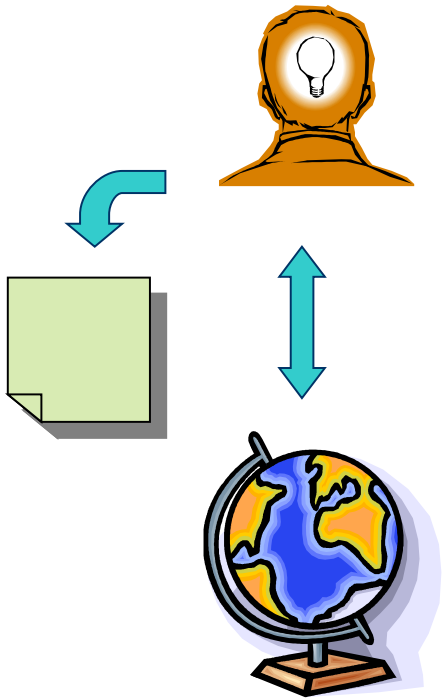
RDF Schema

Mario Arrigoni Neri

Interoperabilità: fase 0

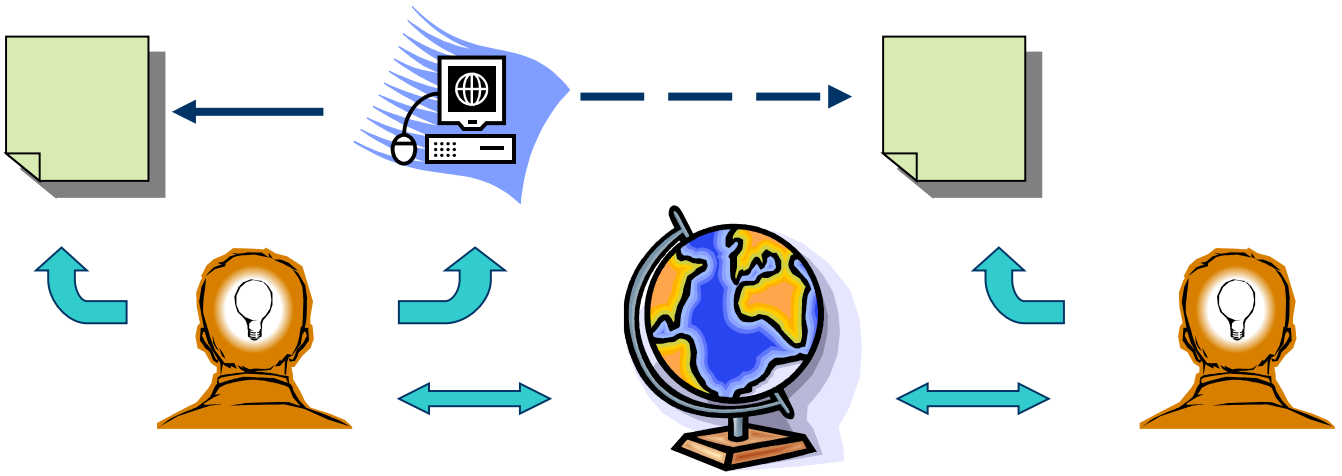
- Ricerca testuale
- Motori di ricerca “classici” : Google, Yahoo, ecc..
- Ricerca su termini specifici: “Parapendio”
  - I risultati sono piuttosto specifici e pertinenti
- Ricerca su termini specifici: “Verdi”
  - Giuseppe Verdi
  - Teatro Verdi
  - Il partito dei Verdi
  - Libri Verdi della commissione europea



- L’interoperabilità si basa sulla comprensione umana dei termini

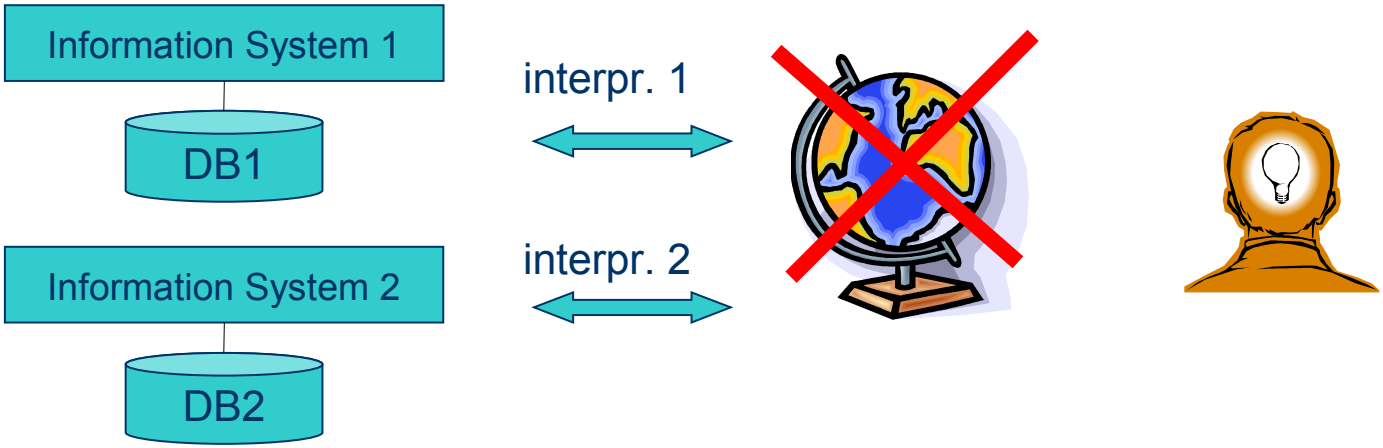
Interoperabilità: fase 1

- XML
- Schemi definiti dall'utente offrono una possibilità di condividere le interpretazioni dei dati con diverse applicazioni
- Diversi repository diversi schemi:
  - Swiss Prot: Find <Species>leech</Species>
  - EMBLChange. Find <Organism>leech<Organism>



Interoperabilità: fase 2

- Semantic Web
- Si elimina il riferimento diretto alla realtà costruendo ontologie composte da termini
- La semantica dei metadati è eterogenea ed inconsistente tra repository diversi

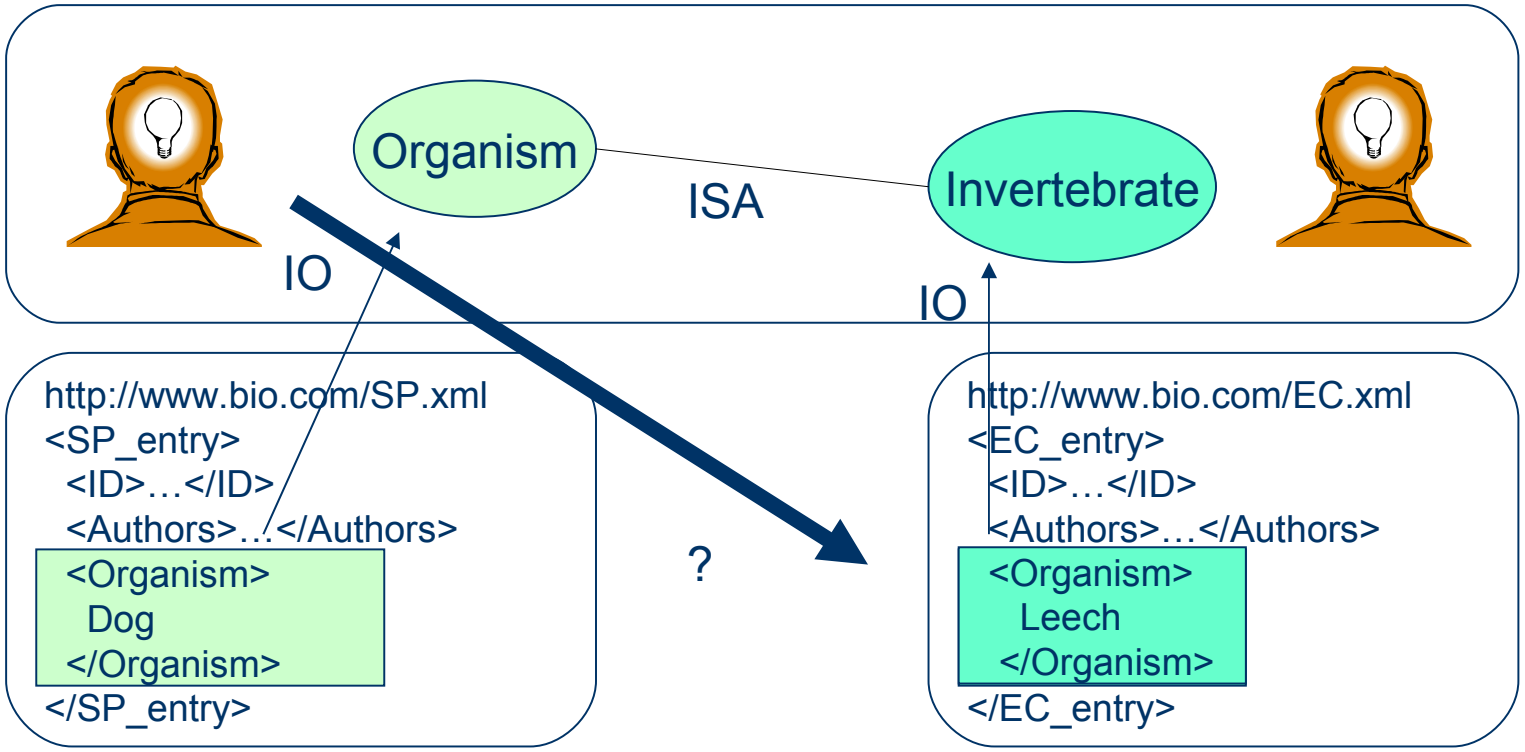


Eterogeneicità semantica

- Standardizzazione: consenso su markup comuni
  - Applicabile se non esistono applicazioni esistenti
  - Applicabile se esistono attori dominanti che “dettano” le regole
- Traduzione: creazione di mappe tra schemi differenti
  - Richiede interpretazione e comprensione
  - La mappatura è complessa e costosa
  - Schemi di traduzione N^2
- Annotazione: creare relazioni con concettualizzazioni comuni
  - Richiede interpretazione e comprensione
  - La creazione delle relazioni può essere complessa
  - Servizi di ragionamento sulla concettualizzazione comune (ontologia) possono fornire valore aggiunto

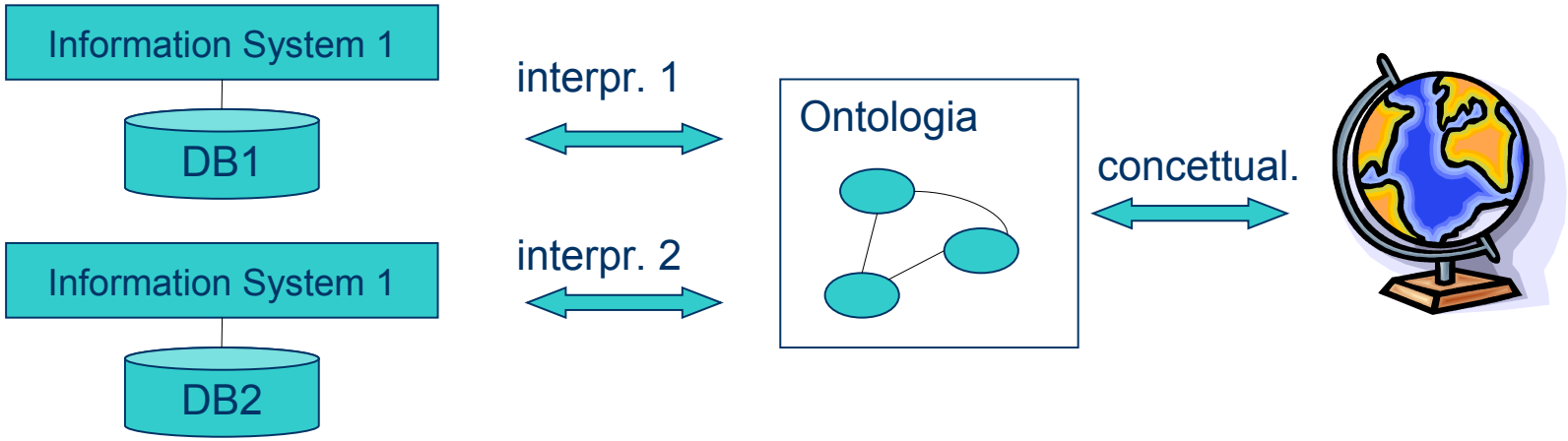
Annotazione

- L'ontologia deve consegnare al sistema il significato dei termini utilizzati



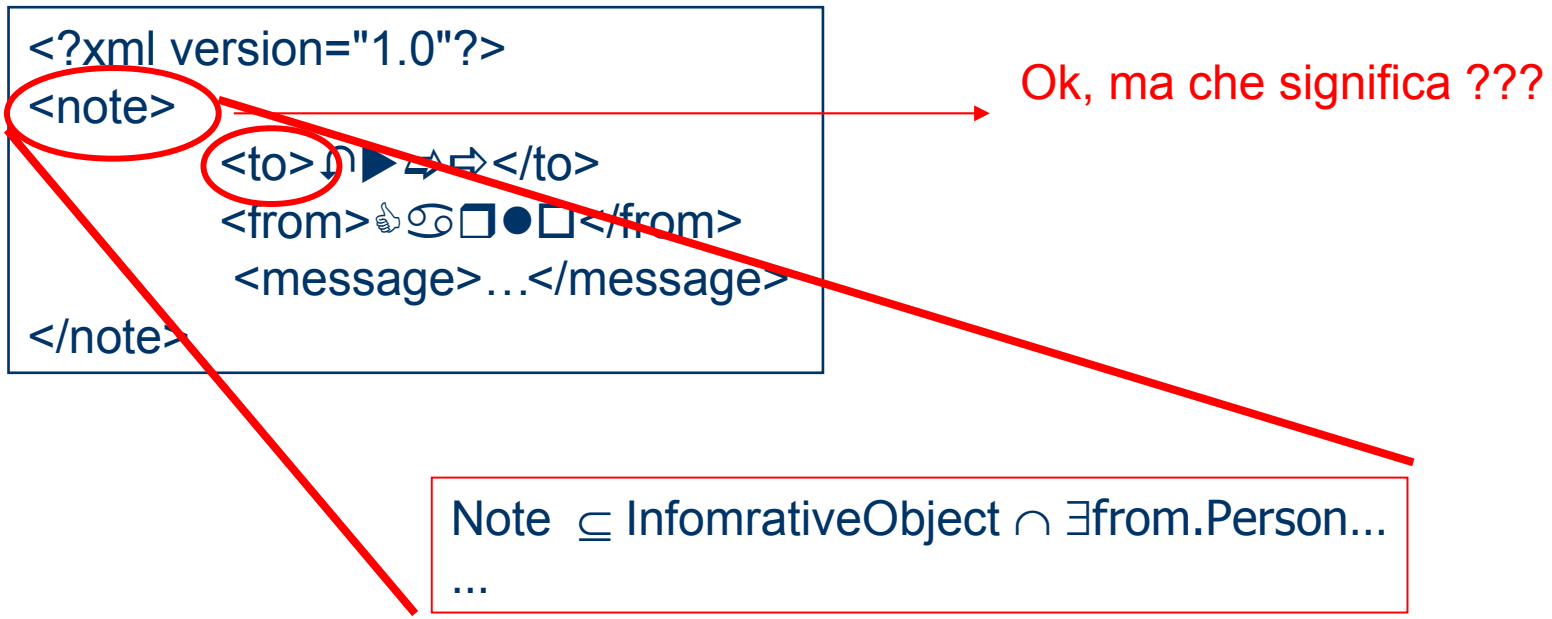
Ontologie

- Una ontologia è una esplicita rappresentazione di una concettualizzazione della realtà [Gruber, 1993]
- Diversi sistemi concordano su una ontologia (direttamente o indirettamente)
- Associano i propri modelli a nodi dell'ontologia condivisa
- Occorre raggiungere un accordo sull'ontologia



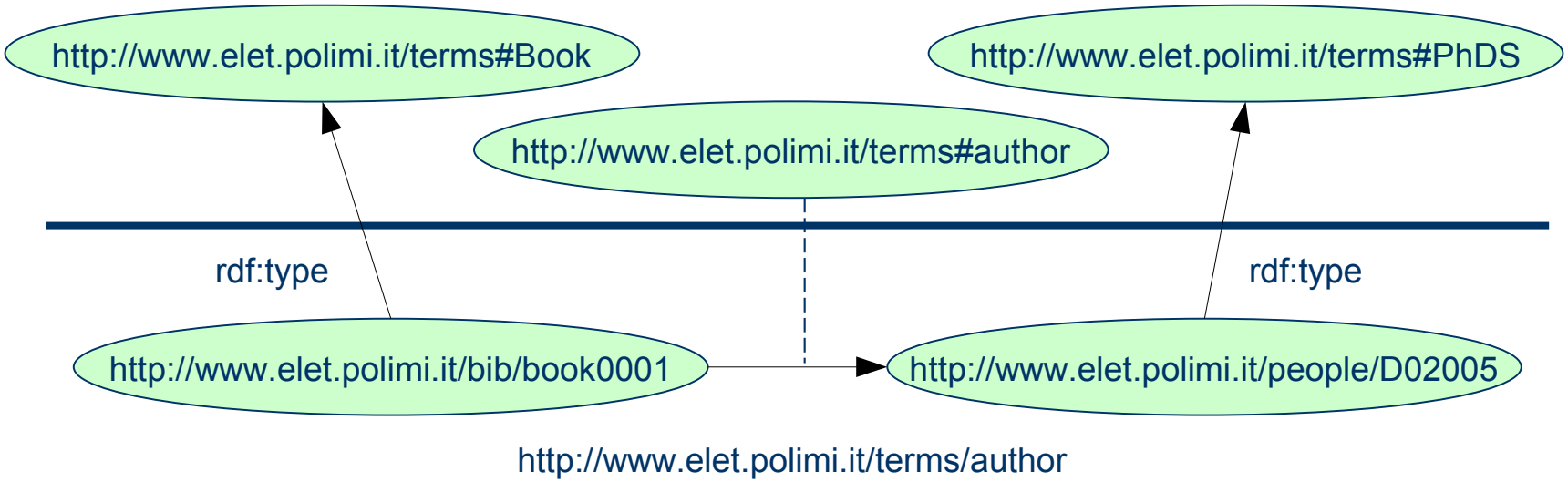
Ontologie

- Da un punto di vista tecnico, l'ontologia permette di dare una semantica ai metadati



Uno schema per RDF

- RDF fornisce un modo per descrivere generiche asserzioni su risorse e proprietà
- E' spesso necessario indicare il fatto che queste si riferiscono a particolari tipi di risorse ed usano specifiche proprietà



RDF(S)

- Per descrivere le classi e le relazioni utilizzate per costruire il particolare modello RDF si utilizza ancora RDF
- Il particolare vocabolario è definito dall' *RDF Vocabulary Description Language*, altrimenti noto come *RDF-Schema*
- Non fornisce un vocabolario specifico per l'applicazione (*terms:Book*, *terms:PhDS*, *terms:author*, ecc..)
- Definisce un meccanismo per specificare classi e proprietà e costruire lo specifico vocabolario
- RDF(S) fornisce un sistema di tipi (semantici) per RDF

Classi – 1

- Le classi sono aggregati di individui. Ogni classe rappresenta un tipo di risorsa su cui si costruisce il modello RDF
- Il *namespace* di riferimento è: *http://www.w3.org/2000/01/rdf-schema#*
- Ogni classe è una risorsa in relazione *rdf:type* con la risorsa *http://www.w3.org/2000/01/rdf-schema#Class*

Classi – 2

terms:PhDS	rdf:type	rdfs:Class

```
<rdf:RDF    xmlns:terms="http://www.elet.polimi.it/terms#"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdf:Description rdf:about="http://www.elet.polimi.it/terms#PhDS">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  </rdf:Description>
</rdf:RDF>
```

Utilizzo la forma abbreviata di RDF ed *rdf:ID* per riferirmi alla descrizione Locale (per fare questo occorre strutturare correttamente il *ns*)

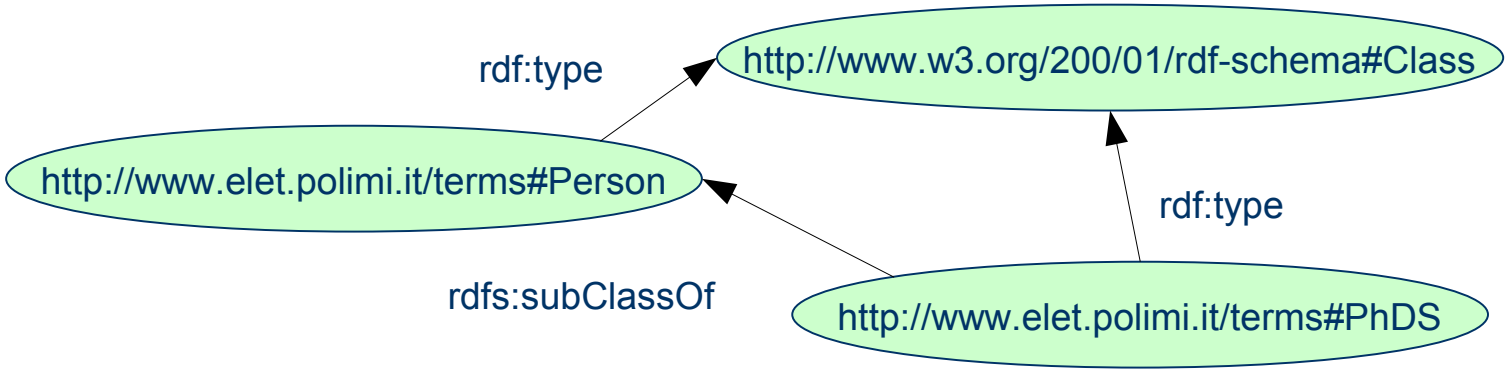
```
<rdf:RDF    xml:base="http://www.elet.polimi.it/terms"
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <rdfs:Class rdf:ID="PhDS"/>
</rdf:RDF>
```

Gerarchie di classi – 1

- Analogamente ai linguaggi OO, RDF(S) permette di organizzare le classi in gerarchie
- Il vocabolario RDF(S) mette a disposizione la relazione *subClassOf*
  - Transitiva
- Es: un PhD Student è (is-a) una Persona

C : insieme dei simboli di classe
Δ^I : insieme di interpretazione
$\bullet^I$: funzione di interpretazione $A^I \subseteq \Delta^I$
$A, B \in C$
A *rdfs:subClassOf* B
$\forall x \in \Delta^I : x \in A^I \Rightarrow x \in B^I$
Ovvero : $A^I \subseteq B^I$
Dove $x \in A^I \leftrightarrow$ <x *rdf:type* A>

Gerarchie di classi – 2




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

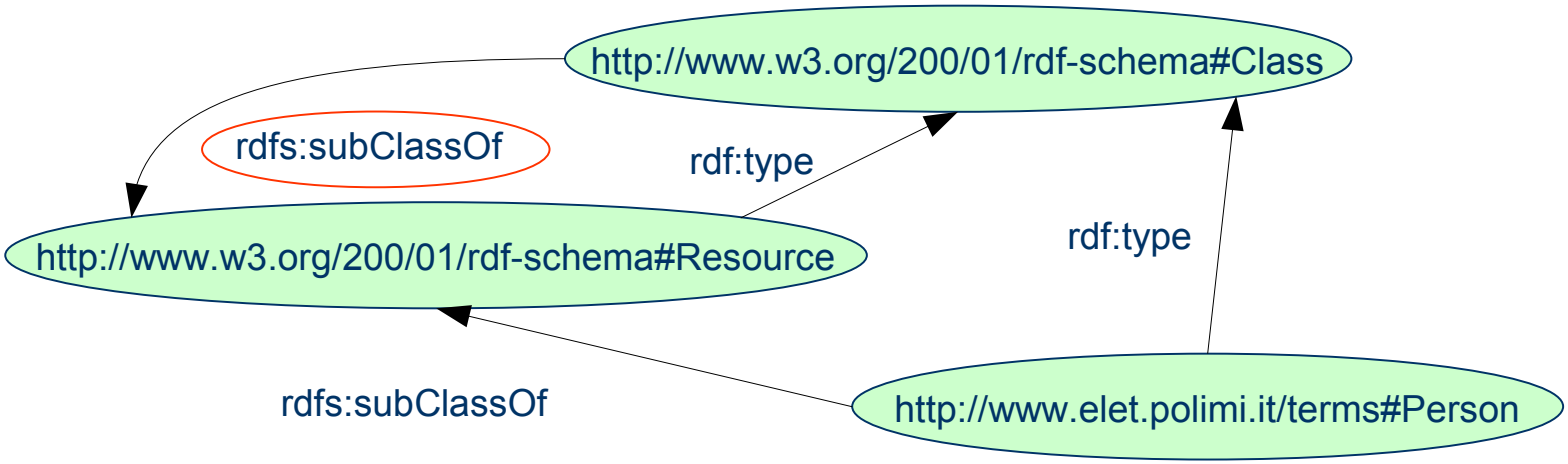
Sintassi alternative

- Dato che RDFS è espresso esso stesso in RDF posso utilizzare tutte le varianti sintattiche

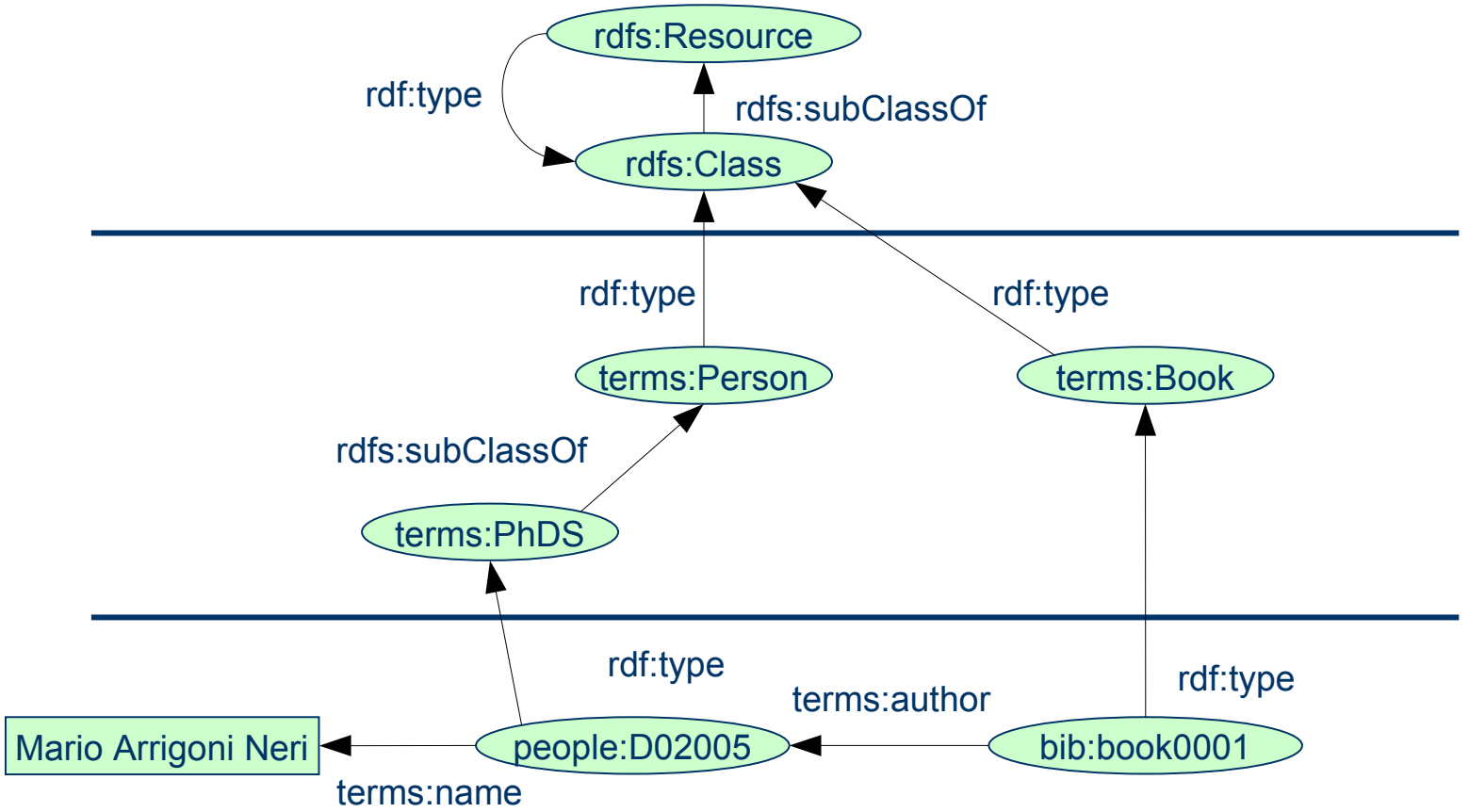
```
<rdfs:Class rdf:ID="PhDS">
        <rdfs:subClassOf>
                <rdfs:Class rdf:ID="Person"/>
        </rdfs:subClassOf>
</rdfs:Class>
```

Il meta-modello RDF(S)

- Le stesse risorse che usiamo per descrivere lo schema sono classi.
- *rdfs:Class* è la (meta) classe a cui appartengono tutte le classi
  - NON è l'analogo dell'Object di Java
- *rdfs:Resource* è la classe "universo" da cui derivano tutte le classi di un modello. Se non indico alcun *rdfs:subClassOf* la classe deriva implicitamente da rdfs:Resource.
  - E' l'analogo dell'Object di Java

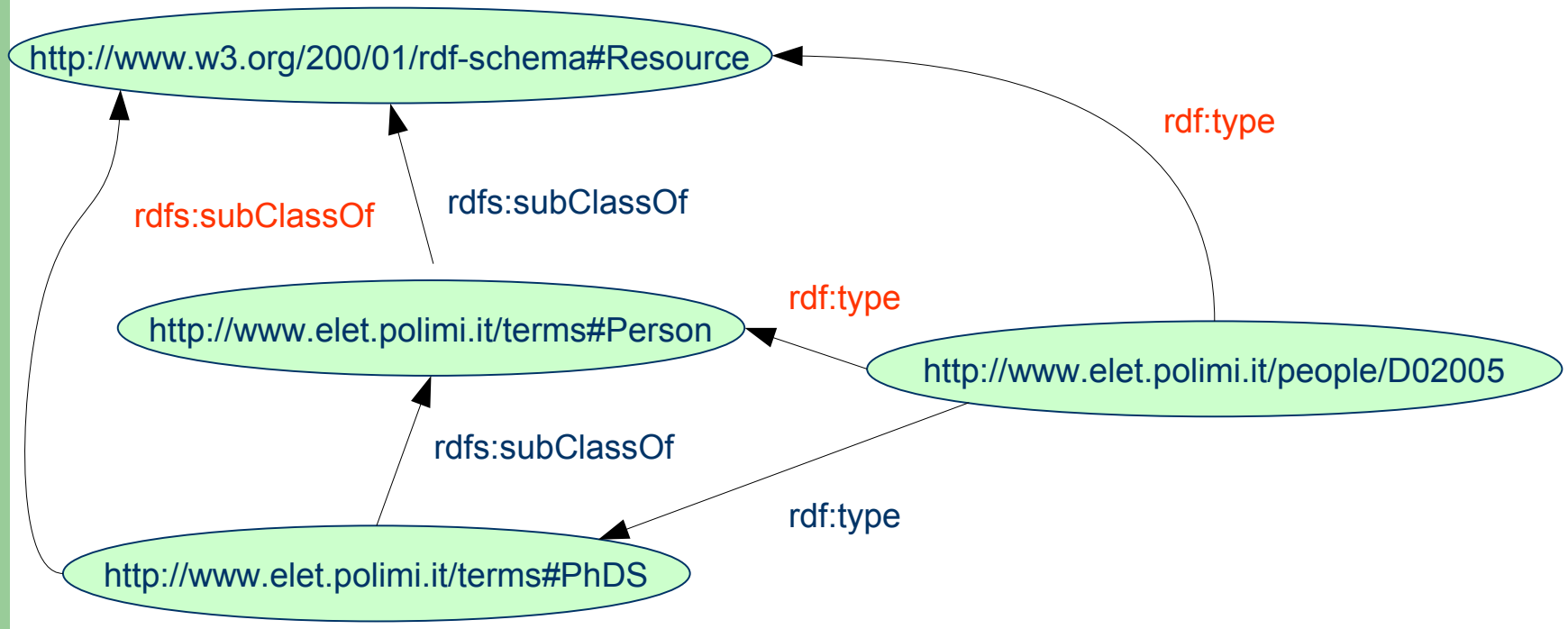


Istanze – classi – metaclassi



Reasoning – 1

- I semplici costrutti rdf:type ed rdfs:subClassOf permettono già un servizio minimale di reasoning



Reasoning – 2

people:D02005	rdf:type	terms:PhDS
terms:PhDS	rdf:type	rdfs:Class
terms:Person	rdf:type	rdfs:Class
terms:PhDS	rdfs:subClassOf	terms:Person

term:Person	rdfs:subClassOf	rdfs:Resource

A	rdfs:subClassOf	B
B	rdfs:subClassOf	C
A	rdfs:subClassOf	C

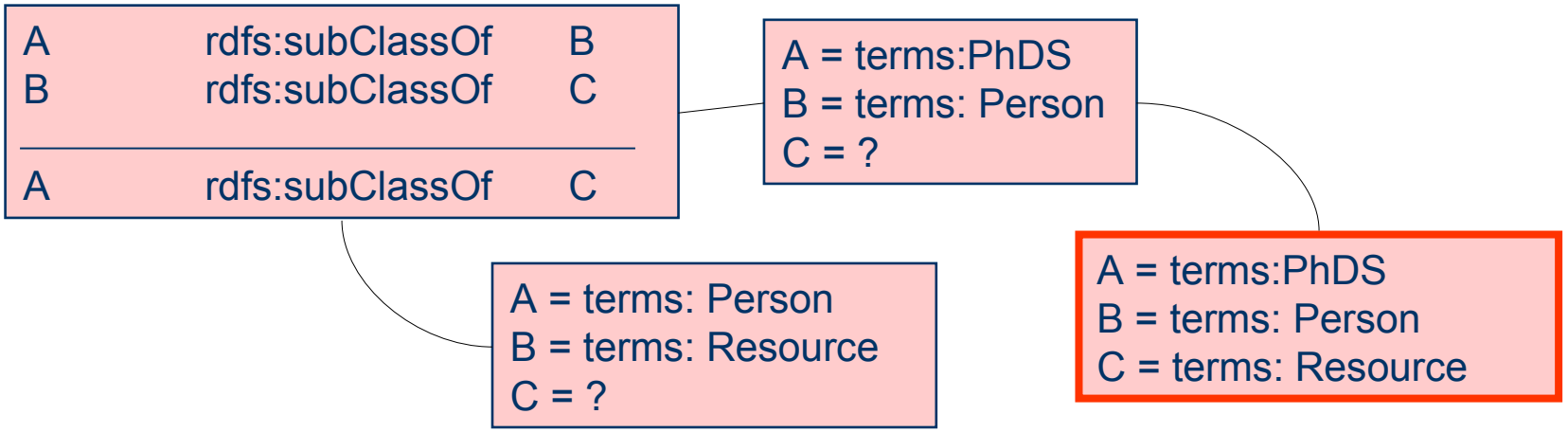
A	rdf:type	B
B	rdfs:subClassOf	C
A	rdf:type	C

Term:PhDS	rdfs:subClassOf	rdfs:Resource
people:D02005	rdf:type	terms:Person
people:D02005	rdf:type	terms:Resource

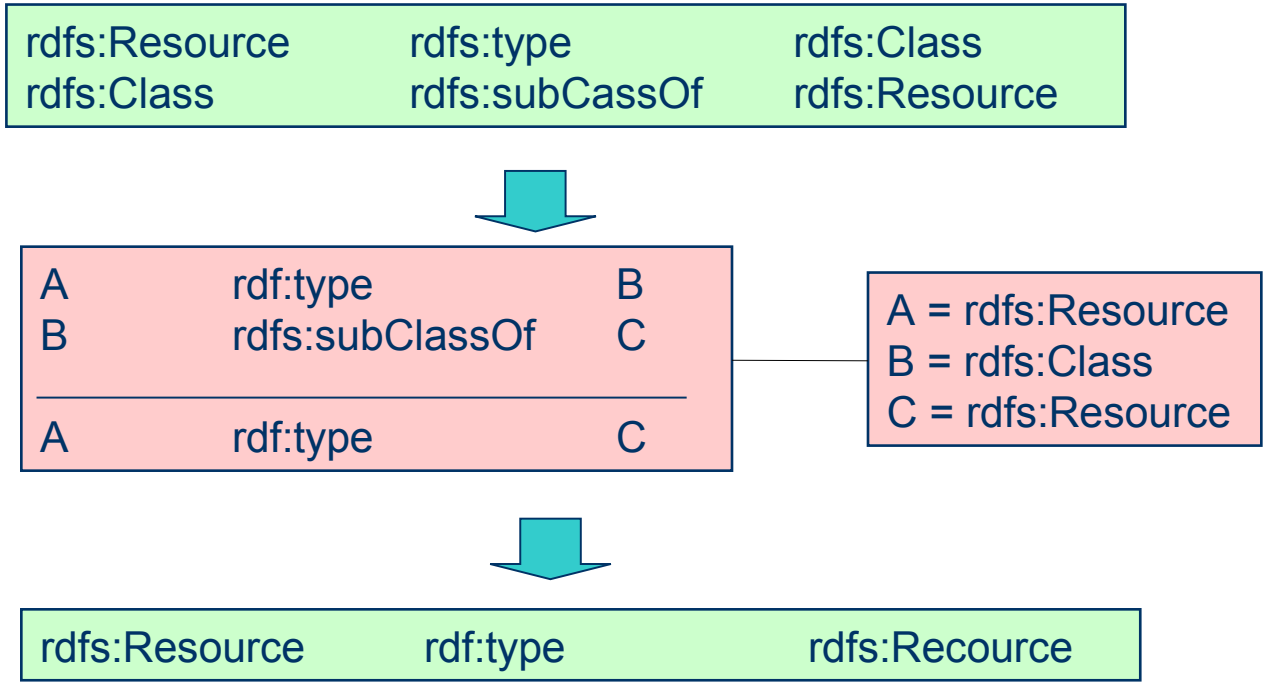
Reasoning – 3

- In questo esempio resoning basato su REGOLE DI INFERENZA
- Il motore inferenziale cerca di combinare gli antecedenti delle regole con i fatti presenti nella base di conoscenza tramite il PATTERN MATCHING

1)	terms:PhDS	rdfs:subClassOf	terms:Person
2)	terms:Person	rdfs:subClassOf	terms:Resource

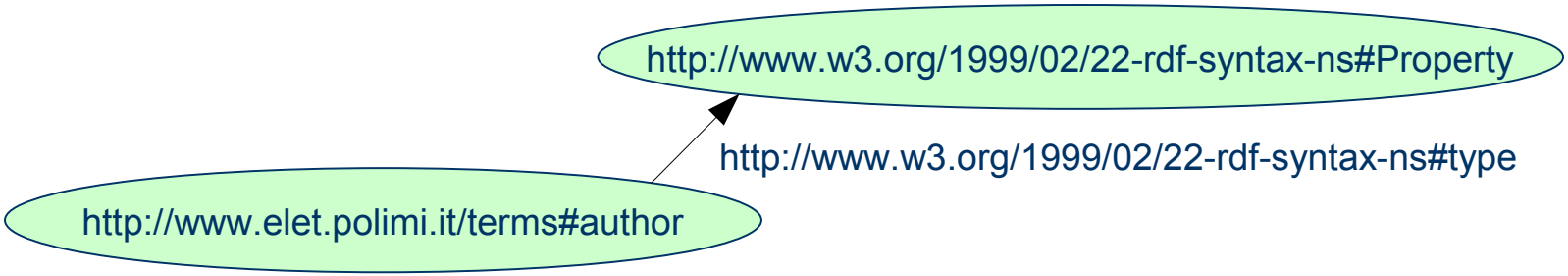


Reasoning sul meta-modello



Proprietà

- Oltre a descrivere le classi a cui appartengono gli oggetti del modello abbiamo bisogno di descrivere specifiche proprietà
- Ogni proprietà RDF è istanza della classe predefinita *rdf:Property*.

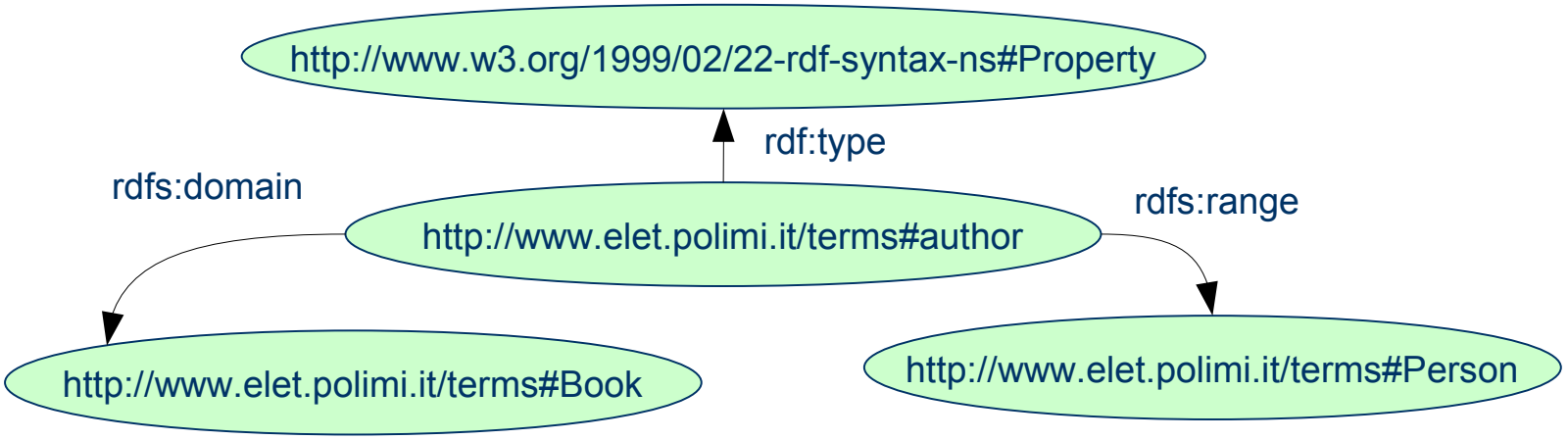


terms:author rdf:type rdf:Property

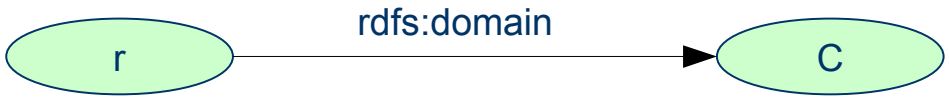
```
<rdf:Property rdf:ID="author"/>
```

Domain e Range – 1

- Rimane il problema di collegare tra di loro classi e proprietà
- RDF(S) fornisce anche un vocabolario per descrivere come ci si aspetta che proprietà e classi si combinino tra di loro
- Le principali informazioni in questo frangente sono date dalle proprietà prdefinite *rdfs:domain* ed *rdfs:range*
- *Es: la relazione author* sussiste tra un libro ed una persona:



Domain e Range – 2

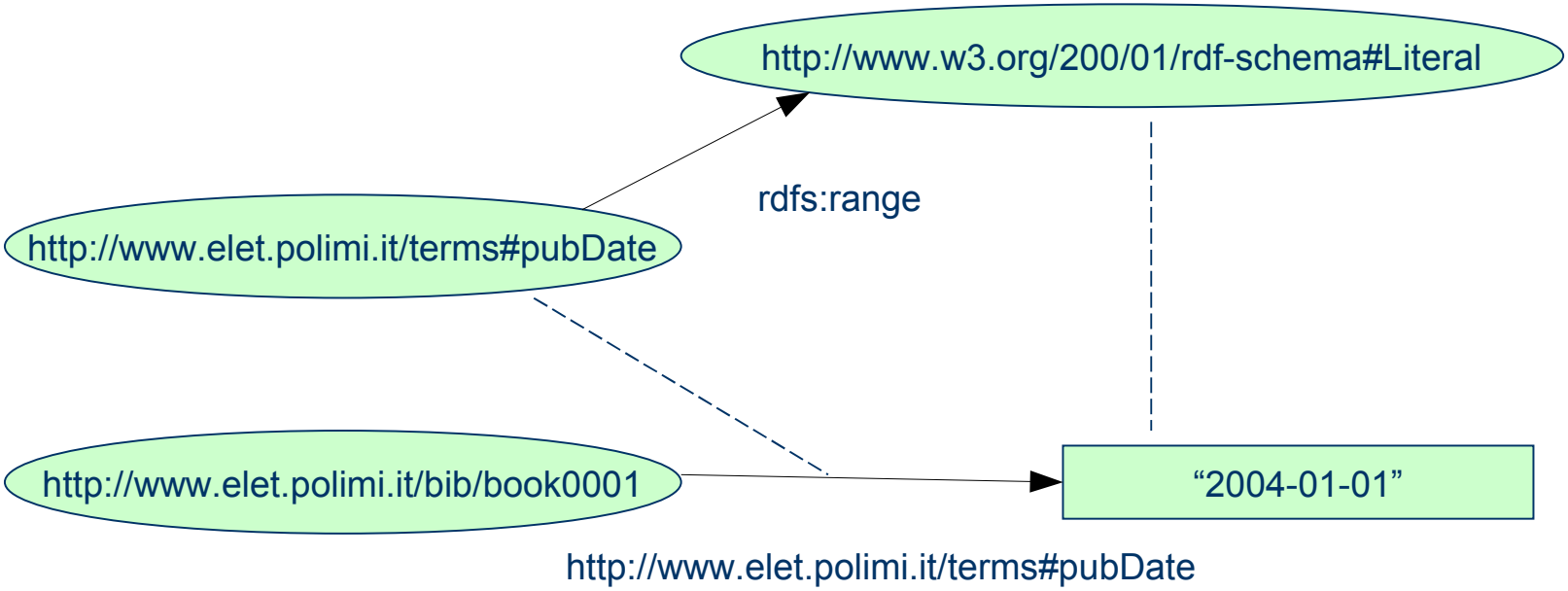


R : insieme dei simboli di relazione
$\bullet^I$: funzione di interpretazione $R^I \subseteq \Delta^I \times \Delta^I$
$\forall x, y \in \Delta^I : \langle x,y \rangle \in r^I \Rightarrow x \in C^I$

$\forall x, y \in \Delta^I : \langle x,y \rangle \in r^I \Rightarrow y \in C^I$

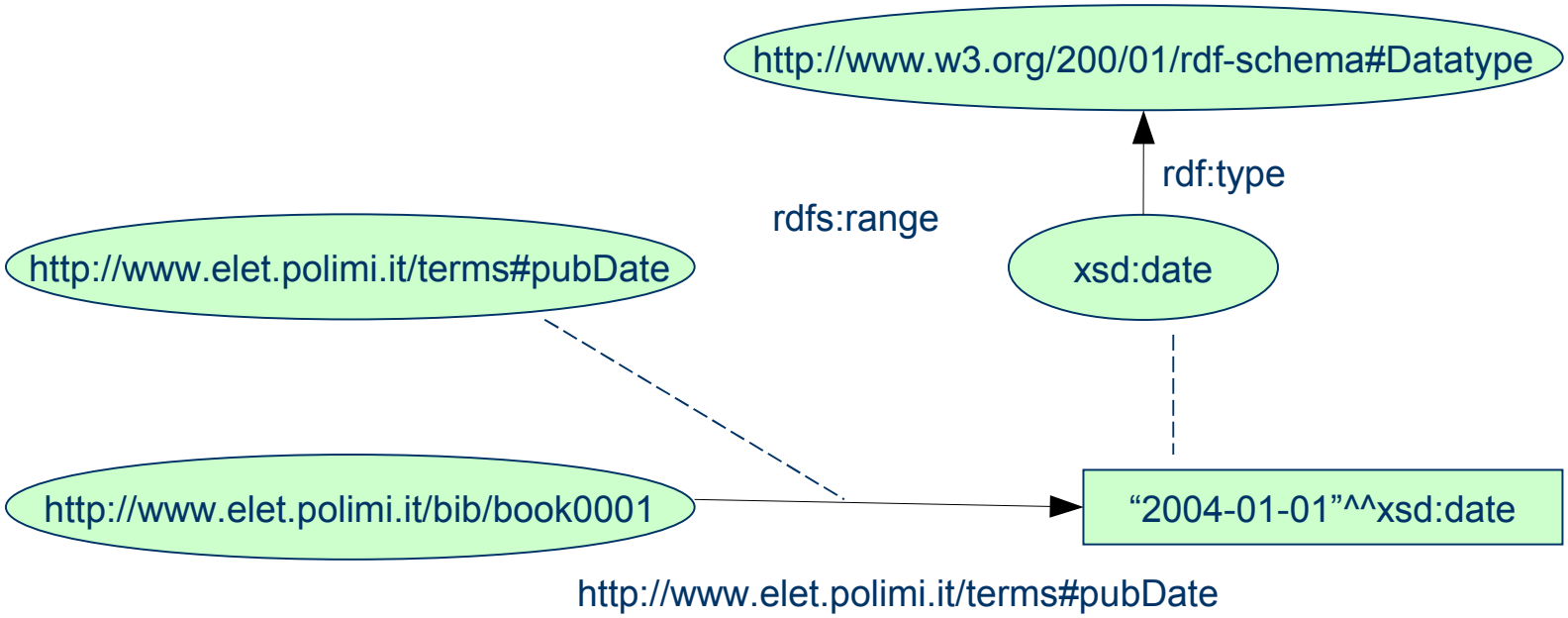
Literal range – 1

- Opzionalmente il range delle proprietà (e solo quello) può essere un literal tipizzato



Literal range – 2

- Opzionalmente può essere un literal tipizzato
- Se si vuole esprimere esplicitamente il fatto che si tratta di un literal si può utilizzare la classe *rdfs:Datatype*



Literal range – 3

```
<rdf:Property rdf:ID="pubDate">
  <rdfs:domain rdf:resource="http://www.elet.polimi.it/terms#Book"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
</rdf:Property>
<rdfs:Datatype rdf:about="http://www.w3.org/2001/XMLSchema#date"/>
```

- RDF non assegna alcun significato ai particolari tipi di literal
- Il controllo di consistenza tra testo formattato e tipo è lasciato all'applicazione
- Quindi il parser RDF non ha modo di "validare" un modello in cui compaiano literal tipizzati
- Si richiede che il literal nel documento sia esplicitamente tipizzato

```
<terms:Book rdf:ID="book0001">
  <terms:author rdf:resource="http://www.elet.polimi.it/bib/book0001"/>
  <terms:pubDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">
                2004-01-01</terms:pubDate>
</terms:Book>
```

Domini e Range multipli

```
<rdfs:Property rdf:ID="author">
  <rdfs:range resource="#Person"/>
  <rdfs:range resource="#Company"/>
</rdfs:Property>
```

```
<rdfs:Property rdf:ID="author">
  <rdfs:range resource="#Person"/>
</rdf:Property>
…
<rdfs:Property rdf:ID="author">
  <rdfs:range resource="#Company"/>
</rdf:Property>
```

$\forall x, y \in \Delta^I : \langle x,y \rangle \in author^I \Rightarrow y \in Person^I$
$\forall x, y \in \Delta^I : \langle x,y \rangle \in author^I \Rightarrow y \in Company^I$

$\forall x, y \in \Delta^I : \langle x,y \rangle \in author^I \Rightarrow y \in Person^I \wedge y \in Company^I$

$\forall x, y \in \Delta^I : \langle x,y \rangle \in author^I \Rightarrow y \in (Person^I \cap Company^I) = \emptyset$

Sottoproprietà

- Analogamente a quanto accade con le classi, ogni proprietà può essere definita come sottoproprietà di una data
- Per questo si utilizza la relazione *rdfs:subPropertyOf*

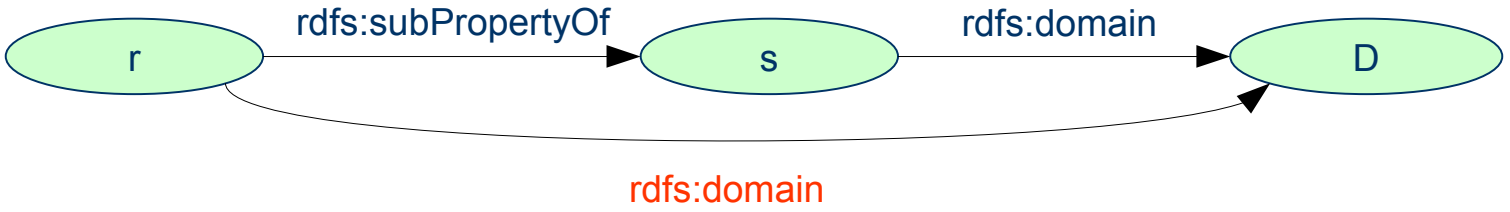
```
<rdf:Property rdf:ID="author">
  <rdfs:subPropertyOf resource="#contributor"/>
</rdf:Property>
```

- La semantica è la solita:
  - L'interpretazione della sottorelazione (sottoproprietà) è un sottoinsieme dell'interpretazione della relazione da cui deriva

R : insieme dei simboli di classe
Δ^I : insieme di interpretazione
$\bullet^I$: funzione di interpretazione $r^I \subseteq \Delta^I \times \Delta^I$
$r, s \in R$
r *rdfs:subPropertyOf* s
$\forall x, y \in \Delta^I : \langle x, y \rangle \in r^I \Rightarrow \langle x, y \rangle \in s^I$

Sottoproprietà, Domain e Range

- Domini e codomini (range) si ereditano lungo le catene di *rdfs:subPropertyOf*



$$\forall x, y \in \Delta^I : \langle x, y \rangle \in r^I \Rightarrow \langle x, y \rangle \in s^I$$
$$\forall x, y \in \Delta^I : \langle x, y \rangle \in s^I \Rightarrow x \in D^I$$

$$\forall x, y \in \Delta^I : (\langle x, y \rangle \in r^I \Rightarrow \langle x, y \rangle \in s^I) \wedge (\langle x, y \rangle \in s^I \Rightarrow x \in D^I)$$

$$\forall x, y \in \Delta^I : \langle x, y \rangle \in r^I \Rightarrow x \in D^I$$

OWA – Open World Assumption

- La semantica di RDF/RDF(S) è studiata in modo da poter effettuare inferenze anche assumendo di non possedere tutta la conoscenza sull'argomento
  - Posso usare nell'RDF classi e relazioni che non compaiono nello schema
- Anche le inferenze rimangono corrette (logica monotonica) con l'aumentare della conoscenza. Non vero ad esempio con interpretazione disgiuntiva di domini e range multipli

```
<rdf:Property rdf:ID="author">
  <rdfs:range resource="#People"/>
</rdf:Property>
<terms:Book rdf:about="bib:book0001">
  <terms:author rdf:resource="#D02005"/>
</terms:Book>
```

```
<terms:People rdf:about="#D02005"/>
```

```
<rdf:Property rdf:ID="author">
  <rdfs:range resource="#Company"/>
</rdf:Property>
```

```
<terms:People rdf:about="#D02005">
  <rdf:type resource="..Company"/>
</terms:People>
```

Corrispondenza con le DL

Costrutto RDF	DL
<A rdfs:subClassOf B>	$A \subseteq B$
<R rdfs:range B>	$T = \forall R.B$
<R rdfs:domain A>	$T = \forall R^{-}.A$ // $\exists R \subseteq A$
<R rdfs:subPropertyOf S>	$R \subseteq S$
<x rdf:type A>	x:A // A(x)
<x R y>	R(x,y)

Però ad RDFS mancano molti costrutti per rientrare nelle logiche AL

Documentazioni

- rdfs:label
  - rdf:type rdf:Property
  - rdfs:domain rdfs:Resource
  - rdfs:range rdfs:Literal
  - Permette di assegnare una stringa human-readable ad ogni risorsa

- rdfs:comment
  - rdf:type rdf:Property
  - rdfs:domain rdfs:Resource
  - rdfs:range rdfs:Literal
  - Permette di assegnare una descrizione human-readable ad ogni risorsa

Annotazioni

- rdfs:seeAlso
  - rdf:type rdf:Property
  - rdfs:domain rdfs:Resource
  - rdfs:range rdfs:Resource
  - Collega la risorsa ad altre risorse che forniscono ulteriori informazioni sulla risorsa o sul suo contesto di uso

- rdfs:isDefinedBy
  - rdf:type rdf:Property
  - rdfs:domain rdfs:Resource
  - rdfs:range rdfs:Resource
  - rdfs:subPropertyOf rdfs:seeAlso
  - Collega a risorse che “definiscono” la risorsa data

Uno schema semantico?

- XML-Schema e DTD specificano la sintassi dei documenti
- RDF Schema specifica la semantica intesa delle risorse utilizzate
- Cioè non vincola la struttura del documento, ma fornisce informazioni utili all'interpretazione del modello rappresentato nel documento

```
<rdf:Description rdf:ID="D02005Name">
        <names:name>Mario</names:name>
        <names:surname>Arrigoni Neri</names:surname>
</rdf:Description>
```

```
<rdf:Description rdf:ID="D02005Name" names:name="Mario">
        <names:surname>Arrigoni Neri</names:surname>
</rdf:Description>
```

Varianti sintattiche per uno stesso modello

RDF(S) e linguaggi OOP

- RDF(S) ha molti punti in comune con i comuni linguaggi di programmazione OOP (es: Java), ma differisce da questi in alcuni punti centrali

- Differente ripartizione delle descrizioni:

Java	RDF(S)

public class Person
{…}
public class Book
{
.. public Person author;
}
	<rdfs:Class rdf:ID="Book"/>
	<rdfs:Class rdf:ID="Person"/>
	<rdf:Property rdf:ID="author">
	<rdfs:domain rdf:resource="#Book"/>
	<rdfs:range rdf:resource="#Person"/>
	</rdf:Property>

Scope delle proprietà

- Ma si tratta solo di una differenza sintattica?
- In un linguaggio di programmazione imperativo la definizione delle relazioni (proprietà) non è semplicemente “sintatticamente” inserita nella definizione della classe, ma è anche “semanticamente” contestuale alla classe
- Se facessi lo stesso in RDF(S) sarei autorizzato a dedurre che l’autore è, contemporaneamente, una persona ed una compagnia
- Posso definire la proprietà “peso” senza indicare dom. e range. Diventa facile riusare proprietà in contesti inattesi.. però

```
public class Book
{
.. Public Person author;
}
…
public class Specification
{
.. Public Company author;
}
```

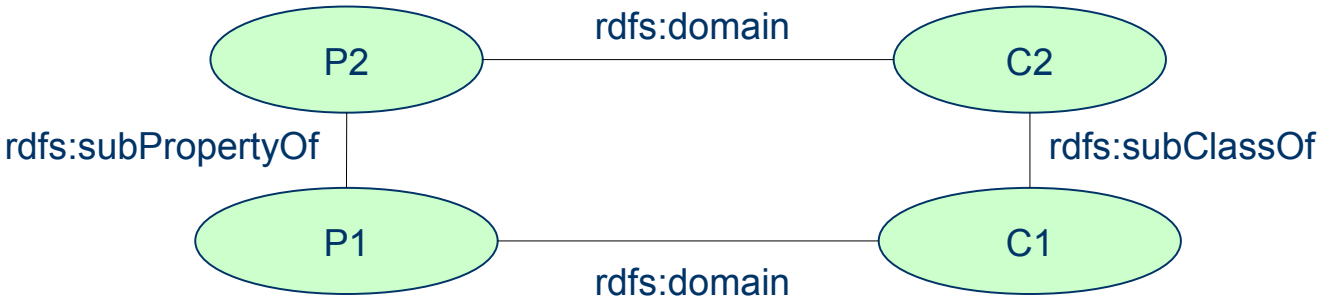
Le due proprietà author sono omonime, ma differenti

Uso dello schema

- Nei linguaggi di programmazione lo schema dei tipi ha sempre uno scopo PRESCRITTIVO. Il programma che non rispetta i vincoli semplicemente è scorretto
- In RDF lo Schema fornisce informazioni aggiuntive, ma si lascia all'applicazione la scelta dell'uso di queste informazioni
- Scenario 1 – uso prescrittivo: l'applicazione interpreta lo schema come dei vincoli (constraints) sui modelli leciti
- Scenario 2 – uso deduttivo sul modello: si possono usare le informazioni sullo schema per dedurre ulteriore conoscenza. Es: ho un libro di cui conosco l'autore, posso dedurre che la risorsa autore è una persona
- Scenario 3 – uso deduttivo sullo schema: incontro un libro che ha per autore una *Company*, in questo caso potrebbe esserci una inconsistenza, oppure posso trovare che Company subClassOf Person

Limiti di RDF – Schema

- Limitato potere espressivo (subClassOf, subPropertyOf, range, domain)
- Semantica non ben definita



- Linguaggi per ontologie
  - Intuitivi / espressivi
  - Sintassi ben specificata, semantica formale, adeguato potere espressivo
  - Compatibili con linguaggi esistenti (RDF)

Dublin Core – 1

- E’ una libreria di metadati definiti da un consorzio di bibliotecari ed archivisti
- Dublino – 1995
- Nato indipendentemente (e prima) del Semantic Web, ma ovviamente ben adatto ad essere integrato con RDF
- Dublin Core 1: quindici (!) categorie di meta-informazioni
- Dublin Core 2: sistema i classe-sottoclasse delle categorie (qualificatori)

Dublin Core – 2

Contenuto
Title
Subject
Description
Type
Source
Relation
Coverage

Copyright
Creator
Publisher
Contributor
Rights

Istanza
Date
Format
Identifier
language

- I modificatori permettono di specificare meglio i contenuti delle proprietà
  - Date : created, valid, ecc..
- Codifica: è possibile indicare tramite modificatori particolari codifiche per i dati:
  - Subject : LCSH (Library of Congress Subject Headings), MeSH, ecc..

Dublin Core – 3