Geospatial data in the Semantic Web

GeoSPARQL

Presenter: Kostis Kyzirakos
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)

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

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."
Example in GeoSPARQL (2/2)

```
gag:OlympiaMunicipality
  rdf:type gag:Municipality;
  rdfs:label "ΔΗΜΟΣ ΑΡΧΑΙΑΣ ΟΛΥΜΠΙΑΣ"@el;
  rdfs:label "Municipality of Ancient Olympia".

```

**Asserted topological relation**
GeoSPARQL Components

Parameters
- Serialization
  - WKT
  - GML
- Relation Family
  - Simple Features
  - RCC8
  - Egenhofer

- Core
- Topology Vocabulary Extension
  - relation family
- Geometry Extension
  - serialization
  - version
- Geometry Topology Extension
  - serialization
  - version
  - relation family
- Query Rewrite Extension
  - serialization
  - version
  - relation family
- RDFS Entailment Extension
  - serialization
  - version
  - relation family

Parameters
- Serialization
  - WKT
  - GML
- Relation Family
  - Simple Features
  - RCC8
  - 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 the top class and has as instances everything that can have a spatial representation.
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: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: An example

Return the names of communities that are near burnt areas

```
SELECT  ?name
WHERE    {
    ?c   rdf:type geo:Feature;
         rdf:type dbpedia:Community;
         geonames:name ?name;
         geo:hasGeometry ?cPoly.
    ?ba  rdf:type geo:Feature;
         rdf:type noa:BurntArea;
         geo:hasGeometry ?baPoly.
    FILTER (geof:distance (?cGeom,?baGeom,
                            uom:metre) < 1500)}
```
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 RCC8 relations e.g., geo:rcc8ec
Example

geonames:Olympia
   rdf:type dbpedia: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".


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.
}
```
GeoSPARQL: An example

Find the municipality that contains the community of Ancient Olympia

```sparql
SELECT ?m
WHERE {
  ?m rdf:type gag:Municipality.
  ?m geo:sfContains geonames:Olympia.
}
```

What is the answer to this query?
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


  • Egenhofer
  • RCC8
GeoSPARQL: An example

Return the names of communities that have been affected by fires

```
SELECT ?name
WHERE {
  ?community rdf:type dbpedia:Community;
    geonames:name ?name;
    geo:hasGeometry ?cPoly.

  ?ba a noa:BurntArea;
    geo:hasGeometry ?baPoly.

  FILTER (geof:sfIntersects(?cGeom, ?baGeom))
}
```
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

\[
\text{ex:f1 geo:hasGeometry ex:g1.}
\]
\[
\text{geo:hasGeometry rdfs:domain geo:Feature.}
\]

we can infer the following triples:

\[
\text{ex:f1 rdf:type geo:Feature .}
\]
\[
\text{ex:f1 rdf:type geo:SpatialObject.}
\]
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

  \[ \text{geof:sfWithin(AthensWKT, GreeceWKT)} \]

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

  \[ \text{dbpedia:Athens geo:sfWithin dbpedia:Greece} \]

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

\[
\forall \ ?f1 \ ?f2 \ ?g1 \ ?g2 \ ?g1Serial \ ?g2Serial \\
(\ ?f1[\text{geo:sfWithin}->\?f2] :- \\
\quad \text{Or} \\
\quad \quad \text{And} \ \ ?f1[\text{geo:defaultGeometry}->\?g1] \\
\quad \quad \quad \ ?f2[\text{geo:defaultGeometry}->\?g2] \\
\quad \quad \quad \ ?g1[\text{ogc:asGeomLiteral}->\?g1Serial] \\
\quad \quad \quad \ ?g2[\text{ogc:asGeomLiteral}->\?g2Serial] \\
\quad \quad \quad \ \text{External}(\text{geo:sfWithin} \ (\?g1Serial,\?g2Serial))) \\
\quad \text{And} \ \ ?f1[\text{geo:defaultGeometry}->\?g1] \\
\quad \quad \ ?g1[\text{ogc:asGeomLiteral}->\?g1Serial] \\
\quad \quad \ ?f2[\text{ogc:asGeomLiteral}->\?g2Serial] \\
\quad \quad \ \text{External}(\text{geo:sfWithin} \ (\?g1Serial,\?g2Serial))) \\
\quad \text{And} \ \ ?f2[\text{geo:defaultGeometry}->\?g2] \\
\quad \quad \ ?f1[\text{ogc:asGeomLiteral}->\?g1Serial] \\
\quad \quad \ ?g2[\text{ogc:asGeomLiteral}->\?g2Serial] \\
\quad \quad \ \text{External}(\text{geo:sfWithin} \ (\?g1Serial,\?g2Serial))) \\
\quad \text{And} \ \ ?f1[\text{ogc:asGeomLiteral}->\?g1Serial] \\
\quad \quad \ ?f2[\text{ogc:asGeomLiteral}->\?g2Serial] \\
\quad \quad \ \text{External}(\text{geo:sfWithin} \ (\?g1Serial,\?g2Serial)))
\]
GeoSPARQL: An example

Discover the features that are inside the municipality of Ancient Olympia

```
SELECT ?feature
WHERE {
}
```
GeoSPARQL: An example

```sparql
SELECT ?feature
WHERE {
  { ?feature geo:sfWithin geonames:Olympia . }
  UNION
    FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
  UNION {
    geonames:Olympia geo:asWKT ?olSerial .
    FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
  UNION {
    FILTER (geof:sfWithin (?featureSerial, ?olSerial)) }
  UNION {
    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 Systems and Applications