Geospatial data in the Semantic Web

stSPARQL

Presenter: Kostis Kyzirakos

Outline

- Main idea
- Early works
- The data model stRDF
- Examples of publicly available linked geospatial data
- The query language stSPARQL

Main idea

How do we represent and query geospatial information in the Semantic Web?

Extend RDF to take into account the geospatial dimension.

Extend SPARQL to query the new kinds of data.

Early works

SPAUK (Kolas, 2007)

- Geometric attributes of a resource are represented by:
  - introducing a **blank node** for the geometry
  - specifying the geometry using **GML vocabulary**
  - associating the blank node with the resource using **GeoRSS vocabulary**
- Queries are expressed in the SPARQL query language utilizing appropriate geometric vocabularies and ontologies (e.g., the topological relationships of RCC8).
- Introduces a new **`PREMISE`** clause in SPARQL to specify spatial geometries to be used in a query
- Use some form of the **`DESCRIBE`** query form of SPARQL for asking queries about geometries

Early works

SPARQL-ST (Perry, 2008)

- Assumes a particular upper ontology expressed in RDFS for modeling **theme**, **space** and **valid time**.

- Spatial geometries in SPARQL-ST are specified by **sets of RDF triples** that give various details of the geometry.

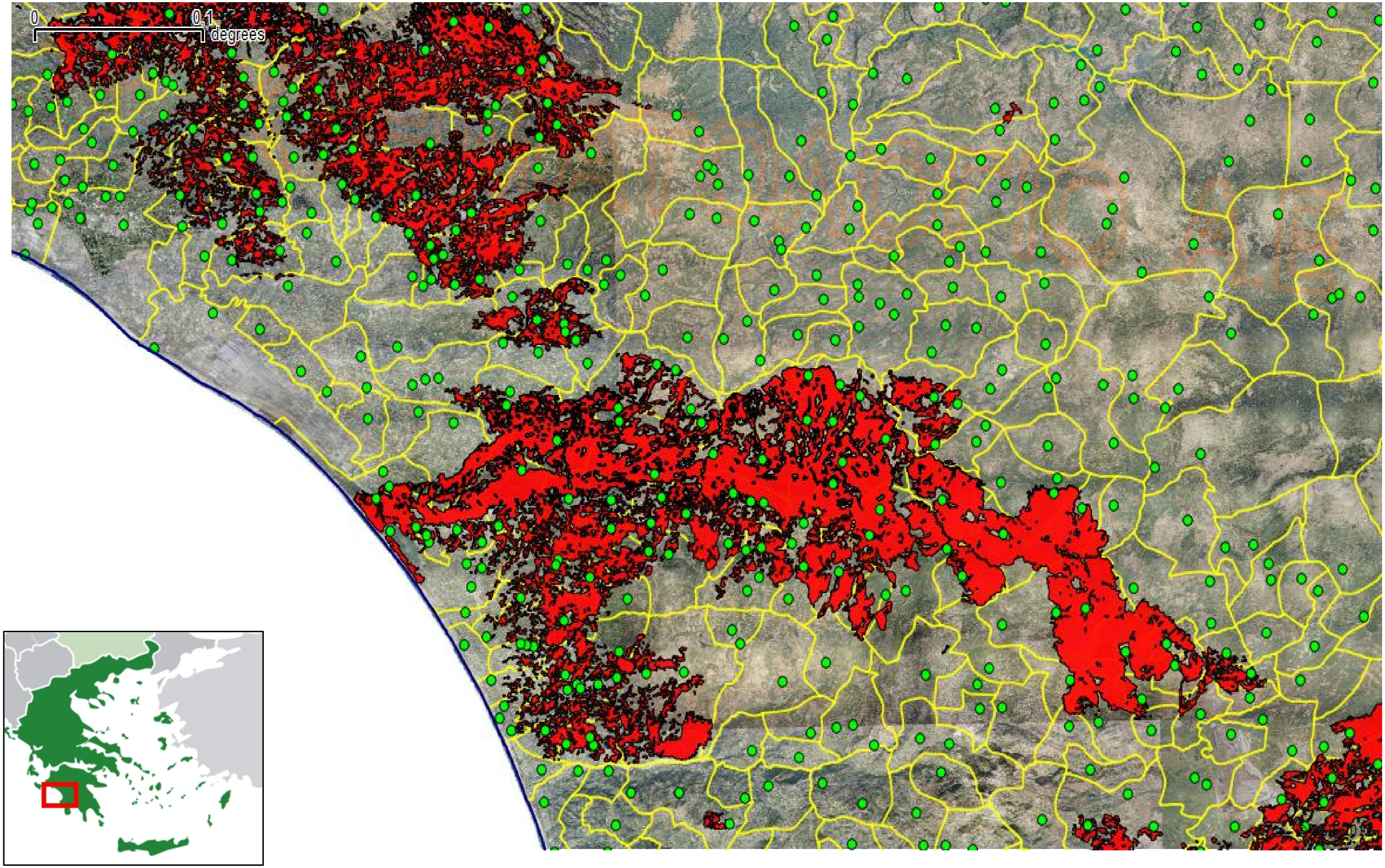
- SPARQL-ST provides a set of built-in spatial conditions that can be used in **SPATIAL FILTER** clauses to constrain the geometries that are returned as answers to queries.

stRDF and stSPARQL

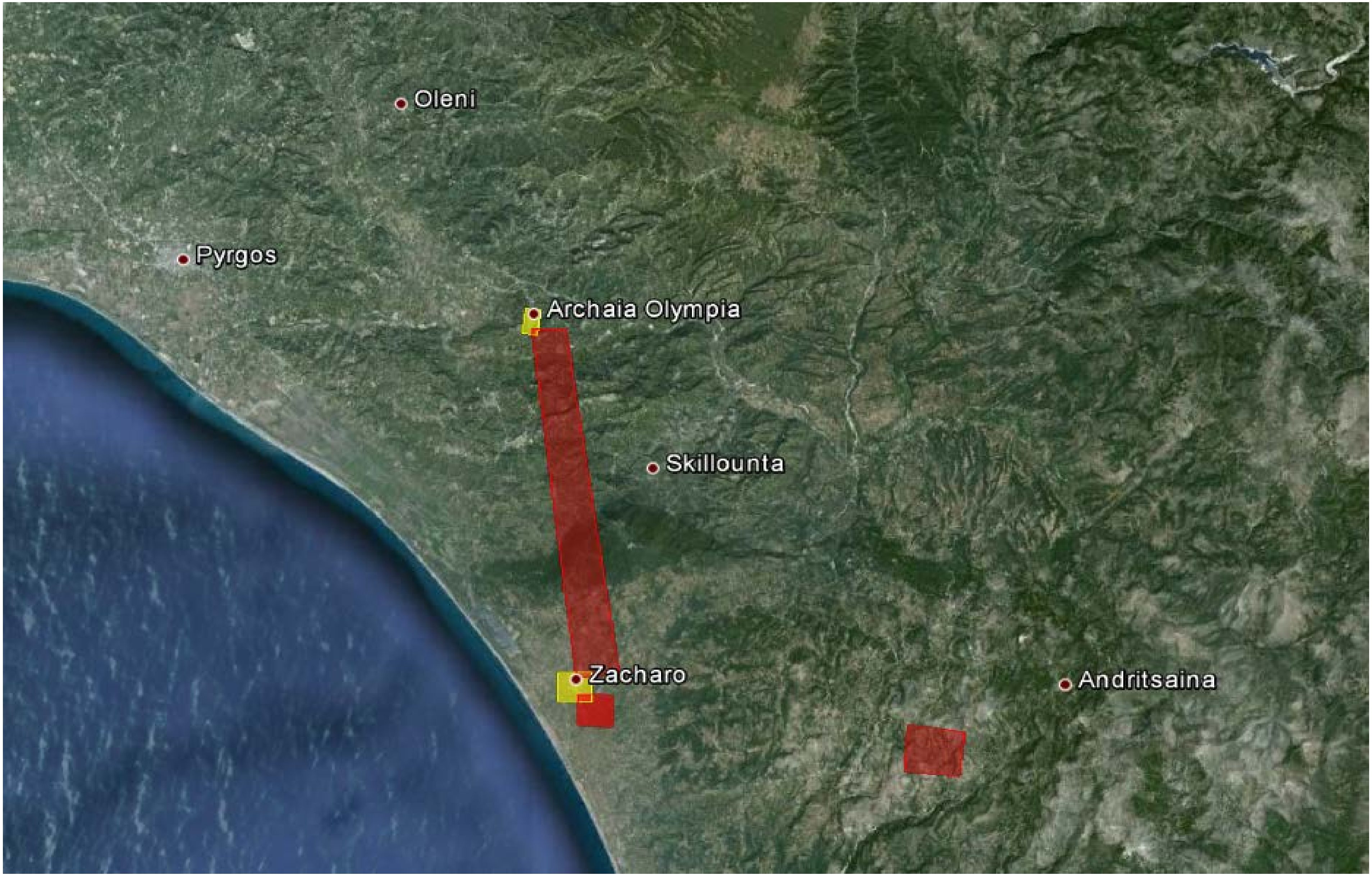
(Koubarakis and Kyzirakos, 2010)

- Similar approach to SPARQL-ST (**theme**, **space** and **valid time** can be represented)
- **Linear constraints** are used to represent geometries
- Constraints are represented using literals of an appropriate datatype
- Formal approach
- New version to be presented today uses **OGC standards** to represent and query geometries

Example



Example with simplified geometries



Example in stRDF

```
geonames:olympia geonames:name "Ancient Olympia";
                 owl:sameAs dbpedia:Olympia_Greece;
                 rdf:type dbpedia:Community .
```

Spatial literal

```
geonames:olympia strdf:hasGeometry
          "POLYGON((21.5 18.5, 23.5 18.5,
                   23.5 21, 21.5 21, 21.5 18.5));
  <http://www.opengis.net/def/crs/EPSG/0/4326>"^^
  strdf:WKT .
```

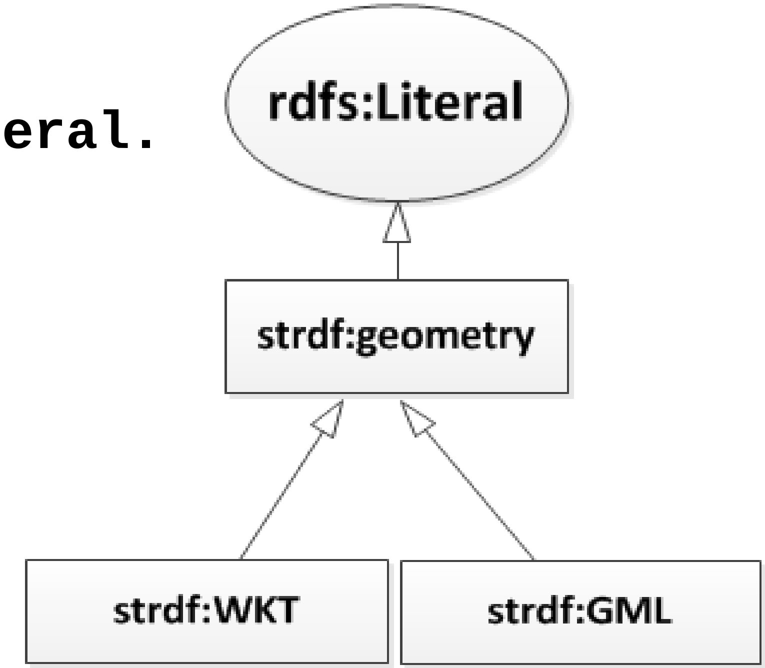
Spatial data type

The stRDF Data Model

```
strdf:geometry rdf:type rdfs:Datatype;
               rdfs:subClassOf rdfs:Literal.

strdf:WKT   rdf:type rdfs:Datatype;
            rdfs:subClassOf   rdfs:Literal;
            rdfs:subClassOf   strdf:geometry.

strdf:GML   rdf:type rdfs:Datatype;
            rdfs:subClassOf   rdfs:Literal;
            rdfs:subClassOf   strdf:geometry.
```



The stRDF Data Model

We define the datatypes **`strdf:WKT`** and **`strdf:GML`** that can be used to represent spatial objects using the WKT and GML serializations.

- **Lexical space:** the finite length sequences of characters that can be produced from the WKT and GML specifications.
  - Literals of type **`strdf:WKT`** consist of an optional URI identifying the coordinate reference system used.

e.g.,
```
"POINT(21 18);
 <http://www.opengis.net/def/crs/EPSG/0/4326>"
^^strdf:WKT
```

The stRDF Data Model

- **Value space**: the set of geometry values defined in the WKT and GML standard that is a subset of the powerset of $\mathbb{R}^2$ and $\mathbb{R}^3$.

- **Lexical-to-value mapping:** takes into account that the vector-based model is used for representing geometries.

- The datatype **`strdf:geometry`** is the union of the datatypes **`strdf:WKT`** and **`strdf:GML`**.

Examples of publicly available linked geospatial data

- Geonames
- Greek Administrative Geography
- Corine Land Use / Land Cover
- Burnt Area Products

Geonames


Analipsi
Gouves Kriti
Goyrnes
Chersonisos
Gouves
Kalo Chorio
Malia
Sisi
Mochos
Potamies
E75
90

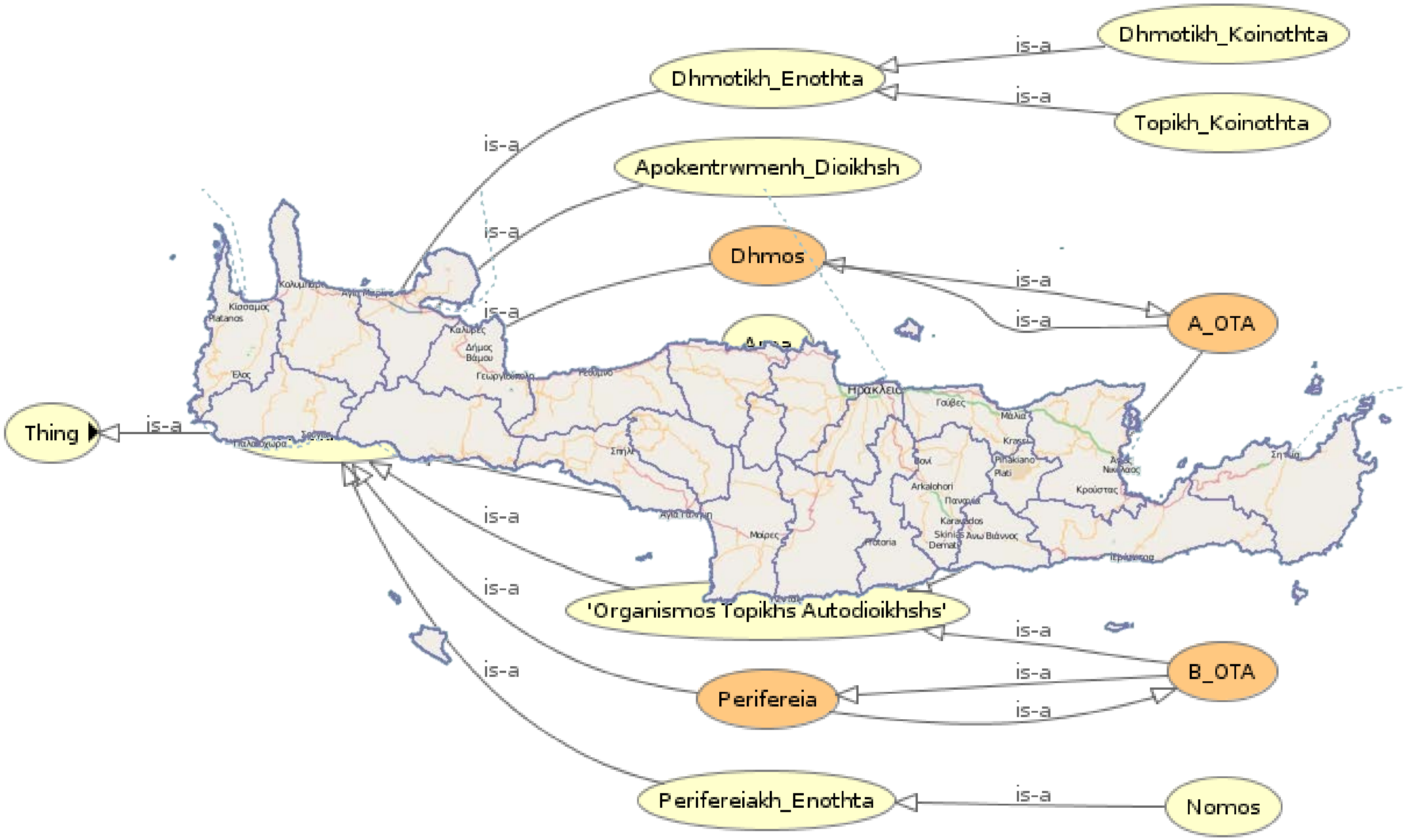
Geonames

```
geonames:260001   rdf:type           gn:Feature;
                  gn:name            "Hersonissos";
                  gn:officialName    "Χερσόνησος"@el;
                  gn:countryCode     "GR";
                  wgs84_pos:lat      "35.30903";
                  wgs84_pos:long     "25.37112";
                  gn:parentCountry   geonames:390903.

geonames:390903   gn:name            "Greece".
```

Greek Administrative Geography

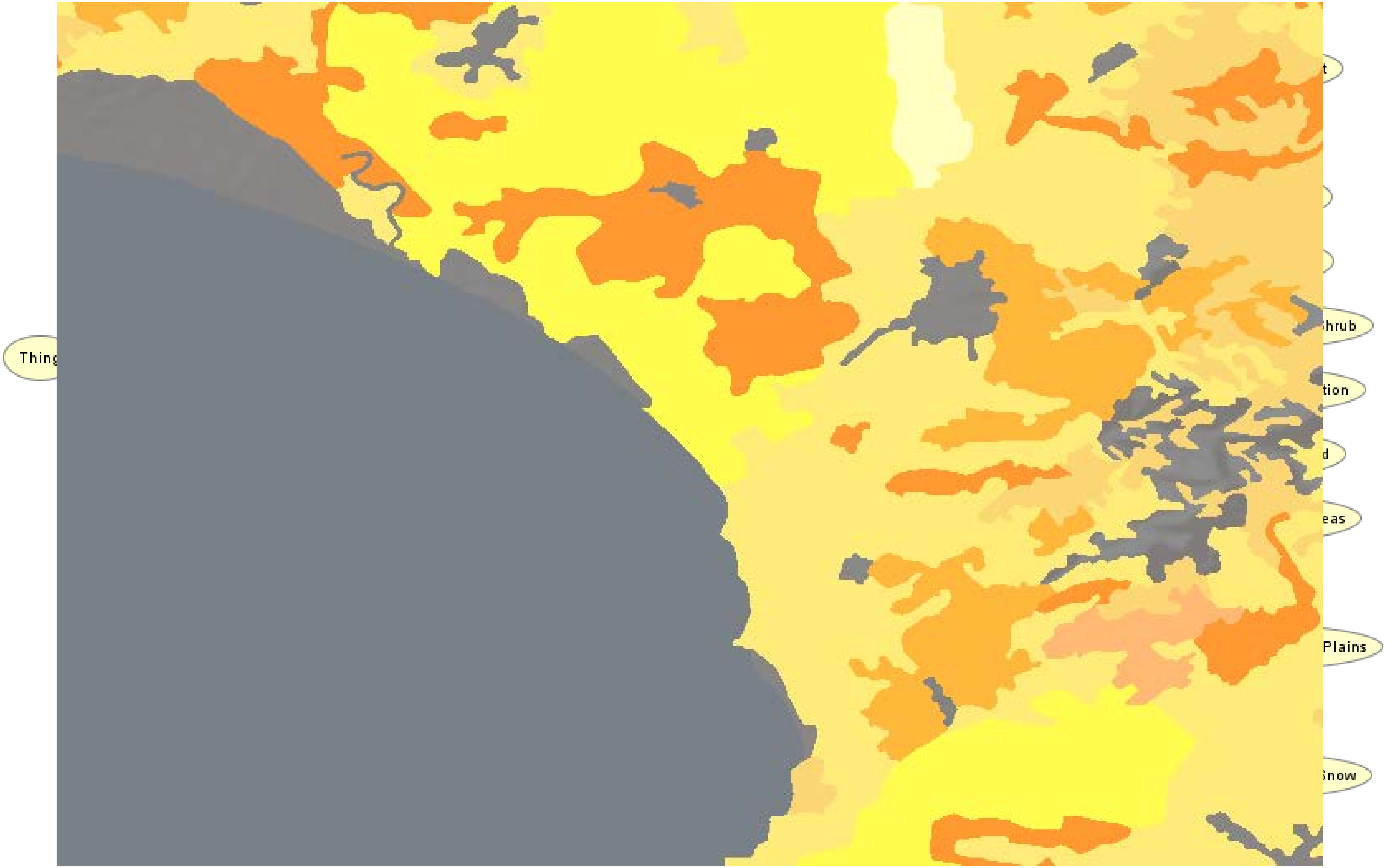
- Kallikrates ontology



Greek Administrative Geography

```
gag:gag003000009002 rdf:type owl:NamedIndividual ;
                    rdf:type gag:Dhmos;
                    rdfs:label "ΔΗΜΟΣ ΧΕΡΣΟΝΗΣΟΥ"@el;
                    rdfs:label "Hersonissos";
                    noa:hasYpesCode "9309"^^xsd:integer;
                    strdf:hasGeometry
                          "MULTIPOLYGON (((
                              25.37 35.34,
                              ...,
                              25.21 35.47)))"^^strdf:WKT;
                    gag:isPartOf gag:gag003000000101.
```

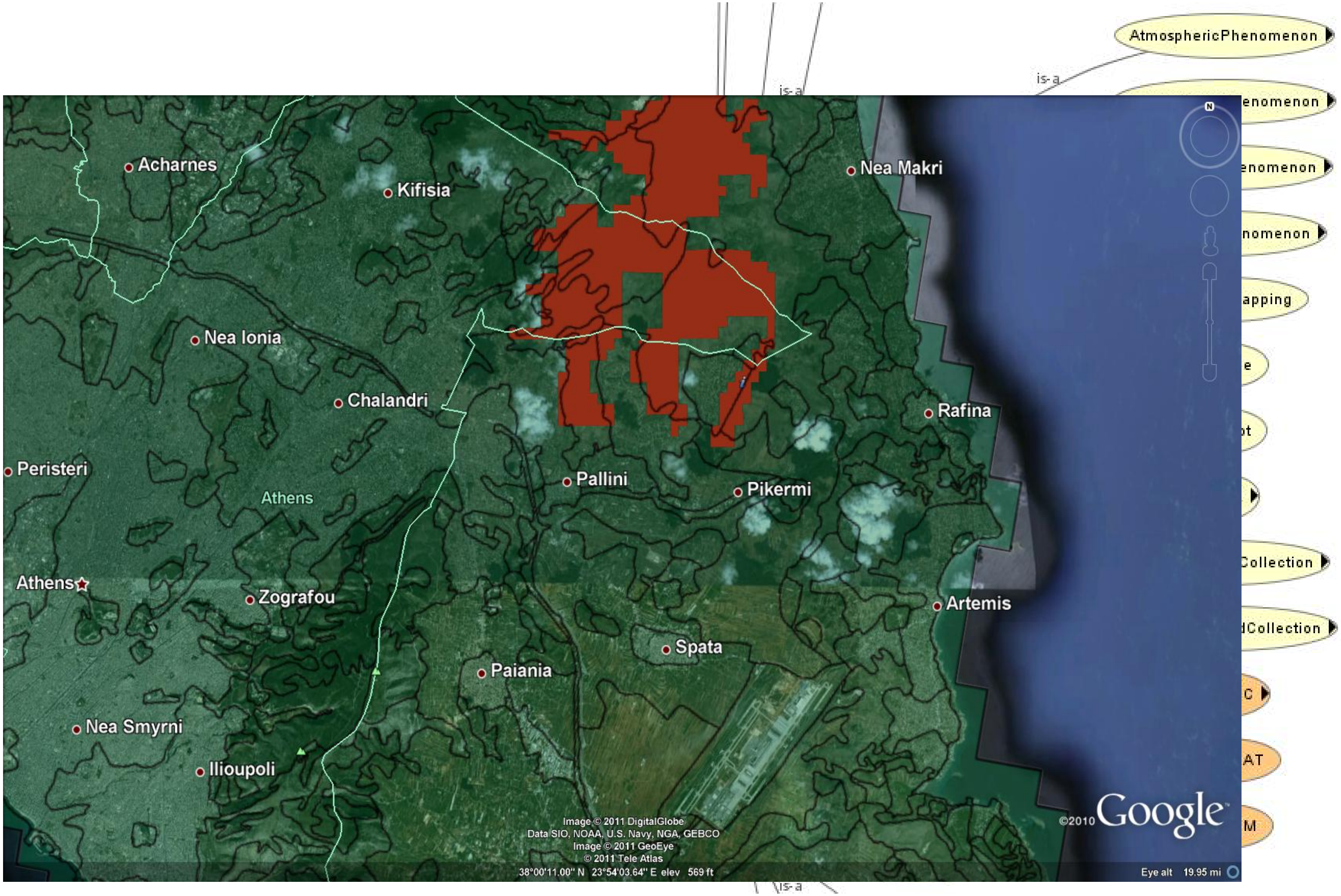
Corine Land Use / Land Cover



Corine Land Use / Land Cover

```
noa:Area_24015134   rdf:type noa:Area ;
                    noa:hasCode "312"^^xsd:decimal;
                    noa:hasID "EU-203497"^^xsd:string;
                    noa:hasArea_ha "255.580790497"^^xsd:double;
                    strdf:hasGeometry "POLYGON((15.53 62.54, ...,
                                     15.53 62.54))"^^strdf:WKT;
                    noa:hasLandUse noa:coniferousForest
```

Burnt Area Products



Burnt Area Products

```
noa:ba_15 rdf:type noa:BurntArea;
          noa:isDerivedFromSatellite "Landsat"^^xsd:string;
          noa:hasAcquisitionTime
                     "2010-08-24T13:00:00"^^xsd:dateTime;
          strdf:hasGeometry
                     "MULTIPOLYGON(((393801.42 4198827.92,
                      ...,
                     393008 424131)));
                        <http://www.opengis.net/def/crs/
                         EPSG/0/2100>"^^strdf:WKT.
```

stSPARQL: Geospatial SPARQL 1.1

We define a SPARQL extension function for each function defined in the OpenGIS Simple Features Access standard

Basic functions

- Get a property of a geometry
```
xsd:int strdf:Dimension(strdf:geometry A)
xsd:string strdf:GeometryType(strdf:geometry A)
xsd:int strdf:SRID(strdf:geometry A)
```
- Get the desired representation of a geometry
```
xsd:string strdf:AsText(strdf:geometry A)
strdf:wkb strdf:AsBinary(strdf:geometry A)
xsd:string strdf:AsGML(strdf:geometry A)
```
- Test whether a certain condition holds
```
xsd:boolean strdf:IsEmpty(strdf:geometry A)
xsd:boolean strdf:IsSimple(strdf:geometry A)
```

stSPARQL: Geospatial SPARQL 1.1

Functions for testing topological spatial relationships

- **OGC Simple Features Access**

```
xsd:boolean strdf:equals(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:disjoint(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:intersects(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:touches(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:crosses(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:within(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:contains(strdf:geometry A, strdf:geometry B)
xsd:boolean strdf:overlaps(strdf:geometry A, strdf:geometry B)

xsd:boolean strdf:relate(strdf:geometry A, strdf:geometry B,
                         xsd:string intersectionPatternMatrix)
```

- **Egenhofer**
- **RCC8**

stSPARQL: Geospatial SPARQL 1.1

Spatial analysis functions

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

```
strdf:geometry strdf:Boundary(strdf:geometry A)
strdf:geometry strdf:Envelope(strdf:geometry A)
strdf:geometry strdf:Intersection(strdf:geometry A, strdf:geometry B)
strdf:geometry strdf:Union(strdf:geometry A, strdf:geometry B)
strdf:geometry strdf:Difference(strdf:geometry A, strdf:geometry B)
strdf:geometry strdf:SymDifference(strdf:geometry A, strdf:geometry B)
strdf:geometry strdf:Buffer(strdf:geometry A, xsd:double distance)
```

- **Spatial metric functions**

```
xsd:float strdf:distance(strdf:geometry A, strdf:geometry B)
xsd:float strdf:area(strdf:geometry A)
```

- **Spatial aggregate functions**

```
strdf:geometry strdf:Union(set of strdf:geometry A)
strdf:geometry strdf:Intersection(set of strdf:geometry A)
strdf:geometry strdf:Extent(set of strdf:geometry A)
```

stSPARQL: Geospatial SPARQL 1.1

Select clause

- Construction of new geometries (e.g., **`strdf:buffer(?geo, 0.1)`**)
- Spatial aggregate functions (e.g., **`strdf:union(?geo)`**)
- Metric functions (e.g., **`strdf:area(?geo)`**)

Filter clause

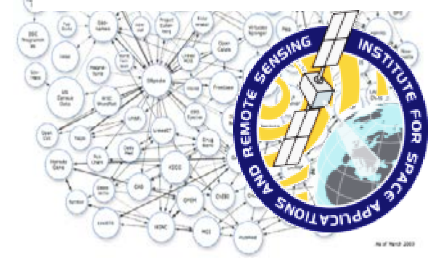
- Functions for testing topological spatial relationships between spatial terms (e.g., **`strdf:contains(?G1, strdf:union(?G2, ?G3))`**)
- Numeric expressions involving spatial metric functions (e.g., **`strdf:area(?G1) ≤ 2*strdf:area(?G2)+1`**)
- Boolean combinations

Having clause

- Boolean expressions involving spatial aggregate functions and spatial metric functions or functions testing for topological relationships between spatial terms (e.g., **`strdf:area(strdf:union(?geo))>1`**)

stSPARQL: An example (1/3)

Return the names of communities that have been affected by fires



```
SELECT   ?name
WHERE {
  ?community rdf:type dbpedia:Community;
             geonames:name ?name;
             strdf:hasGeometry ?comGeom.
  ?ba  rdf:type noa:BurntArea;
       strdf:hasGeometry ?baGeom.

  FILTER(strdf:overlap(?comGeom,?baGeom))
}
```

Spatial Function

stSPARQL: An example (2/3)

Find all burnt forests near communities

```
SELECT ?ba ?baGeom
WHERE {
    ?r rdf:type noa:Region;
       strdf:geometry ?rGeom;
       noa:hasCorineLandCoverUse ?f.
    ?f rdfs:subClassOf clc:Forests.
    ?c rdf:type dbpedia:Community;
          strdf:geometry ?cGeom.
    ?ba rdf:type noa:BurntArea;
        strdf:geometry ?baGeom.

    FILTER( strdf:intersects(?rGeom,?baGeom) &&
            strdf:distance(?baGeom,?cGeom) < 0.02)}
```

Spatial Functions

stSPARQL: An example (3/3)

Isolate the parts of the burnt areas that lie in coniferous forests.

Spatial Aggregate

```
SELECT ?burntArea
(strdf:intersection(?baGeom,
                  strdf:union(?tGeom))
 AS ?burntForest)
WHERE {
   ?burntArea   rdf:type noa:BurntArea;
                strdf:hasGeometry ?baGeom.
   ?forest rdf:type noa:Area;
           noa:hasLandCover noa:coniferousForest;
           strdf:hasGeometry ?fGeom.

   FILTER(strdf:intersects(?baGeom,?fGeom))
}
GROUP BY ?burntArea ?baGeom
```

Spatial Function

Conclusions

- **Geospatial data in the Semantic Web - stSPARQL**
  - Early works
  - The data model stRDF
  - Examples of publicly available linked geospatial data
  - The query language stSPARQL

- **Next topic:** The query language GeoSPARQL