Ontologie computazionali, OWL/RDF e tecnologie collegate

Rossana Damiano
Università di Torino

rossana.damiano@unito.it

Semantic Web: gli standard

- https://www.w3.org/standards/semanticweb (parte del più ampio archivio di recommendations: https://www.w3.org/standards)

- Oggi: "The term "Semantic Web" refers to W3C's vision of the Web of linked data."

- Semantic Web technologies enable people to :
  - create **data stores** on the Web
  - build **vocabularies**
  - write **rules** for handling data.
  - Linked data are empowered by **technologies** such as RDF, SPARQL, OWL, and SKOS.

(da: https://www.w3.org/standards/semanticweb)

Semantic Web: foto di famiglia

- RDF: permette di descrivere risorse.
  - Una risorsa può essere qualsiasi cosa: un documento, una persona, un oggetto fisico, un concetto astratto

- RDFS: permette di descrivere relazioni tra risorse
  - È un linguaggio “property-centric”
  - Propriamente, estensione di RDF

- OWL: permette di descrivere ontologie computazionali
  - Si colloca a un livello superiore di RDF
  - Permette di descrivere relazioni ricche e complesse tra entità (“complex knowledge about things, groups of things, and relations between things”)

Carta d’identità di RDF

- Nato per descrivere risorse digitali
- Il suo data model è basato sul concetto di grafo
  - I nodi rappresentano *entità* e gli archi *relazioni* tra di esse
- Possiede diverse serializzazioni

Specifiche di RDF

- Le specifiche di RDF consistono in una suite di documenti:
- RDF 1.1 Primer (guida di base)
- RDF 1.1 Concepts and Abstract Syntax
- RDF 1.1 XML Syntax
- RDF 1.1 Semantics
- RDF Schema 1.1
- RDF 1.1 Test Cases

Il Data Model di RDF

Il data model di RDF si basa su un insieme di elementi

- Triple
- IRI
- Letterali
- (Blank nodes)
- (Grafi)

L'unità di base di RDF è una tripla, composta di soggetto, predicato e oggetto. Il soggetto e oggetto possono essere IRI o blank nodes, l'oggetto può essere anche un letterale, il predicato può essere solo un IRI

Un insieme di triple forma un grafo

Per raggruppare le triple è possibile identificare singoli grafi

https://www.w3.org/TR/rdf11-concepts/

RDF: triple

- RDF è formato di triple, che hanno la forma

  Soggetto – Predicato – Oggetto

- Esempio:

  Soggetto: Monna Lisa
  Predicato: Fu creata da
  Oggetto: Leonardo da Vinci

Fonte: https://www.w3.org/TR/rdf11-primer/

Esempio: un insieme di triple

	subject	predicate	object
1	range:22571011794154798tt27tt32	faldo:begin	position:22571011794154798tt27
2	range:22571011794154798tt27tt32	faldo:end	position:22571011794154798tt32
3	range:22571011794154798tt27tt32	rdf:type	faldo:Region

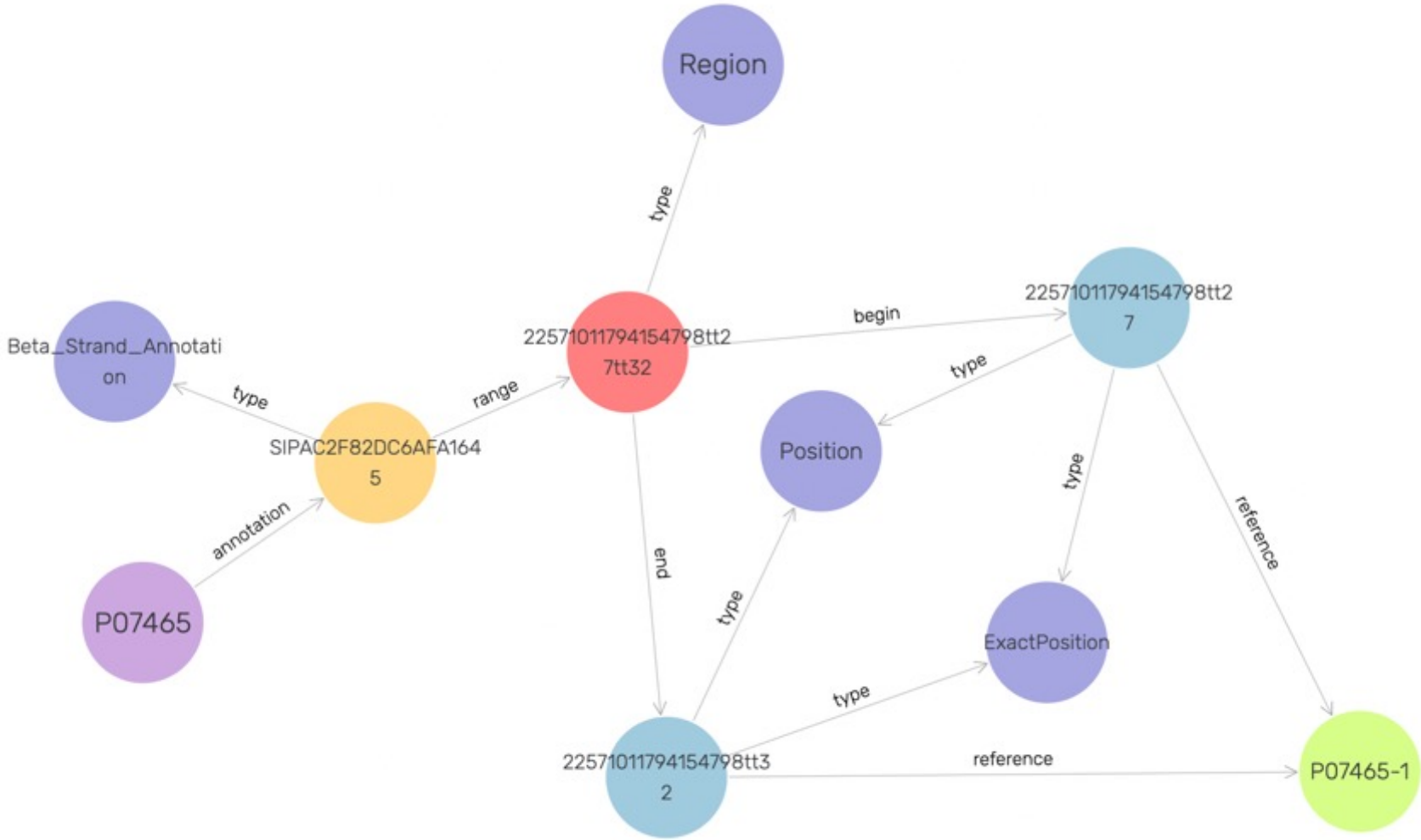
(Fonte: https://www.uniprot.org/uniprot/P07465)

Esempio: un insieme di triple

	subject	predicate	object
1	range:22571011794154798tt27tt32	faldo:begin	position:22571011794154798tt27
2	range:22571011794154798tt27tt32	faldo:end	position:22571011794154798tt32
3	range:22571011794154798tt27tt32	rdf:type	faldo:Region

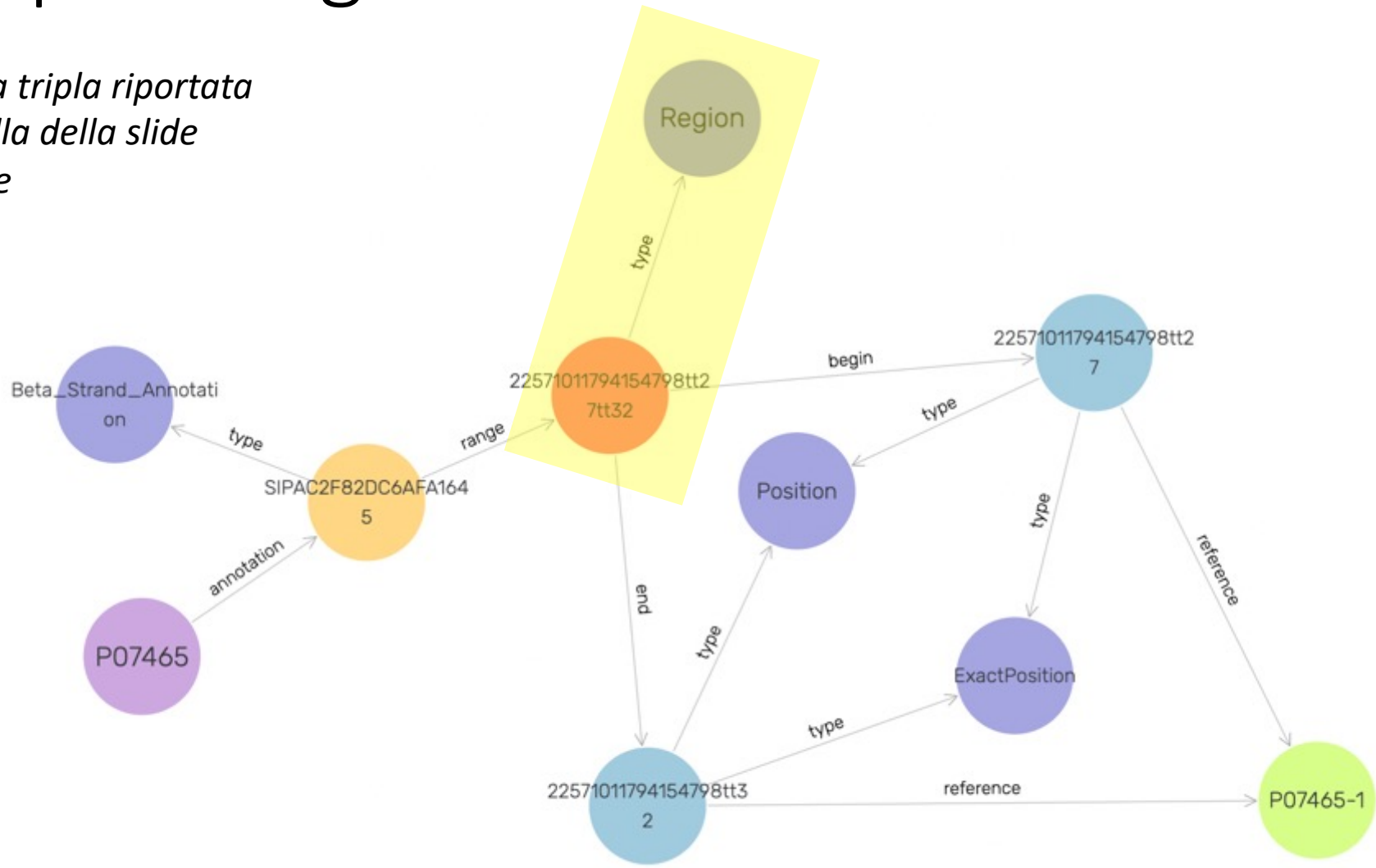
- Soggetto: 22571011794154798tt27tt32
- Predicato: è una (rdf:type)
- Oggetto: Regione

Esempio: un grafo



Esempio: un grafo

In giallo, la tripla riportata nella tabella della slide precedente



Le triple del grafo

<http://purl.uniprot.org/uniprot/P07465#SIPAC2F82DC6AFA1645> <http://purl.uniprot.org/core/range> <http://purl.uniprot.org/range/22571011794154798tt27tt32> .

<http://purl.uniprot.org/range/22571011794154798tt27tt32> a <http://biohackathon.org/resource/faldo#Region> .

<http://purl.uniprot.org/range/22571011794154798tt27tt32> <http://biohackathon.org/resource/faldo#begin> <http://purl.uniprot.org/position/22571011794154798tt27> .

<http://purl.uniprot.org/range/22571011794154798tt27tt32> <http://biohackathon.org/resource/faldo#end> <http://purl.uniprot.org/position/22571011794154798tt32> .

<http://purl.uniprot.org/position/22571011794154798tt27> <http://biohackathon.org/resource/faldo#reference> <http://purl.uniprot.org/isoforms/P07465-1> .

<http://purl.uniprot.org/position/22571011794154798tt27> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://biohackathon.org/resource/faldo#ExactPosition> .

<http://purl.uniprot.org/position/22571011794154798tt27> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://biohackathon.org/resource/faldo#Position> .

<http://purl.uniprot.org/position/22571011794154798tt32> <http://biohackathon.org/resource/faldo#reference> <http://purl.uniprot.org/isoforms/P07465-1> .

<http://purl.uniprot.org/position/22571011794154798tt32> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://biohackathon.org/resource/faldo#ExactPosition> .

<http://purl.uniprot.org/position/22571011794154798tt32<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://biohackathon.org/resource/faldo#Position> .

Il grafo della pagina precedente, espresso come triple RDF

I prefissi

- Tutti gli elementi delle triple, nel grafo di esempio, sono costituiti da URI (o IRI)
  - Gli URI sono *indirizzi http*
- Alcuni indirizzi hanno delle parti in comune
  - Stesso *namespace: http://purl.uniprot.org/uniprot/... oppure e http://purl.uniprot.org/range/...*
- Per rendere la rappresentazione più leggibile, il namespace può essere sostituito da un *prefisso*:

<http://purl.uniprot.org/range/22571011794154798tt27tt32> →

<range: 22571011794154798tt27tt32>

<http://purl.uniprot.org/position/22571011794154798tt27 →

<position:22571011794154798tt27

Perché usare gli IRI?

→ un IRI permette di identificare una risorsa (cliccare http://purl.uniprot.org/range/22571011794154798tt27tt32)

Linked Data: principi cardine

Usare gli URI per denominare (quindi identificare) le entità.

Usare URI HTTP in modo che queste entità possano essere trovate (o interpretate, o "de-referenziate").

Fornire informazioni sulla risorsa quando viene richiesta usando standard come RDF, SPARQL, ecc.

Nel contesto di un'entità, riferirsi sempre alle altre entità usando i loro URI HTTP quando si pubblicano i dati su Web.

Informazioni sulla risorsa

- Non solo la risorsa deve essere de-referenziata (= all'URI della risorsa trovo la risorsa stessa), ma insieme alla risorsa deve essere fornita una descrizione della risorsa stessa.
- La descrizione può essere fornita ad esempio via SPARQL o RDF all'URI della risorsa
  - Vedi esempio della slide precedente: l'IRI rimanda a una query SPARQL
- Dovrebbe includere relazioni con altre risorse, indicate via i loro URI.

Usare vocabolari

- Per descrivere una certa entità, la tripla si basa sull'utilizzo di vocabolari condivisi
  - Invece di «inventare» ogni volta i termini per descrivere le entità di cui parla la tripla, meglio usare tutti gli stessi termini!

<Bob><http://www.my_vocab#like><The Mona Lisa>.

<Bob><http://www.likesanddislikes/loves/adore><The Mona Lisa>.

Meglio concordare in anticipo su un unico termine:

<Bob><http://xmlns.com/foaf/0.1/topic_interest ><The Mona Lisa>.

Nel nostro esempio:

<http://purl.uniprot.org/position/2257101179415479 8tt32>

<http://biohackathon.org/resource/faldo#reference>

<http://purl.uniprot.org/isoforms/P07465-1> .

I vocabolari e i prefissi nell'esempio

Prefisso	Vocabolario
• annotation	http://purl.uniprot.org/annotation/
• range	http://purl.uniprot.org/range/
• position	http://purl.uniprot.org/position/
• faldo	http://biohackathon.org/resource/faldo#

… e molti altri, di cui alcuni di uso generale (SKOS, FOAF, ecc.)

Rappresentazione RDF: i limiti

- La rappresentazione dei dati in formato RDF è adatta a descrivere i dati, ma non dice nulla sul significato dei termini che sono usati per descriverli
- Per aggiungere una semantica ai termini, è necessario utilizzare RDF e OWL

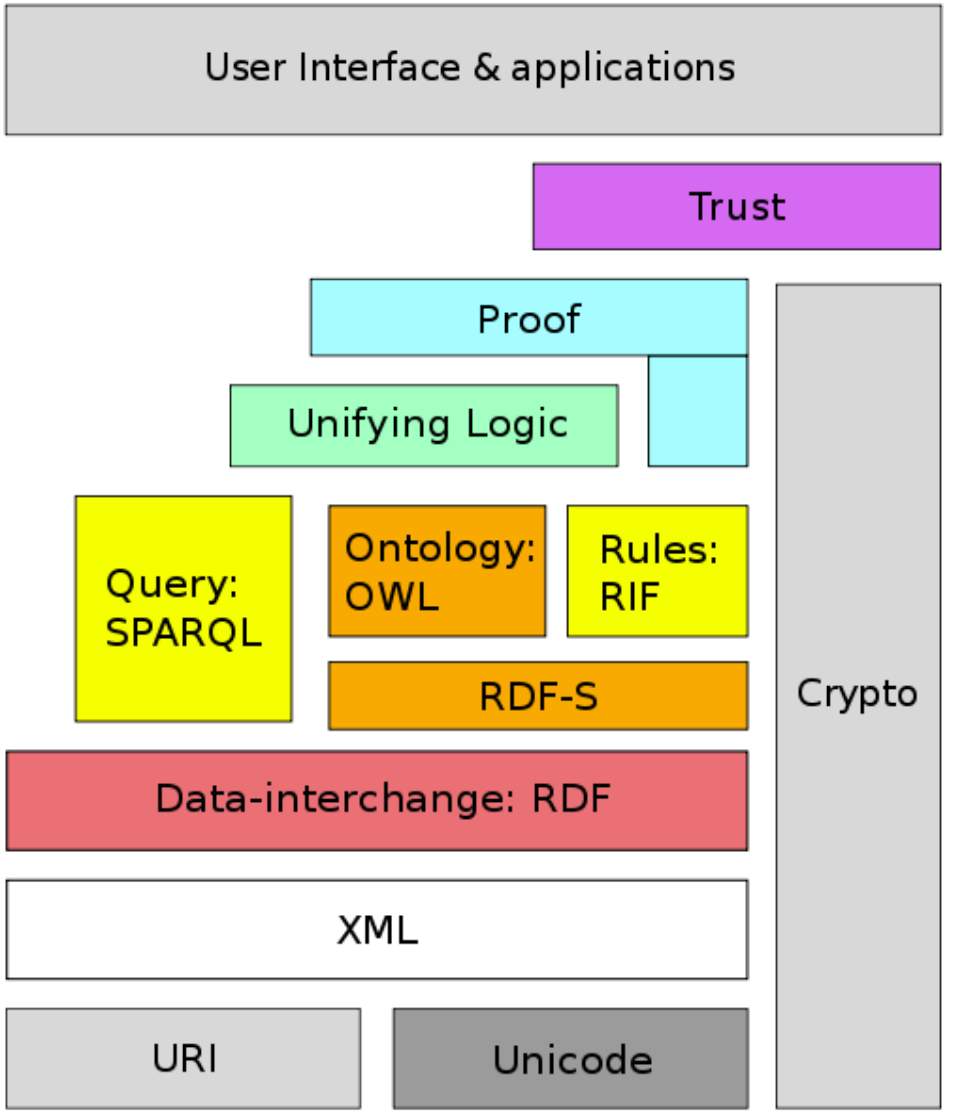
Serializzazione di RDF

- RDF/XML → può avvalersi degli strumenti esistenti per XML
- Turtle → leggibilità e compatibilità con SPARQL
- N-triples → portabilità dei dati
- Json LD → comunicazione client server via API

https://ontola.io/blog/rdf-serialization-formats/

Schemi RDF (RDF-S)

- RDF permette di creare un grafo per descrivere risorse, ma non fornisce nessuna informazione sulle entità descritte né sul vocabolario utilizzato per descriverle.
- La descrizione del vocabolario è affidata al livello successivo, Schemi RDF (RDF-S).
- Attraverso il linguaggio degli schemi RDF è possibile descrivere un semplice vocabolario di tipi entità (classi) e relazioni tra di esse (proprietà)



Semantic Web Layer Cake, 2013

RDFS: Classi e proprietà

Con RDFS si può:

- Definire **classi** e **proprietà**
- Creare **gerarchie** di classi e proprietà

Si definisce *property-centric* perché il focus è sulle *proprietà*:

- le classi compaiono nella definizione delle proprietà, ma non viceversa

→ questo rende possibile aggiungere proprietà senza modificare le classi (estensibilità)

Esempio: Una proprietà

(formato Turtle)

```
:begin rdf:type owl:ObjectProperty ;

        rdfs:comment "The inclusive beginning of a position. Also
        known as start."^^xsd:string ;

        rdfs:label "begin"^^xsd:string ;

        rdfs:range :Position .
```

http://biohackathon.org/resource/faldo.ttl

Esempio: Una proprietà

(formato Turtle)

```
:begin rdf:type owl:ObjectProperty ;

        rdfs:comment "The inclusive beginning of a position. Also
        known as start."^^xsd:string ;

        rdfs:label "begin"^^xsd:string ;

        rdfs:range :Position .
```

http://biohackathon.org/resource/faldo.ttl

Esempio: classi

```
:ExactPosition rdf:type owl:Class ;

                rdfs:comment "A position that is exactly known."^^xsd:string ;

                rdfs:label "Exact position"^^xsd:string ;

                rdfs:subClassOf :Position ;
```

i nomi delle classi sono maiuscoli

http://biohackathon.org/resource/faldo.ttl

Esempio: sottoclassi

```
:ExactPosition rdf:type owl:Class ;

        rdfs:comment "A position that is exactly
        known."^^xsd:string ;

        rdfs:label "Exact position"^^xsd:string ;

        rdfs:subClassOf :Position ;
```

http://biohackathon.org/resource/faldo.ttl

Esempio: proprietà e classi

(formato Turtle)

```
:begin rdf:type owl:ObjectProperty ;

        rdfs:comment "The inclusive beginning of a position. Also
        known as start."^^xsd:string ;

        rdfs:label "begin"^^xsd:string ;

        rdfs:range :Position .
```

http://biohackathon.org/resource/faldo.ttl

Risorse (o individui) e classi

range:2257101179415479 8tt27tt32

rdf:type

faldo:Region .

position:225710117941547 98tt27

rdf:type

faldo:Position

http://biohackathon.org/resource/faldo.ttl

Inferenze RDF

- RDF e RDFs permettono di fare alcune semplici inferenze

- La tripla

  <http://purl.uniprot.org/range/22571011794154798tt27tt32>
  <http://biohackathon.org/resource/faldo#begin>
  <http://purl.uniprot.org/position/22571011794154798tt27>

- Permette di inferire che <http://purl.uniprot.org/range/22571011794154798tt27tt32> appartiene (*rdf:type* oppure *a*) alla classe Position in virtù della definizione di :begin :

  :begin rdfs:range :Position .

http://biohackathon.org/resource/faldo.ttl

Vocabolari e *Knowledge graphs*

- Vocabolario RDF
  - La definizione di un insieme di classi e proprietà costituisce un vocabolario
  - Il vocabolario viene pubblicato a un certo IRI perché possa essere riferito da tutti

- Uno o più vocabolari possono essere usati per descrivere i dati
  - La descrizione di risorse in riferimento a uno o più vocabolari (es. Monna Lisa o <http://purl.uniprot.org/range/22571011794154798tt27tt32>), con proprietà che collegano tra di loro le risorse è denominata Knowledge Graph

- Classi e proprietà → Vocabolario
- Triple → Knowledge graph

Cosa manca al dizionario

- Non si può dire che due classi sono classi *disgiunte*, cioè non hanno elementi in comune
- Non si può dire che la relazione *precede* è transitiva
- Non si può dire che esistono *individui*
  - Esistono solo «risorse», cioè IRI creati esternamente al linguaggio
- Ci serve un linguaggio per ONTOLOGIE

Ontologie computazionali

- Le ontologie computazionali si basano su linguaggi che permettono di descrivere formalmente
  - Le caratteristiche delle classi
  - Le caratteristiche delle relazioni tra le classi
- Questi linguaggi permettono alle macchine di fare inferenze sui concetti e a noi di avere certezza della validità di queste inferenze

Definizione di ontologia

"An engineering artifact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary itself" (Guarino, 1998).

N. Guarino, Formal Ontology and Information Systems. In Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press, pp. 3-15

OWL (OWL2)

OWL 2 (Web Ontology Language) è un linguaggio per creare ontologie per il Web Semantico con un significato definito formalmente.

Le ontologie scritte in OWL 2 contengono **classi**, **proprietà**, **individui**, e **letterali**; sono scritte in documenti il cui formato è definito dal Web Semantico.

Le ontologie OWL 2 possono essere utilizzate in associazione con documenti RDF, anzi sono essere stesse normalmente codificate come documenti RDF.

Semantica di OWL: note

Il linguaggio OWL è una Logica Descrittiva (Description Logic)

- Le logiche descrittive nascono negli anni 80 attingendo elementi delle Reti Semantiche e della Frame Theory
- Le logiche descrittive (o terminologiche) sono orientate alla costruzione di definizioni (=classi) che permettano di classificare le entità di un certo dominio (=individui) sulla base delle loro caratteristiche (=proprietà)

La semantica di OWL si basa sulla Teoria degli Insiemi

- Se una classe è sottoclasse di un'altra, tutti gli individui che appartengono alla prima appartengono anche alla seconda
- Se due classi sono equivalenti, tutti gli individui membri prima sono anche membri della seconda

Struttura di un documento OWL

OWL 2 *non* fornisce strumenti che descrivano in maniera prescrittiva la struttura di un documento OWL.

In particolare, non c'è modo per specificare che una determinata informazione (per esempio, il codice fiscale di una persona) deve essere necessariamente presente.

Elementi dell'ontologia

- Entità: gli elementi che si riferiscono al mondo reale: il concetto di persona, ma anche John e Jack

- Assiomi: le asserzioni generali contenute nell'ontologia (per esempio, il concetto di persona è un concetto più specifico di quello di essere vivente)

- Espressioni: combinazioni di entità che vanno a formare entità più complesse
  - Relativi alle classi e alle proprietà

Tipi di entità

In OWL 2, si descrivono

- Gli oggetti come **individui**
- Le categorie come **classi**
- Le relazioni come **proprietà**.

Le proprietà sono suddivise in:

- **Object properties** che collegano individui ad individui (come una persona al suo coniuge)
- **Datatype properties** che assegnano un dato a un individuo (per esempio l'età a una persona)
- **Annotation properties** che contengono commenti e descrizioni sulle entità

Class axioms: relazioni tra classi

«OWL 2 provides axioms that allow **relationships** to be established between class expressions:»

- The SubClassOf axiom allows one to state that each instance of one class expression is also an instance of another class expression, and thus to construct a hierarchy of classes.
- The EquivalentClasses axiom allows one to state that several class expressions are equivalent to each other.
- The DisjointClasses axiom allows one to state that several class expressions are pairwise disjoint — that is, that they have no instances in common.
- The DisjointUnion class expression allows one to define a class as a disjoint union of several class expressions and thus to express covering constraints.

https://www.w3.org/TR/owl2-syntax/#Class_Expressions

In sintesi

In OWL si distinguono *classi*, *individui* e *proprietà*

Class expressions (e *property expressions*) permettono di descrivere classi (e proprietà)

mediante operatori booleani, quantificazione, restrizioni su proprietà, ecc.

Gli *assiomi di classe* permettono di stabilire relazioni tra classi

sottoclasse
equivalenza
unione
partizione
(con effetti rilevanti per il ragionamento automatico)

Classi e sottoclassi

(formato Turtle)

- Si definisce una classe dichiarando che essa appartiene al tipo "Classe" di OWL

```
obo1:GO_0008150     rdf:type          owl:Class ;
                    rdfs:label        "biological_process" .

obo1:GO_0040007     rdf:type          owl:Class ;
                    rdfs:subClassOf   obo1:GO_0008150 ;
```

Gene Ontology: http://purl.obolibrary.org/obo/go.owl

Classi equivalenti

Due classi possono essere dichiarate equivalenti → tutti gli individui che appartengono a una classe appartengono anche all'altra

Person e Human sono classi equivalenti

:Person owl:equivalentClass :Human .

Person in FOAF e Person in Schema.org sono classi equivalenti

foaf:Person owl:equivalentClass schema:Person

E classi disgiunte

Le classi possono essere dichiarate disgiunte

```
obo1:GO_0040007     rdf:type              owl:Class ;
                    rdfs:label            "growth" ;
                    owl:disjointWith      obo1:GO_0044848 ;

obo1:GO_0044848     rdfs:label            "biological phase" .
```

→ Due classi disgiunte non possono avere individui in comune

Gene Ontology: http://purl.obolibrary.org/obo/go.owl

Object properties

- Le ObjectProperties collegano classi con classi
- Possono essere definite rispetto alle loro caratteristiche.
- Per esempio, è possibile dire
  - che una proprietà è l'inverso di un'altra
  - che una proprietà è transitiva

```
obo1:BFO_0000050 rdf:type owl:ObjectProperty ;
                 owl:inverseOf obo1:BFO_0000051 ;
                 rdf:type owl:TransitiveProperty ;
                 rdfs:label "part of" .
```

Gene Ontology: http://purl.obolibrary.org/obo/go.owl

Data properties

Le data property hanno come domain una classe e come range un tipo di dato

```
faldo:position      rdf:type          owl:DatatypeProperty ;
                    rdfs:label        "position"^^xsd:string ;
                    rdfs:domain       :ExactPosition ;
                    rdfs:range        xsd:nonNegativeInteger
```

http://biohackathon.org/resource/faldo.ttl

Restrizioni

- Le restrizioni sono uno dei meccanismi principali per definire nuove classi a partire da quelle esistenti

- Due tipi di restrizioni:

  - Restrizioni su **classi** mediante operatori insiemistici (intersezione-and, unione-or, complemento-not)

  - Restrizioni poste su **proprietà** (esistenziale, universale, sulla cardinalità)

Quantificatori e proprietà

someValuesFrom esprime la quantificazione esistenziale

```
:Parent  owl:equivalentClass  [
         rdf:type          owl:Restriction ;
         owl:onProperty      :hasChild ;
         owl:someValuesFrom  :Person
 ] .
```

→someValuesFrom = quantificatore esistenziale

→Parent è uguale a chi abbia come figlio (hasChild) una persona (Person)

Manchester syntax: `Parent equivalentTo hasChild some Person`

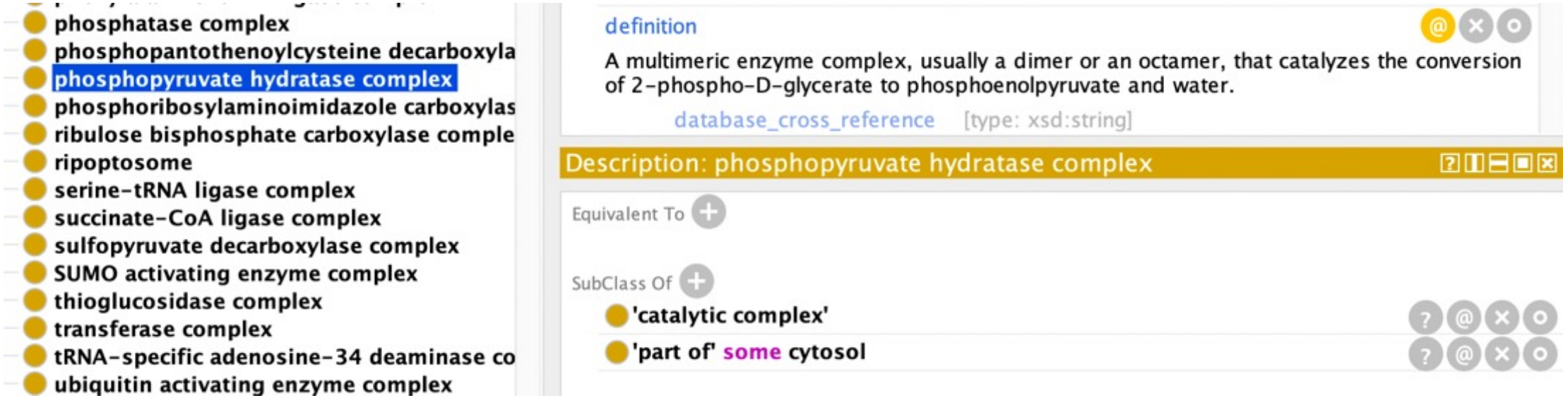
https://www.w3.org/TR/owl2-syntax/#Class_Expressions

Esempio in Gene Ontology

```
obo1:GO_0000015 rdf:type owl:Class ;
        rdfs:subClassOf obo1:GO_1902494 ,
                [ rdf:type owl:Restriction ;
                  owl:onProperty obo1:BFO_0000050 ;
                  owl:someValuesFrom obo1:GO_0005829
                ] ;
          obo1:IAO_0000115 "A multimeric enzyme complex, usually a dimer or
          an octamer, that catalyzes the conversion of 2-phospho-D-glycerate to
          phosphoenolpyruvate and water." ;
          rdfs:label "phosphopyruvate hydratase complex" .
```

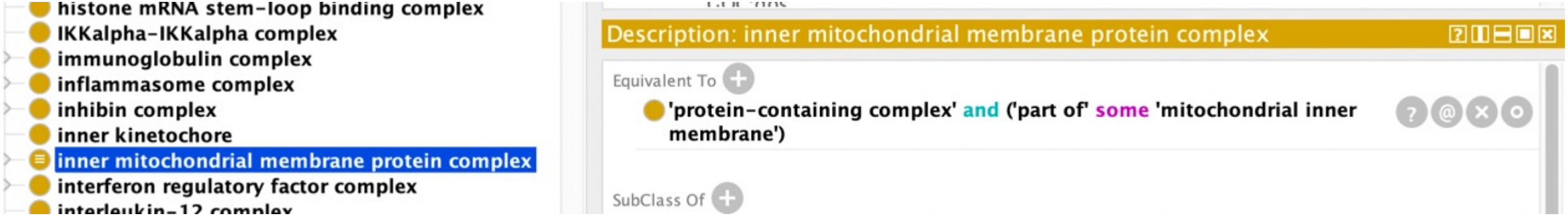
Gene Ontology: http://purl.obolibrary.org/obo/go.owl

Esempio in Protégé



Classi definite

- Create mediante un assioma di equivalenza tra una classe e una class expression
- Associare proprietà necessarie e sufficienti alla classe



Classe equivalente: la definizione

```
obo1:GO_0098800 rdf:type owl:Class ;
        owl:equivalentClass [ owl:intersectionOf ( obo1:GO_0032991
                                [ rdf:type owl:Restriction ;
                                  owl:onProperty obo1:BFO_0000050 ;
                                  owl:someValuesFrom obo1:GO_0005743
                                ]
                              ) ;
                               rdf:type owl:Class
                            ] ;
```

Gene Ontology: http://purl.obolibrary.org/obo/go.owl

OWL: i metadati delle classi

```
obo1:GO_0106210 rdf:type owl:Class ;
        rdfs:subClassOf obo1:GO_0016106 ,
                obo1:GO_0034312 ,
                obo1:GO_0043386 ,
                obo1:GO_0051762 ,
                obo1:GO_1901362 ,
                obo1:GO_1902653 ;
        oboInOwl:created_by "hjd" ;
        oboInOwl:creation_date "2019-05-13T17:30:41Z" ;
        oboInOwl:hasExactSynonym "culmorin anabolism" ,
                        "culmorin biosynthesis" ,
                        "culmorin formation" ,
                        "culmorin synthesis" ;
        oboInOwl:hasOBONamespace "biological_process" ;
        oboInOwl:id "GO:0106210" ;
        rdfs:label "culmorin biosynthetic process" .
```

Gene Ontology: http://purl.obolibrary.org/obo/go.owl

Ontologie fondazionali

- Basic Formal Ontology: ispirata dal lavoro del filosofo Barry Smith
- https://github.com/bfo-ontology/BFO/wiki
- Orientata alle scienze dure e all'ambito biomedico, frutto della ricerca biomedica: fisica, chimica, biologia, medicina, neuroscienze
  - Non contiene concetti relativi ai domini applicativi ma solo gli elementi fondazionali per definirli
  - Riferita anche da GO

GFO 2: Burek, P., Loebe, F., & Herre, H. (2020). Towards GFO 2.0: Architecture, modules and applications. In *Formal Ontology in Information Systems* (pp. 32-45). IOS Press.

Modellazione: prime osservazioni

- Non confondere la relazione di sottoclasse con la relazione mereologica part-of (parte-tutto)!
  - Si deve usare una proprietà per rappresentare le relazioni parte-tutto (X fa parte di Y), non un assioma di sottoclasse (X è una sottoclasse di Y)

- L'ontologia deve essere leggibile
  - Organizzare le classi in una *gerarchia*
  - Attribuire alle entità *nomi* che hanno senso secondo il senso comune o per l'esperto
  - Associare *domain* e *range* alle proprietà