Ontologías y Web Semántica

RDF avanzado

Pablo R. Fillottrani

Depto. Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

marzo 2019

RDF avanzado

Objetivos

- RDF permite expresar propiedades simples sobre recursos, pero las comunidades de usuarios también necesitan definir vocabularios para usar en esas propiedades
- RDF puro no provee formas de definir clases y propiedades específicas de un dominio
- en el documento RDF Vocabulary Description Language se definen extensiones para ello
- todos los términos definidos en este documento tienen URIrefs que comienzan con `http://www.w3.org/2000/01/rdf-schema#` usualmente asociado al prefijo `rdfs`

- RDF Schema no define vocabulario especfíico de un dominio, pero introduce elementos para poder definirlo
- estos elementos permiten establecer cuáles URIrefs son clases, propiedades, y cómo estas clases y propiedades deben estar relacionadas
- en resumen, RDF Schema permite introducir un sistema de tipos para RDF
- pero a diferencia de los lenguajes de programación, los tipos de RDF Schema no definen una camisa de fuerza donde la información debe ser adaptada, sino que proveen información adicional sobre los recursos que describen
- esta información puede luego ser usada en los servicios de inferencia

Lenguaje

- clases: `rdfs:Class`

```
#estudiante  rdf:type  #rdfs:Class .
ex:juan rdf:type #estudiante.
```

- jerarquía de clases: `rdfs:subClassOf`

```
#estudiante  rdfs:subClassOf #person.
```

- propiedades: `rdf:Property`

```
 #nombre  rdf:type     rdf:Property .
```

- jerarquía de propiedades: `rdfs:subPropertyOf`

```
#madre  rdfs:subPropertyOf  #progenitor .
```

- asociación de propiedades con clases: `rdfs:domain`

```
#nombre  rdfs:domain  #person .
```

significa que la propiedad nombre sólo se aplica a personas

- asociación de valores a propiedades: `rdfs:range`

```
#nombre  rdfs:range  xsd:string  .
```

significa que el resultado de nombre siempre es una cadena

- ninguna especificación de rango libera la propiedade; múltiples especificaciones restringen la propiedad a las instancias de la intersección

- literales: `rdfs:Literal`

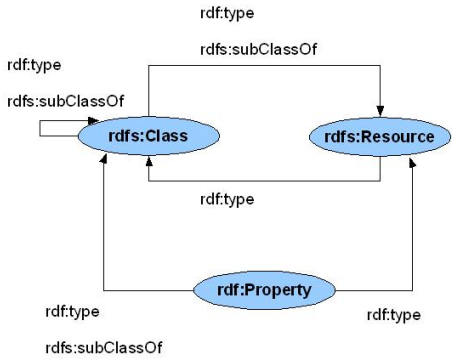
```
:juan  #nombre "Juan" .
:juan  #nombre  _:X .
_:X  rdf:type  rdfs:Literal .
```

- tipos de datos: `rdfs:Datatype`
- todos los tipos de datos XML Schema son instancias de `rdfs:Datatype`, aunque algunos como `QName` no son apropiados para ser usados en RDF
- también pueden usarse otros tipos de datos no XML Schema

Principios de RDFS

- todos los recursos son instancias de `rdfs:Resource`
- las clases son recursos que describen conjuntos de recursos
- se pueden definir jerarquías de clases, y todas las clases son instancias de `rdfs:Class`
- las propiedades son recursos que tienen un rango y un dominio
- se pueden definir jerarquías de propiedades, y todas las propiedades son instancias de `rdfs:Property`

Relación entre el vocabulario de RDFS



Resumen de vocabulario en RDF

- clases:

  `rdf:Property rdf:Statement rdf:XMLLiteral`
  `rdf:Seq rdf:Bag rdf:Alt rdf:List`
- propiedades:

  `rdf:type rdf:subject rdf:predicate`
  `rdf:object rdf:first rdf:rest`
  `rdf:_n rdf:value`
- recursos:

  `rdf:nil`

Resumen de vocabulario de RDFS

- clases

  `rdfs:Resource rdfs:Class rdfs:Literal`
  `rdfs:Datatype rdfs:Container`
  `rdfs:ContainerMembershipProperty`
- propiedades:

  `rdfs:domain rdfs:range rdfs:subPropertyOf`
  `rdfs:subClassOf rdfs:member`
  `rdfs:seeAlso rdfs:isDefinedBy rdfs:comment`
  `rdfs:label`

Metadatos

- los metadatos son datos sobre los datos
- en RDFS es posible definir metadatos sobre recursos, usando algunos de los siguientes términos
  - `rdfs:comment` descripción del recurso orientada a un humano
  - `rdfs:label` nombre del recurso orientado al humano
  - `rdfs:seeAlso` información adicional sobre el recurso
  - `rdfs:isDefinedBy` información sobre el autor

Conceptos básicos

- sean G, G' grafos RDF (conjuntos de triplas), entonces
  - G' es un subgrafo de G si y solo si $G' \subseteq G$
  - G' es una instancia de G si y solo si G' se obtiene a partir de G reemplazando un nodo en blanco por otro nodo en blanco, un literal o un URI
  - G es un grafo totalmente instanciado si no contiene nodos en blanco
- la semántica formal de RDF se define usando la teoría de modelos

Interpretación

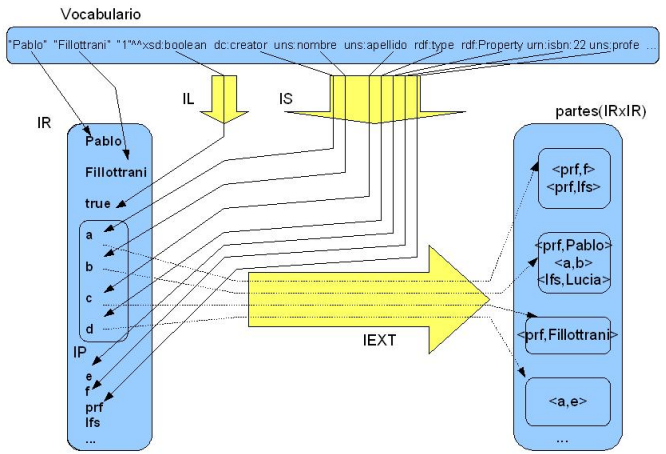
- una interpretación I de un vocabulario RDF está compuesta por
  - un conjunto no vacío I_R de recursos, llamado dominio
  - un conjunto I_P de propiedades
  - un mapeo I_{EXT} de I_P al conjunto de partes de $I_R \times I_R$
  - un mapeo I_S de URIrefs a $I_R \cup I_P$
  - un mapeo I_L de literales tipados a I_R
  - un subconjunto distinguido L de I_R que contiene a los valores de todos los literales

- significado de una expresión F en un grafo totalmente instanciado G en una interpretación I:
  - si E es un literal `"xxx"` entonces $I(E) =$ `aaa`
  - si E es un literal `"xxx"@ttt` entonces $I(E) =<$ `aaa,ttt` $>$
  - si E es un literal tipado, entonces $I(E) = I_L(E)$
  - si E es un URIref, entonces $I(E) = I_S(E)$
  - si E es una tripla `s p o .` totalmente instanciada, entonces $I(E) =$ **verdadero** si `s,p,o` pertenecen al vocabulario y $<$ `s,o` $>\in I_{EXT}(I_P(p))$; sino $I(E) =$ **falso**
  - si E es un grafo totalmente instanciado entonces $I(E) =$ **falso** si existe en E una tripla tal que su interpretación es **falso**; sino $I(E) =$ **verdadero**

Ejemplo

- sea el siguiente grafo escrito en N-triples:

```
# Mi libro en RDF
<urn:isbn:0-345-678> dc:creator uns:profe .
uns:profe uns:nombre "Pablo" .
uns:profe uns:apellido "Fillottrani" .
uns:profe uns:posgrado "1"^^xsd:boolean .
```



Modelos y Consecuencia Lógica

- un modelo de un grafo G es una iterpretación I tal que $I(G) =$ **verdadero**. Se escribe $I \models G$
- el conjunto de modelos de un grafo G se nota $\mathscr{M}(G)$
- esta notación es trivial de extender a conjuntos de grafos
- un grafo G es consecuencia lógica de un conjuntos de grafos S si y solo si $\mathscr{M}(S) \subseteq \mathscr{M}(G)$. Se escribe $S \models G$
- la relación de consecuencia lógica se puede extender a conjuntos de grafos, y es transitiva

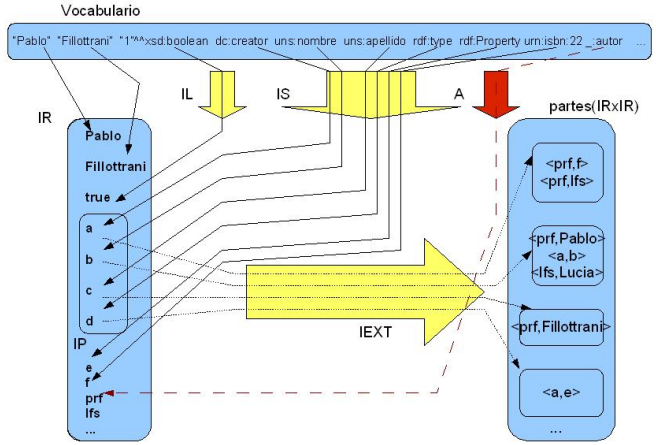
Nodos en blanco

- un nodo en blanco se interpreta como una variable existencial en lógica clásica
- para asignar un significado a los nodos en blancos, se extiende una interpretación I con un mapeo parcial A que va de los nodos en blancos a elementos del dominio
- el concepto de interpretación se extiende de forma que si E es un nodo en blanco y $A(E)$ está definido, entonces $[I+A](E) = A(E)$
- entonces si E es un grafo RDF, $I(E) =$ **verdadero** si existe un mapeo A' tal que $[I+A'](E) =$ **verdadero**; sino $I(E) =$ **falso**

Ejemplo

- sea el siguiente grafo escrito en N-triples:

```
# Mi libro en RDF
<urn:isbn:0-345-678> dc:creator _:autor .
_:autor uns:nombre "Pablo" .
_:autor uns:apellido "Fillottrani" .
_:autor uns:posgrado "1"^^xsd:boolean .
```



Lemas

Lema (del grafo vacío)

El grafo vacío es consecuencia lógica de cualquier grafo.

Lema (del subgrafo)

Un grafo tiene como consecuencia lógica todos sus subgrafos.

Lema (de la instancia)

Una instancia de un grafo tiene como consecuencia lógica al grafo original.

- la operación de mezcla de grafos RDF se define como la unión de triplas, reemplazando posibles nodos en blanco en común

Lema (de la mezcla)

La mezcla de un conjunto de grafos es consecuencia lógica del conjunto, y tiene como consecuencia lógica a todos los miembros del conjunto

- el principal resultado para RDF es el siguiente lema

Lema (de interpolación)

Un conjunto de grafos S tiene como consecuencia lógica un grafo E si y solo si existe un subgrafo de S que es una instancia de E

- este lema muestra que el proceso de encontrar una sustitución de nodos en blanco de E tal que está incluído en S es un proceso sano y completo para ver si $S \models E$

Interpretación del vocabulario RDF

- RDF introduce vocabulario que tiene un significado específico
- una interpretación RDF es una interpretación I que satisface las siguientes condiciones extra sobre el vocabulario RDF:
  - si $x \in I_P$ si y solo si $< x, I(\texttt{rdf:Property}) > \in I_{EXT}(\texttt{rdf:type})$
  - condiciones que deben cumplir los literales con tipo `rdf:XMLLiteral`
  - incluye un conjunto de triplas axiomáticas

Triplas axiomáticas para RDF

```
rdf:type   rdf:type  rdf:Property .
rdf:subject  rdf:type   rdf:Property .
rdf:predicate  rdf:type  rdf:Property .
rdf:object  rdf:type  rdf:Property .
rdf:first  rdf:type  rdf:Property .
rdf:rest  rdf:type  rdf:Property .
rdf:value  rdf:type  rdf:Property .
rdf:_1  rdf:type  rdf:Property .
rdf:_2  rdf:type  rdf:Property .
...
rdf:nil  rdf:type  rdf:List .
```

- no se incluyen definiciones semánticas para la reificación, contenedores y colecciones, con el objetivo de no limitar posibles implementaciones
- en base al concepto de interpretaciones RDF se puede definir el concepto de consecuencias lógicas RDF entre conjuntos de grafos
- para obtener las consecuencias lógicas RDF de un grafo, se transforma al grafo original siguiendo un conjuntos de reglas denominadas reglas de consecuencias RDF

Interpretación del vocabulario RDFS

- RDF Schema tiene un conjunto adicional de vocabulario, con mayores restricciones semánticas
- se define I_C como el conjunto de elementos c de I_R tal que la tripla

  `c  rdf:type  rdfs:Class .`

  pertenece a la interpretación
- $I_{CEXT}(y)$ es el conjunto de elementos r de I_R tal que $< \texttt{r}, \texttt{y} >$ pertence a $I_{EXT}(\texttt{rdf:type})$
- una interpretación RDFS es una interpretación RDF que cumple, entre otras, las siguientes condiciones:
  - $I_C = I_{CEXT}(I(\texttt{rdfs:Class}))$
  - $I_R = I_{CEXT}(I(\texttt{rdfs:Resource}))$
  - si $< x, y > \in I_{EXT}(\texttt{rdfs:domain})$ y $< u, v > \in I_{EXT}(x)$ entonces $u \in I_{CEXT}(y)$

Algunas triplas axiomáticas para RDFS

```
rdf:type  rdfs:domain  rdfs:Resource .
rdfs:domain  rdfs:domain  rdf:Property .
rdfs:range  rdfs:domain  rdf:Property .
rdfs:subPropertyOf rdfs:domain rdf:Property .
rdfs:subClassOf  rdfs:domain  rdfs:Class .
rdf:subject  rdfs:domain  rdf:Statement .
rdf:predicate  rdfs:domain  rdf:Statement .
rdf:object  rdfs:domain  rdf:Statement .
rdfs:member  rdfs:domain  rdfs:Resource .
rdf:first  rdfs:domain  rdf:List .
rdf:rest  rdfs:domain  rdf:List .
...
```

- en base al concepto de interpretaciones RDFS se puede definir el concepto de consecuencias lógicas RDFS entre conjuntos de grafos
- para obtener las consecuencias lógicas RDFS de un grafo, se transforma al grafo original siguiendo un conjuntos de reglas denominadas reglas de consecuencias RDFS

Objetivos

- SPARQL es un lenguaje de consulta para RDF, estilo SQL para base de datos relacionales
- es una recomendación W3C desde enero 2008
- su especificación consta de tres documentos:
  - SPARQL Query Language for RDF define la sintaxis y la semántica de las consultas
  - SPARQL Protocol for RDF define un protocolo para accesso remoto entre una aplicación que emita consultas SPARQL, y un servidor que le envíe resultados
  - SPARQL Query Results XML Format especifica un formato XML para los resultados de las consultas ASK y SELECT de SPARQL.

Sintaxis para consultas

- SPARQL usa para las consultas una sintaxis muy parecida a SQL
- cada consulta puede estar formada por las siguientes clausulas:
  - `PREFIX` define los prefijos para los espacios de nombres
  - `SELECT` identifica las variables que se retornarán como resultados
  - `FROM` nombra los grafos que serán consultados
  - `WHERE` patrón para la consulta en la forma de una lista de triplas

Ejemplo de consulta

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { <http://example.org/book/book1>
          dc:title ?title }
```

Clausula PREFIX

- PREFIX es el mecanismo para usar espacios de nombres
- consta de cero o más de las siguientes cláusulas

  ```
  PREFIX    <prefijo>: <URI>
  ```

- se puede definir un prefijo por defecto

  ```
  PREFIX    : <URI>
  ```

Clausula SELECT

- la consulta puede contener variables, cuyos nombres comienzan con ?. Ejemplo: ?apellido
- pero no necesariamente todas las variables son interesantes en el resultado
- la cláusula SELECT filtra aquellas variables que no aparacen en el resultado
- la sintaxis es

  ```
  SELECT <var1>,<var2>,...
  SELECT <var1>
  ```

Clausula FROM

- las consultas SPARQL se contestan en base a un grafo RDF
- la cláusula FROM selecciona este grafo para una dada consulta
- sintaxis

  ```
  FROM <URI>
  ```

- en caso de múltiples cláusulas FROM, se interpreta que la consulta se realiza sobre la mezcla de todos los grafos nombrados

Clausula `WHERE`

- la cláusula WHERE define el patrón del grafo que forma la respuesta a la consulta
- consta de una serie de tripletas en sintaxis Turtle, encerradas entre `{}`

  `WHERE { <patrón de tripleta> }`
- en el patrón también pueden aparecer nodos en blanco `_:nombre`, variables `?nombre`
- existe una clausula `OPTIONAL` para indicar patrones opcionales, y una clausula `FILTER` para imponer restricciones adicionales al patrón

Ejemplo

```
SELECT ?nombre
WHERE {
_:autor uns:nombre ?nombre .
_:autor dc:creator urn:isbn:222 .
}
```

Ejemplo

```
PREFIX : <http://example.org/data#>
SELECT ?predicado
WHERE { :john ?predicado :mary }
```

Ejemplo

```
PREFIX vc: <http://www.w3.org/vcard-rdf/3.0#>
PREFIX ont: <http://example.org/myOntology#>
SELECT ?y
WHERE {?x vc:FN "John Smith".
?x ont:casadoCon ?y. }
```

Ejemplo

```
PREFIX ex: <http://example.org/#>
SELECT ?persona, ?esposa
WHERE {?persona ex:edad ?edad .
OPTIONAL { ?persona ex:casadoCon
           ?esposa } }
```

Ejemplo

```
PREFIX ont: <http://example.org/myOntology#>
SELECT ?x
WHERE {?x ont:edad ?edad .
FILTER(?edad > 30)}
```

Semántica de las consultas

- una sustitución de variables se hace asignando valores como URIs, nodos en blanco o literales

  `?x=<http://www.example.org/#john>,`
  `?name= :a, ?firstName="John"`

- en las respuestas, no todas las variables tienen que estar necesariamente instanciadas

  `?persona=<http://example.org/#mary>,`
  `?esposa=`

- las respuestas son aquellas sustituciones de variables en el grafo patrón que lo transforman en un subgrafo del grafo base

Repositorios de Ontologías

- Open Ontology Repository `oor.net/`
- Ontohub `ontohub.org`
- Protégé Library `protegewiki.stanford.edu/wiki/Protege_Ontology_Library`
- Vivo Repository `swl.slis.indiana.edu/repository/`
- Bioportal `bioportal.bioontology.org/`
- OBO Foundry `www.obofoundry.org/`
- Linked Open Vocabularies `lov.linkeddata.es/dataset/lov/`
- DERI Vocabularies `vocab.deri.ie/`
- Ontologías de juguete `mowl-power.cs.man.ac.uk/2009/07/sssw/`