Geospatial data in RDF – GeoSPARQL

Presenter: Kostis Kyzirakos

Dept. of Informatics and Telecommunications
National and Kapodistrian University of Athens

TELEIOS

GeoSPARQL

GeoSPARQL is a recently completed OGC standard [Perry and Herring, 2012]

Functionalities **similar to stSPARQL**:

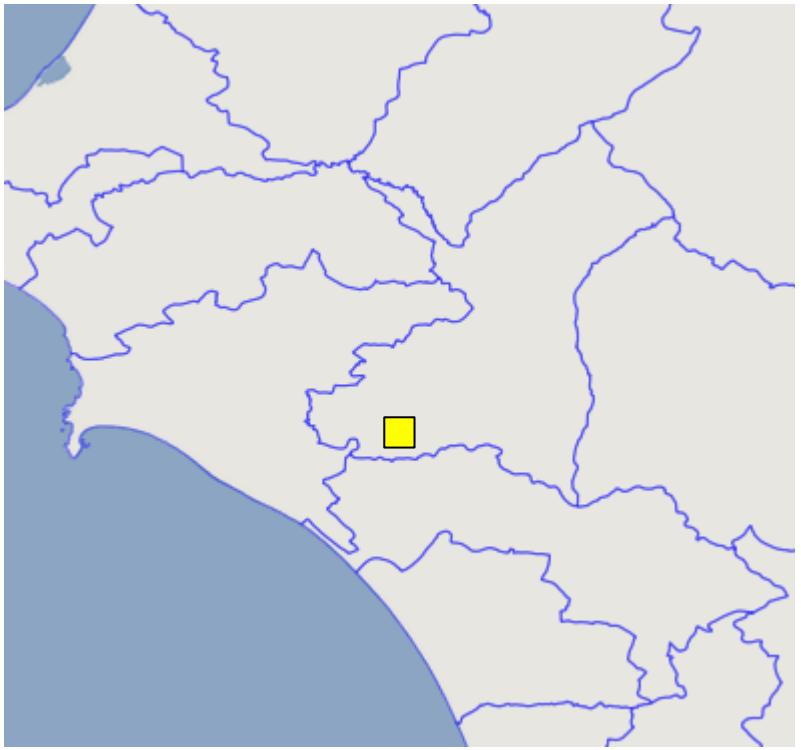
- Geometries are represented using **literals** similarly to stSPARQL.
- The same families of **functions** are offered for querying geometries.

Functionalities **beyond stSPARQL**:

- **Topological relations** can now be **asserted** as well so that reasoning and querying on them is possible.

Example in GeoSPARQL (1/2)

```
geonames:Olympia
   geonames:name "Ancient Olympia";
   rdf:type dbpedia:Community ;
   geo:hasGeometry ex:polygon1.
```



Spatial literal

```
ex:polygon1
    rdf:type geo:Polygon;
    geo:asWKT "POLYGON((21.5 18.5,23.5 18.5,
                        23.5 21,21.5 21,21.5 18.5))
               "^^sf:wktLiteral.
```

Spatial data type

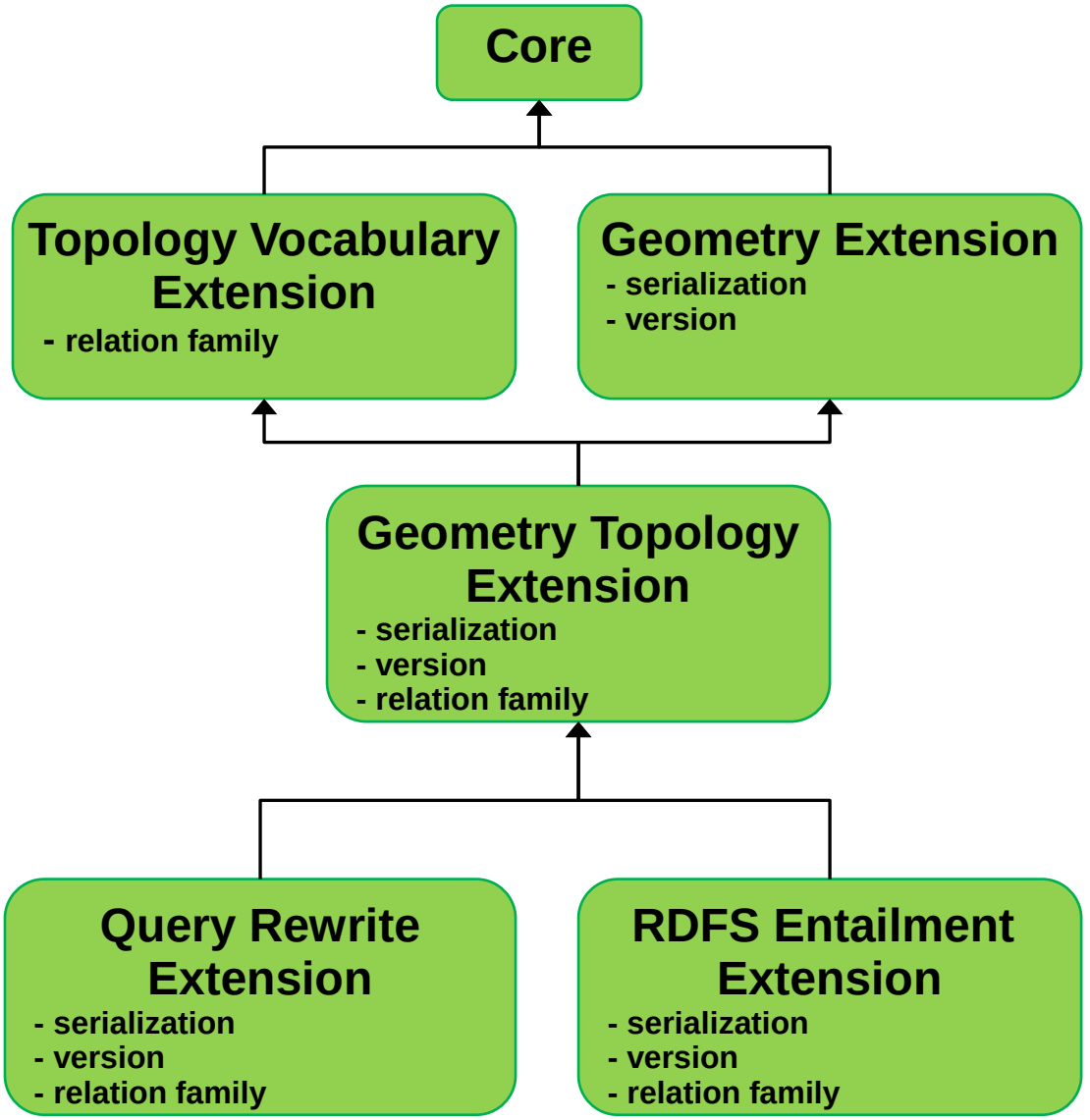
Example in GeoSPARQL (2/2)

```
gag:OlympiaMunicipality

   rdf:type gag:Municipality;

   rdfs:label "ΔΗΜΟΣ ΑΡΧΑΙΑΣ
               ΟΛΥΜΠΙΑΣ"@el;

   rdfs:label "Municipality of
               Ancient Olympia".
```

```
gag:olympiaMunicipality geo:sfContains geonames:olympia .
```

Asserted topological relation

GeoSPARQL Components



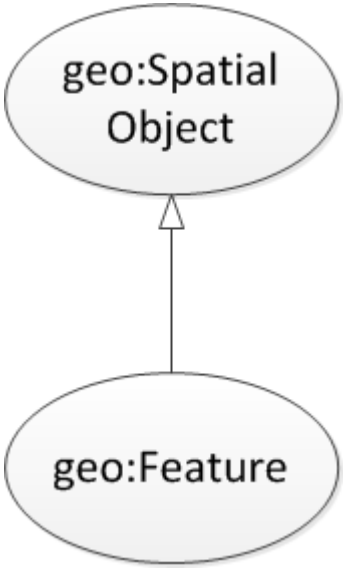
Parameters

- **Serialization**
  - WKT
  - GML
- **Relation Family**
  - Simple Features
  - RCC-8
  - Egenhofer

GeoSPARQL Core

Defines **top level classes** that provides users with vocabulary for modeling geospatial information.

- The class **geo:SpatialObject** is the top class and has as instances everything that can have a spatial representation.
- The class **geo:Feature** is a subclass of **geo:SpatialObject**. Feature is a domain entity that can have various **attributes** that describe **spatial and non-spatial** characteristics.



Example

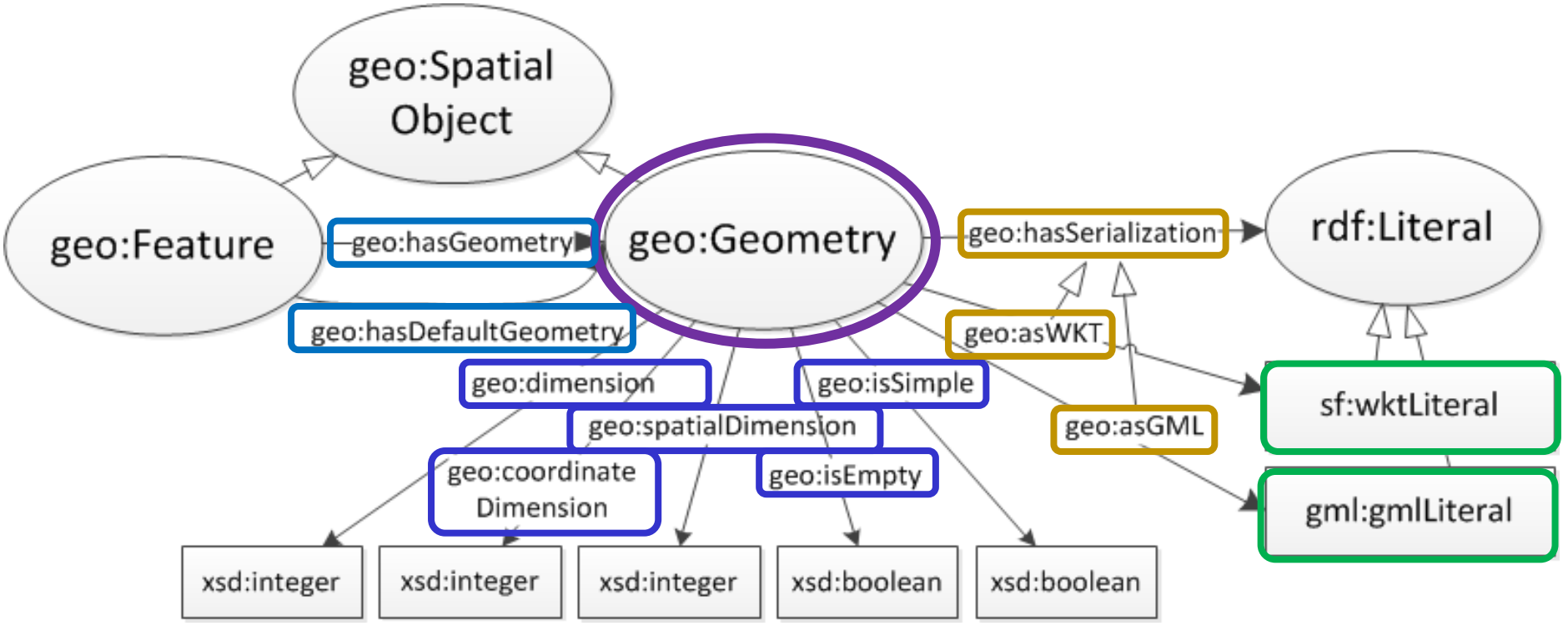
GeoSPARQL representation of the community of Ancient Olympia.

```
dbpedia:Community rdfs:subClassOf geo:Feature .
geonames:Olympia  geonames:name "Ancient Olympia";
                  rdf:type dbpedia:Community .
```

GeoSPARQL Geometry Extension

Provides vocabulary for asserting and querying information about geometries.

- The class **`geo:Geometry`** is a top class which is a superclass of all geometry classes.



Example

GeoSPARQL representation of the community of Ancient Olympia.

```
dbpedia:Community rdfs:subClassOf geo:Feature .
geonames:Olympia  geonames:name "Ancient Olympia";
                  rdf:type dbpedia:Community .


geonames:Olympia  geo:hasGeometry ex:polygon1.

ex:polygon1  rdf:type geo:Polygon;
             geo:isEmpty "false"^^xsd:boolean;
             geo:asWKT "POLYGON((21.5 18.5, 23.5
                        18.5, 23.5 21, 21.5 21,
                        21.5 18.5))"^^sf:wktLiteral.
```

Spatial data type

GeoSPARQL Geometry Extension

Spatial analysis functions

- **Construct new geometric objects from existing geometric objects**

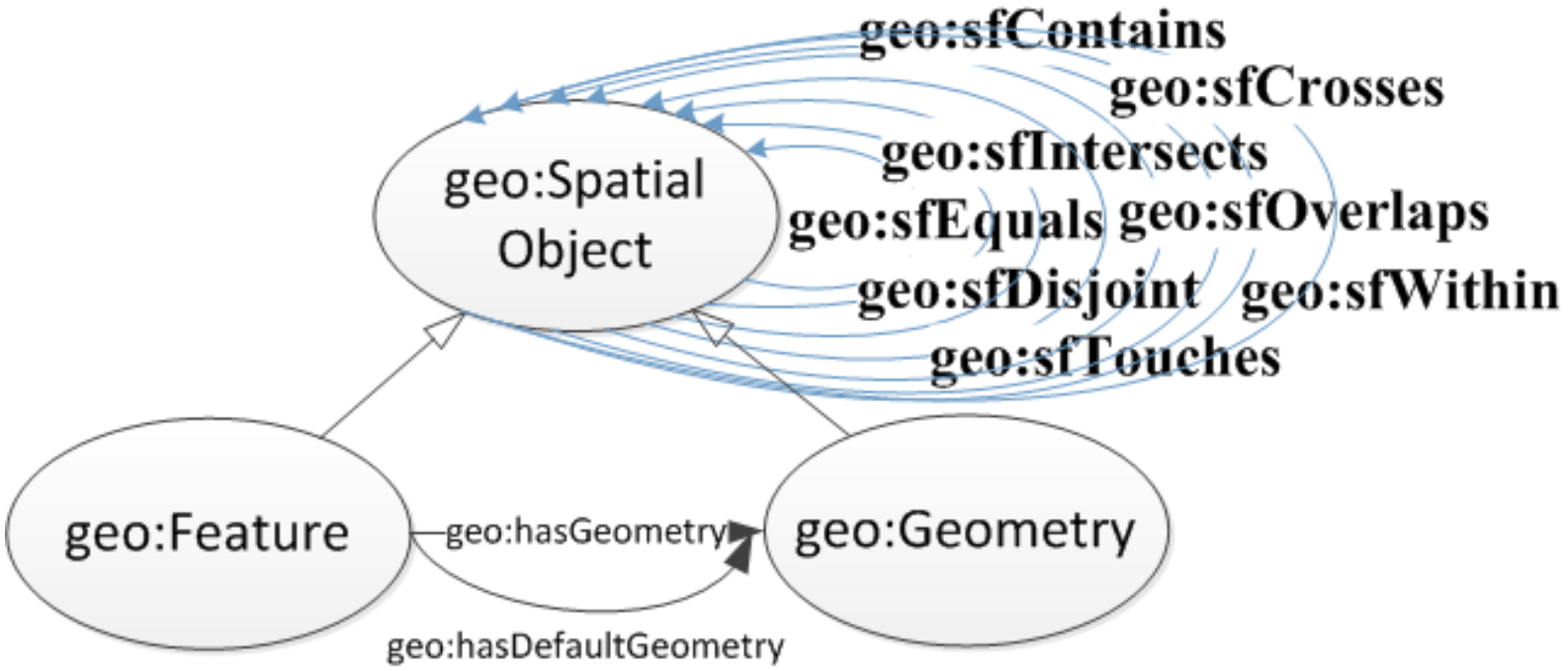
```
geof:boundary (geom1: ogc:geomLiteral): ogc:geomLiteral
geof:envelope (geom1: ogc:geomLiteral): ogc:geomLiteral
geof:intersection(  geom1: ogc:geomLiteral,
                    geom2: ogc:geomLiteral): ogc:geomLiteral
geof:union (  geom1: ogc:geomLiteral,
              geom2: ogc:geomLiteral): ogc:geomLiteral
geof:difference (   geom1: ogc:geomLiteral,
                    geom2: ogc:geomLiteral): ogc:geomLiteral
geof:symDifference (geom1: ogc:geomLiteral,
                    geom2:ogc:geomLiteral): ogc:geomLiteral
geof:buffer(geom: ogc:geomLiteral, radius: xsd:double,
            units: xsd:anyURI): ogc:geomLiteral
geof:convexHull(geom1: ogc:geomLiteral): ogc:geomLiteral
```

- **Spatial metric functions**

```
geof:distance(geom1: ogc:geomLiteral, geom2:
              ogc:geomLiteral, units: xsd:anyURI): xsd:double
```

GeoSPARQL Topology Vocabulary Extension

- The extension is parameterized by the family of topological relations supported.
  - Topological relations for simple features



  - The Egenhofer relations e.g., **`geo:ehMeet`**
  - The RCC-8 relations e.g., **`geo:rcc8ec`**

Example

```
gag:Olympia
  rdf:type gag:Community;
  geonames:name "Ancient Olympia"

gag:OlympiaBorough
  rdf:type gag:Borough;
  rdfs:label "Borough of
              Ancient Olympia".

gag:OlympiaMunicipality
  rdf:type gag:Municipality;
  rdfs:label "Municipality of
              Ancient Olympia".

gag:OlympiaBorough geo:sfContains geonames:Olympia .

gag:OlympiaMunicipality geo:sfContains
                        geonames:OlympiaBorough.
```

Asserted topological relation

GeoSPARQL: An example

Find the borough that contains the community of Ancient Olympia

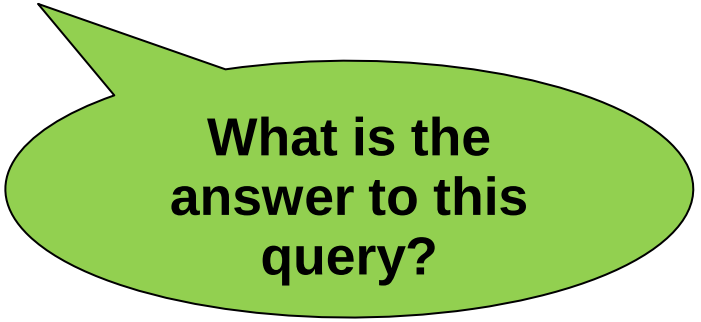
```
SELECT   ?m

WHERE {

 ?m rdf:type gag:Borough.

 ?m geo:sfContains geonames:Olympia.

}
```

Topological Predicate

GeoSPARQL: An example

Find the municipality that contains the community of Ancient Olympia

```
SELECT   ?m

WHERE {

 ?m rdf:type gag:Municipality.

 ?m geo:sfContains geonames:Olympia.

}
```



Example (cont'd)

The answer to the previous query is

`?m = gag:OlympiaMunicipality`

GeoSPARQL does not tell you how to compute this answer which needs **reasoning about the transitivity** of relation **`geo:sfContains`**.

Options:

- Use rules
- Use constraint-based techniques

GeoSPARQL Geometry Topology Extension

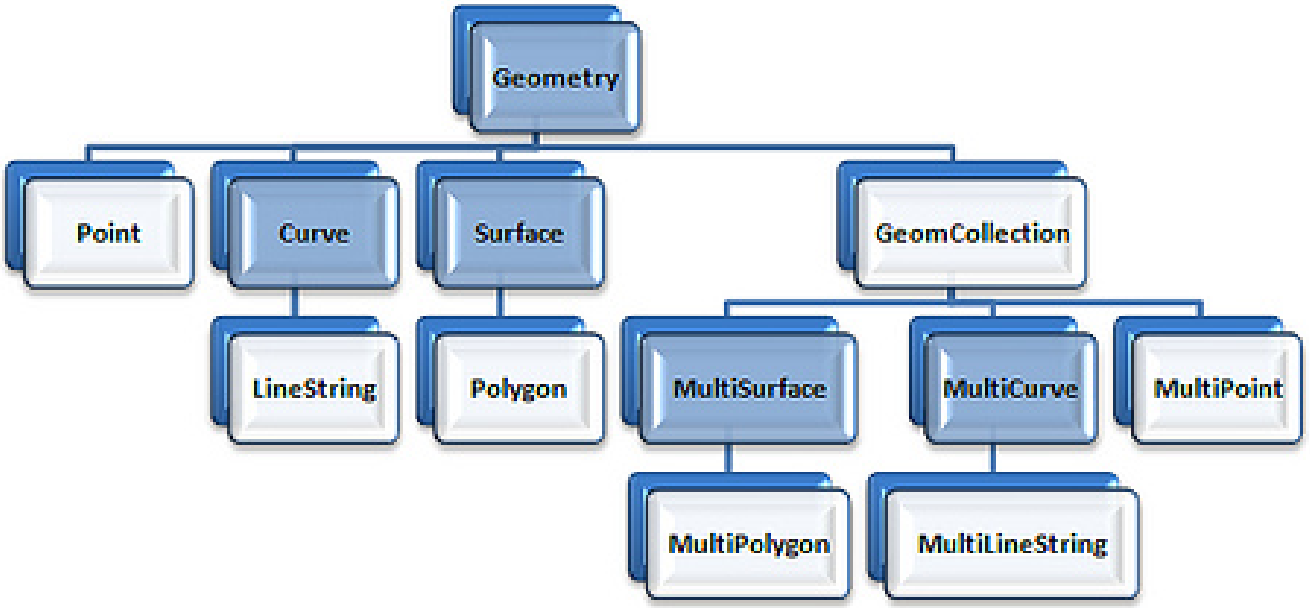
- Defines Boolean functions that correspond to each of the topological relations of the topology vocabulary extension:
  - OGC Simple Features Access

```
geof:sfEquals(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfDisjoint(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfIntersects(geom1: ogc:geomLiteral,geom2: ogc:geomLiteral): xsd:boolean
geof:sfTouches(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfCrosses(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfWithin(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfContains(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
geof:sfOverlaps(geom1: ogc:geomLiteral, geom2: ogc:geomLiteral): xsd:boolean
```

  - Egenhofer
  - RCC-8

GeoSPARQL RDFS Entailment Extension

- Provides a mechanism for realizing the RDFS entailments that follow from the geometry class hierarchies defined by the WKT and GML standards.



- Systems should use an implementation of RDFS entailment to allow the derivation of new triples from those already in a graph.

Example

Given the triples

```
ex:f1 geo:hasGeometry ex:g1 .
geo:hasGeometry rdfs:domain geo:Feature.
```

we can infer the following triples:

```
ex:f1 rdf:type geo:Feature .
ex:f1 rdf:type geo:SpatialObject .
```

GeoSPARQL Query Rewrite Extension

- Provides a collection of **RIF rules** that use topological extension functions to establish the existence of topological predicates.

- Example: given the RIF rule named **`geor:sfWithin`**, the serializations of the geometries of **`dbpedia:Athens`** and **`dbpedia:Greece`** named **`AthensWKT`** and **`GreeceWKT`** and the fact that

`geof:sfWithin(AthensWKT, GreeceWKT)`

returns true from the computation of the two geometries, we can derive the triple

`dbpedia:Athens geo:sfWithin dbpedia:Greece`

- One possible implementation is to re-write a given SPARQL query.

RIF Rule

```
Forall ?f1 ?f2 ?g1 ?g2 ?g1Serial ?g2Serial
  (?f1[geo:sfWithin->?f2] :-
    Or(
      And (?f1[geo:defaultGeometry->?g1]
           ?f2[geo:defaultGeometry->?g2]
           ?g1[ogc:asGeomLiteral->?g1Serial]
           ?g2[ogc:asGeomLiteral->?g2Serial]
           External(geo:sfWithin (?g1Serial,?g2Serial)))

      And (?f1[geo:defaultGeometry->?g1]
           ?g1[ogc:asGeomLiteral->?g1Serial]
           ?f2[ogc:asGeomLiteral->?g2Serial]
           External(geo:sfWithin (?g1Serial,?g2Serial)))

      And (?f2[geo:defaultGeometry->?g2]
           ?f1[ogc:asGeomLiteral->?g1Serial]
           ?g2[ogc:asGeomLiteral->?g2Serial]
           External(geo:sfWithin (?g1Serial,?g2Serial)))

      And (?f1[ogc:asGeomLiteral->?g1Serial]
           ?f2[ogc:asGeomLiteral->?g2Serial]
           External(geo:sfWithin (?g1Serial,?g2Serial)))
    ))
```

Feature - Feature

Feature - Geometry

Geometry - Feature

Geometry - Geometry

GeoSPARQL: An example

Discover the features that are inside the municipality of Ancient Olympia

```
SELECT ?feature
WHERE {
?feature geo:sfWithin
                geonames:OlympiaMunicipality.
}
```

GeoSPARQL: An example

```
SELECT ?feature
WHERE { {?feature geo:sfWithin geonames:Olympia }
        UNION
        { ?feature geo:defaultGeometry ?featureGeom .
          ?featureGeom geo:asWKT ?featureSerial .
          geonames:Olympia geo:defaultGeometry ?olGeom .
          ?olGeom geo:asWKT ?olSerial .
          FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
        UNION { ?feature geo:defaultGeometry ?featureGeom .
          ?featureGeom geo:asWKT ?featureSerial .
          geonames:Olympia geo:asWKT ?olSerial .
          FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
        UNION { ?feature geo:asWKT ?featureSerial .
          geonames:Olympia geo:defaultGeometry ?olGeom .
          ?olGeom geo:asWKT ?olSerial .
          FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
        UNION {
          ?feature geo:asWKT ?featureSerial .
          geonames:Olympia geo:asWKT ?olSerial .
          FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
```

Conclusions

- **Geospatial data in the Semantic Web**
  - The query language GeoSPARQL
    - Core
    - Topology vocabulary extension
    - Geometry extension
    - Geometry topology extension
    - Query rewrite extension
    - RDFS entailment extension

- **Next topic:** Implemented RDF Stores with Geospatial Support

Bibliography

[Perry and Herring, 2012]
Open Geospatial Consortium. *OGC GeoSPARQL - A geographic query language for RDF data*. OGC Candidate Implementation Standard (2012)