Integration of Spatial processing and knowledge Processing through the Semantic Web Stack

Ashish Karmacharya[1,2], Christophe Cruz[2], Frank Boochs[2], Franck Marzani[1]

[1] Institut i3mainz, am Fachbereich 1 - Geoinformatik und Vermessung
Fachhochschule Mainz, Holzstrasse 36, 55116 Mainz
{ashish, boochs}@geoinform.fh-mainz.de

[2] Laboratoire Le2i, UMR-5158 CNRS,
Dep. Informatique IUT Dijon, 7, Boulevard Docteur Petitjean
BP 17867, 21078 Dijon CEDEX, France
{christophe.cruz, franck.marzani}@u-bourgogne.fr

Abstract. This paper presents the integration process of spatial technologies and Semantic Web technologies and its associated tool. The result of this work is a spatial query and rule engine of spatial. To do so, existing ontology with spatial elements is adjusted in order to process the spatial knowledge through spatial technologies. This paper outlines the methods and the processes of these adjustments and how results are returned by our tool. The SWRL and the SPARQL language are extended for spatial purpose and the existing OWL ontology wine is used as an application example.

Keywords: Spatial processing, Knowledge processing, OWL Ontology, Rule language, Query language, SPARQL, SWRL.

1 Introduction

The Semantic Web is a set of technologies complementing the conventional Web tools proposed by Sir Tim Berners-Lee. It is seen as the most likely approach to reach the goal of semantic interoperability. The Semantic Web is envisaged as an extension to the existing web from a linked document repository into the platform where information is provided with the semantic allowing better cooperation between people and their machines. This is to be achieved by augmenting the existing layout information with semantic annotations that add descriptive terms to web content, with meaning of such terms being defined in ontologies [1]. Ontologies play crucial role in conceptualizing a domain and thus play an important role in enabling Web-based knowledge processing, sharing and reuse between applications.
This research attempts to contribute through including the functionalities of the spatial analysis within the Semantic Web framework. Moving beyond the semantic information, it has opened the chapter of inclusion of other form of information within the Semantic Web framework. It is important in the sense of the development of the technology itself. This work should at least provide a certain vision towards the direction the technology should take to integrate new forms of data. It discusses the direction in terms of spatial integration [3, 5].

The Semantic Web stack (e.g. fig. 1.) can be adjusted with a layer of that contains spatial information. The research proposes such an arrangement in the stack. A layer of spatial data mixing seamlessly with the semantic proposition in the layer Ontology through its OWL/RDF based syntax can be envisaged. This layer since uses the standard syntax of OWL/RDF can perform spatial queries through SPARQL or infer rules through standards as SWRL.

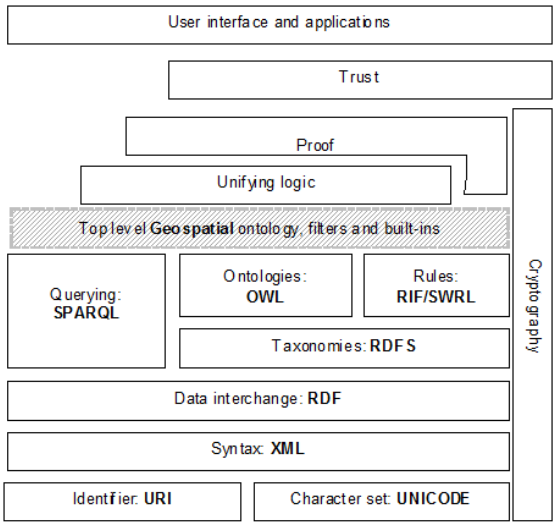

Fig. 1. Snapshot of the wine ontology adjusted with spatial component.

The integration process of the spatial technologies into the Semantic Web stack is undertaken by defining a new kinds of FILTERs for SPARQL queries and new kinds Built-ins for SWRL rules. These new FILTERS and Built-ins allow to process queries and rules with spatial data related to semantic data. The next chapter discusses this adjustment in the Semantic Web stack. Section 3 presents the top level ontology which enables the use of spatial technologies for any OWL ontology. Section 4 presents the translation engine which allows the translation of spatial queries and rules into standard queries and rules. Section 5 gives the complete process of extending an existing ontology in order to take into account the spatial technologies by using the famous wine ontology. Section 6 concludes this paper.

2 Background

The research project of Klein E. M. [2] in a certain degree follows the pattern of existing studies in geo-ontology research domain by focusing on the use of ontology for achieving data interoperability. It follows the pattern through defining the problems of data discovery in WFS (Web Feature Services) and the semantic differences in the data though the features associated to the data have same naming conventions. The problems follow even after the data discovery due to the nature of

information that data represents is not explicitly stored. The research hence plans a mechanism of match making of different SDIs (Spatial Data Infrastructure) through the mediation of semantic rich domain ontology designed through the consultation of the experts. The Domain Ontology as it terms contains explicit information which capture the meanings of real world entities.

As with the case of this research, the research [2] utilizes the inference capabilities of the description logics in the ontology representation language of OWL-DL through inference rules. In addition, it uses a simple hydrological example to semantically annotate the data through the spatial rules. The SWRL representations of the rule are given:

```
Region(?x) ^ hasSlope(?x, Flat) → Lowland(?x)                    (1)

Lowland(?x) ^ River(?river) ^ adjacentTo(?x,?river) ^            (2)
hasAltitude(?x, ?xAlt) ^ hasAltitude(?river, ?riverAlt) ^
swrlb:subtract(?diffAlt, ?xAlt, ?riverAlt) ^
swrlb:lessThan(4, ?diffAlt) → Floodplain(?x)
```

Region, Lowland, River and Floodplain are the concepts and hasSlope, adjacentTo and hasAltitude are the object properties in both feature type's ontology and domain ontology. The idea consists to semantically annotate the concept Floodplain with the rules. The first rule represented by equation 1 forms the lowland if the slope of a region is flat. There are many constraints of a region being lowland but the research uses this rule to demonstrate the usability. A Digital Elevation Model (DEM) is used in background which intersects the inferred information and the dataset is annotated as lowland. In short the object property hasSlope is intercepted and run through an algorithm which combines the DEM dataset to determine the flat slope. Regions inferring these flat slops are then annotated as lowland. Extending the rule to equation 2, it uses object property adjacentTo and built-ins of SWRL to annotate the floodplain. The object property adjacentTo again needs to run an algorithm in collaboration to the spatial dataset to provide the result. This result again infers with the other axioms in the knowledge base to enrich itself. The adjacentTo object property utilizes buffer operation to determine the objects close to it. However, the operation is hidden from the users and is executed inside the algorithm. This execution enriches the knowledge base which could be inferred through standard rule of SWRL. The execution of buffer or any spatial operations are carried out through the spatial operations of ArcGIS. The semantic annotation through these rules is carried out to enrich the Domain Ontology thus negating any short coming of explicit semantics in feature type's ontology.

The method of inferring the rules first through execution of spatial operations at database or application level and then enriching the knowledge base matches with the current research work. However, the implications in both researches are different. The approach that current research undertakes is to enhance the Semantic Web technologies through integrating spatial components into the technology. It differs significantly with the former research [2] as it was conducted to use semantic web tools and techniques to answer specific GIS problems. Hence, the scale of application of Semantic Web techniques is relatively low in the previous research [2]. In other

hand, it could be seen that the spatial operations and functions are used implicitly through object properties like hasSlope or adjacentTo which are terms of natural language. This might give ambiguity to the interpretations of these terms. For example the term adjacentTo can have two or more meanings as rightly quoted in the thesis report. It can be near to each other either through touching or not touching. So, the utilization of spatial operation should be based on these factors. If the adjacent to means that the objects are touching then the spatial operation "Touch" could be directly used instead of Buffer which is more resource dependent.

Contrary to [2], this research has taken the works forward to address these concerns. Instead of using the commonly used terms, it uses the spatial operations and functions terminology standardized by OGC [4]. Standard terms are proposed to formulate rules rather than using domain based terms.

The equation 3 illustrated the adjustment of object property adjacentTo directly through SWRL rules through spatial built-ins, which means that the individuals of River which are in the Buffer of 50m of a individual of the concept Lowland possesses a relationship (ObjectProperty) named adjacentTo that link the Rivers and Lowlands.

```
River(?x)^Lowland(?y)^Buffer(?x, ?y, 50)                    (3)
→adjacentTo(?x,?y)
```

Thus, it could be seen that there is much more flexibility concerning the definition of spatial rules through standard spatial built-ins proposed here. Besides the spatial built-ins for SWRL, this research adds on spatial built-ins to SPARQL, the query language of semantic web tools which is not explicitly researched before. However, before using these spatial built-ins in an existing ontology, it is first necessary to adjust this one with top level concepts. This integration process allows the linking of spatial data to ontologies. This ontology adjustment process is a generic process which allows the adjustment of any existing ontology in order to process spatial queries and rules on it.

3 The Top Level Ontology

This ontology serves as a foundation ontology to which objects can be instantiated during the identification process of spatial elements. The axioms are the building blocks of ontology and hence these axioms in the context of top level ontology of the application should be discussed to provide an overview of the system. The main axioms of this top level ontology are:

Semantic - spatial:Feature
Geometric - shp:Shape
Geometric Relationship - shp:hasShape
Spatial Relationship - sa:hasSpatialRelations
Spatial Database Relationship - doc:hasDBDetails

A shape has a definition in a spatial database. An individual has a shape and has spatial relationships with other individuals which have a shape.

The class axiom spatial:Feature represents the spatial objects. This class axiom is the generalized class of any objects with spatial definition. This class is further specialized into classes representing the different objects such as vin:Winery or vin:Region for instance regarding the example at the end of this paper. The spatial:Feature has to be specialized classes into subclasses. This abstract class cannot be instanced but only the individuals which belong to a subclass of spatial:Feature can have a spatial attribute.

The next important class axiom is shp:Shape which stores the local coordinates of the objects identified in the excavation site. This generalized class is specialized into shp:_3D and shp:_2D sub classes to represent the dimensions of the coordinates. Currently, an orthophoto is used to identify objects on a map and hence the 2D coordinates are returned of the objects. Semantics of objects in the knowledge base are defined through object property feat:objRel. But before that they need to relate to their spatial signature that is to their coordinates. This is managed through the specialized object property of shp:hasShape. As mentioned the coordinate of the object is derived through the digitization are stored as an individual of shp:_2D. This instance stores the coordinate of object. Once both the object and its coordinates are enriched, shp:hasShape provides a relationship between them. For instance, the concept win:Region as a subclass of spatial:Feature has the property shp:hasShape which can be a shp:_2D or shp:_3D.

The annotations to the database are carried out through assigning semantics to the annotations as assigning the relevant database and its relevant table in which the data is stored. It also provides the connection to the spatial column in which geometries of the objects are stored. An object property doc:hasDBDetails under general class doc:hasDocumentDetails provides these attributive connections. The three data properties to address the semantics of spatial annotation part of connecting to the MBRs are doc:dbName, doc:spColumn and doc:tableName, three specialized classes of doc:hasDBDetails.

The spatial functions and operations return geometries on their executions. It is hence important to have provision to store these returned geometries in the ontology. A generalized class sa:spatialOperation is introduced in the top level ontology. Every spatial operation under geoprocessing functions is then adjusted as its subclass. The class hierarchy of sa:spatialOperation reveals that the subclasses within it are the classes which need to represent returned geometries in some form.

The four spatial processing functions which are discussed here are Buffer, Union, Intersection and Difference. These spatial functions compute new spatial geometries. These new geometries are also stored in the spatial database in order to be computed by future spatial functions. As a solution, we definition four new classes called sa:sp_Buffer, sa:sp_Union, sa:sp_Intersection and sa:sp_Difference which are of specialized classes of sa:spatialOperation. The classes here are instantiated when the spatial operation of this category is executed. The result of execution is stored within the instantiated individual as the data property feat:localPlacement.

The functions under this category need to take a feature to execute them. The feature are objects within class feat:Feature. In order to maintain a relationship between the spatial operations representing classes under sa:spatialOperation and features under feat:Feature in the ontology an object property sa:hasSpatialRelations

is added in the top level ontology. The specialized property relates the individuals under sa:spatialOperation and feat:Feature. For example for every instance in class sa:sp_Buffer (sub class ofsa:spatialOperation) be a property sa:hasBuffer (specialized object property of sa:hasSpatialRelations) which relates the sa:sp_Buffer class to the classes specializing feat:Feature. There are also four sa:hasSpatialRelations defined corresponding to each geoprocessing functions (sa:hasBuffer, sa:hasUnion, sa:hasIntersection, sa:hasDifference). Besides theses object properties, data properties to correspond the attributive nature of the relationships are also adjusted in the top level ontology. A generalized data property sa:hasSpatialAttribute is introduced in the top level ontology. Other attributive properties as sa:hasBufferDistance (denotes the buffer distance of the buffer) are specialized properties of it.

Funtions	Concept	ObjectProperty	Execution Method
Buffer	sa:sp_Buffer	sa:hasBuffer(x,c)	$\text{sa: sp}_{\text{Buffer}} \sqsubseteq \exists \text{sa: hasBuffer. feat: Feature} \sqcap \text{sa: hasBufferDistance.}\{c\}$ C is of float value providing the buffer distance
Union	sa:sp_Union	sa:hasUnion(x,c)	$\text{sa: sp}_{\text{Union}} \sqsubseteq \exists \text{sa:}^{2} \text{ hasUnion. feat: Feature} \sqcap (\geq 2 \text{ hasUnion})$
Intersection	sa:sp_Intersection	sa:hasIntersection(x,c)	$\text{sa: sp\_Intersection} \sqsubseteq \exists \text{sa: hasIntersection. feat: Feature} \sqcap (\geq 2 \text{ hasIntersection})$
Difference	sa:sp_Difference	sa:hasDifference(x,c)	$\text{sa: sp\_Intersection} \sqsubseteq \exists \text{sa: hasIntersection. feat: Feature} \sqcap (\geq 2 \text{ hasIntersection})$
Intersection	sa:sp_Intersection	sa:hasIntersection(x,c)	$\text{sa: sp\_Intersection} \sqsubseteq \exists \text{sa: hasIntersection. feat: Feature} \sqcap (\geq 2 \text{ hasIntersection})$

Table 1. The Spatial Processing Functions

Functions	ObjectProperties	Characteristics
Disjoint	sa:hasDisjoint(x,y)	Symmetric
Touches	sa:hasTouch(x,y)	Symmetric
Within	sa:hasWithin(x,y)	Transitive
Overlaps	sa:hasOverlaps(x,y)	■
Equals	sa:hasEqual(x,y)	Symmetric, Transitive
Crosses	sa:hasCrosses(x,y)	Symmetric
Intersects	sa:hasIntersect(x,y)	Symmetric
Contains	sa:hasContain(x,y)	Transitive

Table 2. The Georelationtionship Functions

These functions demonstrate the spatial relations between objects hence they are very straightforward when adjusting in ontology. They can be directly adjusted through object properties within the top level ontology. These functions are adjusted

as specialized object properties of sa:hasSpatialRelations. The execution pattern of every function in this category is executed in similar. The table 2 illustrates the steps of every spatial function following OGC spatial operation standards but this research thesis utilizes four operations to demonstrate the argument. Those functions are Disjoint, Touch, Within and Overlap which are represented through sa:hasDisjoint, sa:hasTouch, sa:hasWithin and sa:hasOverlaps subsequently.

4 The translation engine

The translation engine allows the computation of spatial SPARQL queries and spatial SWRL rules. In both cases, the translation engine interprets the statements in order to parse the spatial components. Once the spatial components are parsed, they are computed through relevant spatial functions and operations by the translation engine through the operations provided at the database level. Then after, the spatial statements are translated to standard statements for the executions through their proper engine namely the SPARQL engine and the SWRL engine. Concerning the inference engine, the enrichment and the population of the ontology regarding the results of the inference process is possibly stored in the ontology.

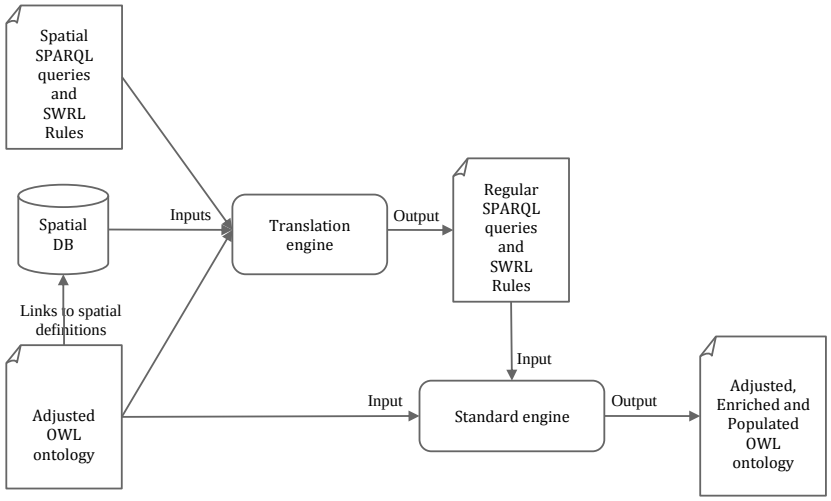

Fig. 2. The spatial processing of the translation Engine translating SPARQL queries and OWL rules

The next sections presents in details the translation engine on more specifically the translation process of spatial SPARQL queries to regular queries. The following one presents the translation process of spatial SWRL rules to regular SWRL rules. These two processes have in common the use of SQL statements to query to the spatial database.

4.1 Spatial SPARQL Queries

The FILTER keyword in SPARQL queries is used to define spatial queries. A FILTER can be used to compare strings and derive results. The functions like regular expression which matches plain literal with no language tag can be used to match the lexical forms of other literals by using string comparison function. In addition, SPARQL FILTER uses the relational operators as = or > or < for the comparison and restrict the result. From this idea, the FILTER principle is extender in order to process georelationship functions.

4.1.1 Geoprocessing FILTER

The following example shows how to select Building which intersect the buffer of 200km of a River. In this example, the keyword FILTER is replaced by the keyword SPATIAL_FILTER in order to be processed by the translation engine

```
SELECT  ?name1 ?name2
WHERE
{
               ?feat1         feat:name      ?name1
               ?feat2         feat:name      ?name2
               ?feat1         rdfs:type      feat:River
               ?feat2         rdfs:type      feat:Building

               SPATIAL_FILTER [buffer (?x, 200 000,?feat1)]
               SPATIAL_FILTER [intersection (?y,?x,?feat2)]
}
```

This process is a selection process, and no inference process is engaged. Once the process is ended, the rule is translated to a standard given in the following example. It can be seen that the SPATIAL_FILTER is replace by standard RDF triples which. Any SPARQL engine is able to run this rule.

```
SELECT  ?name1 ?name2
WHERE
{
       ?feat1  feat:name                    ?name1
       ?feat2  feat:name                    ?name2
       ?feat1  rdfs:type                    feat:River
       ?feat2  rdfs:type                    feat:Building

       ?feat1  sa:hasBuffer                 ?x
       ?x      rdfs:type                    sa:sp_buffer
       ?x      sa:hasBufferDistance  200 000

       ?y      rdfs:type sa:sp_Intersection
       ?y      sa:hasIntersection ?x
       ?y      sa:hasIntersection ?feat2
}
```

The table 3 shows the translation of geoprocessing functions contained in SPATIAL_FILTER into standard triple component of a SPARQL query.

Function	Spatial SPARQL Syntax	Translation
Buffer	SPATIAL_FILTER [buffer (?x, b, ?y)]	?x rel:hasBuffer ?y
	Result: Populated in the knowledge base as	?y rdfs:type sa:sp_buffer
	individuals of class sa:sp_Buffer.	?y sa:hasBufferDistance 200 000
Union	SPATIAL_FILTER [union (?x, ?y1,?y2)]	?x rdfs:type sa:sp_Union
	Result: Populated in the knowledge base as	?x sa:hasUnion ?y1
	individuals of class sa:sp_Union.	?x sa:hasUnion ?y2
Intersection	SPATIAL_FILTER [intersection (?x, ?y1,?y2)]	?x rdfs:type sa:sp_Intersection
	Result: Populated in the knowledge base as	?x sa:hasIntersection ?y1
	individuals of class sa:sp_Intersection.	?x sa:hasIntersection ?y2
Difference	SPATIAL_FILTER [difference (?x, ?y1,?y2)]	?x rdfs:type sa:sp_difference
	Result: Populated in the knowledge base as individuals of class sa:sp_Difference.	?x sa:hasDifference ?y1
		?x sa:hasDifference ?y2

Table 3. The spatial SPARQL syntax and its translation into SARQL syntax.

4.1.2 Georelationship FILTER

The following example shows how to select couples of features which are linked by a touch spatial relationship. In this example, the keyword FILTER is replaced by the keyword SPATIAL_FILTER in order to be processed by the translation engine. The name of features couples are selected with this restriction. The first feature has to be a feat:River which is of kind of feat:feature, and the second feature has to be a feat:Building which is also of kind of feat:feature. The SPATIAL_FILTER selects the couples which are touching spatially.

```
SELECT  ?name1 ?name2
WHERE
{
        ?feat1          feat:name       ?name1
        ?feat2          feat:name       ?name2

        ?feat1          rdfs:type       feat:River
        ?feat2          rdfs:type       feat:Building

        SPATIAL_FILTER [touches (?feat1, ?feat2)]
}
```

This process is a selection process, and no inference process is engaged. The aim of the translate engine consists to compute the touches spatial process of the Cartesian production between the features of the kind feat:River and feat:Building. In the case of a positive result, this new link is stored in the ontology between the couple of feature with the help of a sa:hasTouches relationship which is of the kind of sa:hasSpatialRelations. Once the process is ended, the rule is translated to a standard

given in the following example. It can be seen that the SPATIAL_FILTER is replace by the triple "feat1 sa:touch ?feat2". Thus this rule can be processed by a standard SPARQL engine.

```
SELECT  ?name1 ?name2
WHERE   {
        ?feat1          feat:name       ?name1
        ?feat2          feat:name       ?name2
        ?feat1          rdfs:type       feat:River
        ?feat2          rdfs:type       feat:Building          ?feat1
        sa:touch        ?feat2
}
```

The table 4 shows the translation of georelationship functions contained in SPATIAL_FILTER into standard triple component of a SPARQL query.

Functions	ObjectProperties	Characteristics
Disjoint	sa:hasDisjoint(x,y)	Symmetric
Touches	sa:hasTouch(x,y)	Symmetric
Within	sa:hasWithin(x,y)	Transitive
Overlaps	sa:hasOverlaps(x,y)	■
Equals	sa:hasEqual(x,y)	Symmetric, Transitive
Crosses	sa:hasCrosses(x,y)	Symmetric
Intersects	sa:hasIntersect(x,y)	Symmetric
Contains	sa:hasContain(x,y)	Transitive

Table 4. The spatial SPARQL syntax and its translation into SARQL syntax.

4.1.3 Optimization

The translation engine is time consuming for large spatial database. In order to select the context of execution four options can be given to the SPATIAL_FILTER.

SPATIAL_FILTER_SELECT: No spatial operation is undertaken; the rule is translated without any spatial processing

SPATIAL_FILTER_PROCESS: Spatial operations are processed only for the couples of features which don't have this relationship. If this relation already exists, this one is not computed.

SPATIAL_FILTER_UPDATE: Spatial operations are processed only for the couples of features which have already this relationship in order to update these relationships.

SPATIAL_FILTER_ALL: This is the option by default which consists to compute all relationship for the Cartesian product in order to process it if it doesn't exist or in order or update it.

The following example shows that the selection of features which have the touches relationship is done with the option SPATIAL_FILTER_UPDATE.

```
SELECT  ?name1 ?name2
WHERE
{
            ?feat1          feat:name       ?name1
            ?feat2          feat:name       ?name2
               ?feat1                rdfs:type       feat:River
            ?feat2          rdfs:type       feat:Building

            SPATIAL_FILTER [touches (?feat1, ?feat2)]
            SPATIAL_FILTER_UPDATE
}
```

In addition the spatial filter can be combined by the following manner. It consists to insert news filters and to use the same variable. The following example consists to select building which contains a chimney in order to see if it touches a river. Moreover, no spatial processing is done, only the existing knowledge in the ontology is used to process this query.

```
SELECT  ?name1 ?name2
WHERE{
            ?feat1          feat:name       ?name1
            ?feat2          feat:name       ?name2
            ?feat1          rdfs:type       feat:River
            ?feat2          rdfs:type       feat:Building
            ?feat2          rdfs:type       feat:Chimney

            SPATIAL_FILTER [touches (?feat1, ?feat2)]
            SPATIAL_FILTER [touches (?feat2, ?feat3)]
            SPATIAL_FILTER_ SELECT
}
```

4.2 Inference Rules through SWRL

In an attempt to define the built-ins for SWRL, a list of eight built-ins was proposed during the research work. These eight built-ins reflect four geoprocessing functions and four georelationship functions that are discussed previously. The built-ins reflecting geoprocessing functions are built up in combinations with the spatial classes adjusted in the ontology and their relevant object properties. The built-ins for georelationship functions are in contrast are just object properties and using these object properties in collaboration to the spatial functions in database system.

4.2.1 Geoprocessing Built-ins

The first set of built-ins is the built-ins for geoprocessing functions. They are functions returning geometries and adjusted in the ontology through feat:Feature sa:hasSpatialRelations sa:spatialOperation sequence. This class-property series is illustrated in table 5. The initial step consist the built-ins parsed to be processed by the translation engines. First the spatial built-ins are identified from the statement and parsed. Concurrently, the features on which these built-ins are applied are also

identified.Then after, the SQL statementswith relevant spatial function on the relevantobjects of the featuresare executed at the database level. The results are then enriched in the knowledge base. Once, the knowledge base is enriched, the spatial built-ins are broken down into standard feat:Feature sa:hasSpatialRelations sa:spatialOperation sequence to generate the standard SWRL statement which is executed through standard inference engines.

Functions	Class	Object Property	Data Property	Built-ins
Buffer	sa:sp_Buffer	sa:hasBuffer	sa:hasBufferDistance	Buffer(?x, b, ?y)
Union	sa:sp_Union	sa:hasUnion	-	Union(?x,?y1,y2)
Intersection	sa:sp_Intersection	sa:hasIntersection	-	Intersection(?x,?y1,y2)
Diffrence	sa:sp_Difference	sa:hasDifference	-	Difference(?x,?y1,y2)

Table 5: GeoProcessing built-ins

The execution of every built-in can be elaborated through first running down the spatial operation and then translating the statements with spatial built-in into standard SWRL statements. Simplifying the explanations with an example of

```
feat:Feature(?x) ^ Buffer(?x, b, ?y)
```

suggesting the use of built-in Buffer on objects within the specialized classes of feat:Feature with the buffer distance. This statement is elaborated first through running the SQL statement with the spatial function buffer on each objects of the class to which it meant to run. That is if the statement is related to buffering walls, then each instance of class feat:Wall is taken and buffered through the execution of the SQL statement. The SQL statement with spatial function Buffer would look like:

```
SELECT Buffer(geom::Feature, bufferDistance)
```

Here, the geom are the geometries of the objects within specialized classes of feat:Feature. The result of this execution is then enriched in the knowledge base. Primarily, the rows in result are geometries which indicate the buffers of each object with certain buffer distance. The class sa:sp_Buffer is instantiated with objects representing every row and storing the buffer geometry and the buffer distance within them. Then after, it is time to translate the statement with the spatial built-in into standard form of SWRL statement which would be

```
feat:Feature(?x)^sa:hasBuffer(?x,?y)^sa:sp_Buffer(?y)^
sa:hasBufferDistance(?y,b)
```

Thus, the statement converts the spatial built-in into feat:Feature sa:hasSpatialRelations sa:spatialOperation sequence of standard SWRL statement. The complete list of SQL execution, the result enrichment and statement translation process is illustrated in table 6.

Built-ins	SQL Statements	Translated Built-ins	Built-ins
swrlbspatial:Buffer(?x, b, ?y)	SELECT Buffer(geom::Feature, bufferDistance) Result: Populated in the knowledge base as individuals of class sa:sp_Buffer.	sa:hasBuffer(?x,?y) ^ sa:p_Buffer(?y) ^ sa:hasBufferDistance(?y, b)	swrlbspatial:Buffer(?x, b, ?y)
swrlbspatial:Union(?x,?y1,?y2)	Select Union(geom::Feature1, geom::Feature2) Result: Populated in the knowledge base as individuals of class sa:sp_Union.	sa:sp_Union (?x) ^ sa:hasUnion(?x, ?y1) ^ sa:hasUnion(?x, ?y2)	swrlbspatial:Union(?x,?y1,?y2)
swrlbspatial:Intersection(?x,?y1,?y2)	Select Intersection(geom::Feature1, geom::Feature2) Result: Populated in the knowledge base as individuals of class sa:sp_Intersection.	sa:sp_Intersection(?x) ^ sa:hasIntersection(?x, ?y1) ^ sa:hasIntersection(?x, ?y2)	swrlbspatial:Intersection(?x,?y1,?y2)
swrlbspatial:Difference(?x,?y1,?y2)	Select Difference(geom::Feature1, geom::Feature2) Result: Populated in the knowledge base as individuals of class sa:sp_Difference.	sa:sp_Difference(?x) ^ sa:hasDifference(?x, ?y1) ^ sa:hasDifference(?x,?y2)	swrlbspatial:Difference(?x,?y1,?y2)

Table 6: The SQL statements executions of geoprocessing built-ins for the spatial enrichment

The georelationship built-ins rely on object properties and more straight forward. The built-ins and their linkage to the object properties are presented in table 7.

Functions	Class	Object Property	Built-ins
Disjoint	-	sa:hasDisjoint	Disjoint(?x, ?y)
Touches	-	sa:hasTouch	Touches(?x, ?y)
Within	-	sa:hasWithin	Within(?x, ?y)
Overlaps	-	sa:hasOverlap	Overlaps(?x, ?y)

Table 7: Georelationship Built-ins

However, it is necessary to determine the nature of built-ins from the statement to determine what spatial operation needs to be performed at database level. These statements are hence parsed to identify the spatial built-ins from the statement. Then after, the SQL statement with related spatial operation is executed in the database level. The results are enriched against their specified object properties in the

knowledge base. Now, the statements are ready to get executed. The spatial built-ins are broken down into feat:Feature sa:hasSpatialRelations feat:Feature sequence by the translation engine which is now a standard statement so can be executed.

Built-ins	SQL Statements	Translated Built-ins
swrlbspatial:Disjoint(?x, ?y)	SELECT Feature2 FROM spTable WHERE Disjoint(geom::Feature1, geom::Feature2)	sa:hasDisjoint(?x, ?y)
swrlbspatial:Touches(?x, ?y)	SELECT Feature2 FROM spTable WHERE Touch(geom::Feature1, geom::Feature2)	sa:hasTouch(?x, ?y)
swrlbspatial:Within(?x, ?y)	SELECT Feature2 FROM spTable WHERE Within(geom::Feature1, geom::Feature2)	sa:hasWithin(?x, ?y)
swrlbspatial:Overlaps(?x, ?y)	SELECT Feature2 FROM spTable WHERE Overlap(geom::Feature1, geom::Feature2)	sa:hasOverlaps(?x, ?y)

Table 8: SQL statements executions of georelationship built-ins for the spatial enrichment

It would be helpful to elaborate with an example of built-in

```
Feat:Feature(?x) ^ feat:Feature(?y)^ Touch(?x,?y)
```

It is a spatial operation to determine whether an object is touching another. Generally, the georelationship operations are binary operations and return Boolean values when is executed alone. However, when executed as a conditional parameter of the SQL statement, they yield results. That is if the statement

```
SELECT Touch(geom::Feature1, geom::Feature2)
```

is executed. It returns either true or false determining whether the geometry of feature1 touches geometry of feature2. But if the same operation is executed as

```
SELECT Feature2 FROM spTable WHERE Touch(geom::Feature1,
geom::Feature2)
```

then it returns all the feature2 which touches feature1. Here spTable is the table where the geometries of the features are stored in the database system and has been spatially annotated. The results derived through the execution of the statement with Touch operation is then enriched against sa:hasTouch object property of the specified feature. The last step is to break down the Touch(?x, ?y) built-in into feat:Feature sa:hasSpatialRelations feat:Feature sequence to get the SWRL statement executed. The breakdown of the spatialbuilt-in Touch(?x, ?y) is given as

```
feat:Feature(?x) ^ hasTouch(?x, ?y) ^ feat:Feature(?y)
```

It is a standard SWRL statement which can again be inferred by inference engines. The complete list of SQL statement execution is illustrated in table 7.

5 The wine example

The famous wine ontology is used here to present the principle of spatial ontology adjustment which allows the computation of spatial data on any existing OWL ontology. The wine ontology is selected for several reasons. The wine ontology appears frequently in the literature as an example to define tutorials.

5.1 The Existing Ontology Adjustment

In order to adjust the exiting ontology, two main steps are essential. First, the top level ontology has to be imported into the existing ontology. In this manner, all the components of the spatial layer are available for the existing ontology, which are the annotation and tagging principles of documents and more specifically the spatial definitions. The second step consists to specialized specific concepts of the exiting ontology which have possibly spatial signatures. In the wine ontology, wine regions can be defined as spatial region or polygons in a GIS system. In addition, the wineries can be geolocalized as points in the same GIS system. Since the existing ontology is adjusted, the feed of the spatial database regarding the concepts respectively, wine region and wineries, of the ontology can be undertaken with the help of the individual already defined in the ontology. For instance, the individual vin:ClosDeVougeot, which is a French winery localized in Burgundy, is defined by the geolocalized point 47.174835,4.95544 in the WGS84 coordinate system.

The following figure shows the adjusted wine ontology. On the left side, the tree viewer represents the hierarchy of concept with the top level ontology and spatial:feature concept with the wine ontology specialized concept vin:Region and vin:Winery. All the other concept of the wine ontology can by spatially defined. On the right side, the list of the vin:Winery individuals is given. The individual vin:ClosDeVougeot appears in this list. This list is composed of 43 individuals and the list of vin:Region is composed of 36 individuals.

5.2 Spatial Querying process.

This section presents the benefit of spatial querying on spatial data composed of semantic definition. In the figure 5.16, the individual vin:CoteDOrRegion has a relationship has:adjacentRegion. This relationship defines a symmetric relationship between two regions. In the wine ontology, this information is not feed. Currently, it is no possible to select adjacent regions and regions which are around of 200km to each other. In the case of a spatial definition in a Spatial GIS, the following queries are possible. The first query select all the adjacent regions to vin:CoteDOrRegion. The second query select all the regions which are around of 200km to the region vin:CoteDOrRegion.

```
SELECT  ?adjacent
WHERE
{
       vin:CoteDOrRegion     rdfs:type      vin:Region
       ?adjacent             rdfs:type      vin:Region

       SPATIAL_FILTER [touches (vin:CoteDOrRegion,?adjacent)]
}

SELECT  ?region
WHERE
{
       vin:CoteDOrRegion     rdfs:type      vin:Region
       ?region        rdfs:type      vin:Region

       SPATIAL_FILTER [buffer(?buffer,200000,vin:CoteDOrRegion)]
       SPATIAL_FILTER [intersection (?res,?buffer,?region)]
}
```

The first examples of queries are related to the same kind of individuals, the same can be undertaken on different kind of individuals. For instance, no spatial relationships are defined between regions and wineries. With the adjustment of the ontology and the spatial definition of wine regions and wineries, now the following query can be undertaken easily. I would like to know all the wineries in a specific region.

```
SELECT  ?winery
WHERE
{
       vin:CoteDOrRegion     rdfs:type      vin:Region
       ?winery        rdfs:type      vin:Winery

       SPATIAL_FILTER [within (vin:CoteDOrRegion,?winery)]
}
```

If these relationships were defined in the ontology, then it would be possible to check the spatial consistency of the knowledge based. The individual vin:ClosDeVougeot is a winery located in vin:CoteDOrRegion. In the case of the definition of a symmetric relationship named vin:located between the concept vin:Region and the concept vin:Winery, the individual vin:ClosDeVougeot should be linked to the individual vin:CoteDOrRegion with the help of this relationship. The following query is able to validate this relationship from spatial point of view.

```
SELECT  *
WHERE
{
       vin:CoteDOrRegion     rdfs:type      vin:Region
       vin:ClosDeVougeot     rdfs:type      vin:Winery

       SPATIAL_FILTER [within (vin:CoteDOrRegion, vin:ClosDeVougeot)]
}
```

If the result is false, but the spatial data define and correct than the ontology is inconsistent. The overlap between the semantic links and the spatial data permits to

check the consistency of the knowledge base in the case that the links were not generated from the spatial processing.

5.3 Spatial Inference process.

With the help of the SWRL rules, the enrichment of the ontology is now possible. The following simple example underlines this idea. The winery Clos de Vougeot vin:ClosDeVougeot which is a located in the region of Côte D'Or vin:CoteDOrRegion, and this region is actually a region located in France vin:FrenchRegion. Consequently, the winery Clos de Vougeot vin:ClosDeVougeot is located in France vin:FrenchRegion. The transitive relationship vin:hasSubRegion allows the definition of relationships between regions vin:Region.

This first SWRL rule enriches the ontology with vin:hasSubRegion relations between regions.

```
vin:Region(?x) ^ vin:Region(?y) ^ spatialswrlb:Within(?y, ?x) →
vin:hasSubRegion(?x, ?y)
```

This second SWRL rule enriches the ontology with vin:isLocatedInRegion relations between wineries and regions.

```
vin:Region(?x) ^ vin:Region(?y) ^ vin:Winery(?z) ^
vin:hasSubRegion(?x, ?y) ^  vin:isLocatedInRegion (?z, ?x)
→ vin:isLocatedInRegion (?z, ?y)
```

This third SWRL rule does at the same time the first and the second rule by using spatial built-ins.

```
vin:Region(?x) ^ vin:Region(?y) ^ vin:Winery(?z) ^
swrlbspatial:Within(?y, ?x) ^ swrlbspatial:Within(?z, ?y)
→ vin:isLocatedInRegion (?z, ?y) ^ vin:hasSubRegion(?x, ?y)
```

After the execution of this third rule, new relationships vin:isLocatedInRegion and vin:hasSubRegion are created in the ontology in order to link . Consequently, the ontology is enriched with these new relationships.

6 Conclusion

This research attempts to highlight the possibilities to integrate spatial technology in semantic web framework. It moves beyond the scope of data interoperability while presenting the concept and makes efforts to utilize the potentiality in other areas of the Semantic Web technologies. The underlying technologies of knowledge processing provide to the semantic web the capabilities to process the semantics of the information through close collaboration with the machine. It makes not only the understanding of data easier for achieving interoperability among different data sources, but it also provides valuable knowledge which could enrich the knowledge

base in order to equip it with new knowledge. This helps the users understand the data better. The underlying knowledge technology makes stand out among its contemporaries.

It is important to have standard terms for every built-in that will be developed to process spatial knowledge. With other built-ins in the tools standardized by W3C, the spatial built-ins should also get standardized by the consortium. In addition to W3C, OGC should also get involved in standardizing the built-ins. An effort in this direction should be carried out.

7 References

1. Horrocks, I., Pater-Schneider, P. F., McGuinness, D. L., & Welty, C. A. OWL: a Description Logic Based Ontology Language for the Semantic Web.
2. Klien, E. M.. Semantic Annotation of Geographic Information. Essen: University of Muenster, thesis report, (2008)
3. Karmacharya, A., Cruz, C., Boochs, F., Marzani, F., Use of Geospatial Analyses for Semantic Reasoning, Knowledge-Based and Intelligent Information and Engineering Systems - 14th International Conference, KES 2010, Cardiff, UK, September 8-10, 2010, Proceedings, Part I. Lecture Notes in Computer Science 6276 Springer 2010, ISBN 978-3-642-15386-0.
4. OGC, The Open Geospatial Consortium, Inc.®, http://www.opengeospatial.org/
5. A. Karmacharya, C. Cruz, F. Boochs, F. Marzani, ArchaeoKM: Managing Archaeological data through Archaeological Knowledge, Computer Applications and Quantitative Methods in Archeology - CAA'2010, Granada (Spain) 6-9 April 2010
6. Bechhofer, S., Harmelen, F. v., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., et al. (2004, February 10). OWL Web Ontology Language. Retrieved November 27, 2009, from W3C Recommendation: http://www.w3.org/TR/owl-ref/
7. Berners-Lee, T., Hendler, J., & Lassila, O. (2001, May). The Semantic Web. Scientific AmericaN, pp. 34-43.
8. Berry, J. K. (1999). GIS Technology In Environmental Management: a Brief History, Trends and Probable Future. In D. L. Soden, & B. S. Steel (Eds.), Handbook of Global Environmental Policy and Administration (pp. 49 - 81). Decker, Marcel Inc.