Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Visual Languages and Computing
16 (2005) 331–358
ELSEVIER
Journal of Visual Languages & Computing
www.elsevier.com/locate/jvlc

A critical evaluation of ontology languages for geographic information retrieval on the Internet

Alia I. Abdelmoty*, Philip D. Smart, Christopher B. Jones, Gaihua Fu, David Finch

School of Computer Science, Cardiff University, Wales, UK

Received 15 April 2004; received in revised form 1 October 2004; accepted 11 November 2004

Abstract

A geo-ontology has a key role to play in the development of a spatially aware search engine, with regard to providing support for query disambiguation, query term expansion, relevance ranking and web resource annotation. This paper reviews those functions and identifies the challenges arising in the construction and maintenance of such an ontology. Two current contenders for the representation of the geo-ontology are GML, a specific markup language for geographic domains and OWL, a generic ontology representation language. Both languages are used to model the geo-ontology designed for supporting web retrieval of geographic concepts. The powers and limitations of the languages are identified. In particular, the paper highlights the lack of representation and reasoning abilities for different types of rules needed for supporting the geo-ontology.

1. Introduction

A large amount of geographical information is currently being stored and delivered over the internet. The need for the semantic geospatial web is described in [1]. This paper is concerned with intelligent web-based information retrieval of geographical information. The assumption is that people may wish to find

*Corresponding author.

1045-926X/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2004.11.001

information about something that relates to somewhere. The most common way to refer to a location is to use place names, which may be qualified by spatial relationships (such as in or near). In order to assist in recognizing place names and spatial relationships when they are employed in a search engine query it is proposed to employ an ontology which encodes geographical terminology and the semantic relationships between geographical terms. The idea is that the geographical ontology, henceforth denoted geo-ontology, will enable the search engine to detect that the query refers to a geographic location and to perform a search which will result in the retrieval and relevance ranking of web resources that refer both exactly and approximately to the specified location [2]. This will entail retrieval of resources that refer to alternative versions of a specified place name as well as to places that are spatially associated with it or through relations such as those of containment and adjacency. The geo-ontology will also assist in the production of spatial indexes for web resources as well as metadata extraction of those resources and finally in the relevance ranking process of the retrieved resources.

An ontology representation language is needed to model and maintain the geo-ontology. In this paper, we consider two such languages, namely, Geographic Mark-Up Language (GML) and Web Ontology Language (OWL). GML is being established by the Open GIS Consortium (OGC) [3] as a standard language for encoding and sharing geographic information and OWL has recently been passed as a W3C recommendation for defining and instantiating web ontologies [4]. The research on geo-ontologies presented in this paper has been carried out in the context of the SPIRIT project (Spatially Aware Information Retrieval on the Internet) [2]. An overview of the SPIRIT search engine is presented before introducing the conceptual design of the geo-ontology and the challenges and requirements identified for building and maintaining the ontology. The paper then presents case studies of representing the geo-ontology using GML and OWL. The benefits and limitations of both approaches are examined. Conclusions are then drawn on the suitability of both approaches and the needs for the future ahead.

2. Architecture of the SPIRIT search engine

The SPIRIT search engine consists of the following components: user interface; geographical and domain-specific ontologies; web document collection; core search engine; textual and spatial indices of document collection; relevance ranking and metadata extraction as shown in Fig. 1. Here, a summary is provided of the functionality associated with each of these components and the interactions between the components required to support the functionality.

The user interface allows the user to specify a subject of interest, a place name and a spatial relationship to the place name. The subject term or terms may be non-spatial, as in "database conferences in USA", or spatial, e.g. "hotels", or "cities in the UK". Different types of spatial relationships need to be supported including, topological, e.g. *inside*, *outside*, directional, e.g. *north-of* and *south-of* and proximal, e.g. *near to* or *within a specified distance of*.

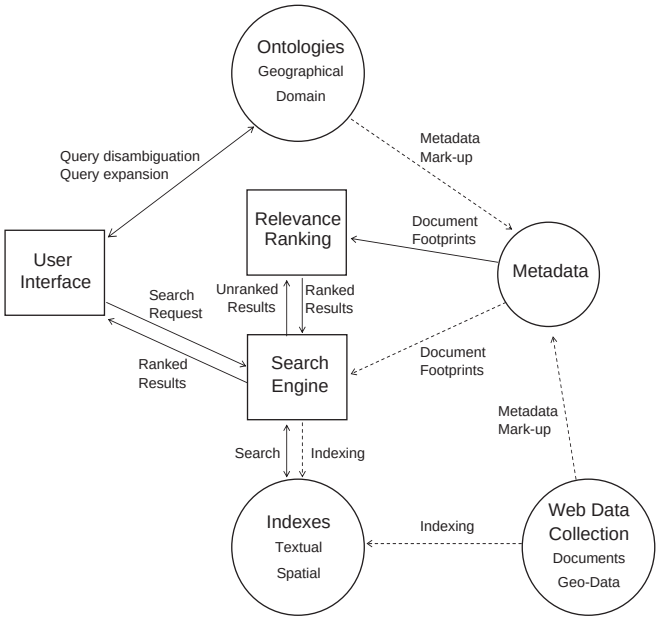


Fig. 1. SPIRIT search engine architecture.

After accepting the query expression, the system starts a process of query disambiguation and expansion. Alternative place names are offered to the user as appropriate to confirm the location of the place of interest. The combination of the place name and spatial relationship are then used to determine a geometric query footprint, i.e. the extent of the location of interest. This footprint may be echoed on a map on the user interface where the user is able to confirm the selected location(s) of interest. Once the user confirms the system's interpretation of the query, it is submitted to the core search engine and a relevance-ranked list of documents is returned to the user.

Web documents are structured to facilitate indexing. SPIRIT supports both pure text indexing and spatio-textual indexing. In order to support spatial indexing of the document collection, each referenced document that contains place names is associated with one or more "document footprints" that are derived from the geographical ontology entries for the respective names. The search engine uses the web document collection and its text and spatio-textual indexes to process the query by comparing the "query footprint" and the "documents' footprints".

The relevance ranking component takes results retrieved from the search engine and relevance ranks them with respect to the non-spatial and spatial elements of the query. Text relevance ranking is based on the BM25 algorithm [5], while spatial relevance is based, initially, on measures of distance between the query footprint and the document footprint and on angular differences from cardinal directions in the case of directionally qualified queries. Various techniques are being explored for

combining textual and spatial relevance scores to produce an integrated score. It is also intended to introduce relevance ranking measures that take account of the parent geographical regions of the query footprint and the document footprint using methods such as those documented in [2].

Initial experiments with the SPIRIT search engine employ a 1 terabyte test collection of web documents (comprising 94 million web pages). The SPIRIT prototype adapts an experimental text search engine GLASS [6] for purposes of building, maintaining and accessing the document collection and indexes. The major modification to the existing search engine functionality concerns the introduction of spatial indexing of the indexes of web documents and facilities to search for geographical context within web documents.

3. A geographical ontology for SPIRIT

The primary ontology component in SPIRIT is a geographical ontology that provides a model of the terminology and structure of the geographic space. The geo-ontology plays a key role in the interpretation of user queries; the formulation of system queries, generation of spatial indexes, relevance ranking and metadata extraction. When interacting with the user, the geo-ontology is used to recognize the presence of place names in a query and then to perform disambiguation. Once the user's query is formulated as a $\langle term, spatialrelationship, place \rangle$ expression, the ontology can be used to generate a polygonal geometric query footprint covering the spatial extent of the query region, based on the interpretation of the spatial relationship with the place. This query footprint is then used to access the spatial index of web documents. The geo-ontology could also be used to "expand" the user's query terms to include alternative names for the same place as well as the names of geographically associated places that may be inside, nearby or contain the specified place. The relevance ranking component accesses the geographical ontology to retrieve geometric footprints of places that are being compared with the query footprint, as well as with associated data providing the geographical context of a place, such as its containing and overlapping places. In the process of metadata extraction from web documents, the ontology is essential in identifying the presence of place names within text.

To support the functions above, actual and alternative place names, including multi-lingual versions need to be supported by the ontology. Geographical containment hierarchies and place types are required for query expansion and disambiguation. A geographic place is associated with possibly multiple geometric footprints. For example, detailed geometric footprints need to be used for accurate spatial indexing of documents. As indexing is a pre-processing operation, no impact on run time performance is expected. However, when generating a query footprint, access to detailed geometries can be expected to introduce processing overheads and hence there is a strong case for supporting generalised polygonal, e.g. a minimum bounding rectangle, or point-based geometries. The same reasoning applies to the use of a footprint for spatial relevance ranking of documents at query time.

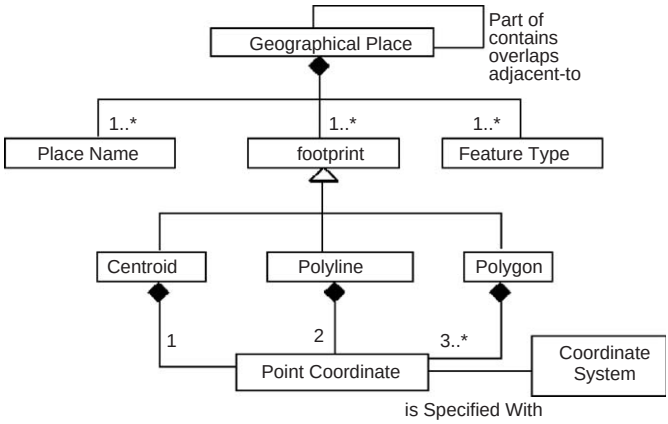


Fig. 2. Conceptual design of the geo-ontology.

Consequently, the design of the geo-ontology supports multiple spatial representations of geographic places, including centre points and minimum bounding rectangles, besides more faithful representations of geometries.

Several types of spatial relationships are stored and supported by the geo-ontology. Part-of relationships are used for maintaining containment hierarchies, e.g. based on different types of administrative hierarchies. Containment hierarchies are important for purposes of place name disambiguation. Part-of, overlap and adjacency relationships can be used to derive similarity metrics that can be exploited for purposes of relevance ranking and for matching places in the process of ontology construction and update using multiple data sets. Hence, the geo-ontology supports part-of, contains, overlap and adjacency relationships between geographic places. The main components of the SPIRIT geo-ontology is shown in Fig. 2.

4. Requirements for a geo-ontology representation language

Ideally, a geo-ontology in SPIRIT would contain references to all places on Earth. The data used to populate the ontology would be obtained from a variety of different sources, including digital maps produced by national mapping agencies, and other data collection organisations, as well as from gazetteers and thesauri. Various complexities with this types of data will need to handled, including

- differences in the semantic as well as the spatial levels of details of representation. For example, different classification hierarchies may exist in different data sets. Also, similar geographic concepts may be represented using different types of geometries in different data sets, e.g. a region may be represented by a polygon in one set and with a point in another set;

- differences in the accuracy of representation. This issue relates to the correctness of the information stored. Any change in the location and the shape of a geographic place can have an effect on the relationship between the place and other places in the data set;
- differences in the underlying geo-referencing system. A unified geo-referencing system needs to be employed to avoid any problems with geometric calculations for example, of distances and orientation that may result due to the use to different grid systems.

Maintaining the consistency and the integrity of the geo-ontology is essential for supporting the correct functionality of the search engine and for ensuring the viability and the quality of the search results produced. Examples of possible maintenance tasks needed when building the ontology base are:

- Ensuring that all mandatory relationships are satisfied, e.g. that every geographic feature belongs to at least one geographic type and has at least one associated footprint.
- A polygon footprint with more than two points, must have at least four points, with the first point being equal to the last point.
- For two features in a containment relationship, the bounding box of the child must be enclosed in the bounding box of the parent.
- For two features in an overlap relationship, the bounding boxes of both must intersect.

Maintenance tools are needed for checking the consistency of stored spatial relations. Such tools can make use of spatial reasoning techniques, e.g. composition tables [7–9], to implement rules for constraining the propagation and derivation of such relationships. Spatial reasoning techniques exploit the inherent properties of relations, such as transitivity and symmetry. Examples of rules for topological relationships include

- $contain(x, y), contain(y, z) \rightarrow contain(x, z)$,
- $inside(x, y), meet(y, z) \rightarrow disjoint(x, z)$,
- $meet(x, y), inside(y, z) \rightarrow inside(x, z)$ or $covered-by(x, z)$ or $overlap(x, z)$.
  Knowledge of size relationships can further enhance the reasoning process; for example, the last rule can be modified with the knowledge that the size of object x is larger than the size of z as follows:
- $meet(x, y), inside(y, z), larger(x, z) \rightarrow overlap(x, z)$.

From the above, the following requirements can be identified for a language used to represent the intended geo-ontology. These are categorized between basic and essential functions and desirable functions.

4.1. Basic requirements

A language for representing the geo-ontology should be capable of the following basic features:

1. Representation of geographic features and their associated types.
2. Representation of spatial and non-spatial properties of geographic features.
3. Representation of conventional (association) as well as spatial relationships between geographic features.
4. Representation of specialization and generalization feature hierarchies.

4.2. Desirable requirements

It is also desirable for the language to support the following functionality:

1. Representation of constraints/properties on the supported types of relationships.
2. Representation of semantic as well as spatial composition hierarchies for geographic features.
3. Representation of different types of rules over types and individuals in the ontology. Rules may be used to derive implicit information, e.g. new spatial relationships between concepts, to express integrity constraints, as well as to represent derived concepts using stored ones.

In the following sections, two ontology representation languages are used to represent the geo-ontology proposed in this paper, namely, GML and OWL. Several other ontology and knowledge representation languages have been developed, for example, Knowledge Interchange Format (KIF) [10] and Simple Html Ontology Extensions (Shoe) [11]. We have chosen to use GML and OWL as test cases for the following reasons.

- GML is an XML-based language which was developed by the OpenGIS Consortium (OGC) [3] as a standardized means for encoding geographical information. It provides a schema with a large vocabulary of geometric data types based on a mature spatial data model for representing vector and raster data. GML 3.0 specification was passed in January 2003 at the OGC and it is anticipated that it will be an ISO TC/211 draft specification sometime in 2004. Its acceptance is growing rapidly within the geographic community, e.g. national mapping agencies such as the Ordnance Survey of Great Britain [12] are now already producing versions of their products in GML. Also, Geographic Information Systems (GIS) such as ESRI's ArcGIS [13] have been introducing functions to import and export data in GML format. It is considered to be well-suited platform for encoding geographic information sent between geospatial web services, which are services for providing access to geographic data and for performing geo-data processing over the internet.

- OWL is a Web ontology language that has recently gained recommendation by the W3C [4]. It is designed to support the architecture of the Semantic Web and is not limited to specific applications. OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes. It builds upon the ontology language DAML + OIL [14], which itself is based on Description Logics (DL) [15–17]. OWL currently comes in three different versions that differ in the degree of expressibility and decidability. For the purpose of this paper, OWL DL [4] is used as it provides for maximum expressiveness, while maintaining decidability.

5. The geo-ontology in GML

The conceptual model in Fig. 3 was used to implement the geo-ontology in GML. Three ontologies are used, a geographical feature ontology, for representing real-world geographic objects, feature type ontology, for representing classes of geographic concepts and spatial relationship ontology for representing different types of relationships, namely, topological, proximal and directional relationships.

A small data set representing administrative units in Wales, UK was used in the implementation of an application schema prototype in GML. The data consisted of Wards and Districts represented by polygons. A containment relationship exist between Wards and Districts, where one Ward is made up of possibly many districts. Wards are normally adjacent to other Wards and similarly Districts are adjacent to other Districts. It is of interest in this application to encode containment and adjacency relationships as well as to store the actual geometrical representation of the polygons representing the data.

GML 3.0 provides several core schemas, written in XML Schema, that define the structure and content of GML instances. For example, the Feature schema provides basic types and elements for creating feature types in GML application schemas. Features can be concrete physical objects, such as roads or rivers, or they can be abstract objects, such as political boundaries. A Geometry schema provides geometric data types to describe the geometric characteristics of features, e.g. points, lines and polygons. Also, pre-defined geometry-valued properties may be used in expressing the geometry of a geographic feature, e.g. position, centerOf, centerLineOf and extentOf.

The following schema fragment shows the declaration of the geo-ontology feature and its type as well as its type definition.

```
⟨element name = "GeographicalFeature"
type = "ont:GeographicalFeatureType"
substitutionGroup = "gml:_Feature"/⟩


⟨complexType name = "GeographicalFeatureType"⟩
  ⟨complexContent⟩
    ⟨extension base = "gml:AbstractFeatureType"⟩
        ⟨sequence⟩
```

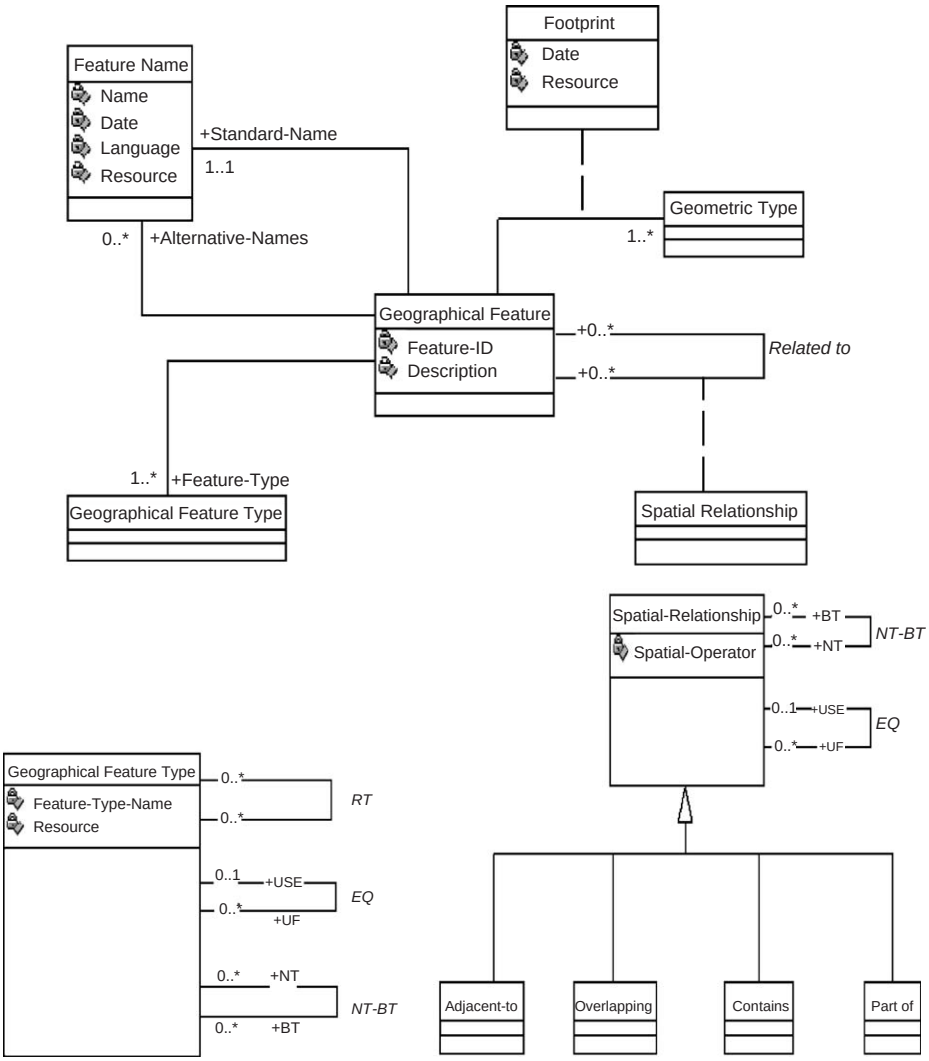


Fig. 3. The three ontologies implemented in GML, feature ontology, feature-type ontology and spatial relations ontology.

```
⟨element name = "StandardName" type = "ont:
FeatureNameType"/⟩
⟨element name = "AlternativeName" type = "ont:
FeatureNameType"
     minOccurs = "0" maxOccurs = "unbounded"/⟩
⟨element ref = "ont:Footprint" maxOccurs
 = "unbounded"/⟩
```

```
                ⟨element ref = "ont:TypeOfFeature" maxOccurs
                 = "unbounded"/⟩
                ⟨element ref = "ont:RelatedTo" minOccurs = "0"
                maxOccurs = "unbounded"/⟩
            ⟨/sequence⟩
        ⟨attributeGroup ref = "gml:AssociationAttributeGroup"/⟩
        ⟨/extension⟩
    ⟨/complexContent⟩
⟨/complexType⟩
```

The following example is a representation of a specific Ward. The exact geometry of the polygon representing the Ward is encoded and every Ward instance is assigned a unique identifier; in this case $id = gf106$. XLinks are used to relate the Ward instance to its feature type ($gft3$) and to adjacent Wards ($gf107$).

```
⟨ont:GeographicalFeature gml:id = "gf106"⟩
⟨ont:StandardName⟩
 ⟨gml:name⟩Vaynor⟨/gml:name⟩
⟨/ont:StandardName⟩
⟨ont:Footprint⟩
 ⟨ont:GeometricType⟩
  ⟨gml:location⟩
   ⟨gml:Polygon⟩
    ⟨gml:exterior⟩
     ⟨gml:LinearRing⟩
      ⟨gml:coordinates⟩300546,214083 300561,214074 300572,214070
      300585,....
      ⟨/gml:coordinates⟩
     ⟨/gml:LinearRing⟩
    ⟨/gml:exterior⟩
   ⟨/gml:Polygon⟩
  ⟨/gml:location⟩
 ⟨/ont:GeometricType⟩
⟨/ont:Footprint⟩


⟨!- - Geographic Feature Type - -⟩
⟨ont:TypeOfFeature⟩
 ⟨gml:featureMember xlink:href = " http://localhost:8080/
 OntologySoftware/Ontology/
 GeographicalFeatureTypeOntology.xml#gft3"/⟩
⟨/ont:TypeOfFeature⟩


⟨!- - Adjacency Relationships - -⟩
⟨ont:RelatedTo⟩
 ⟨gml:featureMember xlink:href = " http://localhost:8080/
```

```
OntologySoftware/Ontology/
SpatialRelationshipOntology.xmlsr3/⟩
⟨gml:featureMember xlink:href = " http://localhost:8080
/OntologySoftware/Ontology/
GeographicalFeatureOntology.xmlgf107"/⟩
⟨/ont:RelatedTo⟩
```

The following extract is from the feature-type ontology and defines Wards and Districts as well as a Zone data type which may be used for (UF) the Wards feature type.

```
⟨ont:GeographicalFeatureType gml:id = "gft3"⟩
    ⟨gml:name⟩Ward⟨/gml:name⟩
    ⟨gml:dataSource⟩Resource Information⟨/gml:dataSource⟩
⟨/ont:GeographicalFeatureType⟩

⟨ont:GeographicalFeatureType gml:id = "gft4"⟩
    ⟨gml:name⟩District⟨/gml:name⟩
    ⟨gml:dataSource⟩Resource Information⟨/gml:dataSource⟩

⟨/ont:GeographicalFeatureType⟩
    ⟨ont:GeographicalFeatureType gml:id = "gft6"⟩
    ⟨gml:name⟩Zone⟨/gml:name⟩
    ⟨gml:dataSource⟩Resource Information⟨/gml:dataSource⟩
    ⟨ont:UFxlink:href = "#gft3"/⟩
⟨/ont:GeographicalFeatureType⟩
```

Similarly, the following extract is from the spatial relationship ontology showing the definition of the "*contains*" and "*adjacentto*" relationships as well as other synonyms for adjacency, namely, "*next to*", "*bordering*" and "*adjoining*".

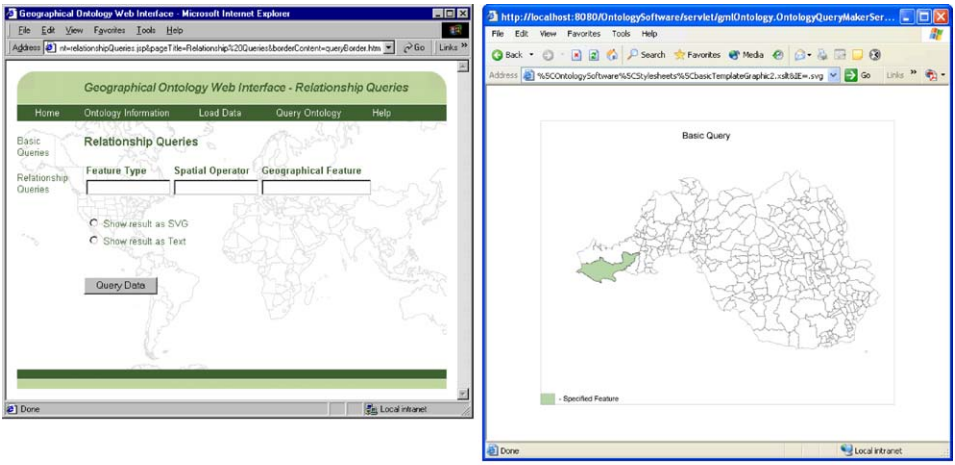
```
⟨ont:SpatialRelationship gml:id = "sr2"⟩
      ⟨ont:Spatial-Operator⟩contains⟨/ont: Spatial-Operator⟩
⟨/ont:SpatialRelationship⟩

⟨ont:SpatialRelationship gml:id = "sr3"⟩
      ⟨ont:Spatial-Operator⟩adjacent to ⟨/ont:Spatial-Operator⟩
      ⟨ont:UF⟩
        ⟨Spatial-Operator⟩next to⟨/Spatial-Operator⟩
      ⟨/ont:UF⟩
      ⟨ont:UF⟩
        ⟨Spatial-Operator⟩bordering⟨/Spatial-Operator⟩
      ⟨/ont:UF⟩
      ⟨ont:UF⟩
        ⟨Spatial-Operator⟩adjoining⟨/Spatial-Operator⟩
      ⟨/ont:UF⟩
⟨/ont:SpatialRelationship⟩
```

The geo-ontology uses XLinks to provide the necessary mechanisms for relating geographical features to other geographical features using the relationships defined in the spatial relationship ontology, and to relate geographical features with their relevant geographical feature types as defined in the geographical feature type ontology. To query the ontology, for example, to find which features are adjacent to which other features, XLinks need to be traversed and resolved in the different schemas. Fig. 4 is a snapshot of the web system developed for querying the GML schemas implemented. Map data for part of Wales in the UK has been used.

5.1. Handling basic requirements

GML provides a rich platform for the definition of geographic features in the geo-ontology. It provides core schemas, based on a mature spatial data model, for the representation of geometric properties of the features. Application schemas, in this case the geo-ontology, need to import necessary core schemas to define their intended structure. The examples above illustrates the definition of relatively simple types. However, various other core schemas are provided in GML 3.0 to define more complex geometries and topologies. For example, the adjacency relationships between Wards and between Districts could have been defined differently using the topological schema in GML. Nodes on the adjacent boundaries could have been identified and linked, as shown in the following example [18]. The example describes edges of polygons by their beginning and end nodes and encodes for each node



(a) (b)

Fig. 4. Snapshots of the system implemented to load, query and display the geo-ontology in GML. (a) Main system interface, and (b) GML data displayed in SVG format.

which edges form its coboundary.

```
⟨gml:Edge gml:id = "e6"⟩
    ⟨gml:directedNode orientation = "-" xlink:href = "#n1"/⟩
    ⟨gml:directedNode orientation = "+" xlink:href = "#n4"/⟩
⟨/gml:Edge⟩
⟨gml:Node gml:id = "n4"⟩
    ⟨gml:directedEdge orientation = "-" xlink:href = "#e4"/⟩
    ⟨gml:directedEdge orientation = "+" xlink:href = "#e6"/⟩
    ⟨gml:directedEdge orientation = "+" xlink:href = "#e3"/⟩
⟨/gml:Edge⟩
```

Features may be arranged in classification hierarchies and their types are determined by tracing the XML Schema inheritance hierarchy back to the base types in the GML core schemas, for example, *AbstractFeatureType*.

Properties of features may be described "in-line" or remotely using Xlinks. Cardinalities and existence qualities of properties may be defined, for example to indicate whether a property is optional or mandatory and to which degree.

Simple relationships between features are defined with properties where the property name designates the role of the target participant with respect to the source in the relationship. Remote properties, using Xlinks, may also be used to encode relationships. The cardinality of the relationship can then be defined on the related features. Spatial relationships can be defined explicitly in a similar fashion to an association relationship, as has been shown in the above examples. Also, some spatial relationships may be inferred if a topological model was used in the description of the data.

5.2. Handling desirable requirements

Composition hierarchies can be defined using feature collections in GML. Thus, for example, Wards could have been defined as collections of Districts as follows.

```
⟨ont:GeographicalFeature gml:id = "gf106"⟩
    ⟨gml:description⟩ Vaynor Ward ⟨/gml:description⟩
    ⟨gml:name ⟩ Vaynor⟨/gml:name⟩
    ⟨gml:boundedBy⟩ .. ⟨/gml:boundedBy⟩
    ⟨gml:featureMember⟩
      ⟨ont:District gml:id = "gf201"⟩
      ⟨/ont:District⟩
    ⟨/gml:featureMember⟩
    ⟨gml:featureMember xlink:href = "#gf202"/⟩
⟨/ont:GeographicalFeature⟩
```

In the above example the Ward *gf*106 was defined as a collection of two districts, *gf*201, defined in-line, and *gf*202, defined remotely. Hence a feature collection can be made of features of different types. GML also offers an array definition to define a collection of features of the same type. The definition in this case is similar to the one

above except for the definition of the "featureMember" element, which is replaced by the definition of a "featureMembers" element as follows.

```
⟨gml:featureMembers⟩
    ⟨ont:District gml:id = "gf201"⟩
    ⟨/ont:District⟩
    xlink:href = "#gf202"
⟨/gml:featureMembers⟩
```

Note the use of the mandatory *boundedBy* property on feature collections that contains an *Envelope* of the collection object, in this case the boundary of the enclosing Ward. Also, feature collections may have other feature collections as members and therefore multiple nesting of composition hierarchies is possible.

GML 3.0 adopts and builds upon a well-defined spatial data model proposed by the OGC (ISO 19107,19108,19109,19117,19123). Along with the definition of geographic features and associated geometric data types, the spatial data model also proposes the definition of spatial operators for the representation of different types of spatial relationships. The current version of GML only supports the representation of features and properties and thus the definition of relationships had to be done using properties as used in the examples above. As the semantics of the spatial relationships are not explicit, only simple, conventional integrity constraints relating to the cardinality and existence properties could be enforced.

Other constraints representing properties of spatial relationships can not be expressed in GML. For example, there are no direct means for stating that feature 1 can not be both inside and adjacent to feature 2. GML does not support operations on features or feature properties, thus limiting the possibilities of derivation of any implicit information, in the form of properties or relationships, in the data.

6. The geo-ontology in OWL

As mentioned earlier, OWL is based on Description Logics (DLs). A DL describes the world in terms of "properties" or "constraints" that specific "individuals" have to satisfy. OWL captures classes and associated properties. These properties can either link classes together or link classes to datatypes. Individuals belong to classes via membership. To belong to a primitive class, an individual must be of the same class type and have at least the same number of associated properties (necessary conditions for class membership). To belong to a defined class, an individual must have exactly the same number of associated properties (necessary and sufficient conditions for class membership), but it does not need to be of the same class type. Hence, defined classes are inferred classes, i.e. the set of all individuals which satisfy its conditions.

As with GML, where any feature is defined as an extension of an abstract feature type (*AbstractFeatureType*), a similar model is used here to represent the geo-ontology in OWL. In OWL every geographical feature is modelled as a subclass of a top level *GeoFeature class*, which itself is a subsumee of an abstract class *Thing*; the

highest level class in OWL. The hierarchical framework of classes in the ontology is constructed and constrained to three levels. The top level contains the GeoFeature class, the ascendant of every geographical feature, along with supplementary classes which are needed for representing properties such as geometric types, etc. The second level splits each class into more specific, but still generic classes or categories of classes, e.g. Transport-Link, Settlement, Topology. Level 3 comprises classes which normally map directly to real world geographical concepts and with which the ontology shall be populated, for example, River, City, Road, Country. Note that levels 2 and 3 can themselves be made of various abstraction levels, e.g. there may be different types of roads. However, only one sub-level is used here for simplicity (Fig. 5).

In the rest of this section, the representational ability of OWL is explored to meet the necessary and desirable requirements identified earlier in the paper.

6.1. Handling basic requirements

6.1.1. Classes and properties

Classes in a class hierarchy may be associated with properties with or without restrictions to model class relationships and structure. OWL has two disjoint property types, *Object properties* and *Datatype properties*. Object properties are used to link classes together, and Datatype properties link classes to XML Schema datatypes. Properties may be constrained to restrict their usage using cardinality constraints: MinCardinality (2..*), MaxCardiniality (0..*) and Cardinality (*). For example the ID of a GeoFeature can be restricted to cardinality 1; i.e. 1 ID per instance of GeoFeature. The following is the OWL representation of the top level

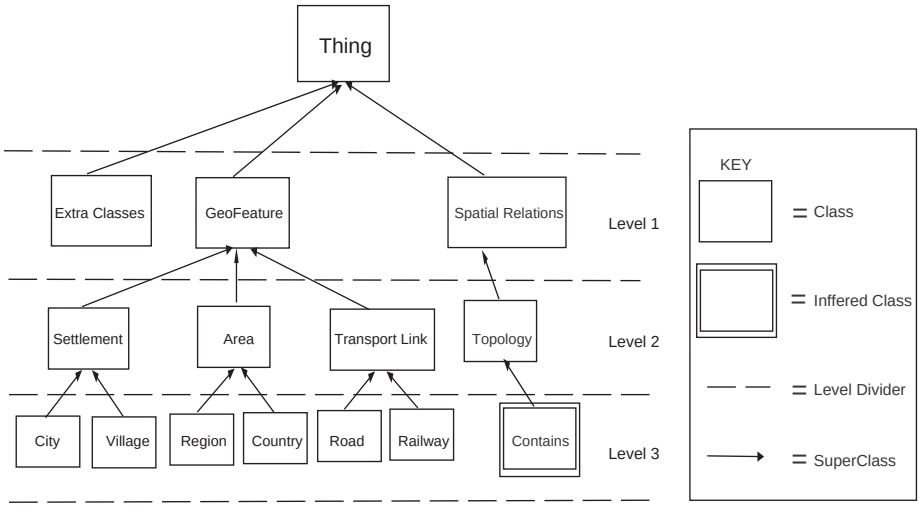


Fig. 5. Hierarchical framework of classes in the owl geo-ontology.

GeoFeature class and subclasses Settlement and City. A GeoFeature can have only one ID, one Name and at least one Footprint.

```
⟨owl:Class rdf:ID = "GeoFeature"⟩
  ⟨rdfs:SubClassof⟩
    ⟨owl:Restriction⟩ ⟨owl: onProperty rdf:resource = "#ID"/⟩
                ⟨owl :cardinality⟩1⟨/owl: cardinality⟩
                ⟨/owl:Restriction⟩
  ⟨/rdfs:SubClassOf⟩
  ⟨rdfs:SubClassof⟩
    ⟨owl:Restriction⟩ ⟨owl:onProperty rdf:resource = "\#Name"/⟩
                ⟨owl :cardinality⟩1⟨/owl: cardinality⟩
                ⟨/owl:Restriction⟩
  ⟨/rdfs:SubClassOf⟩
  ⟨rdfs:SubClassof⟩
    ⟨owl:Restriction⟩ ⟨owl:onProperty rdf:resource
    = "\#Footprint"/⟩
                ⟨owl :mincardinality⟩1⟨/owl :mincardinality⟩
                ⟨/owl :Restriction⟩
  ⟨/rdfs:SubClassOf⟩


⟨owl:Class rdf:ID = "Settlement"⟩
  ⟨rdfs:SubClassOf rdf:resource = "\#GeoFeature"/⟩
⟨/owl:Class⟩


⟨owl:Class rdf:ID = "City"⟩
  ⟨rdfs:SubClassOf rdf:resource = "\#Settlement"/⟩
  ⟨rdfs:SubClassof⟩
    ⟨owl:Restriction⟩ ⟨owl:onProperty rdf:resource
    = "\#PartOf"/⟩
          ⟨owl:allValuesFrom rdf:resource = "\#Region⟩
    ⟨/owl:Restriction⟩
  ⟨/rdfs:SubClassOf⟩
⟨/owl:Class⟩
```

6.1.2. Data properties

Attributes, of types String, Integer, etc., are attached to classes via datatype properties, as shown in table below and the following OWL code.

Level	Class	Property	Type	Cardinality
1	GeoFeature	ID	Integer	1
1	GeoFeature	Name	String	1
1	Geometric-Type	OneOf	Point; Polyline; Polygon	

```
⟨owl:DatatypeProperty rdf:ID = "ID"⟩
    ⟨rdfs:domain rdf:resource = "\#GeoFeature"/⟩
    ⟨rdfs:range rdf:resource = "rdf:resource = " http://
www.w3.org/2001/XMLSchema\#Integer"/⟩
⟨/owl:DatatypeProperty⟩
```

6.1.3. Object properties

Object properties are used to link related classes. For example, a GeoFeature is usually associated with at least one footprint. Hence, a link needs to be established between the GeoFeature class and the geometric class Footprint. The following table shows these properties as used in the geo-ontology.

Domain	Range	Property	Type	Subproperty of
GeoFeature	Geometric-Type	Footprint	Standard (Class to Class)	—
Geometric-Type	XYPoint	Co-ordinates	Standard (Class to Class)	—

The OWL representation of an object property is as follows:

```
⟨owl:ObjectProperty rdf:ID = "Footprint"⟩
    ⟨rdfs:domain rdf:resource = "\#GeoFeature"/⟩
    ⟨rdfs:range rdf:resource = "\#Geometric-Type"/⟩
⟨/owl:ObjectProperty⟩}
```

Spatial relationships are stored as object properties on GeoFeatures. OWL provides three types of object property, namely, transitive, symmetric and inverse. For example, declaring PartOf as a transitive relationship can support the traversal of containment hierarchies. Subproperty axioms allows the creation of a hierarchy of spatial relationships to support query term expansion. Some examples of spatial relationships which may be stored in the geo-ontology are shown in Fig. 6.

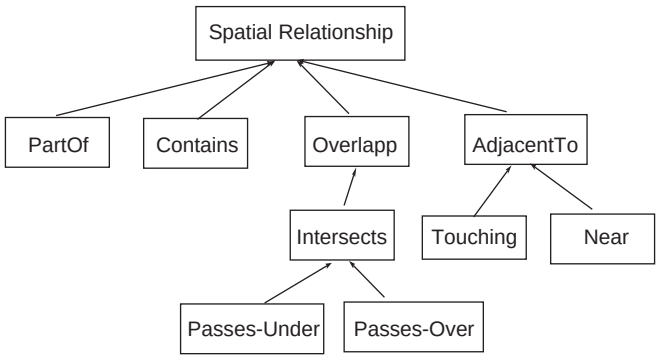


Fig. 6. Spatial relationship property hierarchy.

Domain	Range	Property	Type	Subproperty of
GeoFeature	GeoFeature	Spatial relationship (S.R.)	Standard	—
GeoFeature	N.D.	PartOf	Transitive	S.R.
GeoFeature	N.D.	Contains	Inverse (PartOf) transitive	S.R.
GeoFeature	GeoFeature	AdjacentTo	Symmetric	S.R.
GeoFeature	GeoFeature	Overlap	Standard	S.R.
GeoFeature	GeoFeature	Near	Symmetric	AdjacentTo
GeoFeature	GeoFeature	Touching	Symmetric	AdjacentTo.
GeoFeature	GeoFeature	Intersects	Standard	Overlap.
GeoFeature	GeoFeature	Passes-over	Standard	Intersects.
GeoFeature	GeoFeature	Passes-under	Standard	Intersects.

```
⟨owl:ObjectProperty rdf:ID = "Spatial-Relationship"⟩
 ⟨rdfs:domain rdf:resource = "#GeoFeature"/⟩
 ⟨rdfs:range rdf:resource = "#GeoFeature"/⟩
⟨/owl:ObjectProperty⟩}

⟨owl:ObjectProperty rdf:ID = "partOf"⟩
 ⟨rdfs:subPropertyOf rdf:resource = "#Spatial-Relationship"/⟩
 ⟨rdf:type rdf:resource = "&owl;TransitiveProperty"/⟩
 ⟨rdfs:domain rdf:resource = "#GeoFeature"/⟩
 //no range specified as has locally defined range.
⟨/owl:ObjectProperty⟩}

⟨owl:ObjectProperty rdf:ID = "AdjacentTo"⟩
 ⟨rdfs:subPropertyOf rdf:resource = "#Spatial-Relationship"/⟩
 ⟨rdf:type rdf:resource = "&owl;SymmetricProperty"/⟩
 ⟨rdfs:domain rdf:resource = "#GeoFeature"/⟩
 ⟨rdfs:range rdf:resource = "#GeoFeature"/⟩
⟨/owl:ObjectProperty⟩

⟨owl:ObjectProperty rdf:ID = "Overlaps"
 ⟨rdfs:domain rdf:resource = "http://SPIRIT.com/#GeoFeature"/⟩
 ⟨rdfs:range rdf:resource = "http://SPIRIT.com/#GeoFeature"/⟩
⟨/owl:ObjectProperty⟩

⟨owl:ObjectProperty rdf:ID = "Contains"⟩
 ⟨owl:inverseOf rdf:resource = "#partOf"/⟩
⟨/owl:ObjectProperty⟩
```

In designing the geo-ontology in OWL, the following issues were identified.

1. The PartOf relationship is best modelled with undefined range. Local range restrictions are used per class, using OWL's *allValuesFrom* constructor. For example, a City is restricted to be PartOf a Region and a Region is restricted to be PartOf a Country, etc.
2. One GeoFeature can contain a large number of other GeoFeatures. If the contains relationship was explicitly modelled, the resultant size of the ontology can explode. It is therefore a better strategy to model the inverse relationship part-of, which is of controlled complexity, and from which the contains relationship may be inferred.
3. Spatial Relationships are created as properties, and can therefore be attached to any class in the hierarchy.
4. To model alternative names of geographic features, e.g. Caerdydd is the Welsh name for Cardiff, a separate individual is created in the ontology and equated to the original individual using the *sameAs* axiom. The axiom has well defined semantics within OWL and therefore has greater reasoning potential in comparison with modelling using a conventional data property. For example the following OWL fragment assumes a class City:

```
⟨City rdf:ID = "Cardiff"⟩
...
⟨/City⟩
⟨City rdf:ID = "Caerdydd"⟩
  ⟨owl:sameAs rdf:resource = "\#Cardiff"⟩
⟨/City⟩}
```

5. Composition hierarchies can be modelled using boolean set operators in OWL. For example, class A can be modelled as being made up of individuals from classes *B*, *C* and *D* using the following OWL code.

```
⟨owl:Class rdf:ID = "C"/⟩
⟨owl:Class rdf:ID = "B"/⟩
⟨owl:Class rdf:ID = "D"⟩


⟨owl:Class rdf:ID = "A"/⟩
      ⟨rdfs:subClassOf⟩
            ⟨owl:Class⟩
                  ⟨owl:unionOf rdf:parseType = "Collection"⟩
                    ⟨owl:Class rdf:about = "#B"/⟩
                    ⟨owl:Class rdf:about = "#C"/⟩
                    ⟨owl:Class rdf:about = "#D"/⟩
                  ⟨/owl:unionOf⟩
            ⟨/owl:Class⟩
      ⟨/rdfs:subClassOf⟩
⟨/owl:Class⟩
```

6.2. Handling desirable requirements

It is possible to represent part-of and contain relationships as conventional association relationships and characterising the relationships as being transitive to support their automatic inference.

In the above examples, it was shown how classes, such as a GeoFeature was defined using a collection of properties. OWL provides set operators, namely, union and intersection, for combining collections of properties. This facility provides an opportunity to define new classes of objects using constraints on properties and relationships between individuals of already existing classes. The following is an example of defining a derived class.

6.2.1. Example: defining derived classes using stored properties

Consider the definition of a class of all potential sites for building a new supermarket that may be defined as all sites which are near motorways and urban areas and whose area is $>10\,000\,\mathrm{m}^2$. Assume the existence of a class *Site* with an associated datatype property *Area*. The *near* spatial relationship can be interpreted as synonymous with the relationships adjacent-to or overlaps, defined earlier. Also, assume the definition of an *Over* $1000\,\mathrm{m}^2$ XML datatype. The above definition can be interpreted by the following rule.

$$\begin{aligned} goodSupermarketSite \equiv\ & Site \cap (\exists AdjacentTo.Motorway \cup \exists Overlaps.Motorway) \\ & \cap (\exists AdjacentTo.Urbanarea \cup \exists Overlaps.Urbanarea) \\ & \cap \exists Area.Over1000sqm \end{aligned}$$

The rule can be captured by a derived class that becomes the class of all things satisfying the above conditions, i.e. the set of all good supermarket sites. The OWL extract of a possible implementation of this class is as follows.

```
⟨owl:Class rdf:ID = "Site"⟩
  ⟨rdfs:subClassOf rdf:resource = "GeoFeature"/⟩
    ...
⟨/owl:Class⟩

⟨owl:Class rdf:ID = "Motorway"⟩
  ⟨rdfs:subClassOf rdf:resource = "GeoFeature"/⟩
    ...
⟨/owl:Class⟩

⟨owl:Class rdf:ID = "Urbanarea"⟩
  ⟨rdfs:subClassOf rdf:resource = "GeoFeature"/⟩
    ...
⟨/owl:Class⟩

⟨owl:Class rdf:ID = "goodSuperMarketSite"⟩
    ⟨owl:sameAs⟩
```

```
⟨owl:intersectionOf rdf:parseType = "Collection"⟩
  ⟨owl:Class rdf:about = "\#Site"⟩
  ⟨owl:unionOf rdf:parseType = "Collection"⟩
    ⟨owl:Restriction⟩
       ⟨owl:onProperty rdf:resource = "\#AdjacentTo"/⟩
       ⟨owl:someValuesFrom rdf:resource = "\#Motorway"/⟩
    ⟨/owl:Restriction⟩
    ⟨owl:Restriction⟩
       ⟨owl:onProperty rdf:resource = "\#Overlap"/⟩
       ⟨owl:someValuesFrom rdf:resource = "\#Motorway"/⟩
    ⟨/owl:Restriction⟩
  ⟨/owl:unionOf⟩
  ⟨owl:unionOf rdf :parseType = "Collection"⟩
    ⟨owl:Restriction⟩
       ⟨owl:onProperty rdf:resource = "\#AdjacentTo"/⟩
       ⟨owl:someValuesFrom rdf:resource = "\#Urbanarea"/⟩
    ⟨/owl:Restriction⟩
    ⟨owl:Restriction⟩
       ⟨owl:onProperty rdf:resource = "\#Overlap"/⟩
       ⟨owl:someValuesFrom rdf:resource = "\#Urbanarea"/⟩
    ⟨/owl:Restriction⟩
  ⟨/owl:unionOf⟩
  ⟨owl:Restriction⟩
    ⟨owl:onProperty rdf:resource = "\#Areas"/⟩
    ⟨owl:someValuesFrom rdf:resource = "\#Over10000sqm"/⟩
  ⟨/owl:Restriction⟩
⟨/owl:intersectionOf⟩
⟨/owl:sameAs⟩
⟨/owl:Class⟩
```

The above is an example of relatively complex rule for defining a new class of geographic objects using the stored information in the geo-ontology. It provides a convenient way of automatically deriving new information from the ontology by combining spatial and non-spatial constraints. It is, however, not possible to include in the above definition constraints involving the computation or application of operations on properties as explained in the following example.

6.2.2. Example: using computed properties

The above example involves the definition of individuals to belong to a class by matching restrictions stored on explicitly stored properties. A variation of this example is when the properties need to be computed, as in the following rule.

$$HousesNearMotorways = House \cap Proximity.Motorway = 10.$$

The rule identifies individuals of type house which are within a specific distance of motorways. This rule is more testing for the language, as it involves the computation

of distances between all houses and motorway objects to find those that belong to the new class. If a Proximity property existed which explicitly stored the value of how far every house is from a motorway, then the above rule would boil down to determining class membership by the two restrictions: being a house, and having a Proximity property with value equal 10.

Another way of trying to represent the proximity relationship is by comparing coordinate values. Only comparison of the x co-ordinates is shown for simplicity.

$$(-10 \leqslant (House.XCoord - Motorway.XCoord) \leqslant 10).$$

The above rule states that if the distance between the x coordinates of a house and a motorway was in the range of –10 to 10, then the house qualifies as an individual for our new derived class. OWL does not support operations on datatype properties, nor does it support comparison of different individuals. Hence, this rule cannot be expressed in a straightforward manner in the language.

7. A comparative evaluation of the languages

In this section, a summary is presented of the experiments carried out in both GML and OWL for representing the geo-ontology. The issues identified are grouped under the following headings: representation ability, reasoning power, scalability and query facilities.

7.1. Representational issues

- GML is capable of representing geographic features using a rich built-in vocabulary, based on a well defined geographic and spatial data model. As OWL is a general purpose language, no specific structures are pre-defined to represent geographic features and they are modelled using user-defined classes. It is however, possible to replicate the spatial data model underlying GML in OWL.
- GML is capable of representing data properties on objects that are pre-defined in its vocabulary. OWL can represent data properties using user-defined datatype properties, which take the value of an XML schema datatype. GML's predefined vocabulary provides better meaning to datatype properties than OWL's XML schema datatypes. For example, *gml* : *Name* is well understood, whereas an OWL datatype property with the *ID name* could potentially mean anything.
- In GML properties are used as means of representing relationships between geographic features. *XLinks* are normally used to link objects by representing properties and relationships defined outside the scope of the objects considered. *XLinks* need to be traversed in object hierarchies and between schemas to trace remote object pointers. OWL uses a similar concept to define relationships but also provides better semantics to properties, in particular, properties may be transitive, symmetric or inverse.
- Restrictions, related to the cardinality of defined properties may be defined in both GML and OWL.

- In GML the semantic NT and BT relationships were used to form a hierarchy of objects by linking objects using XLinks. OWL uses the subclass-of axiom to represent the inheritance hierarchy of objects directly. The semantics in the OWL approach are well understood, while tracing XLinks via pointers in GML can be obscure.
- Explicit composition hierarchies can be represented in GML using a defined structure (feature collection). Collections may be members of the same object class or individuals from different classes. Complex composition hierarchies may also be represented, where member features may themselves be feature collections. Using set operators, namely, union and intersection, classes can be defined through collections of individuals from other classes in OWL.

7.2. Reasoning power

Using set operators on properties (union and intersection), OWL was shown to be capable of representing classes made up of boolean combinations of properties and relationships. Since, properties may themselves be references to individuals from other classes, relatively complex rules for defining derived classes using stored properties could be defined. This facility is not available in GML.

Careful construction of the geo-ontology can provide some inherent basic integrity constraints. For example: to constrain every polygon to have at minimum of 4 co-ordinates would require creating a restriction on the polygon class to say that it must have (necessary) 4 or more XY-Coord properties, therefore a $polygon \subseteq (\geqslant 4XY - Coords)$.

Support for checking basic consistency of the ontology in OWL is provided by the DL reasoning engines RACER [19] and FACT [20], both of which are capable of the following:

- Checking class consistency: i.e. checking for inconsistencies in cardinality constraints or value restrictions.
- Inferring subsumption hierarchies: i.e. discovering new super-classes from what has been explicitly stated.

In OWL, the representation of basic spatial integrity constraints is the responsibility of the ontology designers. On the other hand, GML provides built-in definitions of geo-features, their spatial representations and their structure and has, therefore, some constraints already pre-defined. For example, every geo-feature must be associated with a footprint to define its spatial location.

It is, however, not possible, in both languages, to express directly any more complex constraints such as the one shown in Fig. 7. Fig. 7 shows an object B containing another object A. For this fact to be represented completely in an ontology language, we need to also represent the constraints which govern the coordinates of both objects. For example, the following condition must hold for the objects A and B in the figure: $B_{X1} < A_{X1} \cap B_{Y1} < A_{Y1} \cap B_{X4} > A_{X4} \cap B_{Y4} > A_{Y4}$.

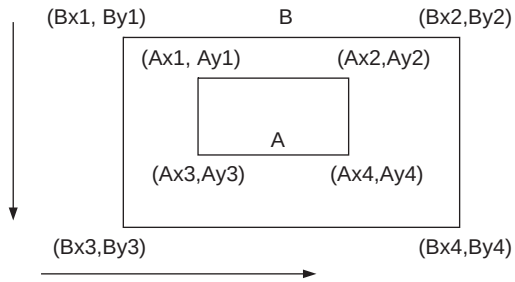


Fig. 7. Relationship between minimum bounding rectangles of objects in containment hierarchies.

The latter constraint can not be simply expressed in either language, since it is not possible to use variables as pointers to specific individuals, or to compare the datatype properties of individuals. Many similar other situations as the example above would be desirable to be represented in the language, e.g. expressing the semantics of spatial relationships (A is west of B, if the maximum x-coordinate of A is less than the minimum x-coordinate of B, etc.).

As they stand, GML and OWL are both concerned with representing the structural model of the geo-ontology. They, however, do not have means for representing operations or rules for manipulating elements in the ontology. It is possible though to envisage the development of specialized tools to serve this purpose for both languages.

Recently such tools have been emerging for OWL, e.g. RuleML [21], XRML [22] and SWRL [22]. For example consider the following spatial reasoning rule: $inside(x, y), meet(y, z) \rightarrow disjoint(x, z)$ as shown in the scene in Fig. 8. The rule can be expressed using Jena [23] and its reasoning subsystem over a geo-ontology in OWL as follows:

$$[disjoint : (?xNS + \text{" } Inside\text{" } ?y), (?yNS + \text{" } meet\text{" } ?z) \rightarrow (?xNS + \text{" } Disjoint\text{" } ?z)],$$

where NS represents the name space used to identify the geo-ontology. Hence, using this rule and given the fact that Cardiff is inside Wales and Wales is adjacent to England, a new fact, namely, Cardiff is outside England, will be asserted in the ontology.

7.3. On querying the ontologies

OWL query engines are still under development [24]. Indeed, most current query engines work only on the RDF level, where only simple triple information can be retrieved, e.g. RQL [25]. There is scope for the development of more powerful query languages for ontologies to support the power offered by the representation languages. GML is primarily a representation language, and does not have associated mechanisms that require subsumption hierarchy inference, or consistency checking. However, spatial query languages for use with XML, have been

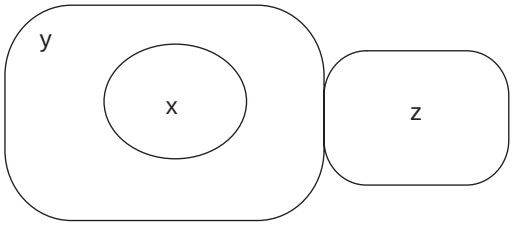


Fig. 8. Example of a spatial reasoning rule. If x is inside y, while not touching its boundary, and y is touching z, then x must be disjoint from z.

investigated by [26], e.g. XQL [27], XML-QL [28] and Lorel [29]. An SQL style query language for use specifically with GML and spatial operations is also being investigated [30].

7.4. On scalability issues

An important factor that needs to be considered when developing ontologies is the issue of scalability. A geographical application domain can potentially contain hundreds of concepts, and many thousands of individuals. For example, the data set used in the GML application created an ontology file size in excess of 14 MB. Detailed polygon geometries were used averaging over 1000 point per polygon. The OWL application created files of almost double the above. In fact, the issue of the level of spatial granularity used when modelling a geo-ontology needs further investigations. Both GML and OWL use an XML encoding, which makes little or no attempt to reduce the size of the information stored. Corcleos and Gonazalez [30] have carried out some work to investigate different approaches for storing GML documents, for efficient querying. However, similar research needs to be considered for OWL. Also, OWL's semantic power have an adverse effect on its scalability. Indeed, it is well known that reasoning with expressive DAML + OIL ontologies, the direct predecessor to OWL, is a difficult and computationally intense problem [31], therefore it follows that such a problem would arise with OWL. Some users are therefore only using a small subset of the language, to guarantee sound and complete reasoning with large deployed ontologies. The AgentCities [32] ontology is one such example, which uses a less expressive subset of the language.

8. Conclusions

The paper described the role of a geo-ontology in a spatially aware search engine SPIRIT. The ontology plays a major role in supporting the different components of the system, in particular, query disambiguation and expansion, spatial indexing and metadata construction as well as in relevance ranking of retrieved resources. Major issues in the construction of such a geo-ontology have been identified. Several requirements were drawn for a ontology representation language needed to support

the representation and manipulation of the geo-ontology. Desirable requirements included the representation and utilization of rules for expressing spatial integrity constraints and spatial reasoning.

To test the abilities of the current generation of ontology languages, prototype applications implementing the proposed geo-ontology were built using two of the most well-known examples, namely, GML and OWL. The abilities of both languages to support the identified requirements were tested and a comparative evaluation is given.

Both languages were capable of similar representation abilities with respect to basic requirements, where geographical features could be modelled with associated properties, and relationships between features could be established. OWL offers better mechanisms for the representation of relationships, as well as facilitates the definition of specific types of relationships such as transitive, symmetric and inverse. GML is based on a well developed geographic data model and therefore offers a large and rich vocabulary for the representation of different types of geographic concepts. However, the syntactical approach to the encoding of a geographical domain in GML, without strong associated semantics, limits the ability of automatic and intelligent reasoning. OWL is based on a sound logical basis offering inferencing potential which can be exploited automatically. Both languages are limited with respect to representing rules over the individuals in the Ontology. Finally, OWL is a generic language and therefore can represent any type of domain specific ontologies. This is compared with GML which can only be useful for specific geographic domains.

Our future work is concerned with various issues related to the actual design of the geo-ontology [33], development of tools for maintaining the consistency of the developed geo-ontology as well as ontology integration and evolvement. These tasks will also involve studying and developing approaches to reasoning over the geo-ontology.

Acknowledgements

This research is supported by the European Commission Framework V project SPIRIT : Spatially-Aware Information Retrieval on the Internet, funded under the Semantic Web Technologies action line. Thanks to Daniel Rome who implemented the geo-ontology in GML as part of his Masters project.

References

[1] M.J. Egenhofer, Toward the semantic geospatial web, in: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems, ACM Press, New York, 2002, pp. 1–7.
[2] C.B. Jones, A.I. Abdelmoty, G. Fu, Maintaining ontologies for geographical information retrieval on the web, in: Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems (ODBASE03), LNCS 2888, 2003, pp. 934–951.
[3] Open Geospatial Consortium, http://www.opengeospatial.org/.

[4] F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein, Owl web ontology language reference, http://www.w3.org/TR/owl-ref/.
[5] S.E. Robertson, S. Walker, M.M. Beaulieu, M. Gatford, A. Payne, Okapi at trec-4, in: Proceedings of the Fourth Text Retrieval Conference (TREC-4), 1995, pp. 73–96.
[6] GLASS: online documentation, http://dis.shef.ac.uk/mark/glass/.
[7] M.J. Egenhofer, A formal definition of binary topological relationships, in: International Conference on Foundations of Data Organization and Algorithms, 1989, pp. 457–472.
[8] B.A. El-Geresy, A.I. Abdelmoty, Towards a general theory for qualitative space, in: Proceedings of the 13th International Conference on Tools with Artificial Intelligence, 2001, pp. 111–120.
[9] B.A. El-Geresy, A.I. Abdelmoty, Sparqs: a qualitative spatial reasoning engine, Journal of Knowledge-Based Systems 17 (2–4) (2004) 89–102.
[10] M. Geneserith, Knowledge interchange format, in: J. Allen (Ed.), Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91), Morgan Kaufman, Los Altos, CA, 1991, pp. 238–249.
[11] J. Heflin, J. Hendler, S. Luke, SHOE: a knowledge representation language for internet applications, Technical Report CS-TR-4078, 1999.
[12] Ordnance survey of Great Britain, http://www.ordnancesurvey.co.uk/oswebsite/.
[13] ESRI-ArcGIS, http://www.esri.com/software/arcgis/.
[14] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, P. Patel-Schneider, Oil: an ontology infrastructure for the semantic web, 2001
[15] D. Nardi, R.J. Brachman, An introduction to description logics, http://www.dis.uniromal.it/nardi/Didattica/IA-VO/dispense/dlhb-01-2pp.pdf.
[16] F. Baader, I. Horrocks, U. Sattler, Description logics as ontology languages for the semantic web, citeseer.nj.nee.com/580806.html.
[17] I. Horrocks, U. Sattler, S. Tobies, Reasoning with individuals for the description logic shiq, in: Proceedings of the 17th International Conference on Automated Deduction, Springer, New York, 2000, pp. 482–496.
[18] R. Lake, D.S. Burggraf, M. Trninic, L. Rae, GML: Geography Mark-up Language, Wiley, New York, 2004.
[19] V. Haarslev, R. Muller, Racer system description, in: Proceedings of the First International Joint Conference on Automated Reasoning, Springer, New York, 2001, pp. 701–706.
[20] I.R. Horrocks, Using an expressive description logic: fact or fiction?, in: A.G. Cohn, L. Schubert, S.C. Shapiro (Eds.), KR'98: Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, San Francisco, CA, 1998, pp. 636–645.
[21] H. Boley, S. Tabet, G. Wagner, Design rationale of ruleml: a markup language for semantic web rules, in: International Semantic Web Working Symposium (SWWS), 2001.
[22] J.K. Lee, M.M. Sohn, The extensible rule markup language, Communications of the Association for Computing Machinery 46 (5) (2003) 59–64.
[23] HP Labs, Jena 2—a semantic web framework, http://www.hpl.hp.com/semweb/jena2.htm, 2004
[24] D.P.A. Magkanaraki, V.C.G. Karvounarakis, T.T. Anh, Ontology storage and querying, Technical Report No. 308, ICS-FORTH, Heraklion, Crete, Greece, April 2002.
[25] G. Karvounarkais, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, RQL: a declarative query language for rdf, in: Proceedings of the Eleventh International Conference on World Wide Web, ACM Press, New York, 2002, pp. 592–603.
[26] J.E. Corcleos, P. Gonazales, Analysis of different approaches for storing gml documents, in: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems, ACM Press, New York, 2002, pp. 11–16.
[27] D.S.J. Robie, J. Lapp, Xml query language (xql), http://www.w3.org/TandS/QL/QL98/pp/xql.html, 1998.
[28] A. Deutsch, M.F. Fernandez, D. Florescu, A.Y. Levy, D. Suciu, Xml-ql: a query language for xml, in: WWW The Query Language Workshop (QL), Cambridge, MA, 1998.
[29] S. Abiteboul, D. Quass, J. McHugh, J. Widom, J.L. Wiener, The Lorel query language for semistructured data, International Journal on Digital Libraries 1 (1) (1997) 68–88.

[30] J.E. Corcleos, P. Gonazalez, A specification of a spatial query language over gml, in: Proceedings of the 9th ACM International Symposium on Advances in Geographic Information Systems, ACM Press, New York, 2001, pp. 112–117.
[31] P. Kogut, J. Heflin, Semantic web technologies for aerospace, cite-seer.nj.nec.com/kogut03semantic.html.
[32] Agent Cities, Daml + oil urban ontology, http://iscte.pt/lhrm/daml/location.
[33] N. Guarino, C. Welty, Evaluating ontological decisions with ontoclean, Communications of the Association for Computing Machinery 45 (2) (2002) 61–65.