Data Models and Query Languages for Linked Geospatial Data

Manolis Koubarakis, Kostis Kyzirakos and Charalampos Nikolaou

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

Tutorial Organization

16:00 – 16:35 Introduction and background in geospatial data modeling

16:35 – 17:20 Representing and querying geospatial data in RDF

17:20 – 17:30 Conclusions, questions, discussion

Introduction and background in geospatial data modeling

Presenter: Charalampos Nikolaou

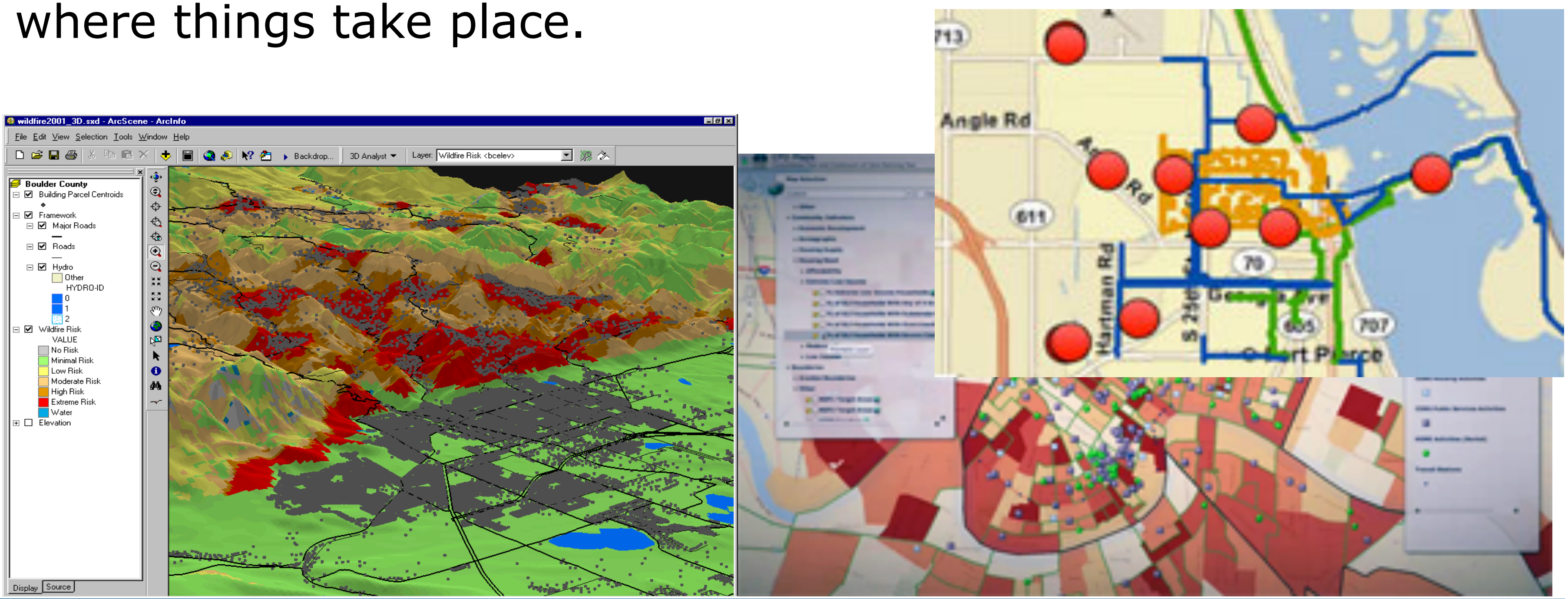
Outline

- Why should you be interested in geospatial information?
- Why should you attend this tutorial?
- Basic GIS concepts and terminology
- Geospatial data standards

Why Geospatial Information?

- **Geospatial,** and in general **geographical,** information is very important in reality: everything that happens, happens somewhere (**location**).

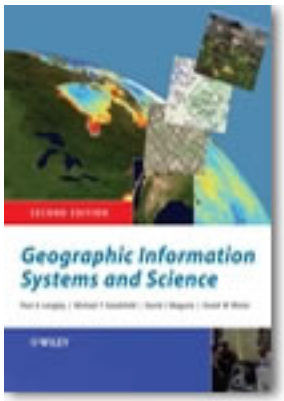
- **Decision making can be substantially improved** if we know where things take place.

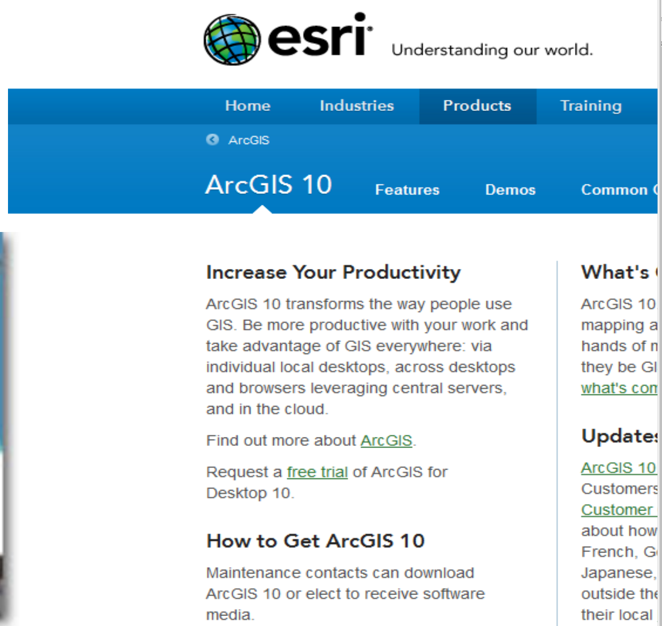


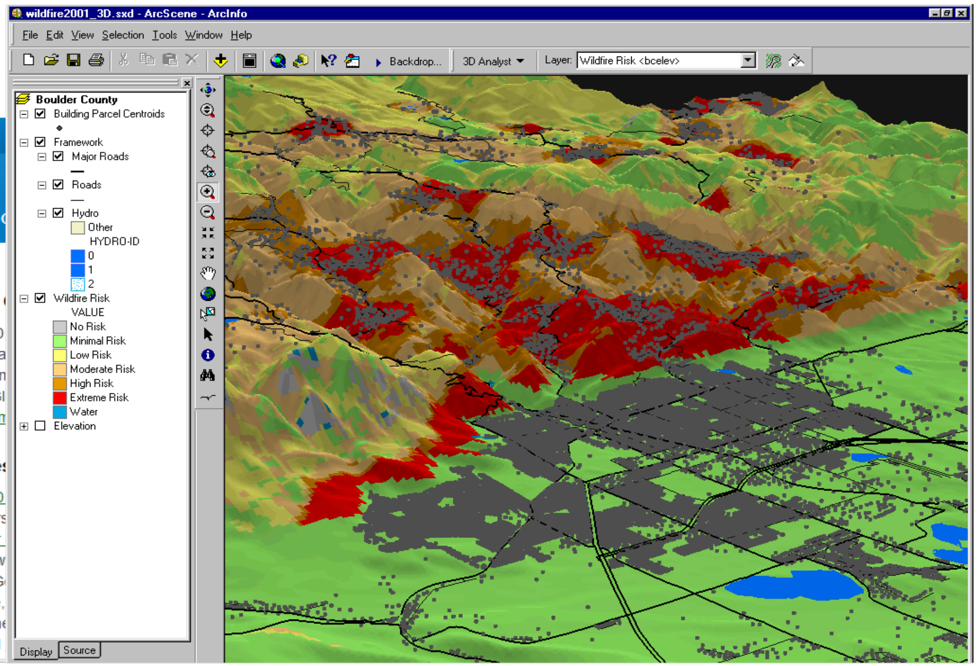
Geographical Information Systems and Science

- A **geographical information system (GIS)** is a system designed to capture, store, manipulate, analyze, manage, and present all types of geographical data.

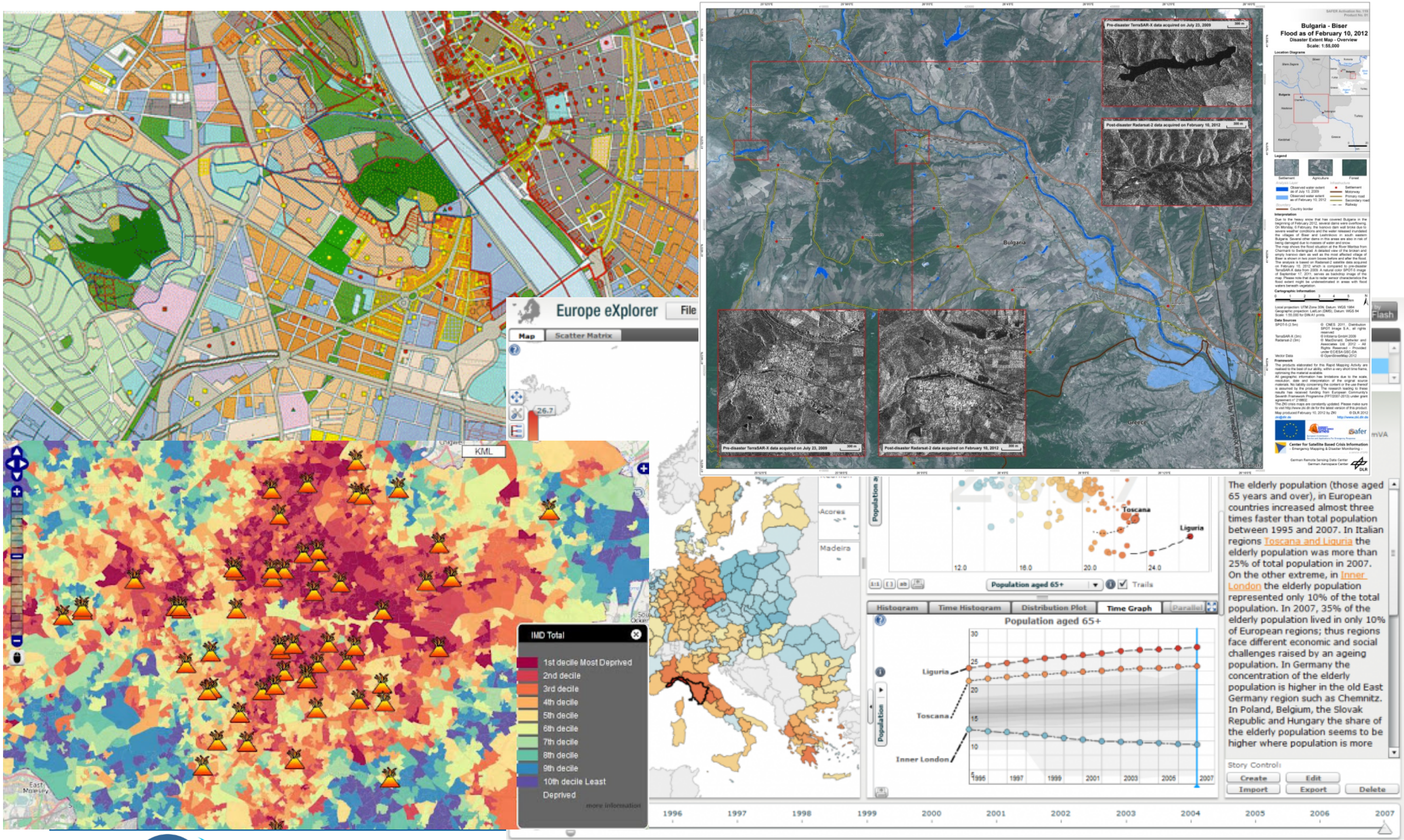
- **GIS science** is the field of study for developing and using GIS.







Combining GIS Data for Decision Making



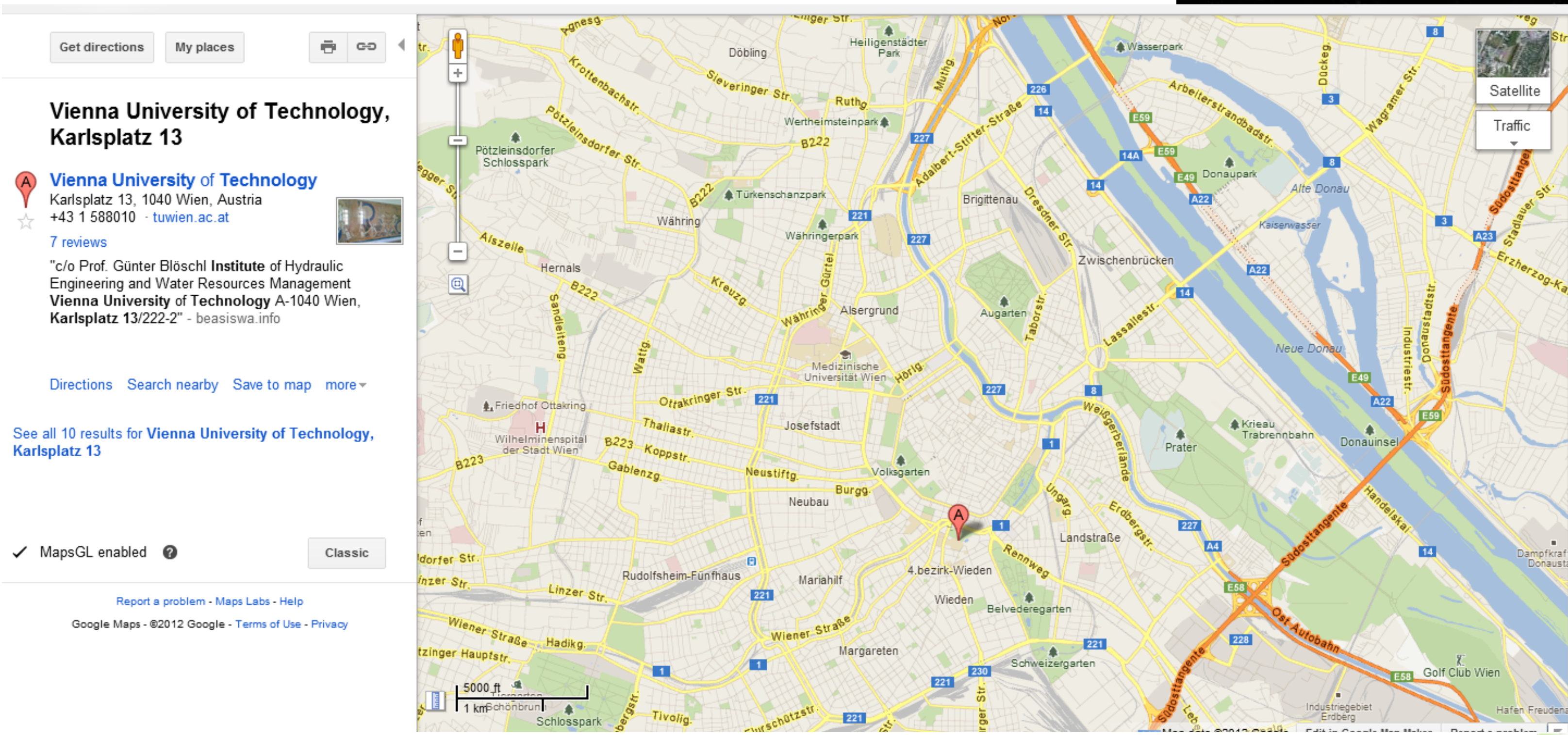
Why this tutorial?

- Lots of **geospatial data is available on the Web** today.
- Lots of **public data** coming out of open government initiatives is **geospatial**.
- Lots of the above data is quickly being **transformed into linked data!**
- People have started building **applications** utilizing linked data.

Geospatial data on the Web

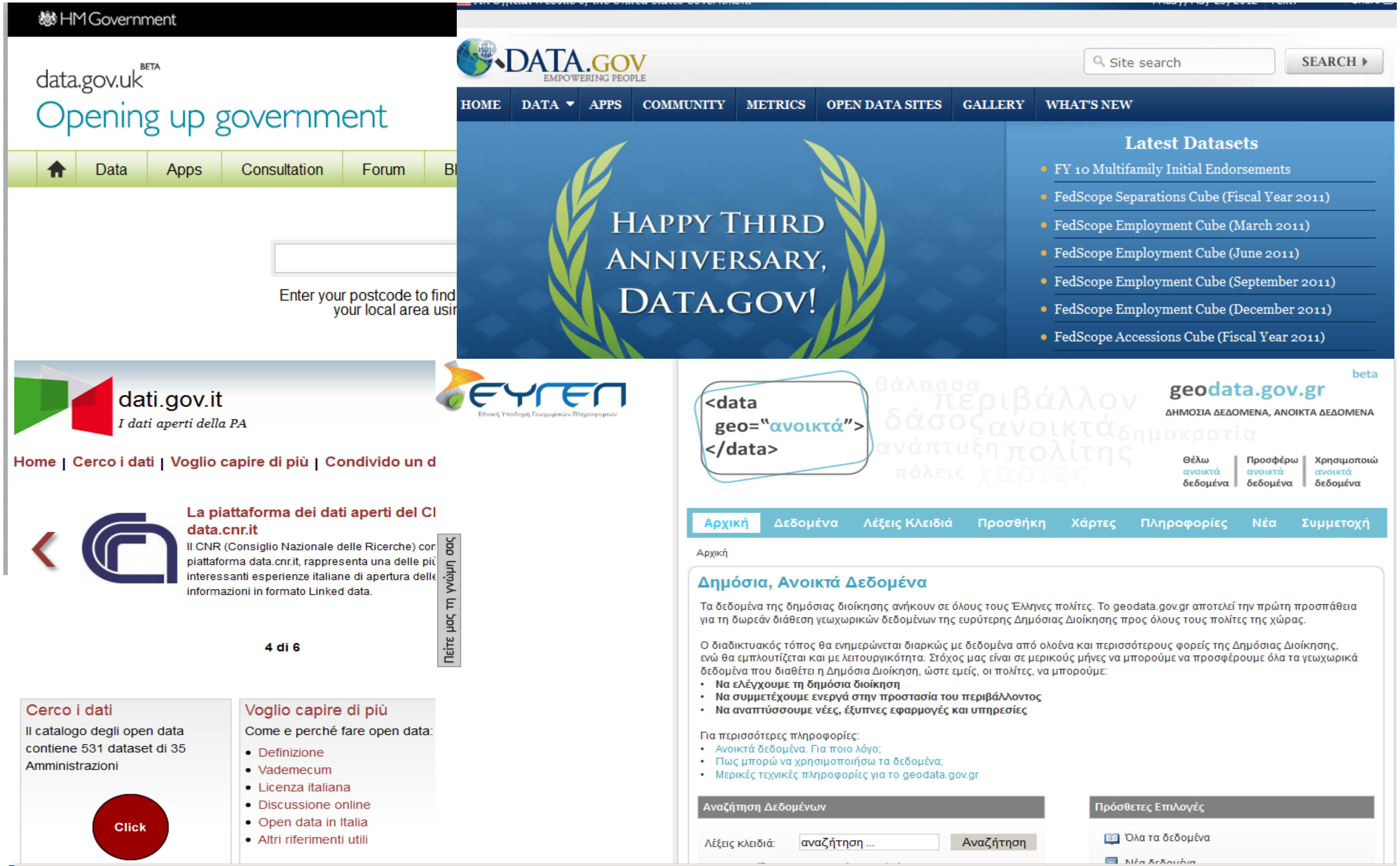
Google Maps

Google earth



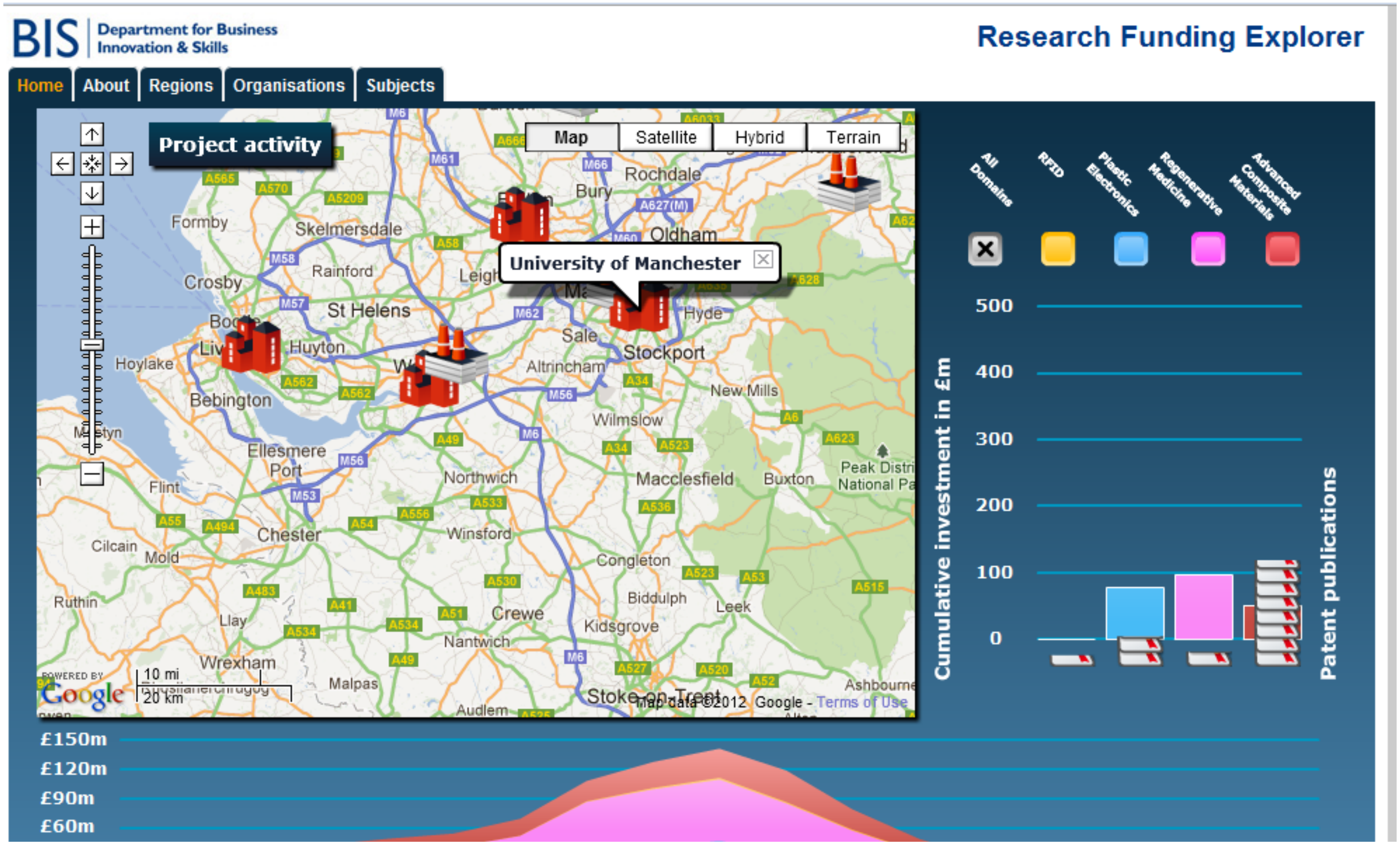
bing maps

Open Government Data



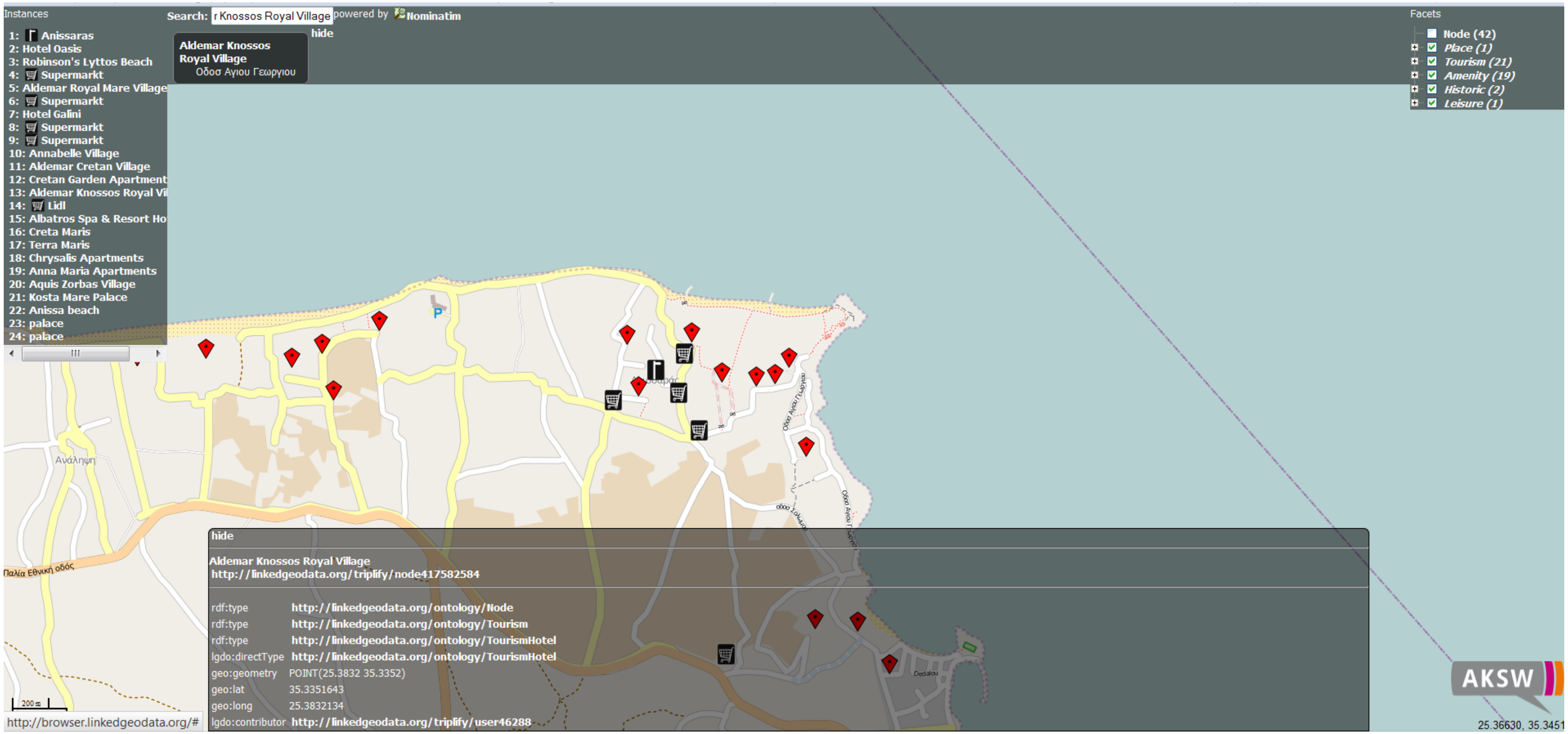
Linked geospatial data – Ordnance Survey

Linked geospatial data – Research Funding Explorer



Linked geospatial data – Spain

Linked geospatial data – Open Street Map



Background in geospatial data modeling

Basic GIS Concepts and Terminology

- **Theme:** the information corresponding to a particular domain that we want to model. A theme is a set of **geographic features.**

- **Example:** the countries of Europe

Basic GIS Concepts (cont'd)

- **Geographic feature or geographic object:** a domain entity that can have various **attributes** that describe **spatial and non-spatial** characteristics.

- **Example:** the country Greece with attributes
    - Population
    - Flag
    - Capital
    - Geographical area
    - Coastline
    - Bordering countries

Basic GIS Concepts (cont'd)

- Geographic features can be **atomic** or **complex.**

- **Example:** According to the Kallikratis administrative reform of 2010, Greece consists of:
  - 13 **regions** (e.g., Crete)
  - Each region consists of **perfectures** (e.g., Heraklion)
  - Each perfecture consists of **municipalities** (e.g., Chersonisos)

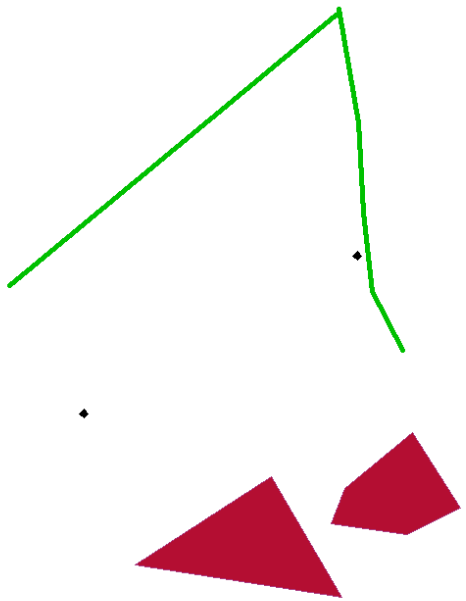
Basic GIS Concepts (cont'd)

- The spatial characteristics of a feature can involve:
  - **Geometric information** (location in the underlying geographic space, shape etc.)
  - **Topological information** (containment, adjacency etc.).

Municipalities of the perfecture of Heraklion:
1. Heraklion
2. Archanes-Asterousia
3. Viannos
4. Gortyna
5. Malevizio
6. Minoa Pediadas
7. Festos
8. Chersonisos

Geometric Information

- Geometric information can be captured by using geometric primitives (**points**, **lines**, **polygons,** etc.) to approximate the spatial attributes of the real world feature that we want to model.



- Geometries are associated with a **coordinate reference system** which describes the coordinate space in which the geometry is defined.

Topological Information

- Topological information is **inherently qualitative** and it is expressed in terms of **topological relations** (e.g., containment, adjacency, overlap etc.).

- Topological information can be **derived from geometric information** or it might be captured by **asserting explicitly the topological relations** between features.

Topological Relations

- The study of topological relations has produced a lot of interesting results by researchers in:
  - GIS
  - Spatial databases
  - Artificial Intelligence (qualitative reasoning and knowledge representation)

The 4-intersection model

- The **4-intersection model** has been defined by Egenhofer and Franzosa in 1991 based on previous work by Egenhofer and colleagues.

- It is based on **point-set topology.**

- **Spatial regions** are defined to be **non-empty, proper subsets of a topological space.** In addition, they must be closed and have connected interiors.

- **Topological relations** are the ones that are invariant under topological homeomorphisms.

4IM and 9IM

- The 4-intersection model can captures **topological relations** between two spatial regions *a* and *b* by considering **whether the intersection of their boundaries and interiors is empty or non-empty**.

- The **9-intersection model** is an extension of the 4-intersection model (Egenhofer and Herring, 1991).

- 9IM captures topological relations between two spatial regions *a* and *b* by considering whether the intersection of their boundaries, interiors and **exteriors** is empty or non-empty.

DE-9IM

- The **dimensionally extended 9-intersection model** has been defined by Clementini and Felice in 1994.

- It is also based on the **point-set topology** of R^2 and deals with **"simple", closed geometries (areas, lines, points).**

- Like its predecessors (4IM, 9IM), it can also be extended to **more complex geometries** (areas with holes, geometries with multiple components).

DE-9IM

- It captures **topological relationships** between two geometries *a* and *b* in R^2 by considering the **dimensions of the intersections of the boundaries, interiors and exteriors** of the two geometries:

$$\text{DE-9IM}(a,b) = \begin{bmatrix} \dim(I(a)\cap I(b)) & \dim(I(a)\cap B(b)) & \dim(I(a)\cap E(b)) \\ \dim(B(a)\cap I(b)) & \dim(B(a)\cap B(b)) & \dim(B(a)\cap E(b)) \\ \dim(E(a)\cap I(b)) & \dim(E(a)\cap B(b)) & \dim(E(a)\cap E(b)) \end{bmatrix}.$$

- The dimension can be **2, 1, 0** and **-1** (dimension of the empty set).

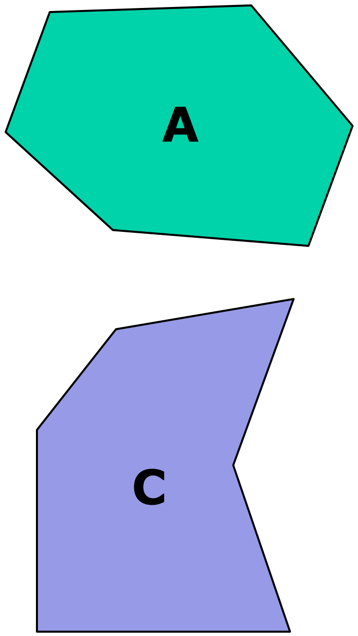
DE-9IM

- **Five jointly exclusive and pairwise disjoint (JEPD)** relationships between two different geometries can be distinguished (**disjoint, touches, crosses, within, overlaps**).

- The model can also be defined using an appropriate **calculus of geometries** that uses these 5 binary relations and boundary operators.

- See the paper: E. Clementini and P. Felice. A Comparison of Methods for Representing Topological Relationships. Information Sciences 80 (1994), pp. 1-34.

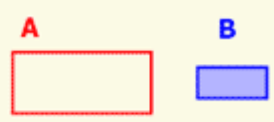
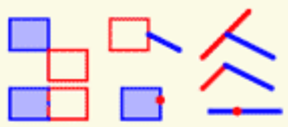
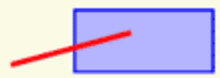
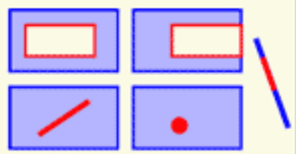
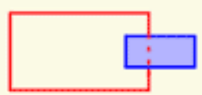
Example: A disjoint C



	I(C)	**B**(C)	**E**(C)
I(A)	F	F	*
B(A)	F	F	*
E(A)	*	*	*

Notation:
- T = { 0, 1, 2 }
- F = -1
- * = don't care = { -1, 0, 1, 2 }

DE-9IM Relation Definitions

Beziehung	Definition	Beispiele
A disjoint B	$\begin{bmatrix} F & F & * \\ F & F & * \\ * & * & * \end{bmatrix}$	
A touches B ($d(A) > 0 \vee d(B) > 0$)	$\begin{bmatrix} F & T & * \\ * & * & * \\ * & * & * \end{bmatrix} \vee \begin{bmatrix} F & * & * \\ T & * & * \\ * & * & * \end{bmatrix} \vee \begin{bmatrix} F & * & * \\ * & T & * \\ * & * & * \end{bmatrix}$	
A crosses B ($d(A) < d(B)$)	$\begin{bmatrix} T & * & T \\ * & * & * \\ * & * & * \end{bmatrix}$	
A crosses B ($d(A) = d(B) = 1$)	$\begin{bmatrix} 0 & * & * \\ * & * & * \\ * & * & * \end{bmatrix}$	
A within B	$\begin{bmatrix} T & * & F \\ * & * & F \\ * & * & * \end{bmatrix}$	
A overlaps B ($d(A) = d(B)$, $d(A) \neq 1$, $d(B) \neq 1$)	$\begin{bmatrix} T & * & T \\ * & * & * \\ T & * & * \end{bmatrix}$	
A overlaps B ($d(A) = d(B) = 1$)	$\begin{bmatrix} 1 & * & T \\ * & * & * \\ T & * & * \end{bmatrix}$	

The Region Connection Calculus (RCC)

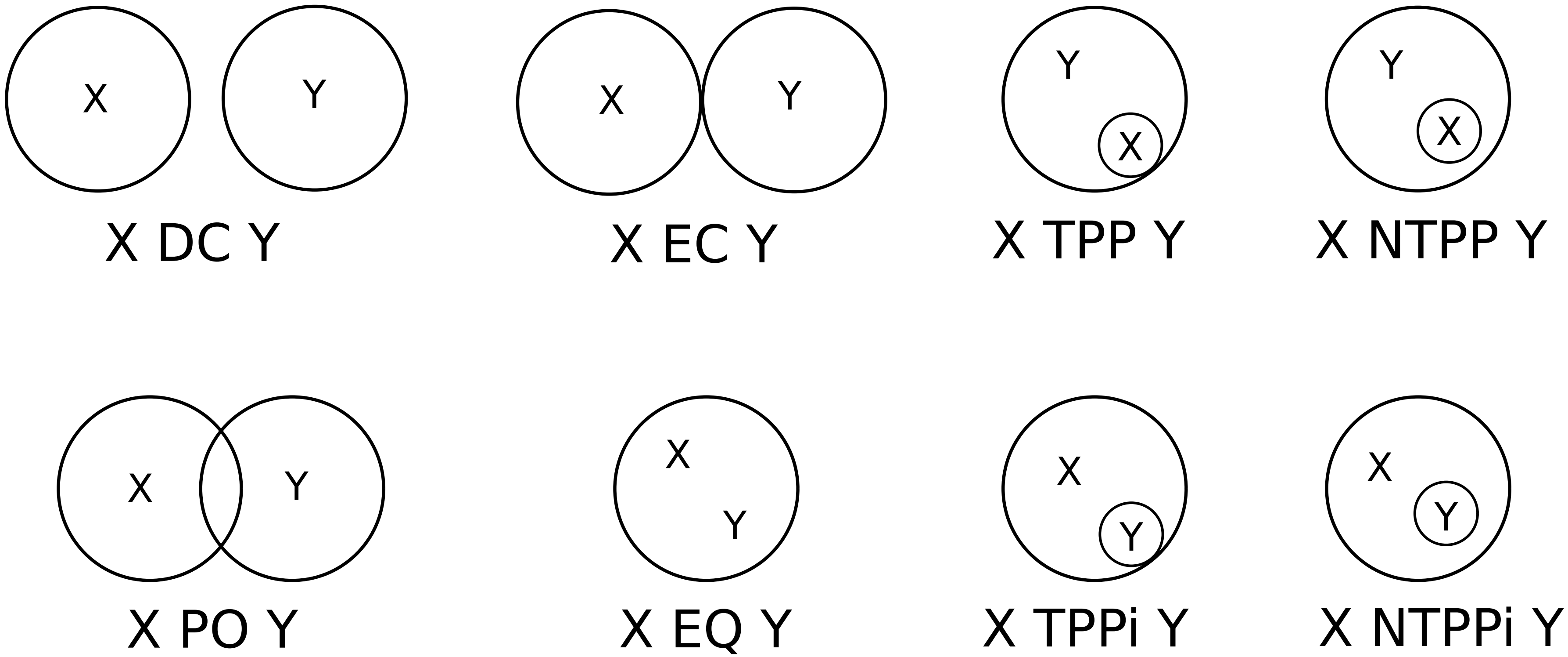
- The primitives of the calculus are **spatial regions**. These are non-empty, regular subsets of a topological space.
- The calculus is based on a single binary predicate *C* that formalizes the "**connectedness**" relation.
  - ***C(a,b)* is true** when the closure of *a* is connected to the closure of *b* i.e., they have at least one point in common.
- It is axiomatized using first order logic.
- See the original paper by Randell, Cui and Cohn (KR 1991).

The Region Connection Calculus (RCC)

Relation	Description	Definition
$C(x,y)$	connects with	primitive relation
$\mathbf{DC}(x,y)$	disconnected	$\neg C(x,y)$
$P(x,y)$	part	$\forall z[C(z,x) \rightarrow C(z,y)]$
$PP(x,y)$	proper part	$P(x,y) \land \neg P(y,x)$
$\mathbf{EQ}(x,y)$	equals	$P(x,y) \land P(y,x)$
$O(x,y)$	overlaps	$\exists z[P(z,x) \land P(z,y)]$
$\mathbf{PO}(x,y)$	partially overlaps	$O(x,y) \land \neg P(x,y) \land \neg P(y,x)$
$DR(x,y)$	discrete	$\neg O(x,y)$
$\mathbf{TPP}(x,y)$	tangential proper part	$PP(x,y) \land \exists z[EC(z,x) \land EC(z,y)]$
$\mathbf{EC}(x,y)$	externally connected	$C(x,y) \land \neg O(x,y)$
$\mathbf{NTPP}(x,y)$	non-tangential proper part	$PP(x,y) \land \neg \exists z[EC(z,x) \land EC(z,y)]$
$Pi(x,y)$	part inverse	$P(y,x)$
$PPi(x,y)$	proper part inverse	$PP(y,x)$
$\mathbf{TPPi}(x,y)$	tangential proper part inverse	$TPP(y,x)$
$\mathbf{NTPPi}(x,y)$	non-tangential proper part inverse	$NTPP(y,x)$

RCC-8

- This is a set of **eight JEPD binary relations** that can be defined in terms of predicate *C*.



RCC-5

- The **RCC-5 subset** has also been studied. The granularity here is coarser. The boundary of a region is not taken into consideration:
  - No distinction among DC and EC, called just DR.
  - No distinction among TPP and NTPP, called just PP.

- RCC-8 and RCC-5 relations **can also be defined using point-set topology**, and there are very close connections to the models of Egenhofer and others.

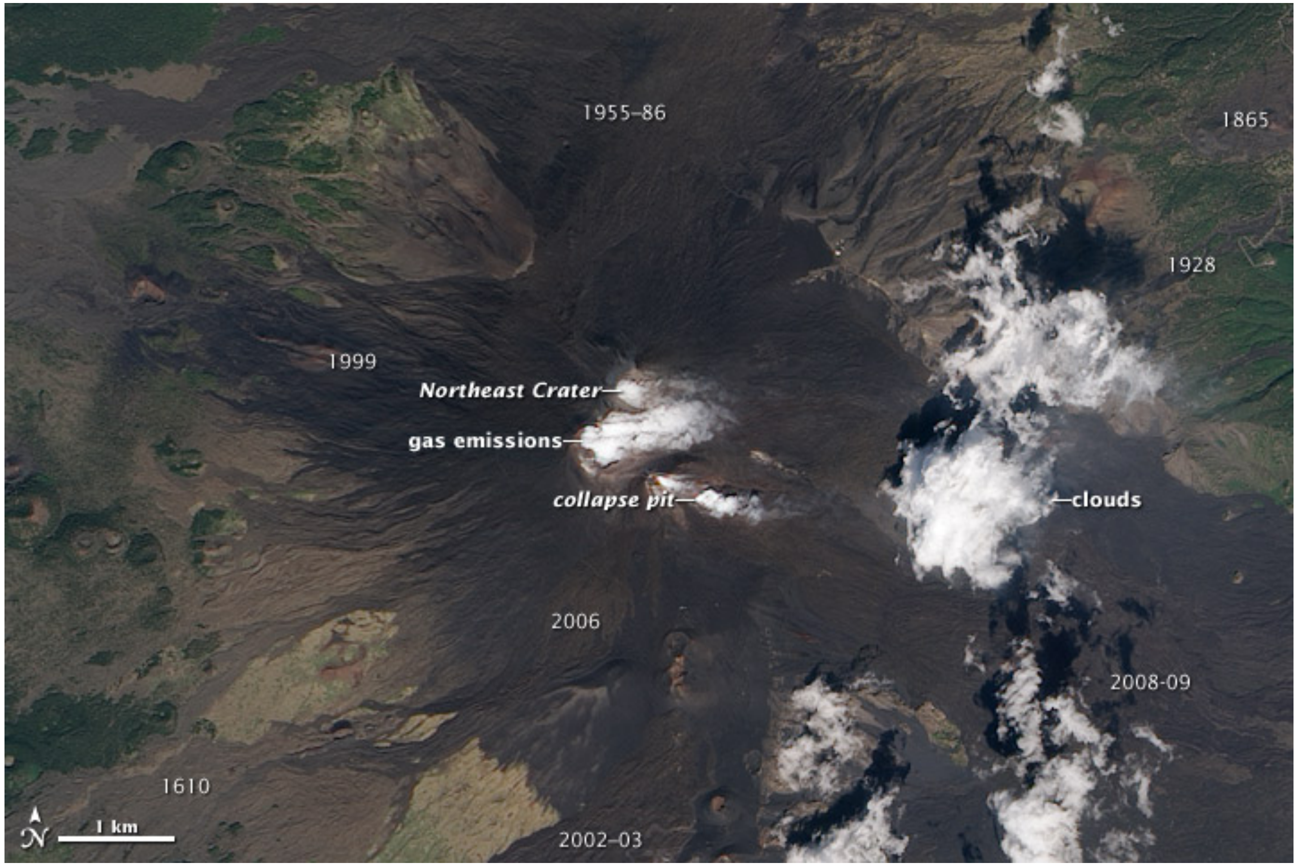
Geographic Space Modeling Paradigms

- **Abstract** geographic space modeling paradigms: discrete objects vs. continuous fields

- **Concrete representations**: tessellation vs. vectors vs. constraints

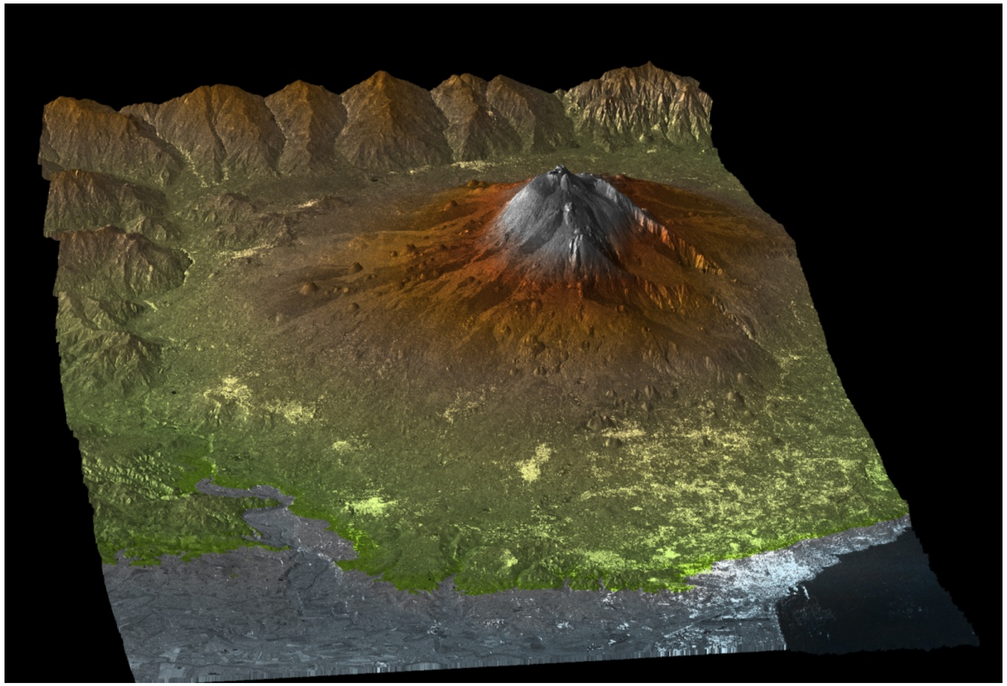
Abstract Modeling Paradigms: Feature-based

- **Feature-based** (or **entity-based** or **object-based**). This kind of modeling is based on the concepts we presented already.



Abstract Modeling Paradigms: Field-based

- Each point *(x,y)* in geographic space is associated with one or several attribute values defined as **continuous functions** in *x* and *y*.
- **Examples:** elevation, precipitation, humidity, temperature for each point *(x,y)* in the Euclidean plane.



From Abstract Modeling to Concrete Representations

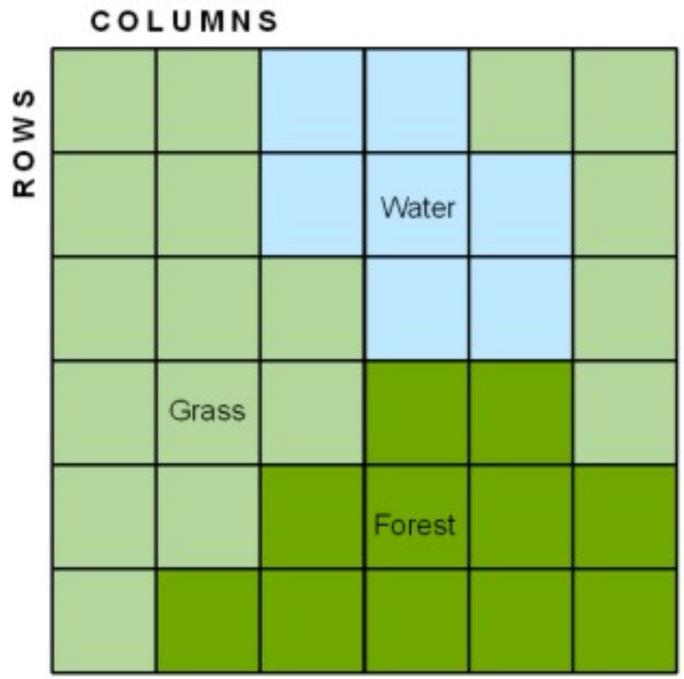
- **Question**: How do we represent the **infinite objects** of the abstract representations (points, lines, fields, etc.) **by finite means** (in a computer)?

- **Answers**:
  - Approximate the continuous space (e.g., R^2) by a discrete one (Z^2)
  - Use **special encodings**

Approximations: Tessellation

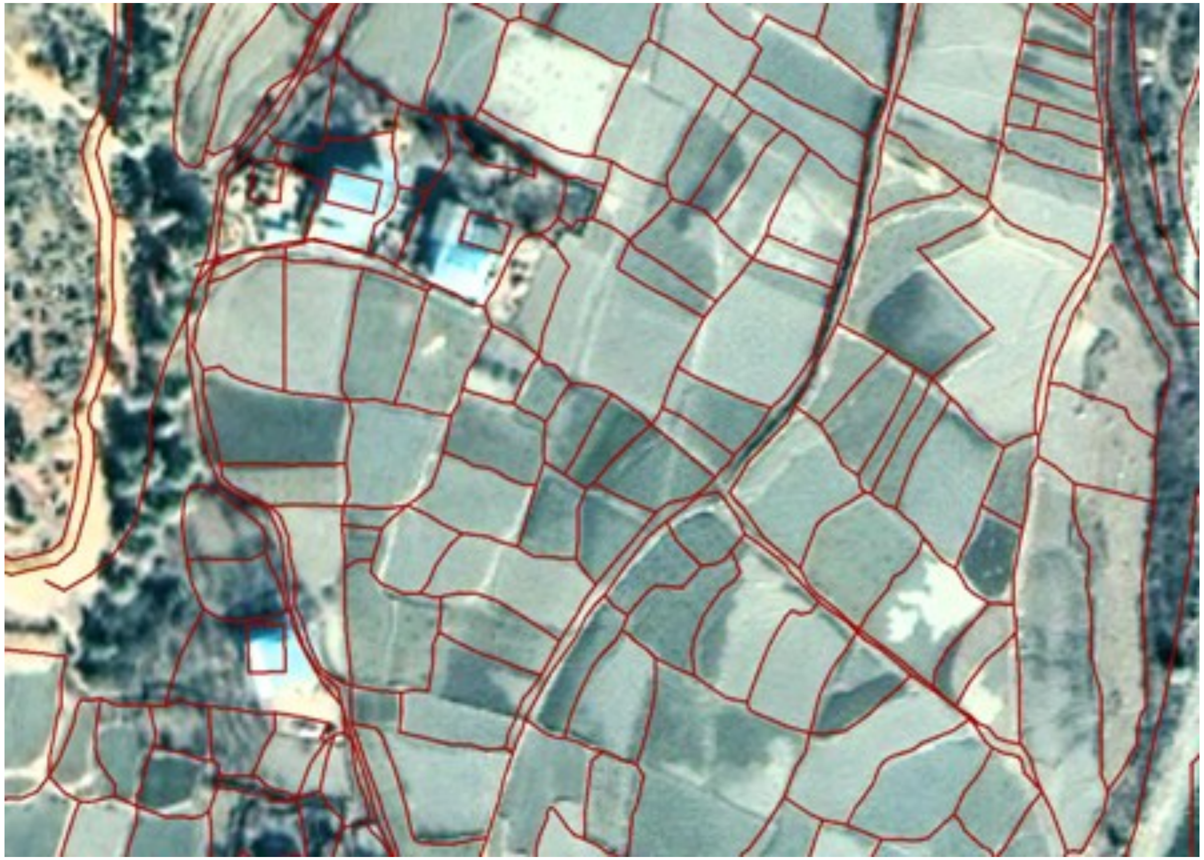
- In this case a **cellular decomposition of the plane** (usually, a grid) serves as a basis for representing the geometry.

- **Example:** raster representation (fixed or regular tesselation)



Example

- **Cadastral map (irregular tessellation)** overlayed on a satellite image.

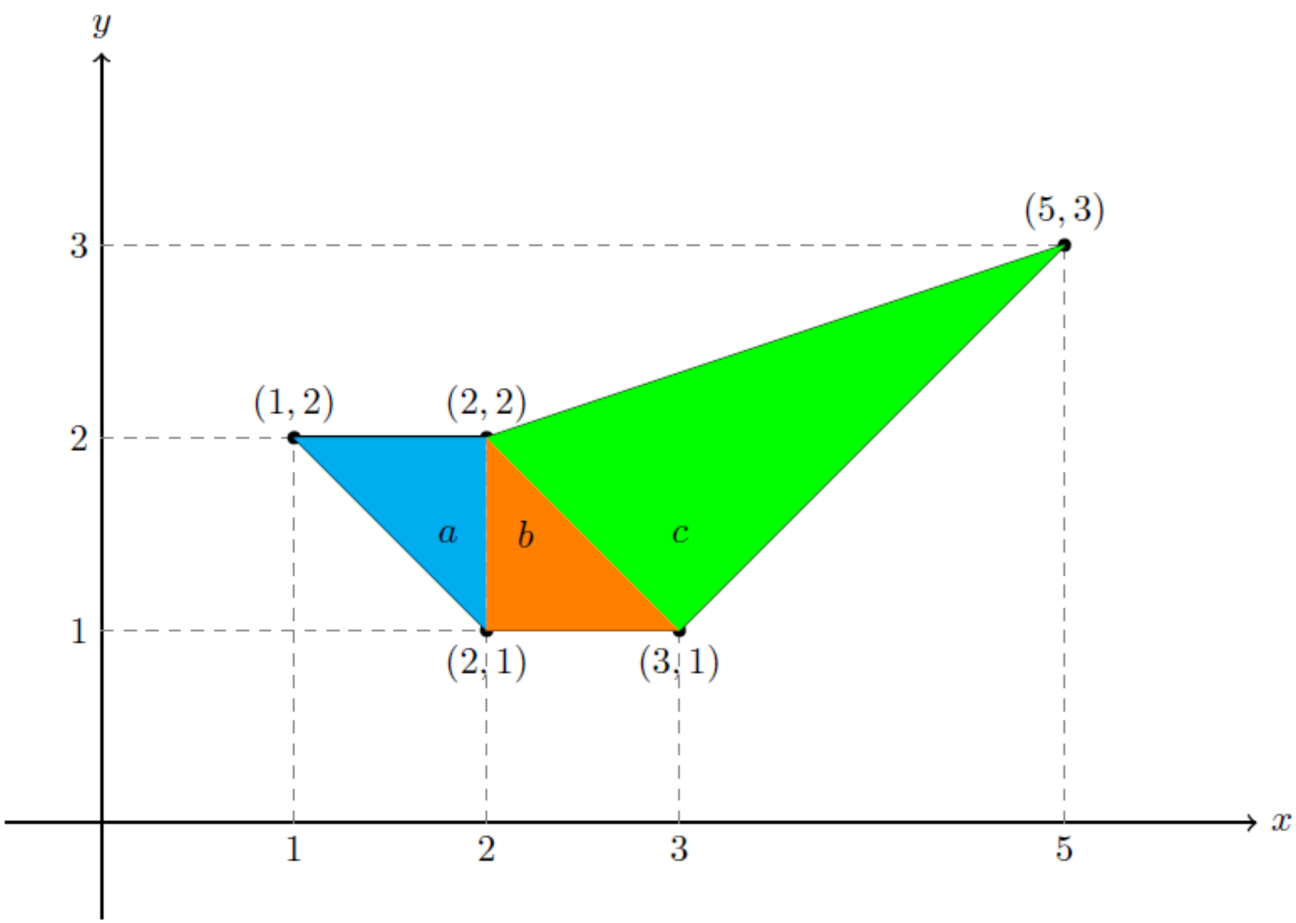


Special Encodings: Vector Representation

- In this case objects in space are represented using **points** as primitives as follows:
  - A **point** is represented by a tuple of coordinates.
  - A **line segment** is represented by a pair with its beginning and ending point.
  - **More complex objects** such as arbitrary lines, curves, surfaces etc. are built recursively by the basic primitives using constructs such as lists, sets etc.

- This is the approach **used in all GIS and other popular systems today. It has also been standardized** by various international bodies.

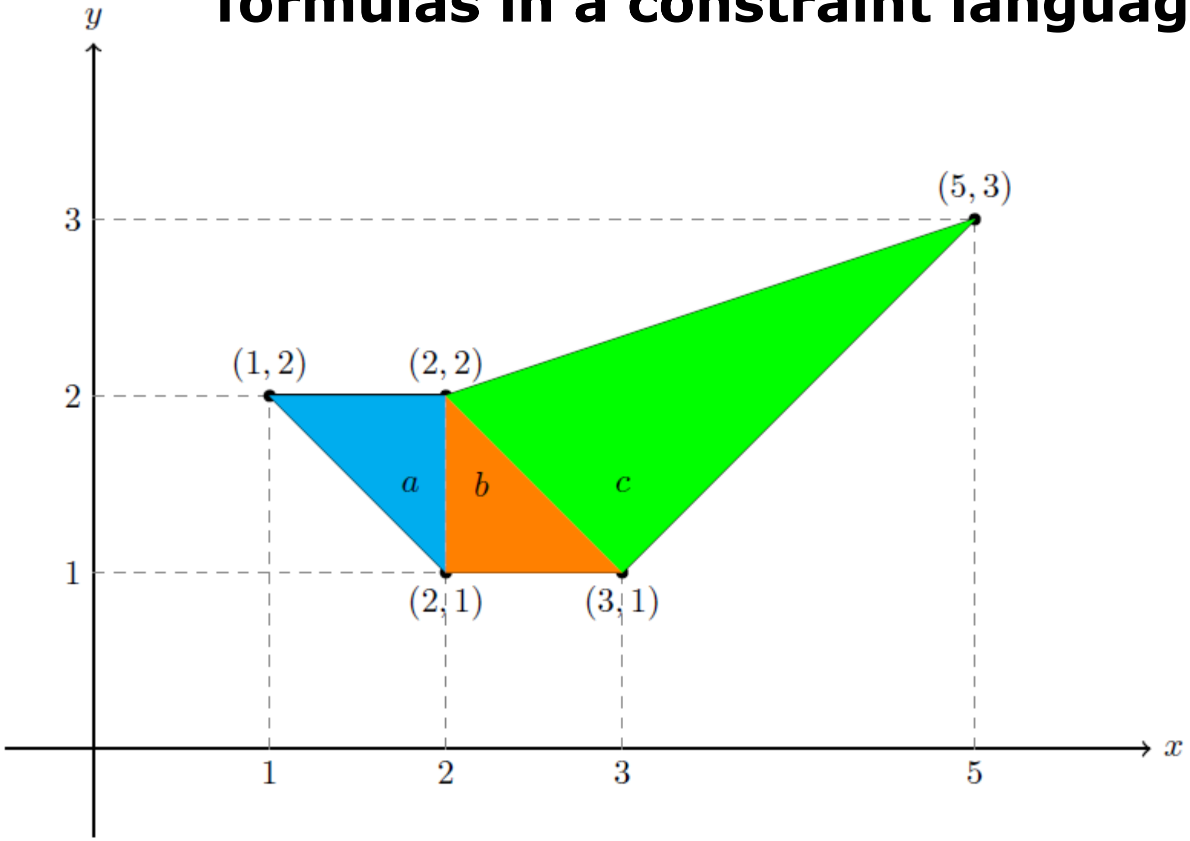
Example




```
[(1,2) (2,2) (5,3) (3,1) (2,1) (1 2)]
```

Special Encodings: Constraint Representation

- In this case objects in space are represented by **quantifier free formulas in a constraint language** (e.g., linear constraints).



$$(y + x \geq 3 \wedge x \leq 2 \wedge y \leq 2) \vee (y + x \leq 4 \wedge x \geq 2 \wedge y \geq 1) \vee (y \geq 3 \wedge x \leq 5 \wedge y - \frac{x}{3} \leq \frac{4}{3})$$

Constraint Databases

- The constraint representation of spatial data was the focus of much work in **databases, logic programming and AI** after the paper by Kanellakis, Kuper and Revesz (PODS, 1991).

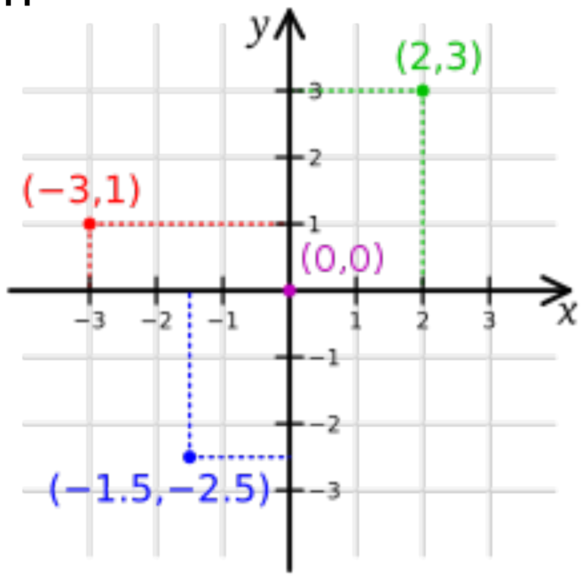
- The approach was very fruitful theoretically but was not adopted in practice.

- See the book by Revesz for a tutorial introduction.

Coordinate Systems

- **Coordinate:** one of *n* scalar values that determines the position of a point in an *n*-dimensional space.
- **Coordinate system:** a set of mathematical rules for specifying how coordinates are to be assigned to points.

- **Example**: the Cartesian coordinate system



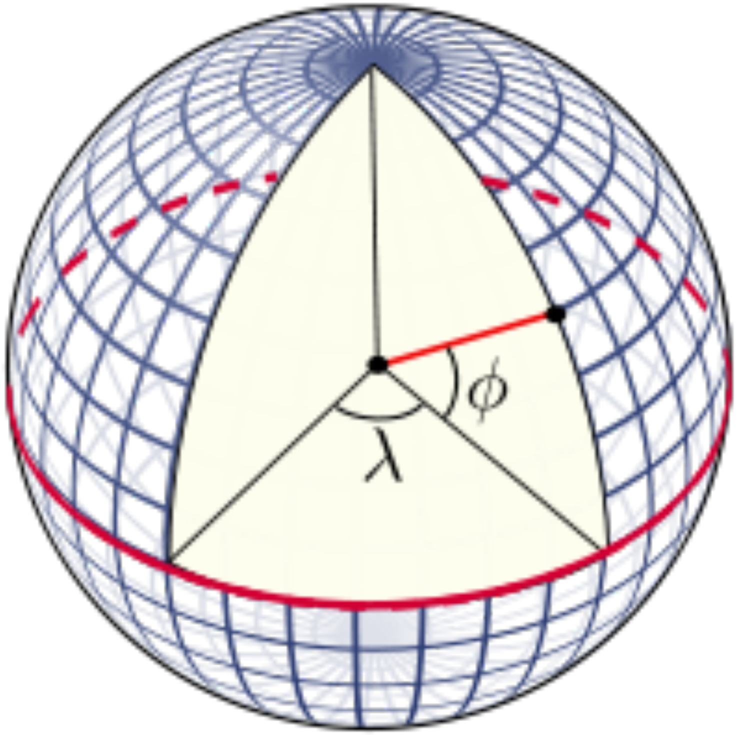
Coordinate Reference Systems

- **Coordinate reference system**: a coordinate system that is related to an **object** (e.g., the Earth, a planar projection of the Earth, a three dimensional mathematical space such as R^3) through a **datum** which specifies its origin, scale, and orientation.

- The term **spatial reference system** is also used.

Geographic Coordinate Reference Systems

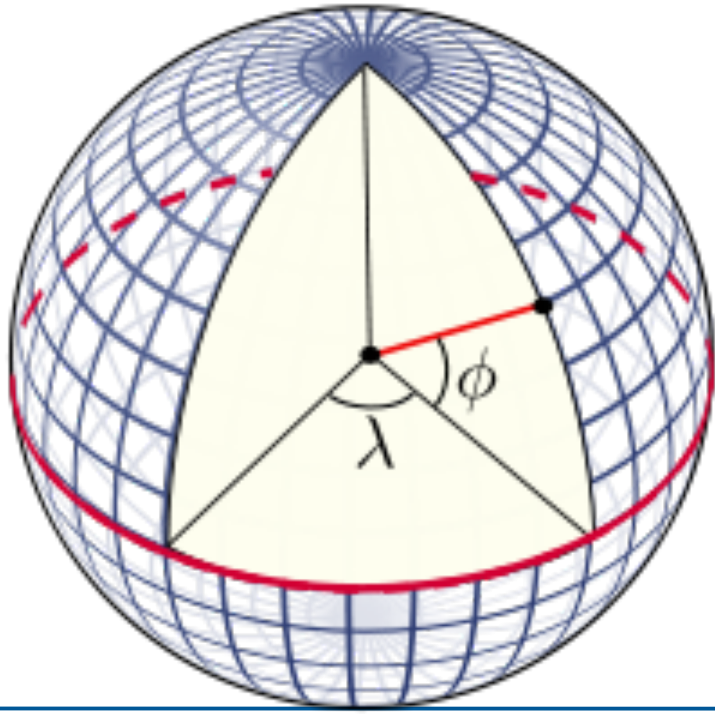
- These are 3-dimensional coordinate systems that utilize **latitude (φ)**, **longitude (λ)** , and optionally **geodetic height (i.e., elevation)**, to capture geographic locations on Earth.



The World Geodetic System

- The **World Geodetic System (WGS)** is the most well-known geographic coordinate reference system and its latest revision is **WGS84.**

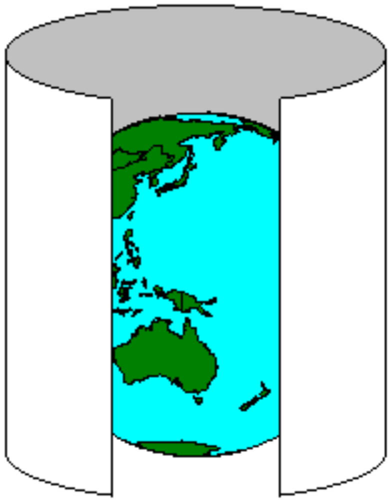
- **Applications:** cartography, geodesy, navigation (GPS), etc.



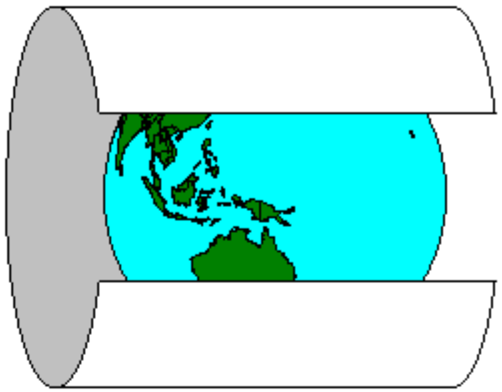
Projected Coordinate Reference Systems

- **Projected coordinate reference system:** they transform the 3-dimensional approximation of the Earth into a 2-dimensional surface (distortions!)

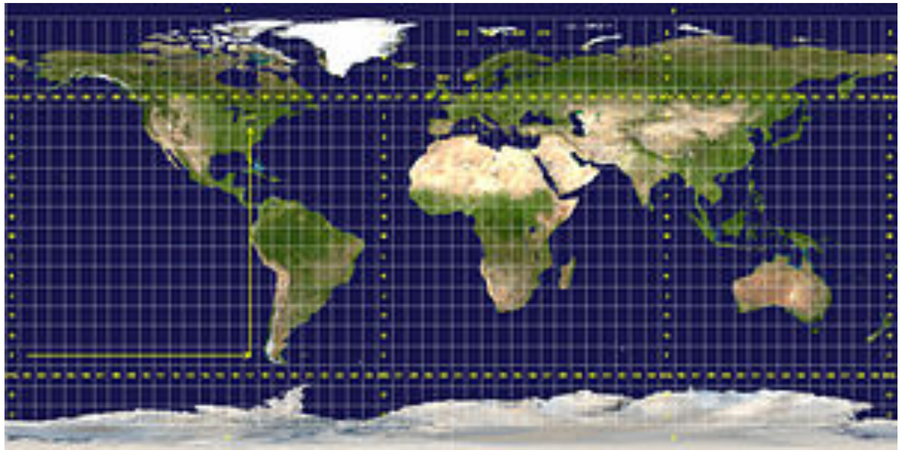
- **Example:** the **Universal Transverse Mercator (UTM)** system



Mercator projection



Transverse Mercator projection



Coordinate Reference Systems (cont'd)

- There are well-known ways to **translate** between co-ordinate reference systems.

- Various authorities maintain lists of coordinate reference systems. See for example:
  - **OGC** http://www.opengis.net/def/crs/
  - **European Petroleum Survey Group** http://www.epsg-registry.org/

Geospatial Data Standards

- The **Open Geospatial Consortium (OGC)** and the **International Organization for Standardization (ISO)** have developed many geospatial data standards that are in wide use today. In this tutorial we will cover:
  - **Well-Known Text**
  - **Geography Markup Language**
  - **OpenGIS Simple Feature Access**

OGC®
Making location count.
www.opengeospatial.org

ISO

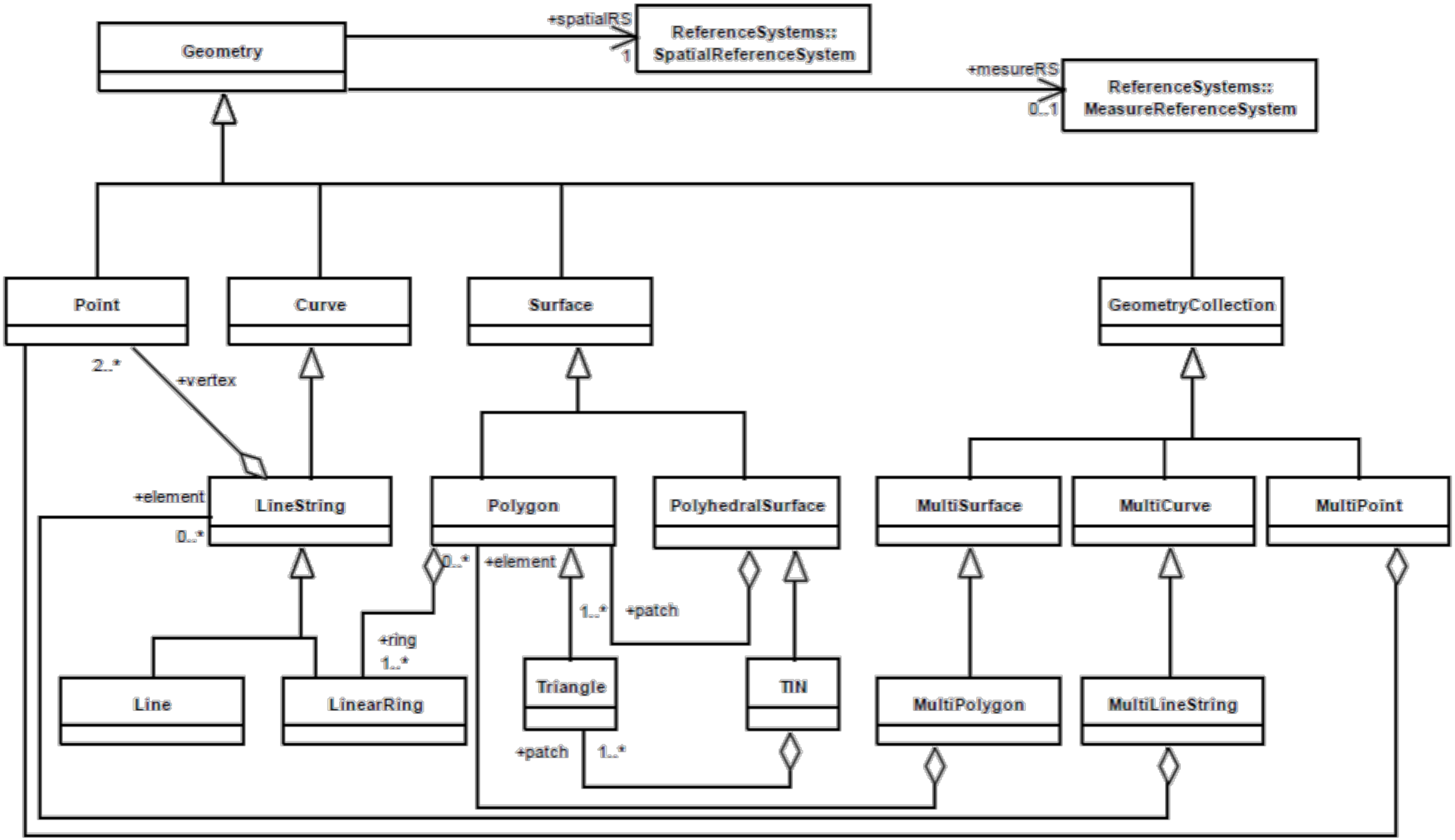
Well-Known Text (WKT)

- WKT is an OGC and ISO standard for representing **geometries**, **coordinate reference systems**, and **transformations** between coordinate reference systems.

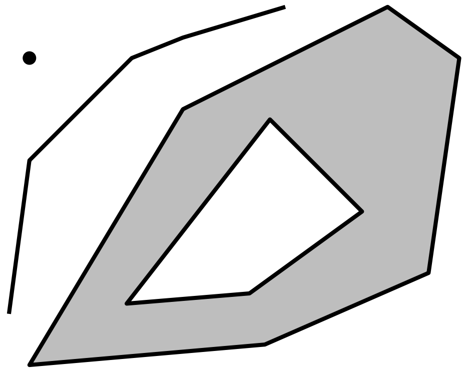
- WKT is specified in **OpenGIS Simple Feature Access - Part 1: Common Architecture** standard which is the same as the **ISO 19125-1** standard. Download from http://portal.opengeospatial.org/files/?artifact_id=25355 .

- This standard concentrates on **simple features:** features with all spatial attributes described piecewise by a straight line or a planar interpolation between sets of points.

WKT Class Hierarchy



Example



WKT representation:

```
GeometryCollection(
  Point(5 35),
  LineString(3 10,5 25,15 35,20 37,30 40),
  Polygon((5 5,28 7,44 14,47 35,40 40,20 30,5 5),
          (28 29,14.5 11,26.5 12,37.5 20,28 29))
  )
```

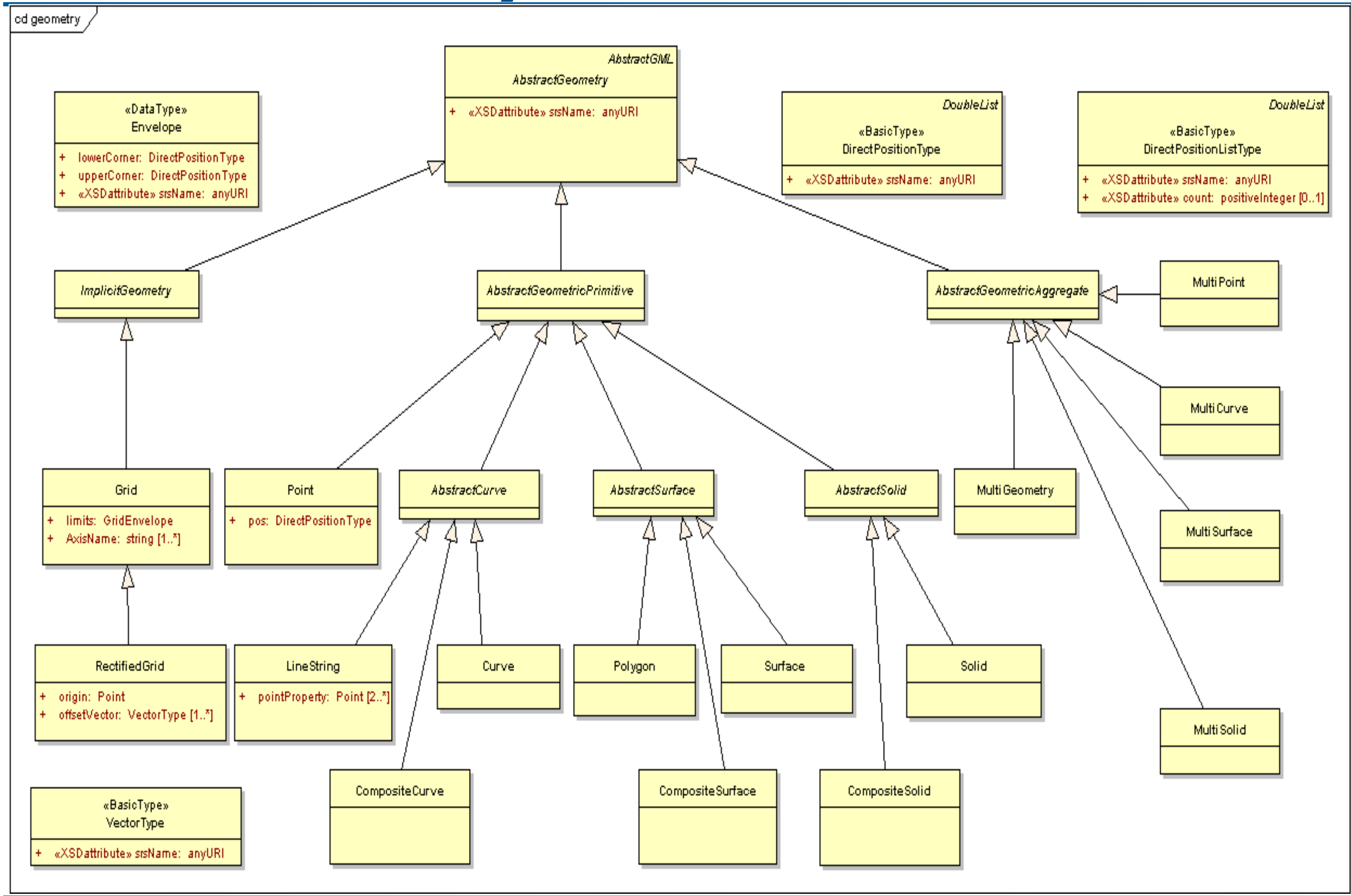
Geography Markup Language (GML)

- **GML** is an **XML-based encoding standard** for the representation of geospatial data.

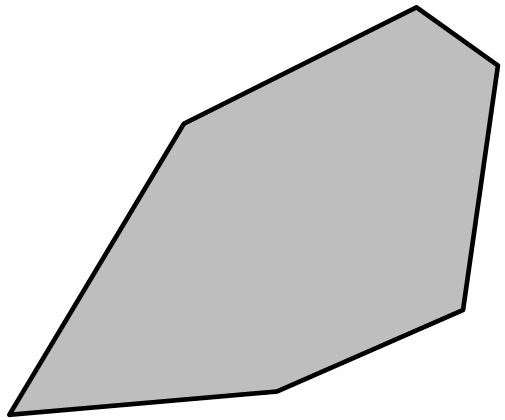
- GML provides XML schemas for defining a variety of concepts: **geographic features, geometry, coordinate reference systems, topology, time** and **units of measurement**.

- **GML profiles** are subsets of GML that target particular applications.
  - **Examples**: Point Profile, GML Simple Features Profile etc.

GML Simple Features: Class Hierarchy



Example



GML representation:

```
<gml:Polygon gml:id="p3" srsName="urn:ogc:def:crs:EPSG:
6.6:4326">
    <gml:exterior>
        <gml:LinearRing>
            <gml:coordinates>
                5,5 28,7 44,14 47,35 40,40 20,30 5,5
            </gml:coordinates>
        </gml:LinearRing>
    </gml:exterior>
</gml:Polygon>
```

OpenGIS Simple Features Access (cont'd)

- OGC has also specified a standard for the **storage, retrieval, query and update of sets of simple features using relational DBMS and SQL.**
- This standard is **"OpenGIS Simple Feature Access - Part 2: SQL Option"** and it is the same as the **ISO 19125-2** standard. Download from http://portal.opengeospatial.org/files/?artifact_id=25354.
- **Related standard**: ISO 13249 SQL/MM - Part 3.

OpenGIS Simple Features Access (cont'd)

- The standard covers two implementations options: (i) using **only the SQL predefined data types** and (ii) using **SQL with geometry types**.

- **SQL with geometry types:**
  - We use the WKT geometry class hierarchy presented earlier to define **new geometric data types for SQL**
  - We define new **SQL functions on those types**.

SQL with Geometry Types - Functions

- Functions that **request or check properties** of a geometry:
  - `ST_Dimension(A:Geometry):Integer`
  - `ST_GeometryType(A:Geometry):Character Varying`
  - `ST_AsText(A:Geometry): Character Large Object`
  - `ST_AsBinary(A:Geometry): Binary Large Object`
  - `ST_SRID(A:Geometry): Integer`
  - `ST_IsEmpty(A:Geometry): Boolean`
  - `ST_IsSimple(A:Geometry): Boolean`

SQL with Geometry Types – Functions (cont'd)

- Functions that test **topological relations** between two geometries using the **DE-9IM**:
  - `ST_Equals(A:Geometry, B:Geometry):Boolean`
  - `ST_Disjoint(A:Geometry, B:Geometry):Boolean`
  - `ST_Intersects(A:Geometry, B:Geometry):Boolean`
  - `ST_Touches(A:Geometry, B:Geometry):Boolean`
  - `ST_Crosses(A:Geometry, B:Geometry):Boolean`
  - `ST_Within(A:Geometry, B:Geometry):Boolean`
  - `ST_Contains(A:Geometry, B:Geometry):Boolean`
  - `ST_Overlaps(A:Geometry, B:Geometry):Boolean`
  - `ST_Relate(A:Geometry, B:Geometry, Matrix: Char(9)):Boolean`

SQL with Geometry Types – Functions (cont'd)

- Functions for **constructing new geometries** out of existing ones:
  - `ST Boundary(A:Geometry):Geometry`
  - `ST_Envelope(A:Geometry):Geometry`
  - `ST_Intersection(A:Geometry, B:Geometry):Geometry`
  - `ST_Union(A:Geometry, B:Geometry):Geometry`
  - `ST_Difference(A:Geometry, B:Geometry):Geometry`
  - `ST_SymDifference(A:Geometry, B:Geometry):Geometry`
  - `ST_Buffer(A:Geometry, distance:Double):Geometry`

Geospatial Relational DBMS

- The OpenGIS Simple Features Access Standard is today been used in all **relational DBMS with a geospatial extension.**
  - The **abstract data type mechanism** of the DBMS allows the representation of all kinds of geospatial data types supported by the standard.
  - The query language (SQL) offers the **functions** of the standard for querying data of these types.

Conclusions

- **Background in geospatial data modeling**:
  - Why geographical information?
  - Geographical Information Science and Systems
  - Geospatial data on the Web and linked geospatial data
  - Abstract geographic space modeling paradigms: discrete objects vs. continuous fields
  - Concrete representations: tessellation vs. vectors vs. constraints
  - Geospatial data standards

- **Next topic:** Representation and querying of geospatial data in RDF

Representing and Querying Geospatial data in RDF: stSPARQL and GeoSPARQL

Presenter: Kostis Kyzirakos

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

Outline

- Main idea
- Vocabularies and Ontologies
- The data model stRDF
- Examples of publicly available linked geospatial data
- The query language stSPARQL
- The query language GeoSPARQL

Main idea

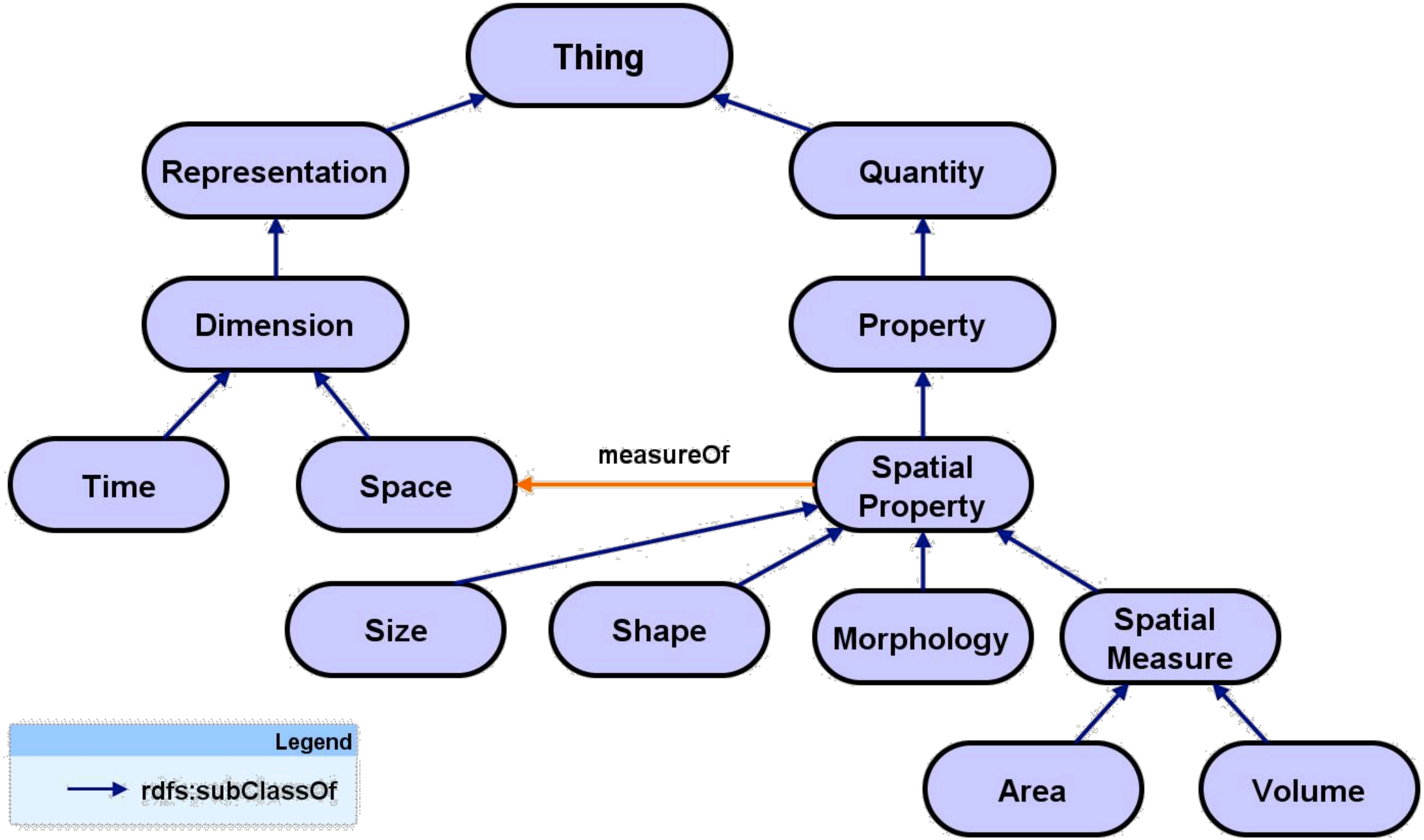
How do we **represent** and **query geospatial information** in the Semantic Web?

Develop appropriate **vocabularies** and **ontologies**

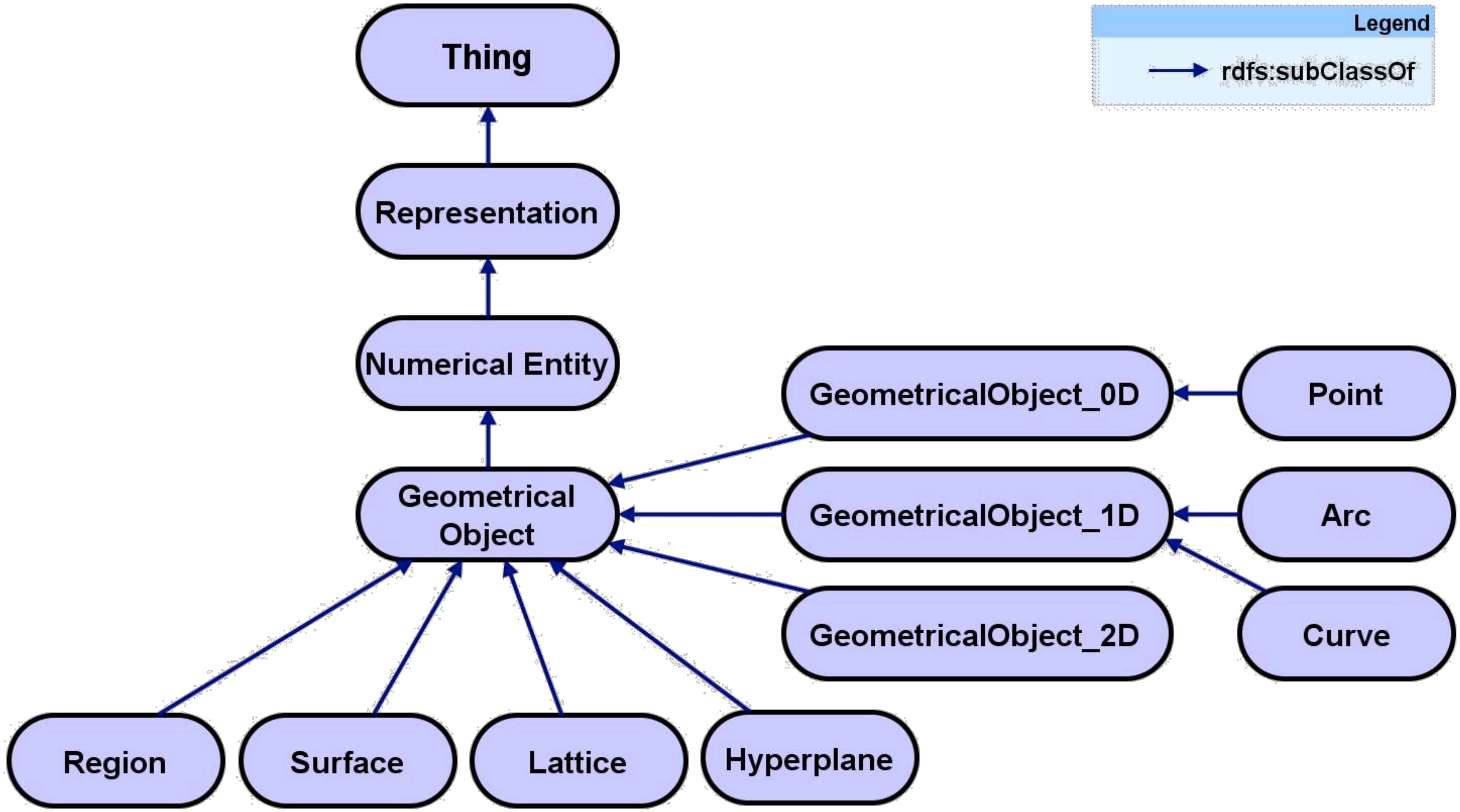
Extend RDF to take into account the geospatial dimension.

Extend SPARQL to query the new kinds of data.

Develop appropriate vocabularies and ontologies: Sweet Ontology

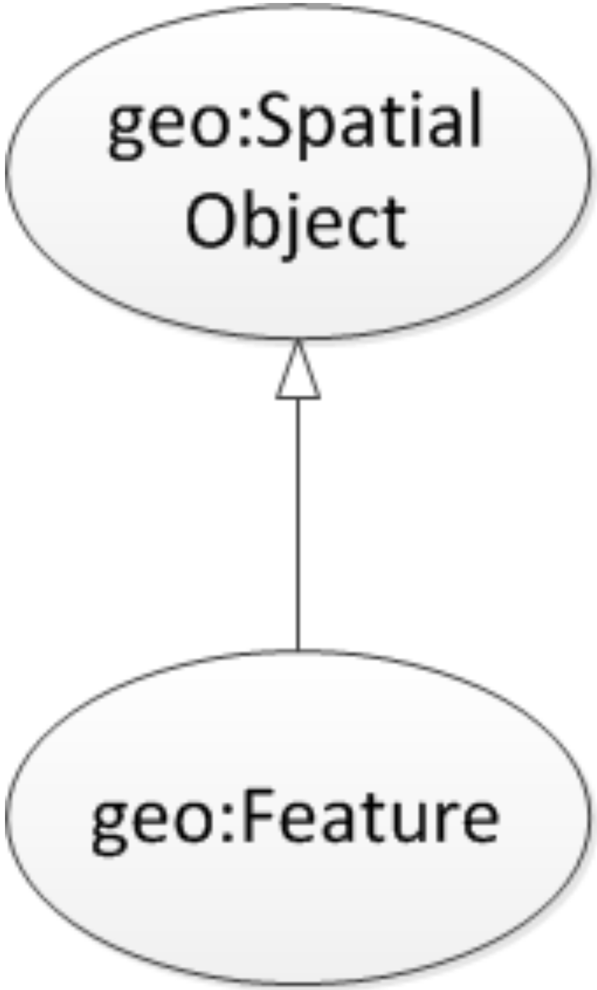


Develop appropriate vocabularies and ontologies: Sweet Ontology (cont'd)



GeoSPARQL Core

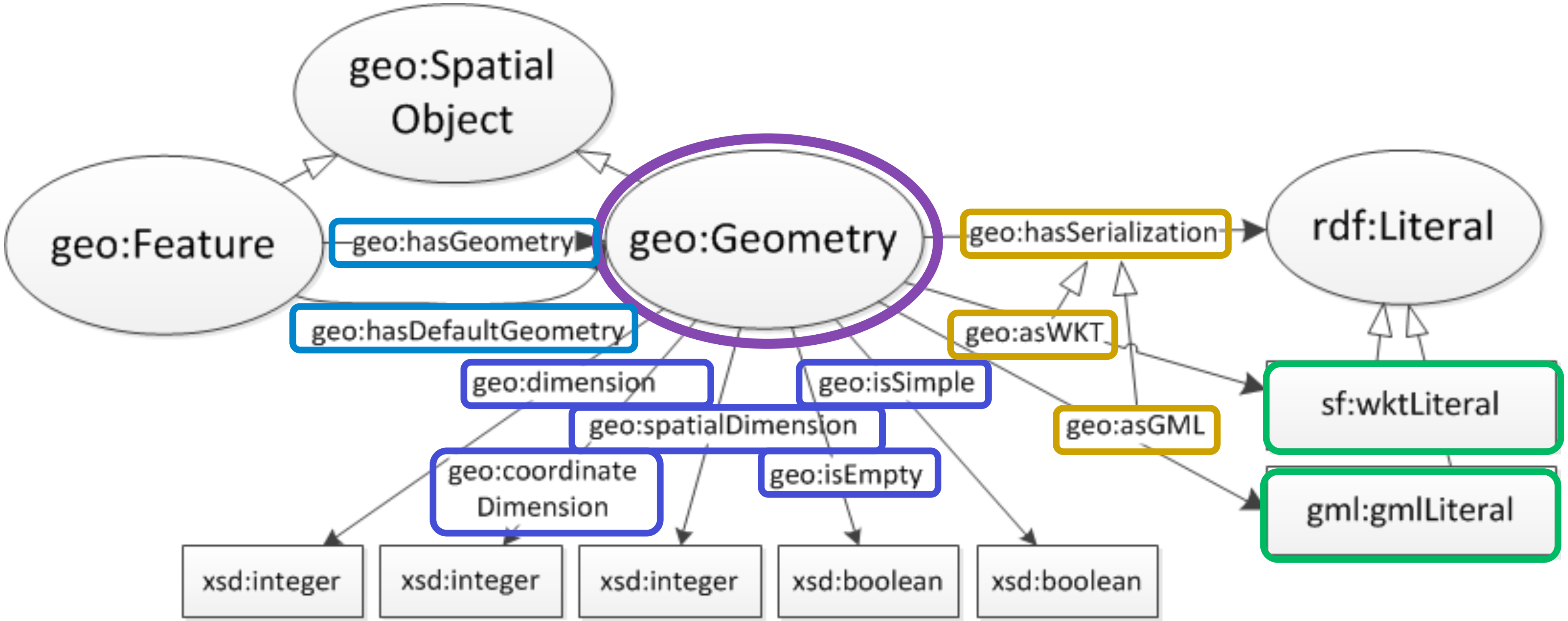
Defines **top level classes** that provides users with vocabulary for modeling geospatial information



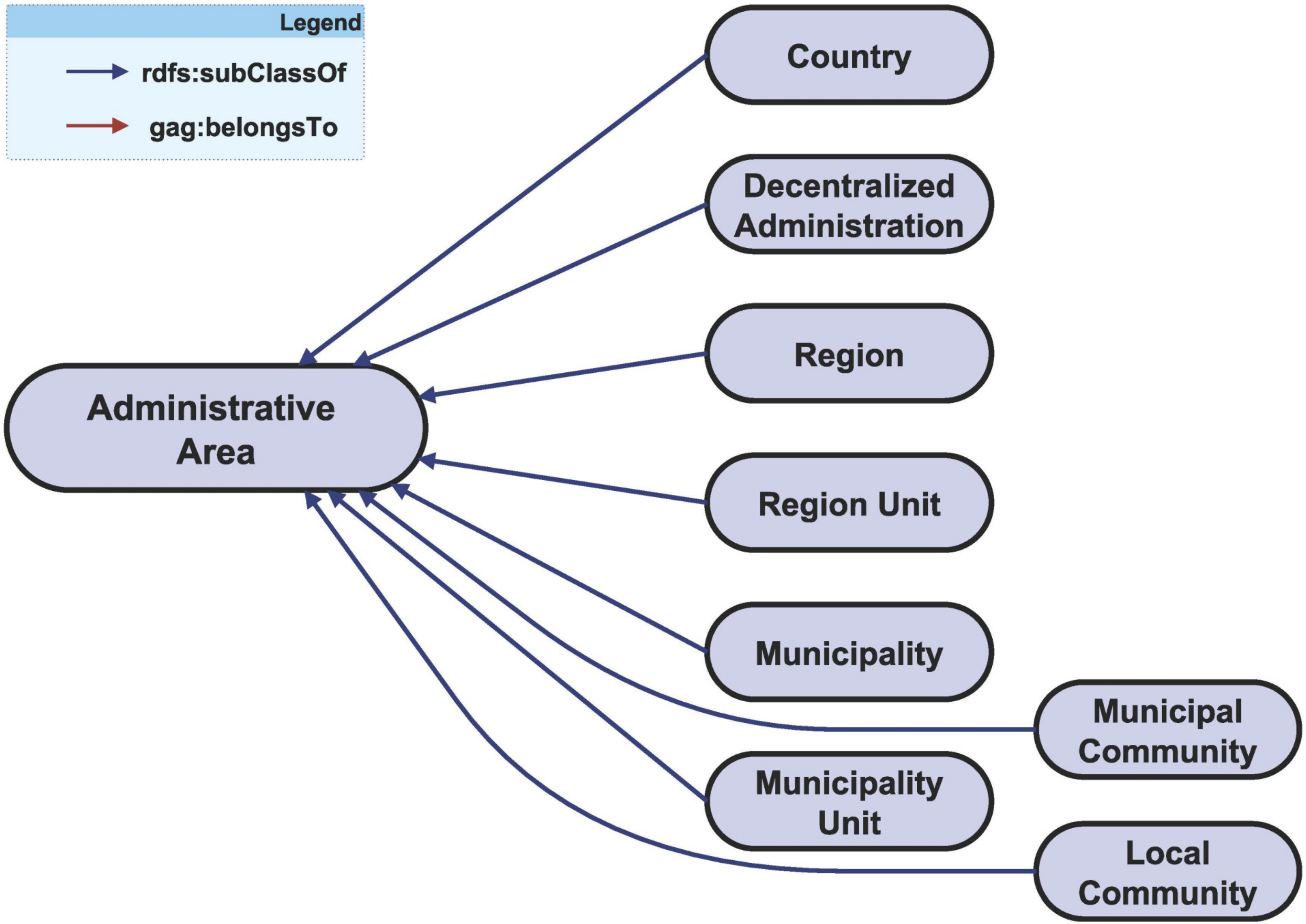
GeoSPARQL Geometry Extension

Provides vocabulary for asserting and querying information about geometries.

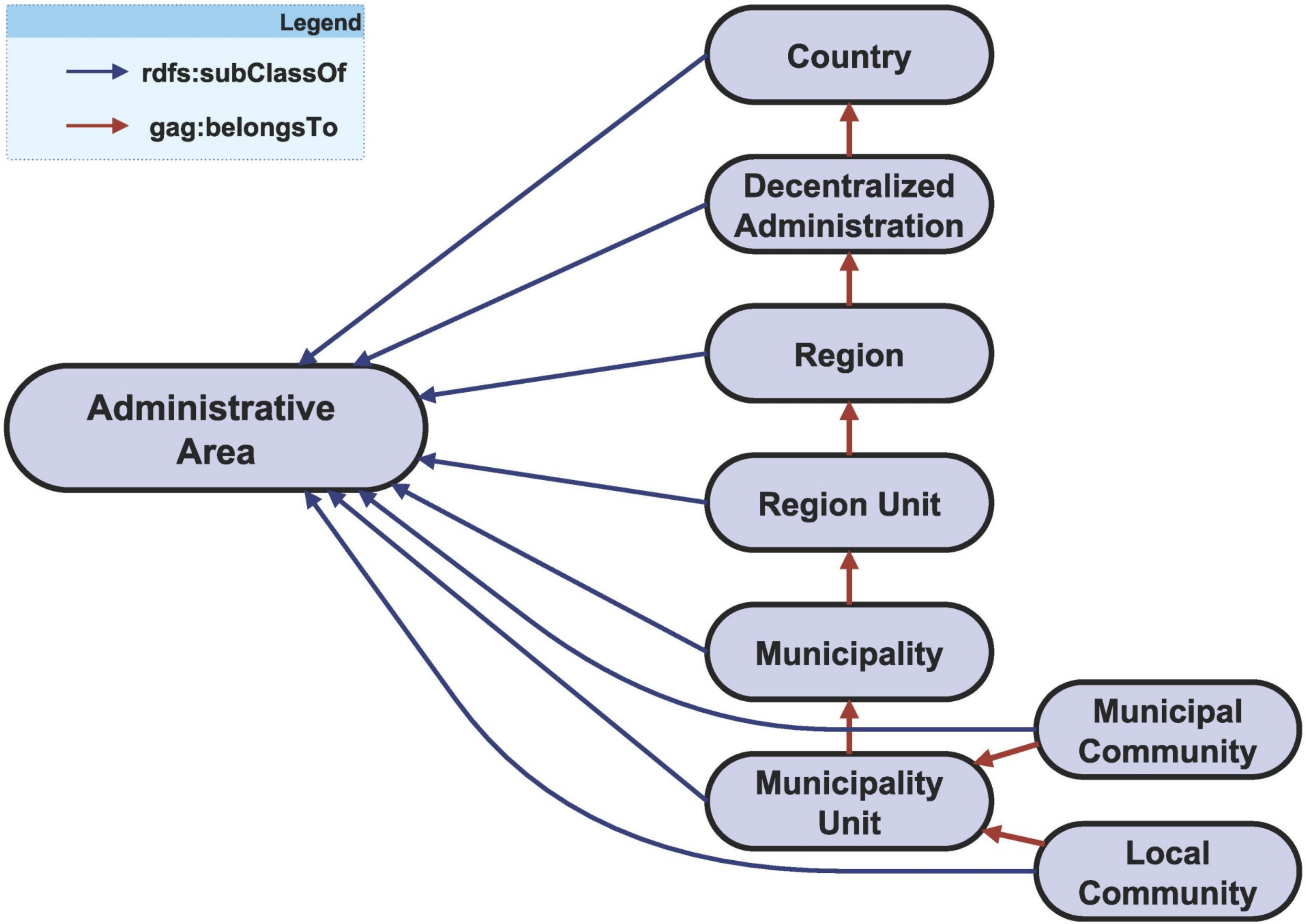
- The class **`geo:Geometry`** is a top class which is a superclass of all geometry classes.



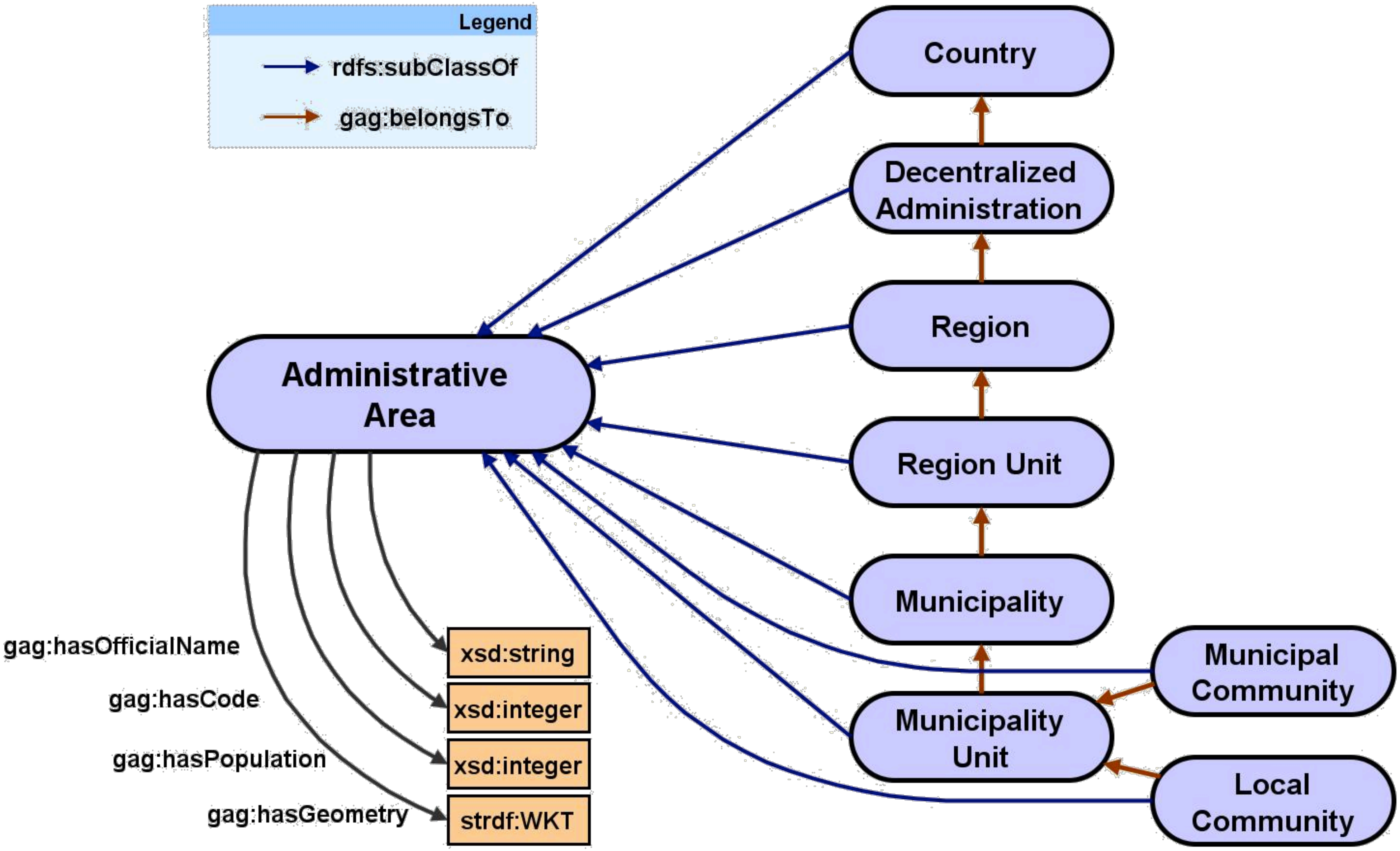
Greek Administrative Geography



Greek Administrative Geography



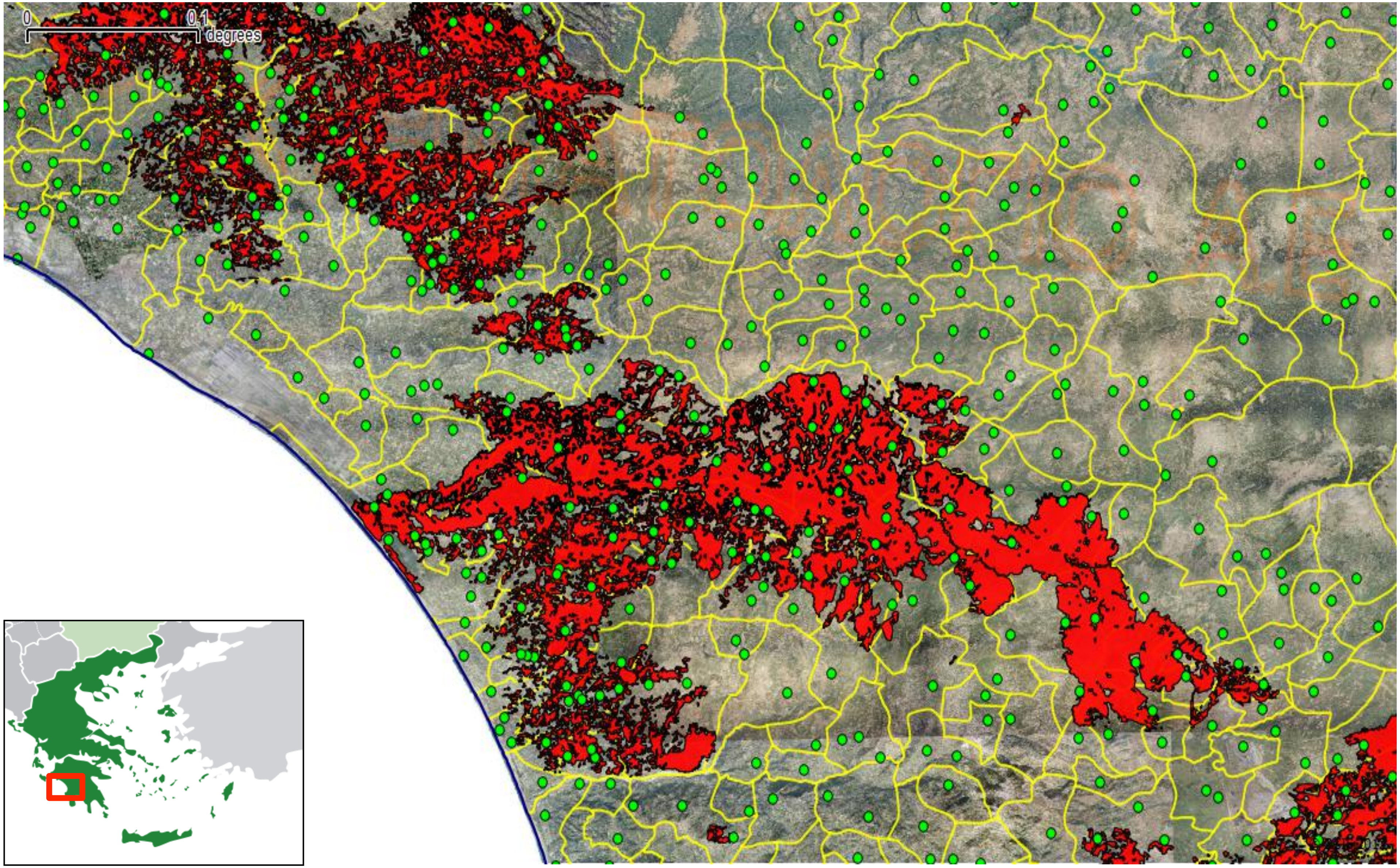
Greek Administrative Geography



stRDF and stSPARQL

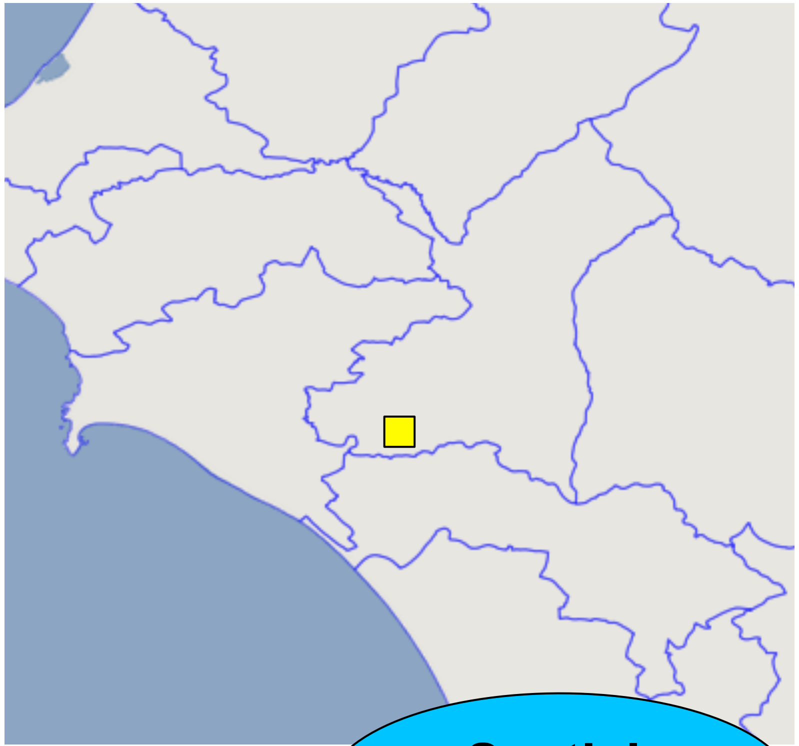
- **Theme**, **space** and **valid time** can be represented [Koubarakis and Kyzirakos, 2010]
- **Linear constraints** are used to represent geometries
- Constraints are represented using literals of an appropriate datatype
- Formal approach
- New version to be presented today uses **OGC standards** to represent and query geometries

Example



Example in stRDF

```
gag:Olympia
  gag:name "Ancient Olympia";
  rdf:type gag:MunicipalCommunity .
```



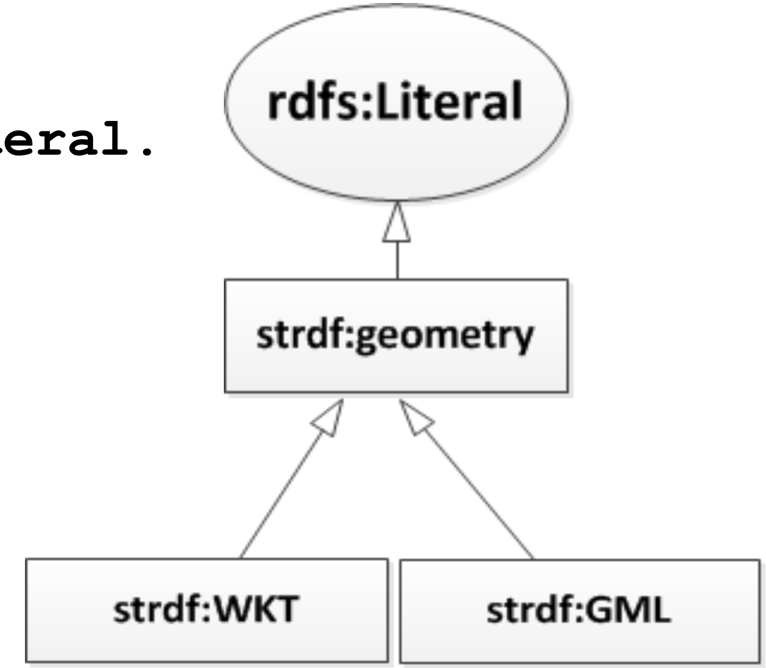
Spatial literal

```
gag:Olympia gag: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   strdf:geometry.

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

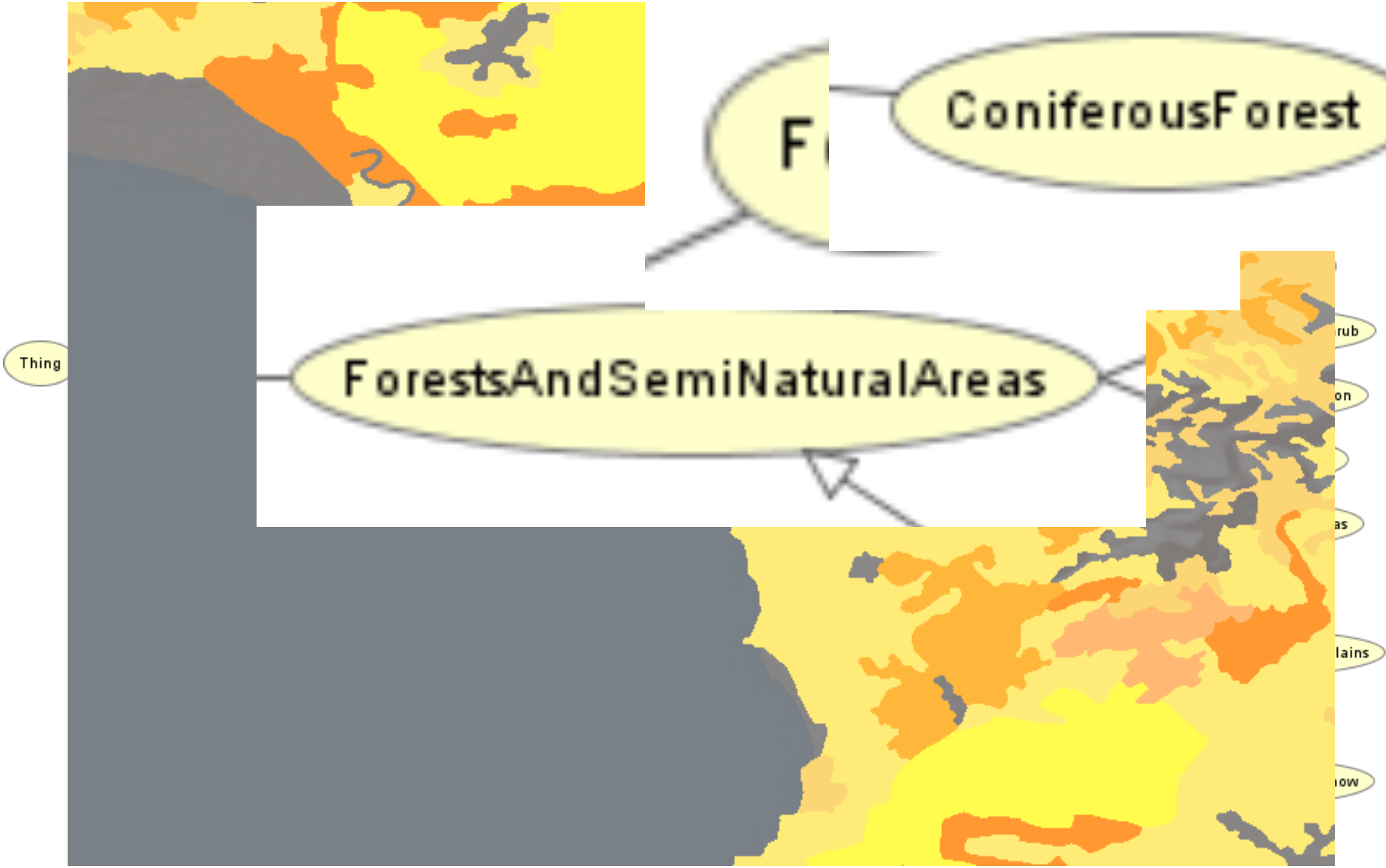
Examples of publicly available linked geospatial data

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

Greek Administrative Geography

```
gag:Olympia
  rdf:type gag:MunicipalCommunity;
  gag:name "Ancient Olympia";
  gag:population "184"^^xsd:int;
  gag:hasGeometry "POLYGON
  (((25.37 35.34,…)))"^^strdf:WKT.

gag:OlympiaMUnit
  rdf:type gag:MunicipalityUnit;
  rdfs:label "Municipality Unit of
              Ancient Olympia".

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

gag:Olympia gag:isPartOf gag:OlympiaMUnit .

gag:OlympiaMUnit gag:isPartOf gag:OlympiaMunicipality.
```

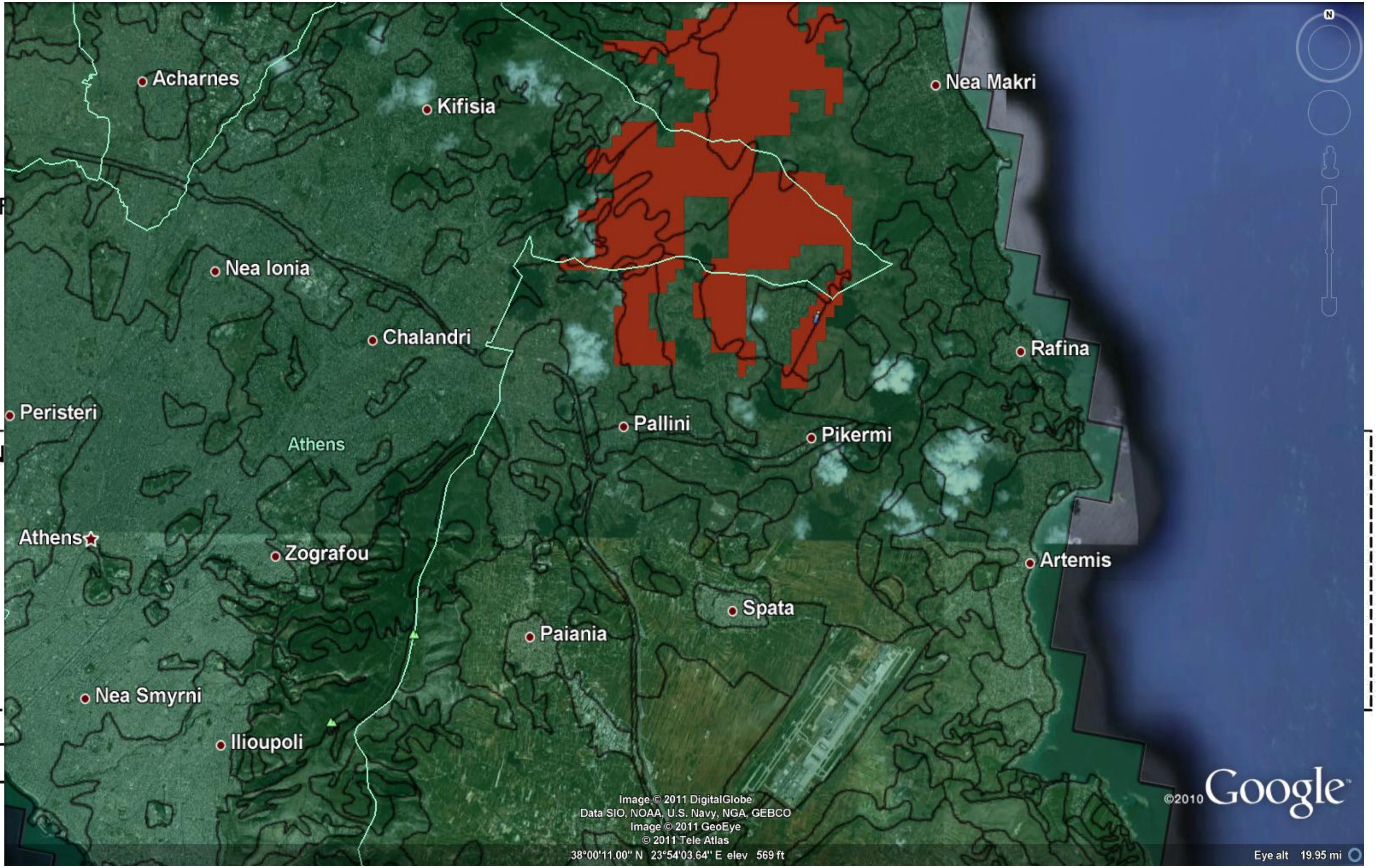
Corine Land Use / Land Cover



Corine Land Use / Land Cover

```
clc:Area_24015134
  rdf:type clc:Area ;
  clc:hasCode "312"^^xsd:decimal;
  clc:hasID "EU-203497"^^xsd:string;
  clc:hasArea_ha "255.5807904"^^xsd:double;
  clc:hasGeometry "POLYGON((15.53 62.54,
                              …))"^^strdf:WKT;
  clc:hasLandUse clc:ConiferousForest .
```

Burnt Area Products



Burnt Area Products

```
noa:ba_15
  rdf:type noa:BurntArea;
  noa:isProducedByProcessingChain
         "static thresholds"^^xsd:string;
  noa:hasAcquisitionTime
         "2010-08-24T13:00:00"^^xsd:dateTime;

  noa: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**
- **RCC-8**

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 {
   ?muni  rdf:type gag:LocalCommunity;
          gag:name ?name;
          gag:hasGeometry ?muniGeo .
   ?ba  rdf:type noa:BurntArea;
        noa:hasGeometry ?baGeo .

   FILTER(strdf:overlap(?muniGeo,?baGeo))
}
```

Spatial Function

stSPARQL: An example (2/3)

Find all burnt forests near communities

```
SELECT ?ba ?baGeom
WHERE {
    ?r rdf:type clc:Region;
       clc:hasGeometry ?rGeom;
       clc:hasCorineLandCoverUse ?f.
    ?f rdfs:subClassOf clc:Forest.
    ?c rdf:type gag:LocalCommunity;
       gag:hasGeometry ?cGeom.
    ?ba rdf:type noa:BurntArea;
        noa: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.

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

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

Spatial Aggregate

Spatial Function

GeoSPARQL

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

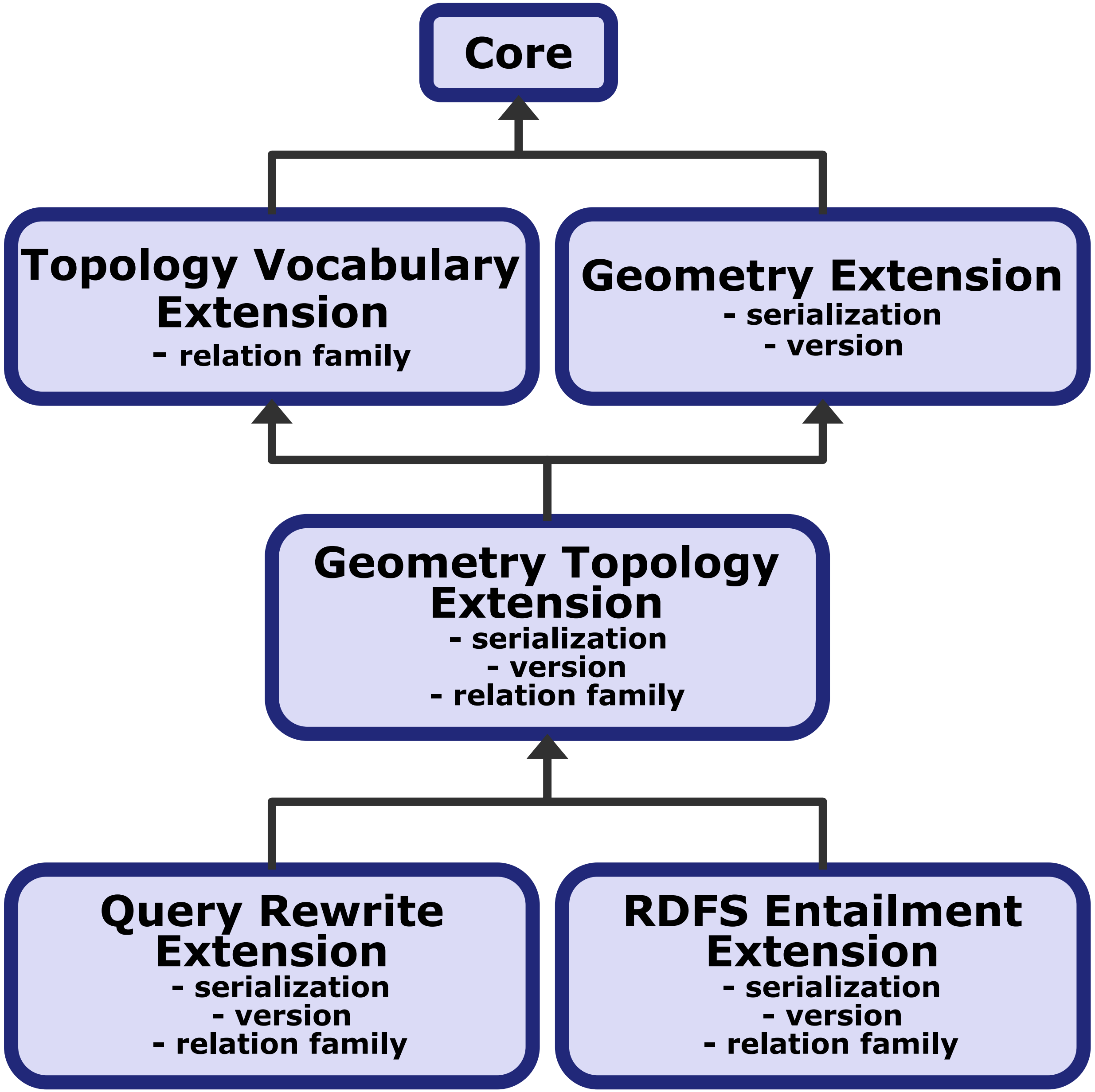
Functionalities **similar to stSPARQL**:

- 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.

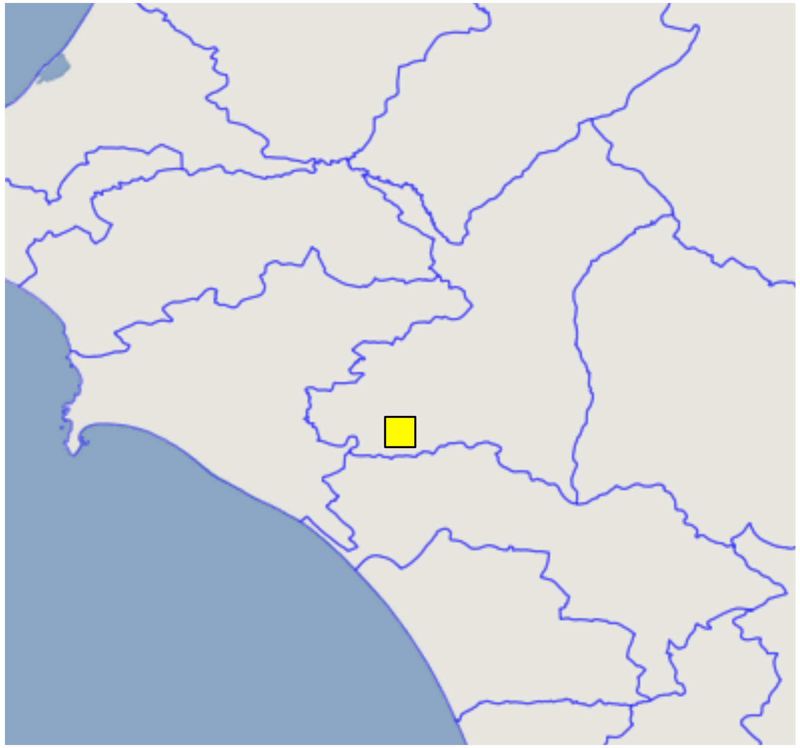
GeoSPARQL Components



Parameters

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

Example in GeoSPARQL (1/2)



```
gag:Olympia
  rdf:type gag:MunicipalCommunity;
  gag:name "Ancient Olympia";
  gag:population "184"^^xsd:int;
  geo:hasGeometry ex:polygon1.
```

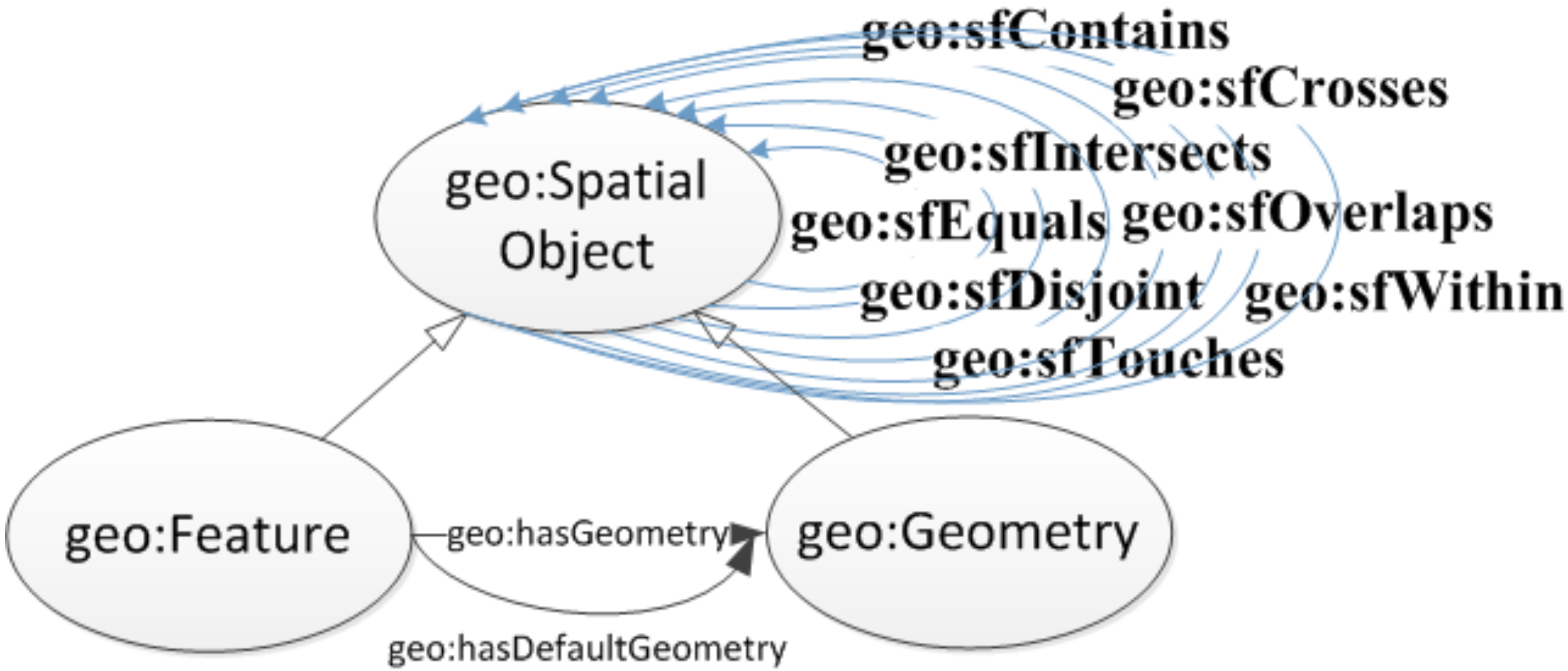
Spatial literal

```
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:wktLi
```

Spatial data type

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`**

Greek Administrative Geography

```
gag:Olympia
  rdf:type gag:MunicipalCommunity;
  gag:name "Ancient Olympia".

gag:OlympiaMUnit
  rdf:type gag:MunicipalityUnit;
  rdfs:label "Municipality Unit of
              Ancient Olympia".

gag:OlympiaMunicipality
  rdf:type gag:Municipality;
  rdfs:label "Municipali
              Ancient
```

Asserted topological relation

```
gag:Olympia geo:sfWithin gag:OlympiaMUnit .

gag:OlympiaMUnit geo:sfWithin gag:OlympiaMunicipality.
```

GeoSPARQL: An example

Find the municipality unit that contains the community of Ancient Olympia

```
SELECT   ?m

WHERE {

 ?m rdf:type gag:MunicipalityUnit.

 ?m geo:sfContains gag: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 gag:Olympia.
}
```

What is the answer to this query?

Example (cont'd)

The answer to the previous query is

```
?m = gag:OlympiaMunicipality
```

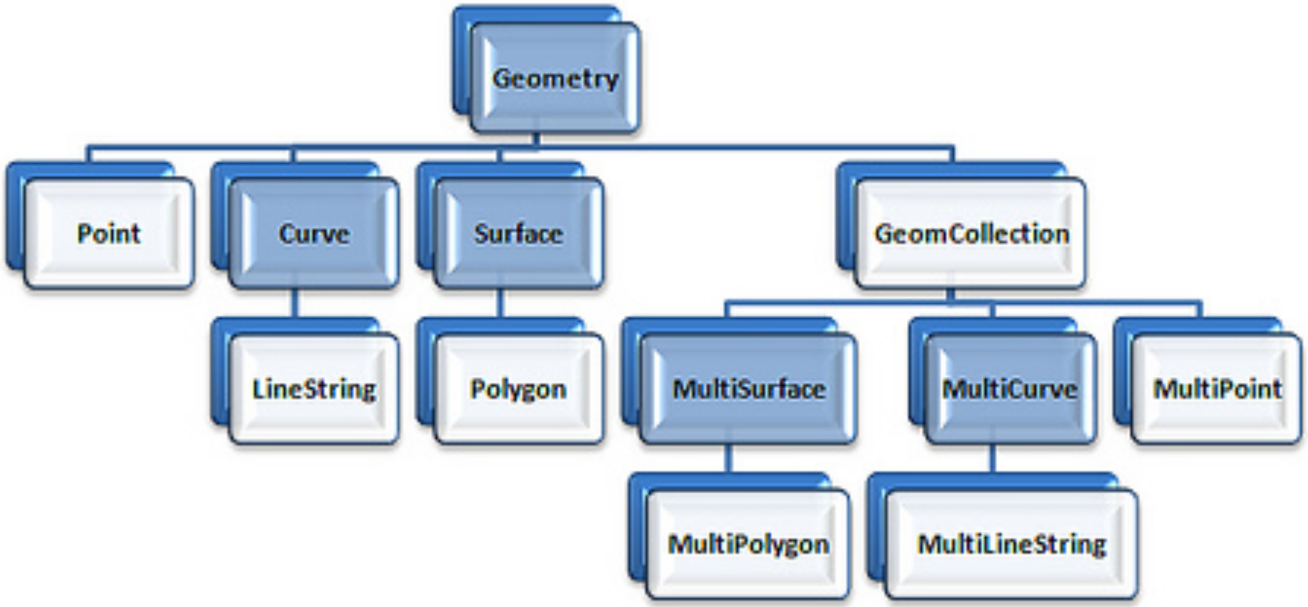
GeoSPARQL does not tell you how to compute this answer which needs **reasoning about the transitivity** of relation `geo:sfContains`.

Options:

- Use rules
- Use constraint-based techniques

GeoSPARQL 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 **`gag:Athens`** and **`gag: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

```
gag:Athens geo:sfWithin gag: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)) }
```

System	Language	Index	Geometries	CRS support	Geospatial Function Support
Strabon	stSPARQL/ GeoSPARQL*	R-tree-over-GiST	WKT / GML support	Yes	• OGC-SFA • Egenhofer • RCC-8
Parliament	GeoSPARQL*	R-Tree	WKT / GML support	Yes	•OGC-SFA •Egenhofer •RCC-8
Oracle	GeoSPARQL*	R-Tree, Quadtree	WKT / GML support	Yes	•OGC-SFA •Egenhofer •RCC-8
Brodt et al. (RDF-3X)	SPARQL	R-Tree	WKT support	No	OGC-SFA
Perry	SPARQL-ST	R-Tree	GeoRSS GML	Yes	RCC-8
AllegroGraph	Extended SPARQL	Distribution sweeping technique	2D point geometries	Partial	•Buffer •Bounding Box •Distance
OWLIM	Extended SPARQL	Custom	2D point geometries	No	•Point-in-polygon •Buffer •Distance
Virtuoso	SPARQL	R-Tree	2D point geometries	Yes	SQL/MM (subset)
uSeekM	SPARQL	R-tree-over GiST	WKT support	No	OGC-SFA

Bibliography

[Kolas and Self, 2007]
Kolas, D., Self, T.: *Spatially Augmented Knowledgebase*. In: Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007). Lecture Notes in Computer Science, vol. 4825, pp. 785-794. Springer Verlag (2007)

[Perry, 2008]
Perry, M.: *A Framework to Support Spatial, Temporal and Thematic Analytics over Semantic Web Data*. Ph.D. thesis, Wright State University (2008)

[Koubarakis and Kyzirakos, 2010]
Koubarakis, M., Kyzirakos, K.: *Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL*. In: ESWC. pp. 425-439 (2010)

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

Conclusions

What we talked about

- Introduction
- Background in geospatial data modeling
- Geospatial data in the Semantic Web (extensions to RDF, stSPARQL and GeoSPARQL)
- Implemented systems (RDF stores)

What we did not talk about: Tools

- Tools for **translating** GIS data (e.g., shape files or tables from a geospatial DBMS) into the geospatial extensions of RDF that we presented.

What we did not talk about: Representational issues

- Is the GeoSPARQL vocabularies/ontologies always appropriate?

- Is using the WKT/GML encoding of a spatial object always a good idea?

What we did not talk about: Theory

- **Semantics:** How do we extend the semantics of SPARQL, to give semantics to stSPARQL and GeoSPARQL?

- **Computational complexity of query processing:** What is the complexity of stSPARQL or GeoSPARQL querying?

- How we can model/query geospatial information using **Description Logics**?

Thank you for Attending!

- Questions?
- Feedback?