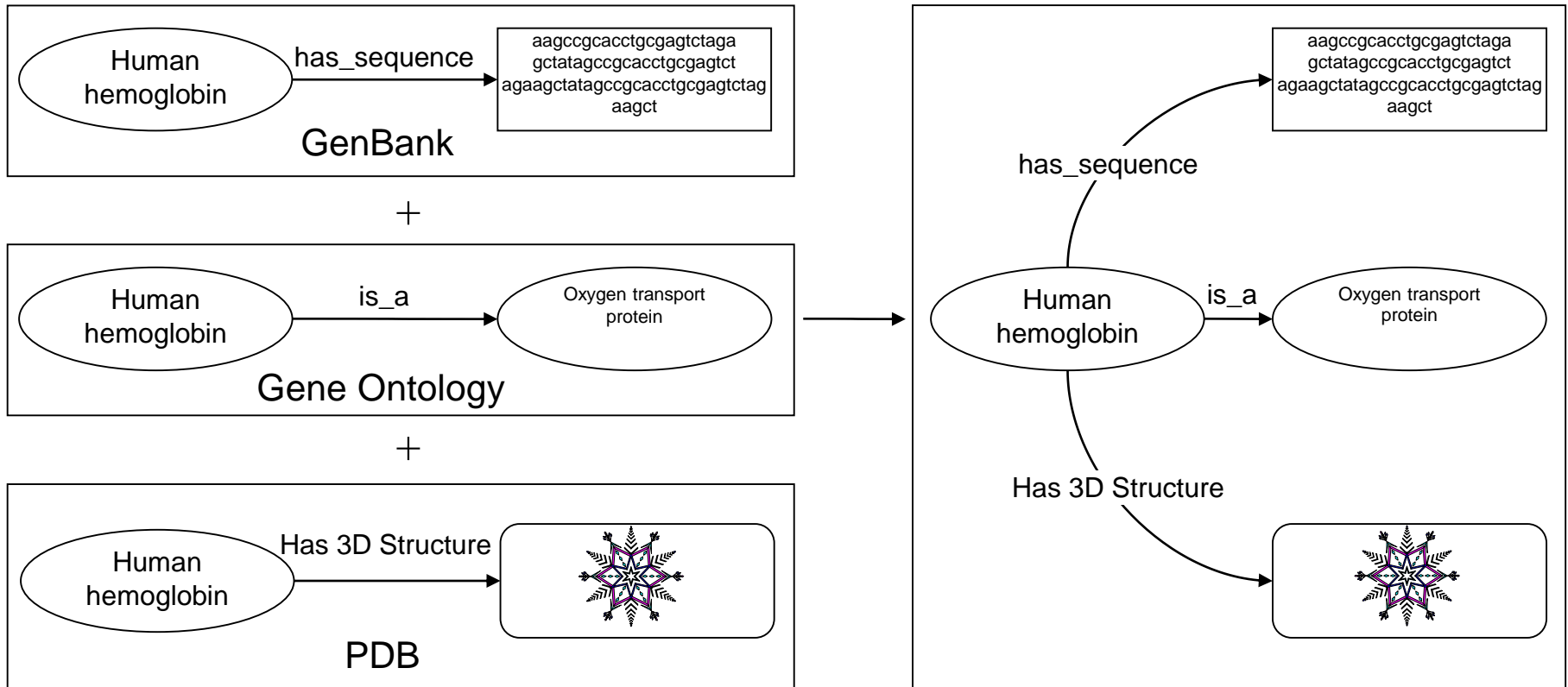
- I detrattori di RDF hanno spesso sostenuto la sua inutilità rispetto a quanto già fornito da XML e XML Schema (e altri, più recenti, linguaggi di schemi per XML)

RDF è visto da alcuni come una tecnologia eccessivamente complessa, che prova a risolvere un problema che XML e HTTP già risolvono (Van Dijck, 2003)

- RDF aggiunge però, pur attraverso un modello estremamente semplice, una semantica condivisa per interpretare i dati
- Vantaggi:
 - Nessuna variazione sintattica (es. attributo o elemento annidato?): la serializzazione XML di un grafo RDF non è unica, tuttavia... a chi importano le diverse serializzazioni, se tutte possono essere ricondotte univocamente allo stesso grafo RDF che le ha originate?
 - La semantica esplicita permette una migliore e più immediata integrazione di differenti risorse distribuite

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

+

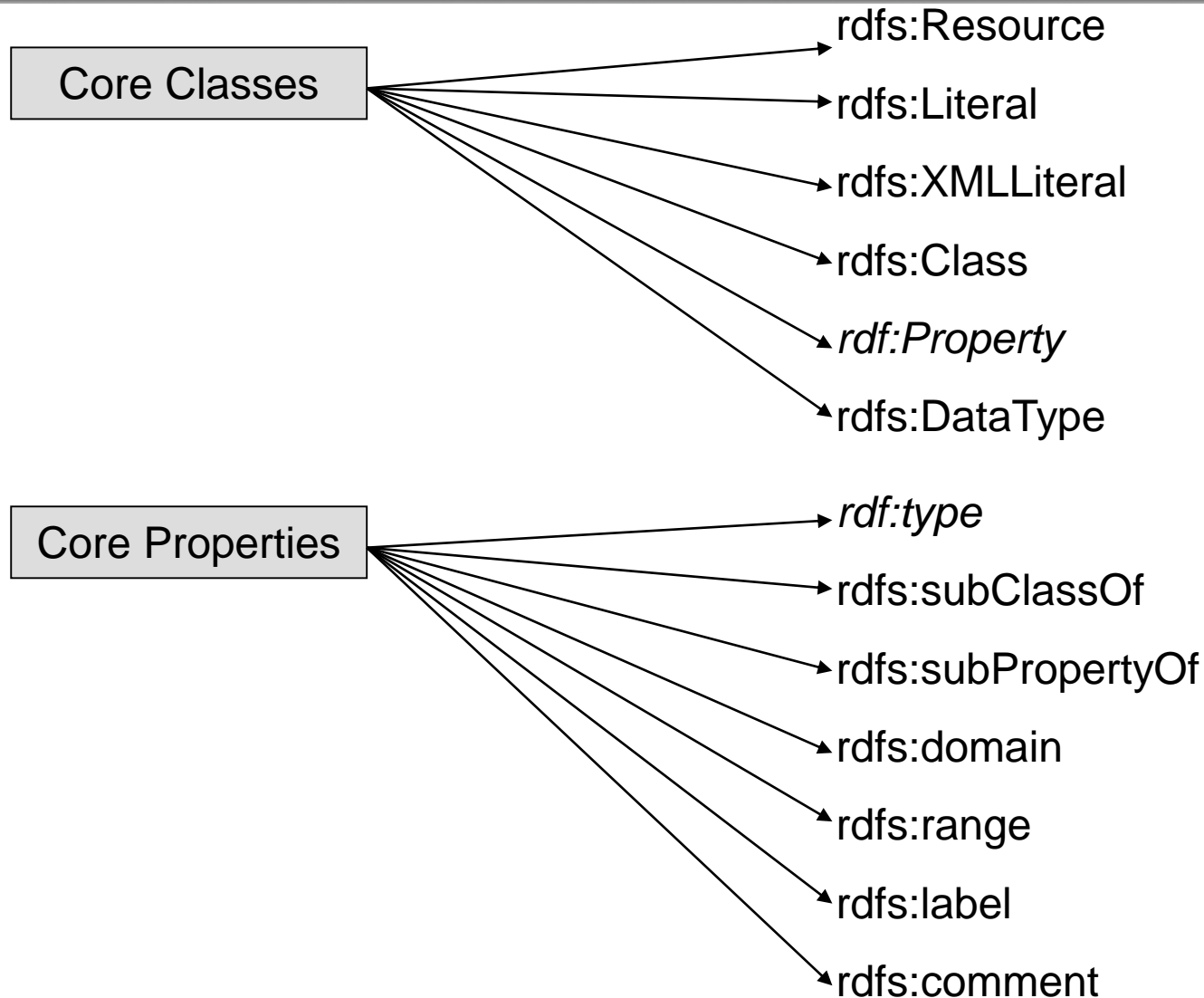
```
<structure><protein name="hh434"/><atom x="-30" y="40"  
elem="H"/>...</structure>
```

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.

- RDF (così come è) manca della possibilità di:
 - specificare diversi livelli di astrazione
 - organizzare le risorse in categorie esplicite
 - definire restrizioni sulle proprietà
- RDFS estende RDF con un vocabolario per definire schemi, es.:
 - Class, Property
 - type, subclassOf, subPropertyOf
 - range, domain
- con questa estensione, RDF(S) può essere considerato a tutti gli effetti un linguaggio per la rappresentazione della conoscenza

RDFS : Vocabolario principale



RDFS : Vocabolario principale

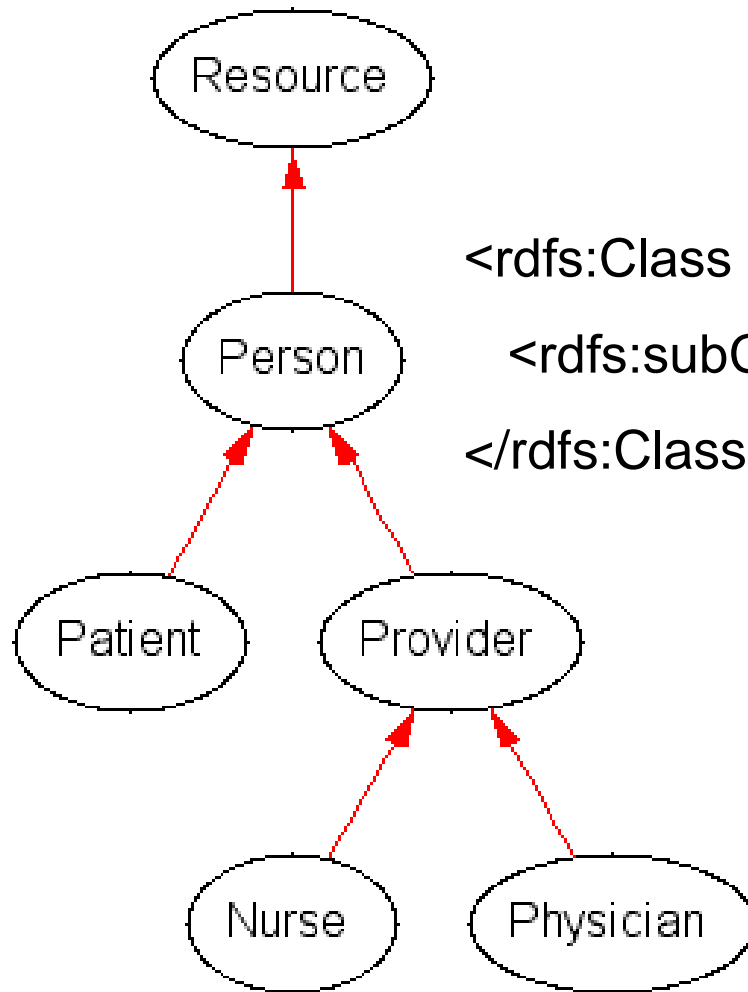
- ***rdfs:Resource*** – Tutte le cose descritte attraverso espressioni RDF sono risorse e sono considerate *istanze* della classe *rdfs:Resource*
- ***rdfs:Class*** – rappresenta il generico concetto di tipo o categoria. Può essere usato per definire ogni tipo di cosa, e.g. pagine web, persone, documenti, tipi di documento...
- ***rdf:type*** – Questo elemento esiste già nel vocabolario RDF, ma in RDFS lega le risorse alle categorie (classi) cui appartengono. È analogo al costruito *instance-of* del *design orientato agli oggetti (OO)*

- ***rdf:Property*** – anche questa risorsa viene dal vocabolario RDF, e rappresenta il sottoinsieme di tutte le risorse RDF che sono proprietà
- ***rdfs:subClassOf*** – questa proprietà definisce una relazione di sopra/sottoinsieme tra classi. È una proprietà transitiva (si noti che la caratteristica di essere transitiva viene dalla semantica del linguaggio RDFS ed è hard-wired nella proprietà *subClassOf*; non può essere assegnata ad alcuna proprietà)
- ***rdfs:subPropertyOf*** – Questa proprietà è usata per indicare che una proprietà è una specializzazione di un'altra proprietà

RDFS : Vocabolario principale

- ***rdfs:range*** – dichiara che tutti i valori per una certa proprietà appartengono ad una data classe
- ***rdfs:domain*** – dichiara che tutte le risorse caratterizzate da una certa proprietà appartengono ad una data classe
- ***Annotation properties*** – non giocano alcun ruolo nella semantica del linguaggio ma forniscono un utile mezzo per commentare/documentare un repository di dati
 - *rdfs:comment*: il modo più generale per commentare qualcosa. In genere fornisce una definizione in linguaggio naturale della risorsa che la contiene
 - *rdfs:label*: fornisce termini per descrivere una risorsa.
 - *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 punta alla risorsa che descrive la proprietà soggetto

RDFS: Schema di esempio

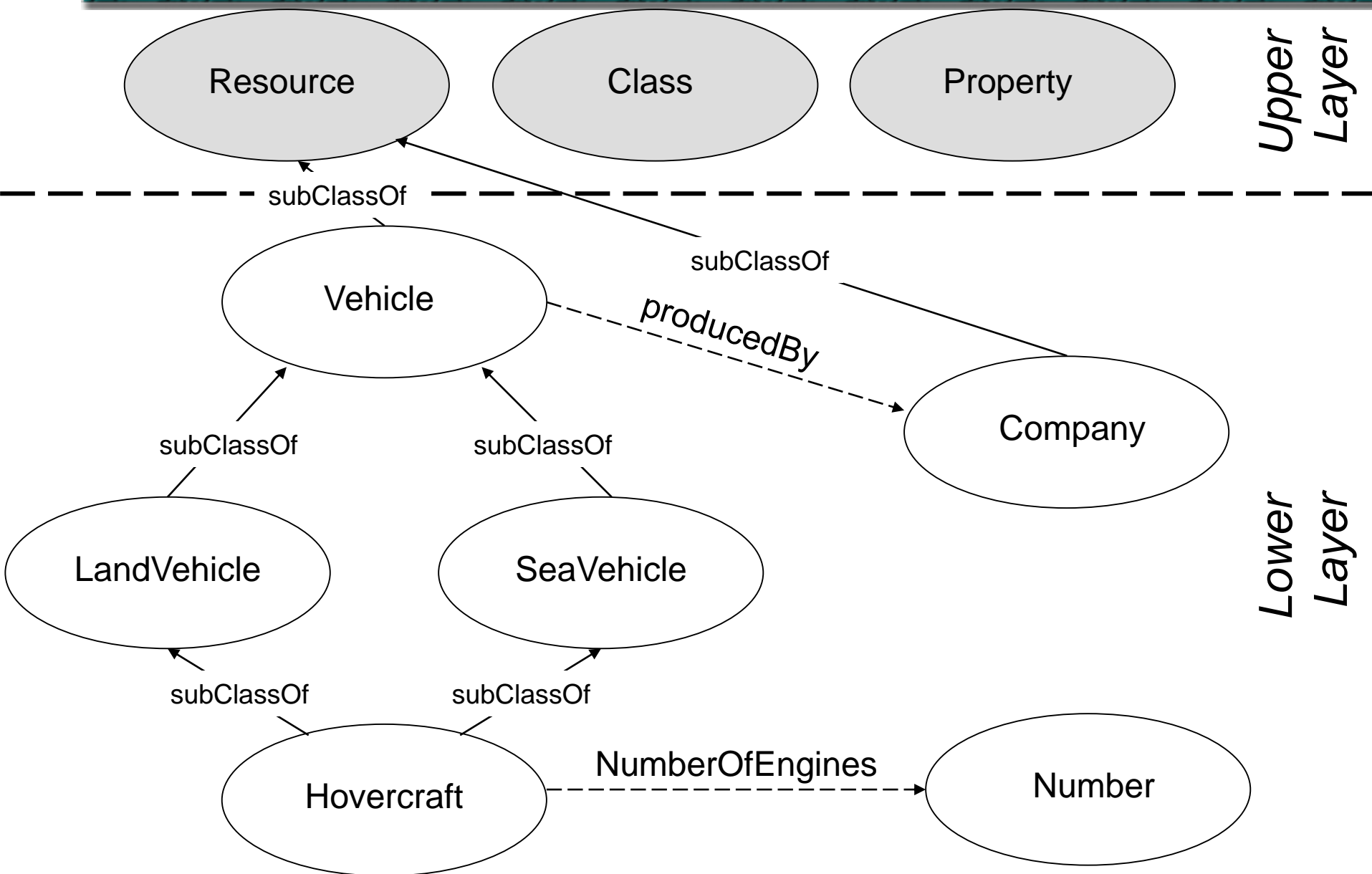


```
<rdfs:Class rdf:ID="Provider">
```

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

```
</rdfs:Class>
```

RDFS: Schema di esempio



RDFS: Schema di esempio

```
@prefix : <http://...> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:Vehicle a rdfs:Class ;
  rdfs:subClassOf rdf:Resource .

:LandVehicle a rdfs:Class ;
  rdfs:subClassOf :Vehicle .

:SeaVehicle a rdfs:Class ;
  rdfs:subClassOf :#Vehicle .

:Hovercraft a rdfs:Class ;
  rdfs:subClassOf :LandVehicle , SeaVehicle .

:Company a rdfs:Class ;
  rdfs:subClassOf rdf:Resource .

:producedBy a rdf:Property ;
  rdfs:domain :Vehicle ;
  rdfs:range :Company ;
  rdfs:label "Vehicle Producer"@en .

:numberOfEngines a rdf:Property ;
  rdfs:domain :Hovercraft ;
  rdfs:range xsd:integer ;
  rdfs:comment "This property states how many engines the hovercraft has"@en .
```

Limitazioni di RDFS

- RDFS è **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 è 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 tutte le persone hanno esattamente due genitori
 - Nessuna proprietà **transitive, inverse or symmetriche**
 - Non si può affermare che `isPartOf` è una proprietà transitiva, che `hasPart` è l'inversa di `isPartOf` o che `touches` è simmetrica
 - ...

- Due linguaggi sviluppati estendendo (parte di) RDFS
 - **OIL**: sviluppato dal gruppo di (per lo più) ricercatori europei (molti dal progetto europeo OntoKnowledge)
 - **DAML-ONT**: sviluppato dal gruppo di (per lo più) ricercatori statunitensi (nel programma DARPA **DAML**)
- Impegno congiunto per produrre **DAML+OIL**
 - Lo sviluppo fu realizzato dal “Joint EU/US Committee on Agent Markup Languages”
 - Estende (il «sottoinsieme DL» di) RDF
- **DAML+OIL** sottomesso al W3C come base per la standardizzazione
 - Formazione del Web-Ontology (**WebOnt**) Working Group
 - Il gruppo WebOnt sviluppa il linguaggio **OWL** (Web Ontology Language) basato su **DAML+OIL**
 - Il linguaggio OWL è ora una **Raccomandazione W3C** (correntemente alla versione 2)

- 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 ritrattare informazione precedente) alla teoria e/o alla descrizione degli oggetti per far sì che questi siano ancora un modello per la *teoria*
- Semantica RDFS/OWL: due caratteristiche importanti
 - **Open World Assumption** (OWA)
 - **No Unique Name Assumption**

- La CWA (**Closed World Assumption**), tipica dei database tradizionali, e la NF (**negation-as-failure**), che caratterizza i 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 forma atomica ground A , la NF dice che:
 - Se non è possibile dimostrare A in una base di conoscenza KB , allora A è falsa in quella KB

Non monotonicità della CWA e NF

- CWA e NF sono naturalmente connesse ad una visione non-monotona del mondo
- Es: abbiamo un DB con il solo fatto: `woman(MarilynMonroe)` ed eseguo delle query in Prolog
 - Query: `?- woman(MarilynMonroe).`
 - Risposta: sì
 - Query: `?- man(MarilynManson)`
 - Risposta: no (usando la CWA/NF).
 - Successivamente aggiorniamo il DB con `man(MarlynManson).`
 - Query: `?- man(MarlynManson)`
 - Risposta: sì

- 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**.
- Un po' di esempi:

```
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 incompetita anche se mincard per author è 1, perché l'autore potrebbe essere definito "da qualche altra parte nel mondo" (Open World Assumption)

- 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**.
- Un po' di esempi:

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

I due valori per *eg:copyrightHoder* destano problemi? No, potrebbero esistere due nomi per indicare la stessa cosa, così inferiamo che *institute1* e *institute2* denotano lo stesso oggetto

- 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**.
- Un po' di esempi:

```
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.
```

Si inferisce che Duffy è ANCHE una Person (a causa dell'asserzione di equivalentClass sulla restrizione allValuesFrom Person)

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

- RDF 1.1 Primer (<https://www.w3.org/TR/rdf11-primer/>)
- 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/>)

- Questo argomento può essere approfondito su:
 - 2° chapter of the Description Logics Handbook
 - <https://bscw.deri.org/pub/bscw.cgi/d3851/dlhb-02.pdf>
 - https://cdn.preterhuman.net/texts/thought_and_writing/philosophy/Bader%20et%20al%29%20%28ed.%29%20-%20The%20Description%20Logic%20Handbook%20-%20Theory,%20Implementation%20and%20Applications%20%282003%29.pdf
 - OWL W3C Guide (nota: non aggiornata a OWL 2)
 - <https://www.w3.org/TR/owl-guide/>