Taller de Lenguajes y Tecnologías de la Web Semántica.
Prueba Final - 2021
Solución

10 de diciembre de 2021

1. Considere un endpoint sparql. Asuma que Ud no conoce nada sobre su contenido.

   a) Escriba una consulta que le permita recuperar las uris de todas las clases usadas en ese sitio.
   SOLUCION:
   Por **usadas** se entiende que hay algún individuo asignado a esa clase. Por lo tanto con la siguiente consulta alcanza:

```
select ?c
where {
    [] a ?c.
}
```

   b) Escriba una consulta que le permita recuperar todas uris de las propiedades de ese sitio, con su dominio y rango si es que los tienen declarados. En caso que no tenga uno de los dos datos anteriores (dominio o rango), la propiedad igual debería aparecer en el resultado.
   SOLUCION:
   En este caso, interesan las propiedades usadas y su dominio y rango si existen. Esto lleva a usar dos `OPCIONAL` separados:

```
select ?p ?d ?r
where {
    [] ?p [].
    OPTIONAL {
        ?p rdfs:domain ?d.
    }
    OPTIONAL {
        ?p rdfs:range ?r.
```

```
    }
}
```

Si se utiliza un patrón de grafos sin `OPTIONAL` **SOLO** devuelve las propiedades que fueron usadas y tienen los dos datos: dominio y rango.

c) Escriba una consulta que le permita listar todas las uri de los grafos con nombre.
SOLUCION:
La claúsula **GRAPH** es la que itera sobre los grafos con nombre, por lo que las dos siguientes consultas deberían funcionar:

```
select ?g
where {
    GRAPH ?g {
        [] ?p [].
    }
}

select ?g
where {
    GRAPH ?g {}
}
```

La variable *?p* puede ser sustituida por un nodo blanco según el triplestore. La diferencia entre las dos consultas es que la segunda debería devolver incluso los grafos vacíos, mientras que la primera devuelve sólo grafos con contenido.
Las dos soluciones se consideran correctas.

d) Escriba una consulta `construct` que dadas las propiedades que tienen la propiedad `rdfs:domain` declarada, devuelve el conjunto de ternas que involucran los individuos en esa relación y reflejan la restricción que impone la regla de `rdfs:domain` en RDFS.
SOLUCION:
La regla de inferencia correspondiente al `rdfs:domain` obliga a que los individuos que aparezcan en la primer componente de la propiedad, deban pertenecer a la clase indicada como domain. Las ternas que reflejan eso se pueden generar con la siguiente consulta:

```
construct {
    ?obj a ?d
}
where {
    ?p rdfs:domain ?d.
    ?obj ?p [].
}
```

2. Considere el siguiente grafo que incluye esquema e instancias.

```
@prefix arq: <http://www.fing.edu.uy/inco/tsw/arquitectura#> .
@prefix ev: <http://www.fing.edu.uy/inco/tsw/eventos#> .
@prefix res: <http://www.fing.edu.uy/inco/tsw/resources/> .

    ev:Evento a owl:Class.
    arq:EdificiosFamosos a owl:Class.
    arq:EdificiosFamosos owl:hasKey ( ev:nombre ev:inicio ).

    res:fing  a arq:EdificiosFamosos;
            arq:nombre  "Facultad de Ingeniería".

    res:evento1 a ev:Evento;
                ev:nombre "Tiza 2021";
                ev:inicio "2021-11-1 20:00"^^xsd:datetime;
                ev:lugar  res:fing;
                ev:duracion "2 horas".

    res:evento2 a ev:Evento;
                ev:nombre "Premios 2021";
                ev:inicio "2021-11-1 20:00"^^xsd:datetime;
                ev:lugar  res:fing;
                ev:duracion "3 horas".
```

a) Considere además que el grafo está cargado en un triplestore **sin razonamiento**.Cuál sería el resultado de la siguiente consulta ?

```
select ?nombre
where {
    [] ev:lugar res:fing,
        ev:nombre ?nombre.
}
```

SOLUCION:
El resultado es:

”Tiza 2021”
”Premios 2021”

b) Considere ahora que está cargado en un triplestore **con el máximo nivel de razonamiento** activado. Cuál sería el resultado de la consulta anterior ? Justique su respuesta.
SOLUCION:
Con esa consulta, el resultado es exactamente el mismo.
La restricción `owl:hasKey` impone que si los valores de las propiedades indicadas en dos objetos son iguales, entonces hay un `owl:sameAs` entre los objetos. Por lo que **la ontología sigue siendo consistente** y no hay ningún error.
Supongamos que la consulta fuera distinta:

```
select ?s ?nombre
where {
    ?s ev:lugar res:fing,
        ev:nombre ?nombre.
}
```

El resultado de la parte anterior sería:

ev:evento1	”Tiza 2021”
ev:evento2	”Premios 2021”

En este caso particular, la segunda consulta daría el mismo resultado porque la clave está definida sobre `arq:EdificiosFamosos`. Supongamos que la clave estuviera definida sobre `ev:Evento` de la siguiente forma :

```
arq:Evento owl:hasKey ( ev:nombre ev:inicio ).
```

En este caso el resultado en el segundo caso tendría 4 tuplas al funcionar con un razonador:

ev:evento1	”Tiza 2021”
ev:evento1	”Premios 2021
ev:evento2	”Tiza 2021”
ev:evento2	”Premios 2021”