OWL based formalisation of geographic databases specifications

Nathalie Abadie, Ammar Mechouche, Sébastien Mustière

OWL based formalisation of geographic databases specifications

Nathalie Abadie, Ammar Mechouche, Sébastien Mustière

Institut Géographique National, Laboratoire COGIT,
73 Avenue de Paris,
94160 Saint-Mandé, France

{nathalie-f.abadie, ammar.mechouche, sebastien.mustiere}@ign.fr

Abstract. The ability to share and combine geographic data from different information sources in a consistent way is a key issue for enabling successful implementation of Spatial Data Infrastructures. This can only be done through a deep understanding of databases structure and content. In this paper, we propose to do that through the elicitation and formalisation of geographic database specifications, relying on OWL ontologies, as recommended in the semantic Web community. We thus propose a general ontology for eliciting key concepts manipulated by data specifications, and rules to build local ontologies representing knowledge contained in specific data specifications.

Keywords: Database Specification, Ontologies, Data semantics, OWL

1 Why formalising specifications?

Since the early days of geographic information systems, the increase of geographic data acquisition campaigns has resulted in a huge amount of diverse, heterogeneous and distributed geographic data sources having a great potential in different application fields such as environment, agriculture, urban planning or risks management. However, even if these data represent the same geographic real world, they frequently vary due to technical, historical and political reasons [1]. Consequently, the ability to share and combine geographic data from different information sources in a consistent way is a key issue for enabling their efficient usability.

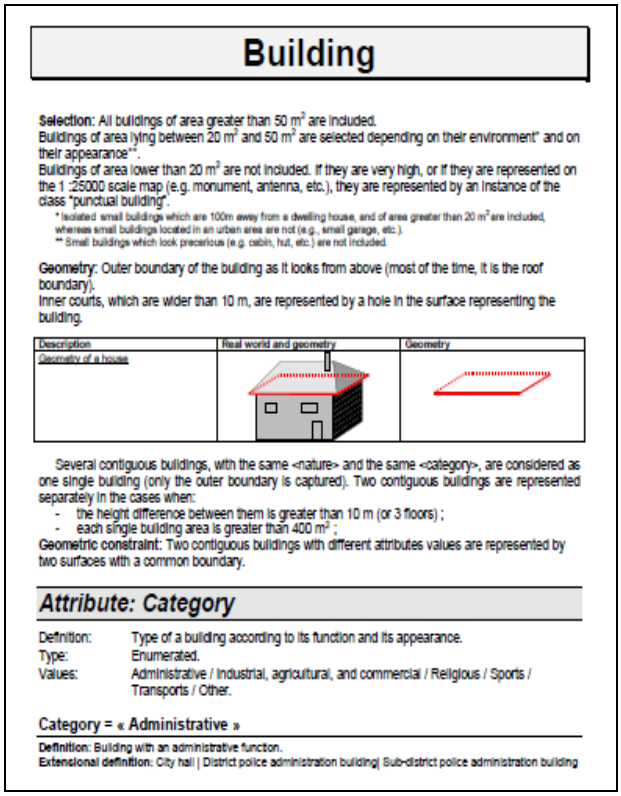
Previous efforts in the field of geo-data integration mainly focused on syntactic heterogeneities solving through the use of standards. Semantic interoperability of geo-data, which address more complex problems, is still investigated. Actually, recent works mainly focused on geographic data discovery and retrieval in the context of Spatial Data Infrastructures. However, they do not consider combining and merging heterogeneous geographic databases. Such a task obviously necessitates a good understanding of databases structure and content, i.e. their specifications. This

understanding is indeed necessary to identify and solve different kinds of heterogeneities or mismatches [2].
In this paper, we propose to support this deep understanding of database structure and content through the elicitation and formalisation of geographic databases specifications. Such formal specifications could be then exploited in an automatic schema matching process. They could also be used for data quality evaluation, among which checking if databases respect their specifications is a key issue.
Several formal models for geographic database specifications have already been proposed [3][4]. As formalisation of data specifications in SDIs is a kind of elicitation of data semantics in a Web environment, we propose to rely on semantic Web standards to do so: the widely shared approach for eliciting semantics is to develop ontologies expressed in the Ontology Web Language (OWL [5]), the *de facto* standard language for ontologies in the semantic Web.
In the next section we describe data specifications from traditional geodata producers like National Mapping Agencies. Then we present some related work in the field of geographic data integration. Our model for formalising specifications is described in the following sections: section 4 gives the general principles of the model; section 5 details the content of a general ontology containing main representation constructs required to formally depict data specifications; and section 6 presents rules for formalising a data specification. Conclusions are then given in section 7.

2 Geographic data integration issues

Like any database, geographic databases are described by their schema. Classes are named by common geographic words, which usually refer to geographic concepts used by human to categorize the represented geographic entities. Their instances, geographic features, are information objects that represent geographic entities, i.e. entities that are located in the geographic space. They are described by attributes and a geometrical representation (usually point, line or polygon).
Each geo-data producer has its own rules for data capture, and its own point of view about the geographic real world [6]. As an example, if a feature class is named 'Building', it may actually designate only permanent buildings, or include precarious buildings, such as cabins, or huts. Moreover, it may even designate only habitations. Besides, a geographic database is produced at a specific scale of analysis and is therefore associated with a specific level of detail. Geographic features are then captured in the database consistently with its level of detail. For example, only buildings of area greater than 50 m^2 may be captured. Besides, in vector databases, the geometric representation of a given geographic feature may vary. As an example, a building may be represented by a polygon representing the surface covered by this building or by a point captured at the center of the building. Since data capture for a given database is often done by several persons, homogeneity of data meaning within the database is ensured by storing all selection and representation criteria in specific

textual documents, used as guideline for data capture, namely the database specifications (Fig. 1)[1].



Building

Selection: All buildings of area greater than 50 m² are included.
Buildings of area lying between 20 m² and 50 m² are selected depending on their environment* and on their appearance**.
Buildings of area lower than 20 m² are not included. If they are very high, or if they are represented on the 1 :25000 scale map (e.g. monument, antenna, etc.), they are represented by an instance of the class "punctual building".

* Isolated small buildings which are 100m away from a dwelling house, and of area greater than 20 m² are included, whereas small buildings located in an urban area are not (e.g., small garage, etc.).
** Small buildings which look precarious (e.g. cabin, hut, etc.) are not included.

Geometry: Outer boundary of the building as it looks from above (most of the time, it is the roof boundary).
Inner courts, which are wider than 10 m, are represented by a hole in the surface representing the building.

Description	Real world and geometry	Geometry
Geometry of a house		

Several contiguous buildings, with the same <nature> and the same <category>, are considered as one single building (only the outer boundary is captured). Two contiguous buildings are represented separately in the cases when:
- the height difference between them is greater than 10 m (or 3 floors) ;
- each single building area is greater than 400 m² ;

Geometric constraint: Two contiguous buildings with different attributes values are represented by two surfaces with a common boundary.

Attribute: Category

Definition: Type of a building according to its function and its appearance.
Type: Enumerated.
Values: Administrative / Industrial, agricultural, and commercial / Religious / Sports / Transports / Other.

Category = « Administrative »

Definition: Building with an administrative function.
Extensional definition: City hall | District police administration building| Sub-district police administration building

Fig. 1. Excerpt of the specification of the BDTOPO database [7]

Geographic databases capture process necessarily leads to numerous kinds of heterogeneities or mismatches between different datasets [2]. Some of them, like syntactic heterogeneity have been addressed and can be overcome through the use of standards such as those developed by the Open Geospatial Consortium. However, schematic and semantic heterogeneities or scale conflicts, cannot be identified and solved without a good understanding of databases structures and contents (i.e. their specification), and still pose challenges.

1 All examples in this paper originate from actual specifications (from IGN France and IGN Belgium © institut géographique national), originally in French and translated here for the sake of clarity.

3 Related work

The use of ontologies as tools for formalizing consensual knowledge within a community has been acknowledged as a valid approach for semantic integration of heterogeneous data sources. Different ontology-based approaches that address the specific problem of geo-data semantic integration exist.

[8] proposes a similarity measure between feature classes described by ontologies, based on concepts labels, attributes, relations and semantic neighborhood. The G-Match algorithm and geo-ontology matcher developed by [9] computes similarity between concepts of two different ontologies by analyzing their names, their attributes, their taxonomies and their relations including their topological relations.

[10] focuses on integrating diverse spatial data repositories. Geo-data discovery and retrieval is performed through a service oriented architecture that uses a global ontology as information broker. To ensure semantic interoperability, application ontologies of data provider must be written using the shared vocabulary of the global ontology. In the GEON project, geo-data sources retrieval and integration is performed thanks to a semantic registration procedure [11]. Data providers are asked to describe mappings between their source schemas and one or more domain ontologies that are used to query all datasets in a uniform fashion. More generally, the issue of semantic annotation of geo-data as the elicitation of relations between a data schema and a domain ontology by defining mappings between them is central in [12]. A rule-based approach combining semantic Web technologies and spatial analysis methods is introduced for automating this critical task. Rules are used to define conditions for identifying geospatial concepts. Spatial analysis procedures derived from these rules are used to determine whether a feature of a dataset represents an instance of a geospatial concept. These works rather aim at enabling geo-data discovery and retrieval. For example, if a user is looking for data that represent '*buildings'*, they aim at determining in which feature classes of the available datasets '*buildings'* are represented, even if these classes have different names.

As part of geo-data integration issues, schema translation from a source to a target schema has also been addressed. Recent approaches [13] [14] [15] provide users with graphical interface to help them in describing their schemas and defining mappings between source and target schemas. Schema matching task is then performed manually by geo-databases experts. Actually, schema translation does not only require to identify source and target schema classes that refer to the same geographic concepts, but it often implies complex (and often geometric) transformations of the source dataset to fit target dataset specification requirements [16]. Heterogeneities between feature classes representing the same geographic concepts must be identified and solved. Let us consider two different databases covering the same geographical space. The first one has a feature class named 'Building' which represents only "buildings of area greater than 20 m^2", while the second one has a feature class named 'Built area' which represents "buildings of area greater than 50 m^2". The mapping rule between these classes should state that 'Building' instances of area greater than 50 m^2 represent the same real world buildings as 'Built area' instances. This shows

that mappings between source and target schemas cannot be defined without a good understanding of source and target datasets specifications. However, as they are mainly written for human readers, these specifications are only available in natural language. Consequently, they are not directly tractable. Annotating databases schemas with formal specifications would enable to take advantage of the knowledge they contain for automatic schema matching purpose.

4 Formalising specifications: two levels of ontologies

In this section, we describe general principles guiding our proposal for a formal model of data specifications. Following principles of the semantic Web, "*the semantics of a given domain is usually encapsulated, elucidated and specified by an ontology*" [17], we thus propose to formalise each database specification by means of an application ontology, named "local specification ontology" (LSO). This ontology contains information such as selection criteria used to populate the database. For example, it formalises the fact that "only watercourses that are permanent and wider than 10 meters are represented in the feature class 'River' of the database", but also that "the geometry of features 'River' corresponds to the centreline of the modelled watercourses".

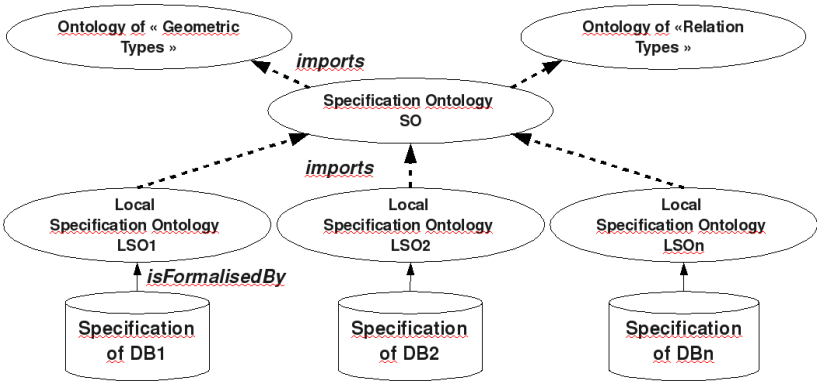


Fig. 2. Global framework for formalising database specifications

A key issue for successful use of formal specifications in an integration process is then to ensure enough homogeneity in the way to formalise them. This will be ensured by two means. The first one is to define unambiguously key concepts manipulated by the different local specification ontologies. In other words, we define a domain ontology, named "Specification Ontology" (SO), on which each LSO relies (taking the OWL vocabulary, we say that each LSO imports SO, cf. Fig. 2). This ontology SO only contains concepts specific to geographic data specifications. It relies in turn on more general ontologies, for example for defining basic geometric types [18]. For example, this domain ontology SO formalises the concepts of data source and centreline, which are commonly used in many data specifications.

The second mean to ensure homogeneity between local specification ontologies is to define common rules to fill them. For example, we require that feature classes such as 'River' are modelled as "classes" in the OWL language, and that selection constraints are modelled as "axioms" including rules that restrict the possible interpretations for the defined term, those axioms being defined by means of concepts and relations defined in SO and LSO (see section 6 for details).

5 The Specification Ontology

This section details the Specification Ontology (SO), which is considered here as a common semantic model represented in OWL and providing a unified view of the formal geographic database specifications concepts to be shared. The elements (concepts and properties) of the SO ontology are referred to when building Local Specification Ontologies (LSO), as explained more in details in the next section. The structure of the SO ontology is depicted in Fig. 3 and described below.

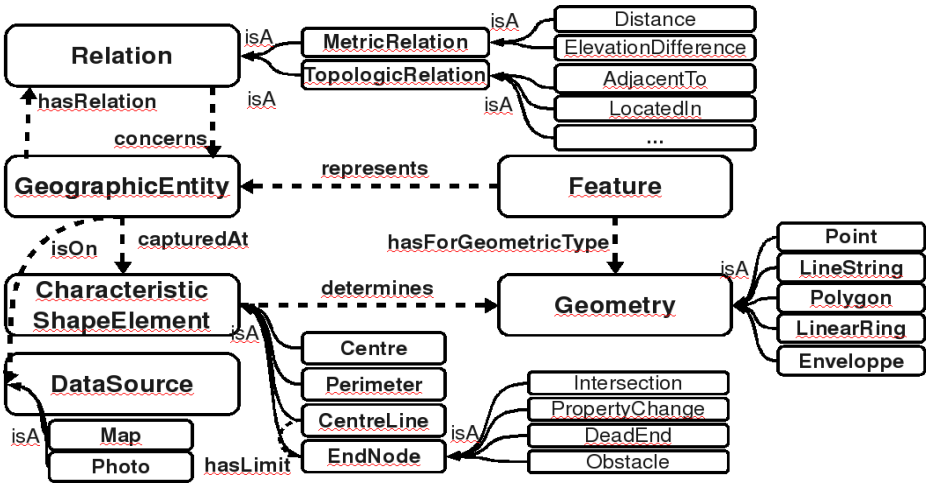


Fig. 3. General structure of the Specification Ontology (SO).

The SO ontology consists of a set of classes that are related either by a subsumption relationship 'is-a' or by other relations we introduced. As reusing existing domain ontologies is recommended in order to enable interoperability between computing applications [19] some of its classes and relations are imported from existing ontologies adopted in the geographic domain. The main classes and relations of the SO ontology are summarized in the following subsections.

5.1 The 'Feature' and '*GeographicEntity'* Classes

The key idea guiding our model is that "semantics provides meaning by associating the *representing* to what is *represented* in the real world" [17]. A clear distinction between concepts manipulated in the data schema and concepts describing the real world must be made, even if textual specifications may use the same words to designate them. Let us take a simple example to illustrate that. If a specification states that the definition of the feature class 'Buildings' is defined by "Buildings bigger than $50m^2$", it is clear that the first 'Building' refers to the name of the feature class representing only some particular buildings in a particular database, while the term '*Building'* in the definition refers to a general and shared concept of the real word. Those two buildings have very different meaning that must be separated for an efficient formalisation of semantics. The SO ontology thus includes two base classes which are: 'Feature' and '*GeographicEntity'* (Fig. 3). The class 'Feature' is imported from GeoRSS-Simple[2], a simple serialization of the GeoRSS feature model [18] following the general principles of the ISO General Feature Model, and recommended by the W3C Geospatial Ontologies incubator group for the description of geospatial resources on the Web. This ontology is used, in a first approach, as a simple way to model the schema classes of a particular geographic database. The class '*GeographicEntity'* models the real world entities and their classical properties like their height, width, length, surface, perimeter, etc. Real world geographic entities properties are represented in OWL by DatatypeProperties with '*GeographicEntity'* as a domain and OWL data type xsd:double as a range. The relation between the class 'Feature' and the class '*GeographicEntity'* is materialized by the relationship *represents*, which is an OWL ObjectProperty with 'Feature' as a domain and '*GeographicEntity'* as a range. In OWL, this relation is expressed with the following axiom associated with the 'Feature' class[3] :

```
Class: gml:Feature
  SubClassOf: so:represents some so:GeographicEntity
```

5.2 The *Relation* class

Other relations, which are OWL ObjectProperties, are used to connect the class '*GeographicEntity'* to the other classes in the ontology SO. OWL ObjectProperties are binary predicates; they relate two classes or two instances. However, we need to use also ternary relations, such as *distance,* which relates three elements: two geographic entities and a restriction on a DatatypeProperty. In order to allow ternary relations, and in general n-ary relations, to be represented in OWL, we used the reification in its traditional form[4]. The reification consists in introducing a new class (for example *Distance* on Fig. 3) with two properties: a DatatypeProperty called *value* specifying the value of the distance, and an ObjectProperty *concerns* referring to a

2 http://mapbureau.com/neogeo/neogeo.owl
3 Axioms are represented with the OWL language Manchester Syntax.
4 http://www.w3.org/TR/swbp-n-aryRelations/

geographic entity. Then, we obtain a binary relation between a geographic entity and the reified class *Distance*.
Topological relations, like the relation *locatedIn*, can be represented as OWL ObjectProperties, since they are binary relations. This is the case in the spatial relations ontology implemented by the Ordnance Survey[5]. In our SO ontology, we propose to use both solutions: on the one hand, topological relations can be modeled as binary relations through object properties, and on the other hand, the same relations can be modeled as n-ary relations. The main interest of this double formalisation is that reification represents relations as classes associated with a rich semantics. Then it makes it possible to explain exactly what we mean by each relation, especially in the context of spatial relations which are very complex [20].

5.3 The *Geometry* and *GeometricModeling* Classes

Every instance of a geographic vector database feature class has a geometrical representation, which describes the location and the shape of its corresponding real world entity, consistently with the database level of detail. Therefore, a specification also details, for each feature class, how instances geometry shall be captured.
The *Geometry* class is imported from the GeoRSS-Simple ontology. It models the different types of geometries used for geographic data, like *Polygon*. The *Geometry* class is also related to the 'Feature' class with the *hasForGeometry* OWL ObjectProperty:

```
Class: gml:Feature
  SubClassOf: so:hasForGeometry some gml:GeographicEntity
```

The *CharacteristicShapeElement* class models the characteristic shape elements of geographic entities, which shall be captured to instantiate database feature class instances geometry, like *Centre*, *Centreline* or *Perimeter*. For example, an instance of a feature class 'Building' can be captured by drawing the perimeter of the corresponding real world '*building'* in order to be stored in the database as a polygon. The *CharacteristicShapeElement* class is related to the class *Geometry* with the OWL ObjectProperty *determines*; each sub-class of *CharacteristicShapeElement* has an axiom specifying the type of geometry it determines. For example, the class *Perimeter* is defined as follows:

```
Class: so:Perimeter
  SubClassOf: so:determines some gml:Polygon
```

5.4 The *DataSource* Class

The *DataSource* class models the different digital or paper supports on which a geographic entity can be visible or present. Now, it mainly includes two sub-classes, (but new classes could be added): *Photograph* and *Map*, both associated with an

5 http://www.ordnancesurvey.co.uk/ontology/SpatialRelations.owl

OWL DatatypeProperty specifying their *scale*. These classes serve to express specifications of the form "… present on the map", or "… visible on the photo".

```
Class: so:DataSource
  SubClassOf: so:scale some string
```

6 HowTo? Steps to formalise a database specification

For each database to be integrated, a local specification ontology (LSO) is created. It describes the database schema and the rules used to populate its feature classes by capturing geographic entities. This local specification ontology imports the specifications ontology (SO), described in section 4, and uses it to formalise the specification of that specific database.

6.1 How to represent database schema entities?

A first step to formalise a specification consists in translating the database schema into OWL formalism. This is done according to a fairly intuitive strategy already presented in [21]. Each schema class represents an object-oriented abstraction of real world entities. Therefore, in our local specifications ontology, each feature class will be translated into an OWL class, whose label corresponds to the prefix "db" associated with the feature class name. As they represent schema feature classes, these OWL classes will be created as sub-classes of the SO class 'Feature'. OWL class datatype properties, object properties and *subClassOf* relations are straightly derived from their respective feature class attributes, associations, and inheritance relation.

Moreover, it is a common modeling practice for geographic databases to simplify the schema structure by merging semantically close feature classes into a single class. In such cases, the specific nature of each instance of the feature class is defined more accurately by an attribute (usually named 'nature' or 'type'). Most of the time, this attribute's values are terms that designate geographic concepts. As an example, a feature class 'Water Point' has an attribute 'nature' with possible values 'cistern', 'fountain', 'spring' or 'well'. Besides, it happens frequently that instances of such feature classes, having different natures, have different specifications, e.g. different selection criteria. We propose to translate the values of these specific attributes into OWL classes, subsumed by the OWL class derived from their respective feature class in the database schema, in order to make their specification formalisation easier.

We have implemented a generic translator, developed with the protégé-owl API[6]. It takes an ISO 19109 [22] schema as input and converts it into OWL ontology elements [5], according to the strategy presented above. Figure 4 shows how a piece of BDCARTO® [23] schema is translated into OWL format.

6 http://protege.stanford.edu/plugins/owl/api/

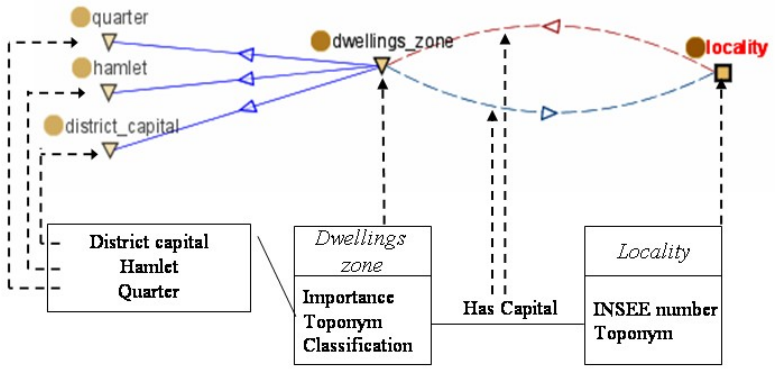


Fig. 4. Translating BDCARTO® schema (at the bottom of the image) into OWL ontology (piece of ontology visualized with Protégé, at the top of the image)

6.2 How to represent real world geographic entities?

As specifications detail how real world geographic entities are captured in a given database, we need to represent these geographic entities in our local specification ontology, which consists in making explicit what Partridge [24] calls the domain ontology which underlies the database.

The geographic entities of this underlying domain ontology can be derived from the specification text. Actually, a specification describes the database structure and content by using geographic vocabulary. An intuitive method to build this domain ontology consists in retrieving in the specification text the specific terms used to designate real world geographic entities and to use them as labels for our OWL classes. These OWL classes are represented in our local specification ontology as sub-classes of '*GeographicEntity*'.

Building such an domain ontology can be done in a more or less structured manner. In order to go a step further, relations between concepts, like subsumption or meronymy relations, can be created by analysing the linguistic relationships between geographic terms provided by the specification text. This may be done semi-automatically thanks to natural language processing tools [25]. Finally, this domain ontology can be improved by a manual analysis. This is a longer and harder task but provides a semantically richer ontology and therefore better integration results.

6.3 How to link Features to Geographic Entities?

As explained in section 4, the relationship between database features and geographic entities is formalised thanks to an ObjectProperty named *represents*. It enables us to instantiate several kinds of specification rules. As an example, selection constraints,

which specify, for each feature class, the nature of real world entities that must be represented are formalised by *someValuesFrom* restrictions:

```
Class: lso:db_Spring
   EquivalentTo: so:represents some (lso:Spring or
                 lso:Resurgence or lso:Outcropping)
```

Moreover a geographic database is associated with a given level of detail. Therefore real world entities must be captured consistently with this level of detail. Thus specifications express geometric constraints on the size of geographic entities that shall be captured in a specific feature class. As an example, a specification precises that '*basins'* are captured in the database class 'Water Body' if their length is greater than 10 m. In OWL the range of a DatatypeProperty is a simple built-in data type like a *double*, *integer* or *string*. However, in this case, we need to make a different cardinality restriction. This becomes possible with the new version of OWL, namely OWL 2 [26] which is based on a more expressive Description Logic (*SROIQ*) and which provides some new interesting features [27], one of which is the data type restriction construct, which allows new data types to be defined by restricting the built-in data types in various ways [28].The constraints on '*basins''s* size presented above can then be formalised as follows:

```
Class: lso:db_WaterBody
  EquivalentTo: so:represents some (lso:Basin and
                so:length some double[>10.0])
```

Contextual constraints state that real world entities are captured in the database if they are really significant in the landscape depending on their environment, or if they are mentioned on a reference data source. In the former case, the constraints used will deal with real world geographic entities relationships. As an example, a specification of the feature class 'GuardedShelter' states that: "…'*mountain hotels'* which are *located in* a '*National Park'* or in its vicinity (less than 2 km away from the park) are also included". These kinds of constraints are formalised with restrictions on metric relations and topologic relations defined in the specifications ontology:

```
Class: lso:db_GuardedShelter
  EquivalentTo: so:represents some ((lso:Mountain_Hotel
           and so:locatedIn some lso:National_Park) or
           (lso:Mountain_Hotel and so:hasRelation some
           (so:Distance and so:concerns some
           lso:National_Park and so:value some double
           [<2000.0])))
```

In the latter case, when geographic entities are required to be mentioned on a specific data source, this constraint can be formalised thanks to a *someValueFrom* restriction: instances of a given geographic entity are related to instances of *DataSource* via the *isOn* ObjectProperty. For example, the 'Water Point' feature class specification, which precises that "*'Springs'* are represented if they are mentioned on the 1:25000 map", will be translated into:

```
Class: lso:db_WaterPoint
  EquivalentTo: so:represents some (lso:Spring and
      so:isOn some (so:Map and so:scale value "1/25000"))
```

Besides, specifications define constraints on specific real world entity properties, such as "only *outdoor* '*swimming-pools'* are captured". Geographic entities properties are defined in our model by DatatypeProperties and constraints on their values are formalised thanks to restrictions on these properties:

```
Class: lso:db_SwimmingPool
  EquivalentTo: so:represents some (lso:Swimming_pool and
                lso:isOutdoor value true)
```

6.4 How to formalise geometry instantiation rules?

Geometry instantiation rules, such as "'Watercourse segments' geometry is represented by a line drawn along '*rivers'* centreline", define what geometric type shall be used for this feature class, and what characteristic shape elements of real world entities shall be depicted. Both aspects are taken into account in our SO ontology, so that the geometry instantiation rule presented above can be formalised with two someValueFrom restrictions:

```
Class: lso:db_WatercourseSegment
  EquivalentTo: so:hasForGeometry some gml:LineString and
                so:represents some (lso:River and
                so:capturedAt some so:CentreLine)
```

6.5 How to formalise attribute instantiation rules?

A feature class specification also defines the meaning of class attributes and explains how their values shall be filled. However, by attribute instantiation rules, we do not mean cardinality or data type constraints, but rather rules which define precisely how attribute values shall be determined for each feature class instance, like: "The attribute 'width' of the feature class 'Hydrographic segment' takes the value 'small' if this '*river section's width'* lies between 0 and 10 meters". Such rules can typically not be directly represented in OWL since they are constraints between fillers of two different properties. As a consequence, we propose to use the Semantic Web Rule Language (SWRL) [29] rules to formalise them. As a matter of fact, SWRL was designed to add additional expressivity to OWL. The specification rule presented above will be formalised as follows:

```
lso:river(?x) ∧ so:width(?x,?y) ∧
swrlb:GreatherThan(?y,0) ∧ swrlb:LesserThan(?y,10) ∧
lso:db_HydrographicSegment(?z) ∧ so:represents(?z,?x) ⇒
lso:size(?z, "not wide")
```

6.6 How to formalise network section definition rules?

Network section definition rules often combine constraints on real world entities properties and database integrity constraints. As an example, a road section

specification states that "A new 'Road segment' is created if the value of the 'Road segment' feature class attribute 'number of lanes' must be changed, in order to be consistent with the '*number of lanes'* of this '*road segment'* in the real world. The resulting 'Road segment' must be longer than 1000m". Such rules are formalised thanks to someValuesFrom restrictions on the *hasLimite* ObjectProperty:

```
Class: lso:db_RoadSegment
  EquivalentTo: so:represents some lso:Road_segment and
                so:capturedAt some (so:CentreLine
                and(so:hasLimite some (so:Intersection
                or so:PropertyValueChange)))
Class: lso:bd_RoadSegment
  SubClassOf: lso:length some double [>1000.0]
```

6.7 Discussion

The geographic database specifications formalisation model that we propose has been implemented on the specifications of different databases in order to check whether it really enables to represent most of specifications contents. However, even if this model proved to be generic enough to represent different specifications, there remain specifications rules that are not formalised now in our proposal, such as fuzzy specifications rules. This is due to the fact that actual standard OWL version does not handle uncertainty. Second, vague rules that must be formalised can usually have many subjective interpretations: "'*Basins'*, '*Wells'* and '*Wash-houses'* are captured if they are *exceptional*". As a consequence, we propose to keep such vague rules as annotations in natural language (represented by OWL annotation properties).

Once the specification of each database that we want to integrate has been formalised in a LSO ontology, we can compare these LSO ontologies to automatically derive mappings between database schemas. For that purpose, a tool is being implemented in Java with the OWL API[7]. It takes two LSO ontologies as input and outputs expressive mappings based on the Geo Ontology Mapping Language [15] defined in the HUMBOLDT project as a geographic databases specific extension of the Ontology Mapping Language [30]. The use of a reasoner (Hermit[8]) enables us first to check the consistency of each formal specification. More, when both LSO ontologies are merged, it can infer equivalentClass and subClassOf relations between feature classes of our databases schemas. However, most of the time, relations between geographic databases feature classes are not direct equivalence or subsumption relations, but rather equivalence relations between instances of subsets of each databases feature class, which represents the same geographic entities. In order to find such relations, an application is being implemented for comparing axioms of both LSO ontologies, and derive expressive schema mappings from comparison results.

7 http://owlapi.sourceforge.net/
8 http://hermit-reasoner.com/

7 Conclusion

In this paper we proposed an OWL 2 based model for geographic database specification formalisation, which aims at eliciting geographic databases semantics by describing the link between data and what they represent. Two levels of formalisation are distinguished: the key concepts used in data specifications are specified in a specifications domain ontology (SO), whereas knowledge contained in one given database specification is described in a specification application ontology (LSO) which uses the concepts of the specifications ontology. This model is intended to be used in a global schema matching process. A tool enabling automatic comparison of formal specifications is being implemented. It aims at providing us with expressive schemas mappings between geographic heterogeneous databases, for schema translation or schema integration purposes.

Acknowledgement

This work is partly founded by the French Research Agency through the GeOnto project ANR-O7-MDCO-005 on "creation, alignment, comparison and use of geographic ontologies" (http://geonto.lri.fr/). The authors would also like to thank IGN-Belgium and Anne Féchir for providing us with additional data specifications.

8 References

1. Sheth, A., Larson, J.: Federated database systems for managing distributed, heterogeneous and autonomous databases. ACM Computing Surveys, 22(3), pp 183-236, (1990)
2. Bishr, Y.: Overcoming the Semantic and Other Barriers to GIS Interoperability. International Journal of Geographical Information Science, 12(4), 299--314, (1998)
3. Gesbert N.: Etude de la formalisation des spécifications de bases de données géographiques en vue de leur intégration. Phd thesis, University of Marne-la-Vallée, (2005)
4. Christensen, J. V.: Formalizing Specifications for Geographic Information. Proceedings of the 9th AGILE Conference on Geographic Information Science. College of Geoinformatics, University of West Hungary, pp. 186--194, (2006)
5. W3C: OWL Web Ontology Language, Overview, W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-features/
6. Fonseca, F., Clodoveu, D., Camara, G.: Bridging Ontologies and Conceptual Schemas in Geographic Information Integration. GeoInformatica, 7(4), pp 355--378, (2003)
7. IGN: BD Topo Pays, Version 1.2, Institut Géographique National, Paris, France,(2002)
8. Rodriguez, M.A., Egenhofer, M.J.: Determining Semantic Similarity Among Entity Classes from Different Ontologies, IEEE Transactions on Knowledge and Data Engineering, 15(2), 442--456, (2003).
9. Hess, G.N., Iochpe, C., Castano, C.: An Algorithm and Implementation for GeoOntologies Integration, Procedings 8th Symposium on GeoInformatics, Campos De Jordao, Brasil, pp 129--140, (2006).

10. Paul, M., Ghosh S.K.: An Approach for Service Oriented Discovery and Retrieval of Spatial Data, Procedings International Workshop on Service Oriented Software Engineering (IW-SOSE'06), Shangay, China, pp 84—94, (2006).
11. Nambiar, U., Ludscher Ludäscher, B., Lin K., Baru, C.: The GEON portal: accelerating knowledge discovery in the geosciences, Procedings 8th ACM International Workshop on Web Information and Data Management, Aarlington, Virginia, USA, pp 83—90, (2006).
12. Klien E.M.: Semantic Annotation of Geographic Information. Phd thesis, Institute for Geoinformatics, University of Muenster. Muenster, Germany, (2008).
13. Balley, S.: Aide à la restructuration de données géographiques sur le Web - Vers la diffusion à la carte d'information géographique. PhD Thesis, University of Marne-La-Vallée, France, (2007).
14. Schade, S.: Translation of Geospatial Data, Challenges, Solution and Vision, *12*th International Conference on Geographic Information Science (AGILE'09), Pre-Conference Workshop "Challenges in Spatial Data Harmonisation", Hannover, Germany, (2009).
15. Reitz, T., de Vries, M., Fitzner, D.: A5.2-D3[3.3]Conceptual Schema Specification and Mapping, HUMBOLDT Technical Report. (2009).
16. Fichtinger, A., Klien, E., Giger, C.: Challenges in Spatial Data Harmonisation: Examples and Approaches ferom the HUMBOLDT Project, *12*th International Conference on Geographic Information Science (AGILE'09), Pre-Conference Workshop "Challenges in Spatial Data Harmonisation", Hannover, Germany, (2009).
17. Kavouras M., Kokla M.:Theories of geographic concepts : Ontological Approaches to Semantic Integration. CRC Press, Taylor & Francis Group, Boca Raton, FL, USA, (2008)
18. Lieberman J., Singh R., Goad C.: W3C geospatial ontologies, W3C incubator group report, 23 october 2007. (2007)
19. Simperl E.P.B.: Reusing ontologies on the Semantic Web: A feasibility study. Data Knowledge Engineering. 68(10), 905-925 (2009).
20. Hudelot C., Atif J., Bloch I.: Fuzzy spatial relation ontology for image interpretation. Fuzzy Sets and Systems 159(15): 1929-1951 (2008)
21. Abadie, N.: Schema Matching Based on Attribute Values and Background Ontology, 12th International Conference on Geographic Information Science (AGILE'09), Hanover, Germany, (2009)
22. ISO TC/211: Geographic Information – Rules for application schema, 2001.
23. IGN: BD Carto, Version 3, Spécification de Contenu, Edition 1, Institut Géographique National, 175 p, Paris, France, (2005)
24. Partridge, C.: The role of ontology in integrating semantically heterogeneous databases. Technical Report 05/02 LADSEB-CNR, Padoue, (2002)
25. Kamel M., Aussenac-Gilles N.: Ontology Learning by Analysing XML Document Structure and Content, Knowledge Engineering and Ontology Development (KEOD), Madère, Portugal (2009)
26. W3C: OWL 2 Web Ontology Language, Document Overview, W3C Recommendation 27 October 2009. http://www.w3.org/TR/owl2-overview/
27. W3C: OWL 2 Web Ontology Language, New Features and Rationale Overview, W3C Recommendation 27 October 2009. http://www.w3.org/TR/2009/REC-owl2-new-features-20091027/
28. Cuenca Grau B., Horrocks I., Motik B., Parsia B., Patel-Schneider P., Sattler U.: OWL 2: The next step for OWL. J. of Web Semantics, 6(4), 309--322 (November 2008).
29. SWRL: A Semantic Web Rule Language combining OWL and RuleML, W3C Member submission 21 May 2004. http://www.w3.org/Submission/SWRL/
30. Euzenat, J., Scharffe, F., Zimmermann, A.: D2.2.10: Expressive alignment language and implementation. Project deliverable 2.2.10, Knowledge Web NoE (FP6-507482), (2007).