Formalization of 2-D Spatial Ontology and OWL/Protégé Realization

Kulsawasd Jitkajornwanich
Computer Science and Engineering Department
Univ. of Texas at Arlington
P.O. Box 19015
Arlington, TX 76019
Tel. +1 253 205 1250
kulsawasdj@hotmail.com

Ramez Elmasri
Computer Science and Engineering Department
Univ. of Texas at Arlington
P.O. Box 19015
Arlington, TX 76019
Tel. +1 817 272 0067
elmasri@cse.uta.edu

Chengkai Li
Computer Science and Engineering Department
Univ. of Texas at Arlington
P.O. Box 19015
Arlington, TX 76019
Tel. +1 817 272 0162
cli@cse.uta.edu

John McEnery
Department of Civil Engineering
Univ. of Texas at Arlington
P.O. Box 19308
Arlington, TX 76019
Tel. +1 817 272 0234
mcenery@uta.edu

ABSTRACT
Ontology specification is a core component of the Semantic Web, and facilitates interoperability among different systems that use distinct models. Developing a spatial ontology will allow many applications that have spatial objects to interact. In this paper, we formalize 2-D spatial concepts and operations into a spatial ontology. We show how these concepts can be realized in Protégé [30]. The Protégé spatial ontology provides spatial built-ins that can be used to provide a spatial dimension to other ontologies when needed. We give some examples of the use of our ontology, which is based on a standard Geometry class hierarchy [23], with a few modifications.

Categories and Subject Descriptors:
H.2.8 [**Database Management**]: Database Applications – *spatial databases and GIS*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods – *semantic networks*.

General Terms: Design

Keywords: Ontology, spatial database, semantic web, Protégé

1. INTRODUCTION
Ontology [10, 20] is considered to be a core component of the Semantic Web [7]. With the reasoning, inferencing, and representation mechanisms associated with an ontology, it becomes possible that systems with different definitions of the same concepts can interoperate with each other. In addition, a nearly complete description of concepts in a particular area of knowledge becomes readily available for interested users. In this paper we focus on spatial ontology. Representing spatial knowledge is a basic problem in many applications, such as GIS and map applications. In the past few years, work on spatial ontologies has focused on two main areas: spatial database integration [8, 9, 12] and spatial ontology creation and its use in the semantic web [5, 29, 11, 14]. In spatial database integration, a spatial ontology is used as a tool to integrate different spatial databases. In spatial ontology creation, there are two different major approaches. First, by analyzing a collection of existing spatial databases and methodologies, a spatial ontology model is defined based on those databases [5]. However, this leads to the problem that the created spatial ontology will be limited to those databases and consequently will not be sufficient to be a standard for representing a complete formal spatial ontology. The second approach in spatial ontology creation is to define a complete spatial ontology model. For example, in [29], they propose to create a spatio-temporal ontology based on the MADS model, which allows a regular database to model spatial and temporal characteristics [24, 25]. However, this approach has not been materialized in an implemented system, and there is no formal specification of spatial ontology developed from this approach. In addition, it is limited to the polygon data type only. Thus, the complete set of operations among point, line, and polygon is lacking. Finally, in [11, 14], they propose using the RCC8 calculus [26] for spatial reasoning on regions, but they do not propose a complete spatial ontology, which is the ultimate goal of our work.

Our work gives a formal specification of spatial ontology, defines a complete collection of spatial operations, and provides a general spatial ontology implemented in Protégé. Many researchers have worked in the area of Temporal and Spatial Ontology [19, 29, 8, 5, 9, 18, 22, 13, 6, 12, 1, 4]. A specification of temporal ontology was introduced in [13]. It clearly discussed temporal ontology formalization, and comprehensively defines temporal concepts and operations. It is based on the temporal logic developed by Allen [2, 3]. The following is an example of the *Meet* operation between two time intervals formalized by [13], assuming that T_1, T_2 are two time intervals and t is a time instant.

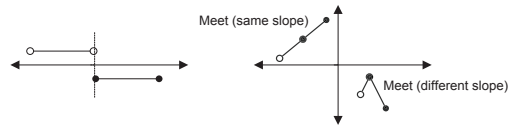
$$Meet(T_1,T_2) \equiv (\exists t)[ends(t,T_1) \wedge begins(t,T_2)]$$

A complete formalization of ontology forms the basis and reference for ontology implementation. In addition, since the temporal ontology specification in [13] was intended to capture all temporal reasoning on web pages, it is gradually becoming the standard for temporal ontology specification.

One of the main applications of spatial ontology is GIS applications. Although spatial concepts and operations have been specified in many works [23, 24, 25, 28, 31], there are few attempts at specifying a complete formal ontology for spatial concepts. Spatial operations are more complex than temporal operations, and can be defined over multiple dimensions, especially two and three dimensions, whereas temporal operations are only on one dimension. Figure 1 shows how the *Meet* operation is different in one dimension and two dimensions. Additionally, temporal operations have only two directions

SWIM 2011, June 12, 2011, Athens, Greece.

(before and after) whereas for two dimensional spatial operations, there are continuous directions along 360º of a two dimensional space. Figure 2 shows eight directions, at 45º intervals. (East is 0º, north is 90º, west is 180º, etc.).



a) Temporal operation (1-D) b) Spatial operation (2-D)

Figure 1. The difference of *Meet* operation in 1-D and 2-D

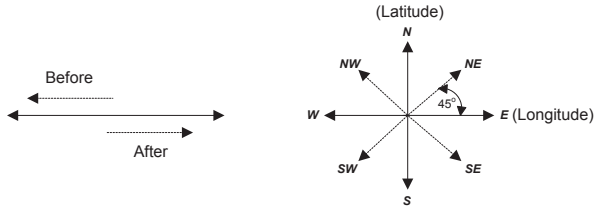


Figure 2. The difference of direction between temporal operation (1-D, left) and spatial operation (2-D, right)

In recent years, spatial or location-based applications have received a lot of attention. Examples include Google Maps and other location-based applications. One of the main advantages of having a spatial ontology specification is that such a specification acts as a conversion medium among various applications, which allows communication among different systems without requiring one system to know the exact description used by other systems.

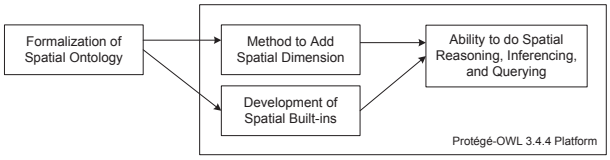


Figure 3. An overview of our methodology

In this paper, we present preliminary results towards our long-term goal of a complete two-dimensional spatial ontology specification. We will first formalize the spatial ontology concepts and operation definitions, then discuss a method to add spatial dimension to existing ontologies. Particularly we use a technique similar to the "lightweight" temporal ontology introduced in [22] so that we can incorporate a spatial layer into existing ontologies without requiring significant changes to the original ontology. We also implement spatial built-ins in Protégé based on the concept and operation definitions in the spatial ontology formalization to do spatial reasoning, inferencing and querying on ontology. Figure 3 shows an overview of our methodology for spatial ontology development.

As mentioned earlier, spatial operations are much more complex than temporal operations. Therefore in this initial effort we will only focus on a subset of two-dimensional spatial operations. Considering spatial relationships, there are six major types as follows: (1) Point VS Point, (2) Point VS Line, (3) Point VS Polygon, (4) Line VS Line, (5) Line VS Polygon, and (6) Polygon VS Polygon. In this paper only spatial relationships between Point and Line (i.e., (1), (2), and (4)) are covered. Other relationships will be covered in future work.

The rest of the paper is organized as follows. Section 2 discusses the spatial ontology formalization consisting of spatial concept definitions and spatial operation definitions. Section 3 discusses how the defined ontology is implemented in Protégé, along with some small examples to show how it works. Section 4 presents conclusion and future work.

2. SPATIAL ONTOLOGY FORMALIZATION

2.1 Formalization of Concept Definitions

In [23], a geometry class hierarchy is proposed for 2-D objects. The hierarchy shown in Figure 4 is based on the one in [23], with some minor modifications to allow our formalization.

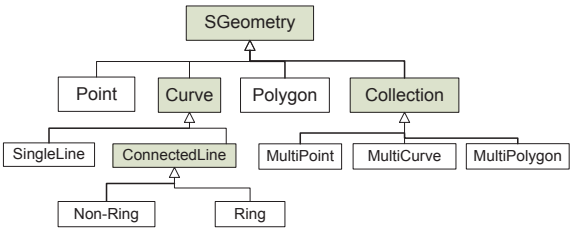


Figure 4. Geometry class hierarchy

Considering the leaf nodes in the hierarchy of Figure 4, the geometry objects can be categorized into 8 types as shown in Figure 5.

1. Point (*p*)
2. Single Line (*sl*)
3. Connected Line (*cl*): Non-Ring (*nr*)
4. Connected Line (*cl*): Ring (*r*)
5. Polygon (*a*)
6. MultiPoint (*mp*)
7. MultiCurve (*mc*)
8. MultiPolygon (*ma*)

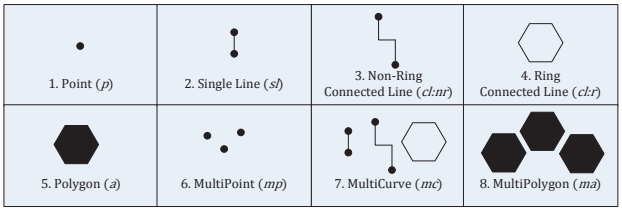


Figure 5. Types of geometry object in 2-D space

Our proposed spatial ontology consists of two parts: concept definitions and operation definitions. The following is the formalization of concept definitions.

In order to be used in the ontology, the concept definitions of all geometry object types in two-dimensional space (see Figure 4 and 5) have to be defined. For simplicity, we shall assume the 2-D coordinate system based on longitude and latitude, although the formalization can be adapted to other 2-D coordinate systems.

1. Point (*p*)

Point can be defined by longitude(x) and latitude(y).

$Point(p) \equiv (x, y)\ where\ x = longitude\ and\ y = latitude$

2. Single Line (*sl*)

Single line can be defined by any two points.

$SingleLine(sl) \equiv (p_1, p_2)\ ;\ p_1, p_2\ are\ points\ and\ p_1 \neq p_2.$

3. Connected Line (*cl*): Non-Ring (*nr*)

Non-ring connected line can be defined by a sequence of points $(p_1, p_2, \dots, p_N)$, $N \geq 3$, and $p_i \neq p_j\ for\ i \neq j$, which in turn defines a sequence of single lines $(sl_1, sl_2, \dots, sl_{N-1})$ and each $sl_i = (p_i, p_{i+1})$.

In other words, the sequence must contain at least two single lines and each p_i is connected to p_{i+1} for $i<N$.

$NonRingConnectedLine(cl{:}nr) \equiv (p_i|1 \le i \le N, N \ge 3, p_i \ne p_j$
$for\ i \ne j, sl_i = (p_i, p_{i+1}), Meet(sl_i, sl_{i+1}) = true\ for\ i = 1, \dots, N-1)$

4. Connected Line (*cl*): Ring (*r*)
Ring connected line is defined as a sequence of points $(p_1, p_2, \dots, p_N)$, $N \ge 3$, and $p_i \ne p_j\ for\ i \ne j$, which in turn defines a sequence of single lines $(sl_1, sl_2, \dots, sl_N)$ where each $sl_i = (p_i, p_{i+1})$ for *i=1,2,…,N-1* and $sl_N = (p_N, p_1)$. However, the area inside the ring is not part of the connected line.

$RingConnectedLine(cl{:}r) \equiv (p_i|1 \le i \le N, N \ge 3, p_i \ne p_j\ for\ i \ne j,$
$sl_i = (p_i, p_{i+1})\ for\ i = 1,2, \dots, N-1\ and\ sl_N = (p_N, p_1),$
$Meet(sl_i, sl_{i+1}) = true\ for\ i = 1, \dots, N-1, Meet(sl_N, sl_1) = true,$
$and\ area(r) = 0)$

5. Polygon (*a*)
Polygon is defined as a sequence of points $(p_1, p_2, \dots, p_N)$, $N \ge 3$, and $p_i \ne p_j\ for\ i \ne j$, which in turn defines a sequence of single lines $(sl_1, sl_2, \dots, sl_N)$ where each $sl_i = (p_i, p_{i+1})$ for *i=1,2,…,N-1* and $sl_N = (p_N, p_1)$, and the area within is part of the polygon.

$Polygon(a) \equiv (p_i|1 \le i \le N, N \ge 3, p_i \ne p_j\ for\ i \ne j, sl_i = (p_i, p_{i+1})$
$for\ i = 1,2, \dots, N-1\ and\ sl_N = (p_N, p_1), Meet(sl_i, sl_{i+1}) = true$
$for\ i = 1,2, \dots, N-1, Meet(sl_N, sl_1) = true, Cross(sl_i, sl_j) = false$
$for\ all\ i,j\ where\ i \ne j, and\ area(a) \ne 0)$

6. MultiPoint (*mp*)
MultiPoint can be defined as a set of two or more points.

$MultiPoint(mp) \equiv \{p_i|1 \le i \le N, N \ge 2\}$

7. MultiCurve (*mc*)
MultiCurve can be defined as a set of single line(s) or connected line(s).

$MultiCurve(mc) \equiv \{c_i|c_i = sl \vee c_i = cl; 1 \le i \le N, N \ge 2\}$

8. MultiPolygon (*ma*)
MultiPolygon is defined as a set of two or more polygons.

$MultiPolygon(ma) \equiv \{a_i|1 \le i \le N, N \ge 2\}$

2.2 Formalization of Operation Definitions

As mentioned earlier, we only present the relationships and operations related to lines and points in this paper. We can divide these operation definitions into six different categories depending on the pair of spatial objects.

1. Point and Point
2. Point and Single Line
3. Single Line and Single Line
4. Point and Connected Line
5. Single Line and Connected Line
6. Connected Line and Connected Line

2.2.1 Point and Point

There is only one relationship between point and point.

1. Equal
Two points are equal if and only if they have exatly the same longtitude and latitude respectively.

$Equal(p_1, p_2) \equiv p_1.x = p_2.x \wedge p_1.y = p_2.y$

2.2.2 Point and Single Line

Considering point and single line, there are two possible spatial operations: *Endpoint* and *Ontheline*.

1. Endpoint
Endpoint is a relationship between point and single line. In the formalization, we sometimes need to distinguish unambiguously the two endpoints so we define two operations on a line: *EndpointNe* and *EndpointSw*. North and south will have precedence in distinguishing the endpoints and east and west will be used only if a line is horizontal. That is, the end point of the line with higher latitude will be classified as *EndpointNe* regardless of whether the longitude is either east or west. Figure 6(a) shows some examples of how the endpoints are categorized.

A point p will be an endpoint of single line sl if and only if p is equal to either one of the single line endpoints.

$Endpoint(p, sl) \equiv p = sl.p_1 \vee p = sl.p_2$

$EndpointNe(p, sl) \equiv Endpoint(p, sl) \wedge (\exists p_1)[Endpoint(p_1, sl) \wedge$
$p \ne p_1 \wedge ((sl.p.y > sl.p_1.y) \vee (sl.p.y = sl.p_1.y \wedge sl.p.x > sl.p_1.x))]$

$EndpointSw(p, sl) \equiv Endpoint(p, sl) \wedge (\exists p_1)[Endpoint(p_1, sl) \wedge$
$p \ne p_1 \wedge ((sl.p.y < sl.p_1.y) \vee (sl.p.y = sl.p_1.y \wedge sl.p.x < sl.p_1.x))]$

The relationship between *EndpointNe* and *EndpointSw* can be specified as follows:

$EndpointNe(p_1, sl) \wedge EndpointSw(p_2, sl) \rightarrow$
$[(sl.p_1.y_1 > sl.p_2.y_2) \vee (sl.p_1.y_1 = sl.p_2.y_2 \wedge sl.p_1.x_1 > sl.p_2.x_2)]$

We can now define two unary operations on single line, *slope* and *distance*, as follows (these are used in our specification of some of operations):

$Slope(m, sl) \equiv EndpointNe(p_1, sl) \wedge EndpointSw(p_2, sl) \wedge$
$m = [(p_1.y - p_2.y)/(p_1.x - p_2.x)]$

$Distance(d, sl) \equiv EndpointNe(p_1, sl) \wedge EndpointSw(p_2, sl) \wedge$
$d = [squareroot((p_1.y - p_2.y)^2 + (p_1.x - p_2.x)^2)]$

2. Ontheline
Ontheline (see Figure 6(d)) is a relationship between point and single line. If a point p is on the single line sl then the slope of the point p to one of the endpoints of the single line must be equal to the slope of the single line. In addition, the point p has to fall on the single line. We also have a function to create a single line given two points: *CreateSingleLine(sl,x,y)* where sl is the created single line and x,y are the two endpoints.

$Ontheline(p, sl) \equiv$
$Slope(m_1, sl) \wedge CreateSingleLine(sl_1, p, sl.p_1) \wedge Slope(m_2, sl_1) \wedge$
$m_1 = m_2 \wedge (sl.p_1.x_1 \le p.x \le sl.p_2.x_2 \vee sl.p_2.x_2 \le p.x \le sl.p_1.x_1)$

An endpoint is also considered to be on the line.

$Endpoint(p, sl) \rightarrow Ontheline(p, sl)$

2.2.3 Single Line and Single Line

There are five main relationships between single lines.

1. Equal
Two single lines are equal if and only if they have exatly the same points.

$Equal(sl_1, sl_2) \equiv sl_1.p_1 = sl_2.p_1 \wedge sl_1.p_2 = sl_2.p_2$

2. Meet
Two single lines meet when they share one endpoint. We can further divide the operation into two more cases: *MeetSameSlope* and *MeetDiffSlope* as follows (see Figure 6(b,c)).

$Meet(sl_1, sl_2) \equiv (\exists p \in \{sl_1.p_1, sl_1.p_2, sl_2.p_1, sl_2.p_2\})$
$[Endpoint(p, sl_1) \wedge Endpoint(p, sl_2)]$

$MeetSameSlope(sl_1, sl_2) \equiv Meet(sl_1, sl_2) \wedge Slope(m_1, sl_1) \wedge$
$Slope(m_2, sl_2) \wedge m_1 = m_2$

$MeetDiffSlope(sl_1, sl_2) \equiv Meet(sl_1, sl_2) \wedge Slope(m_1, sl_1) \wedge Slope(m_2, sl_2) \wedge m_1 \neq m_2$

3. Cross

Two single lines cross when their slopes are different and the intersecting point fall on both lines (see Figure 6(e)). To make formalization easier, we will first define *Between* and *ProperBetween* relations as follows. We also have an operation *IntersectPoint(p,sl1,sl2)* that returns the point of intersection *p* between two single lines sl_1, sl_2, if the two lines do not have the same slope. We do not show this because of limited space.

$Between(x, x_1, x_2) \equiv (x_1 \leq x \leq x_2 \vee x_2 \leq x \leq x_1)$

$ProperBetween(x, x_1, x_2) \equiv (x_1 < x < x_2 \vee x_2 < x < x_1)$

$Cross(sl_1, sl_2) \equiv Slope(m_1, sl_1) \wedge Slope(m_2, sl_2) \wedge m_1 \neq m_2 \wedge IntersectPoint(p, sl_1, sl_2) \wedge Between(p.x, sl_1.p_1.x, sl_1.p_2.x) \wedge Between(p.x, sl_2.p_1.x, sl_2.p_2.x)$

$ProperCross(sl_1, sl_2) \equiv Slope(m_1, sl_1) \wedge Slope(m_2, sl_2) \wedge m_1 \neq m_2 \wedge IntersectPoint(p, sl_1, sl_2) \wedge ProperBetween(p.x, sl_1.p_1.x, sl_1.p_2.x) \wedge ProperBetween(p.x, sl_2.p_1.x, sl_2.p_2.x)$

In this operation, we have one special case when the intersecting point is also one of the endpoints of the line. In other words, one of the endpoints of a line lies on the other line. We will call this case of operation *TCross* (see Figure 6(f)).

$TCross(sl_1, sl_2) \equiv (\exists p)[(Endpoint(p, sl_1) \wedge Ontheline(p, sl_2) \wedge \neg Endpoint(p, sl_2)) \vee (Endpoint(p, sl_2) \wedge Ontheline(p, sl_1) \wedge \neg Endpoint(p, sl_1))]$

4. Overlap

Two single lines overlap if and only if their slopes are equal and there is one endpoint lying in another line (see Figure 6(g)).

$Overlap(sl_1, sl_2) \equiv CreateSingleLine(sl_3, sl_1.p_1, sl_2.p_2) \wedge Slope(m_3, sl_3) \wedge Slope(m_1, sl_1) \wedge Slope(m_2, sl_2) \wedge m_1 = m_2 = m_3 \wedge (\exists p)[\big(Endpoint(p, sl_1) \wedge Between(p.x, sl_2.p_1.x, sl_2.p_2.x)\big) \vee (Endpoint(p, sl_2) \wedge Between(p.x, sl_1.p_1.x, sl_1.p_2.x))]$

5. Within

For *Within* operation, we can divide it into two more types: *CompleteWithin* and *SharedEndpointWithin* (see Figure 6(h,i)).

$CompleteWithin(sl_1, sl_2) \equiv CreateSingleLine(sl_3, sl_1.p_1, sl_2.p_2) \wedge Slope(m_3, sl_3) \wedge Slope(m_1, sl_1) \wedge Slope(m_2, sl_2) \wedge m_1 = m_2 = m_3 \wedge (\forall p)[\big(Endpoint(p, sl_1) \wedge Between(p.x, sl_2.p_1.x, sl_2.p_2.x)\big) \vee (Endpoint(p, sl_2) \wedge Between(p.x, sl_1.p_1.x, sl_1.p_2.x))]$

$SharedEndpointWithin(sl_1, sl_2) \equiv CreateSingleLine(sl_3, sl_1.p_1, sl_2.p_2) \wedge Slope(m_3, sl_3) \wedge Slope(m_1, sl_1) \wedge Slope(m_2, sl_2) \wedge m_1 = m_2 = m_3 \wedge (\exists p_1, p_2)[\big(Endpoint(p_1, sl_1) \wedge Endpoint(p_1, sl_2)\big) \wedge \big(\big(Endpoint(p_2, sl_1) \wedge Between(p_2.x, sl_2.p_1.x, sl_2.p_2.x)\big) \vee (Endpoint(p_2, sl_2) \wedge Between(p_2.x, sl_1.p_1.x, sl_1.p_2.x))\big)]$

Within operation is also considered as *Overlap* and *SharedEndpointWithin* can be considered as *CompleteWithin*.

$CompleteWithin(sl_1, sl_2) \vee SharedEndpointWithin(sl_1, sl_2) \rightarrow Overlap(sl_1, sl_2)$

$SharedEndpointWithin(sl_1, sl_2) \rightarrow CompleteWithin(sl_1, sl_2)$

Meet, Cross, Overlap, and *Within* operations are also considered as *Intersect*.

$Meet(sl_1, sl_2) \vee Cross(sl_1, sl_2) \vee Overlap(sl_1, sl_2) \vee Within(sl_1, sl_2) \rightarrow Intersect(sl_1, sl_2)$

For the remaining sections, we list all the operations, *but give formalization for only some due to space limitations*. The full formalization is in [17].

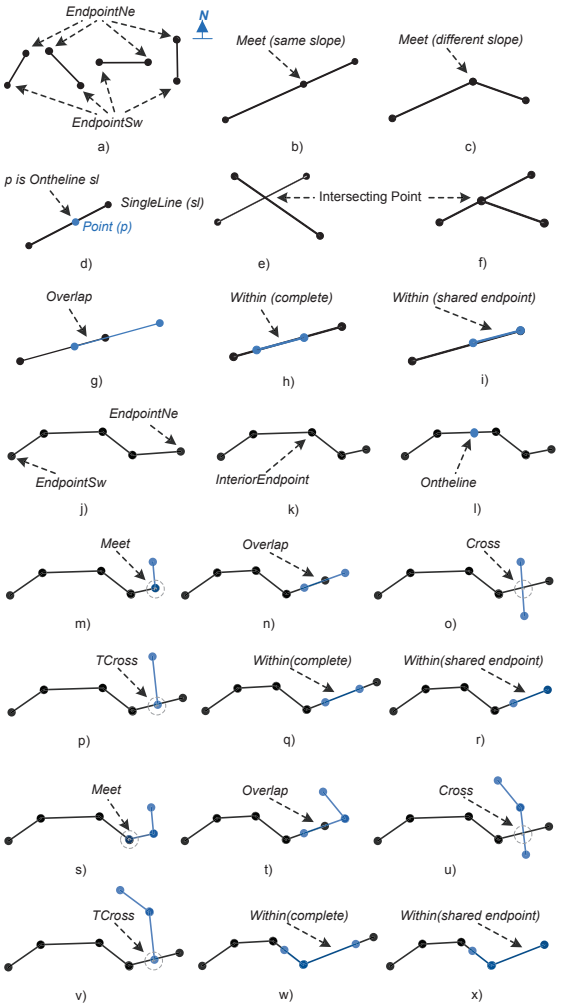


Figure 6. Types of spatial operations

2.2.4 Point and Connected Line

In the following, we will discuss the formalization of spatial operations between point and connected line. Considering point and connected line, there are three possible spatial operations: *Endpoint*, *InteriorEndpoint* and *Ontheline*.

1. **Endpoint** (see Figure 6(j))
2. **InteriorEndpoint**

InteriorEndpoint (see Figure 6(k)) is another relationship between point and connected line when a point falls in the joint between two single lines of connected line.

$InteriorEndpoint(p, cl) \equiv (\exists p_1 \in \{cl.p_i; i \neq 1, N\})[p = p_1]$

3. **Ontheline** (see Figure 6(l))

2.2.5 Single Line and Connected Line

1. **Meet** (see Figure 6(m))
2. **Cross**

A single line *sl* and connected line *cl* cross if and only if there is a single line sl_i of connected line *cl* crossing the single line *sl* (see Figure 6(o)).

$Cross(sl, cl) \equiv (\exists sl_1 \in cl.sl_i)[Cross(sl, sl_1) \vee (TCross(sl, sl_1) \wedge (\exists p_1)[Endpoint(p_1, sl_1) \wedge Ontheline(p_1, sl) \wedge \neg Endpoint(p_1, cl)])]$

3. **TCross** (see Figure 6(p))
4. **Overlap** (see Figure 6(n))
5. **Within:**

5.1 CompleteWithin (see Figure 6(q))
5.2 SharedEndpointWithin (see Figure 6(r))

2.2.6 Connected Line and Connected Line

1. **Equal**
2. **Meet** (see Figure 6(s))
3. **Cross** (see Figure 6(u))
4. **TCross** (see Figure 6(v))
5. **Overlap**

A connected line cl_1 and connected line cl_2 overlap when their single lines are overlapped (see Figure 6(t)).

$$Overlap(cl_1, cl_2) \equiv$$
$$(\exists sl_1 \in cl_1.sl_i, sl_2 \in cl_2.sl_j)[Overlap(sl_1, sl_2) \vee Equal(sl_1, sl_2)]$$

6. **Within:**

6.1 CompleteWithin (see Figure 6(w))
6.2 SharedEndpointWithin (see Figure 6(x))

3. OWL/Protégé Realization

Protégé [30] is a well-known and widely-used open-source platform for ontology management including creation, visualization, and manipulation of ontology [15]. Moreover, Protégé is also user-friendly and domain-customizable because it allows user-defined or imported plug-ins.

3.1 Adding Spatial Dimension to Ontology

We use our formal specification of spatial ontology and adopt one of the approaches proposed in [22] for adding temporal dimension to existing ontologies. Figure 7 gives an overview of our approach.

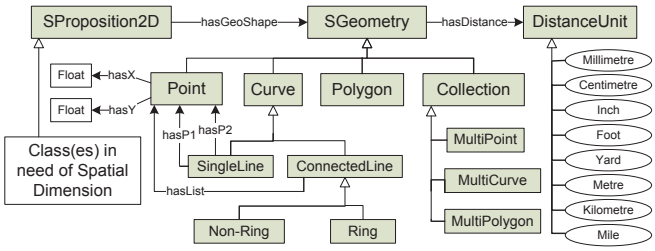


Figure 7. Method to add spatial dimension to existing ontolgies

In Figure 7, SProposition2D class has *hasGeoShape* relationship with SGeometry class, which contains geometry types: point, single line, non-ring connected line, and ring connected line. Point consists of two numbers: longitude and latitude. Single line consists of exactly two points. Connected line consists of three points or more. There are two types of connected lines: non-ring and ring. *hasDistance* is a relationship between SGeometry class and DistanceUnit class. DistanceUnit contains the units of distance measurements such as km., mile, yard, etc.

The principle is that any class in need of spatial dimension will be added as a subclass of SProposition2D class. Consequently, the class will automatically have *hasGeoShape* property that enables a class entity to be modeled as one of the spatial data types.

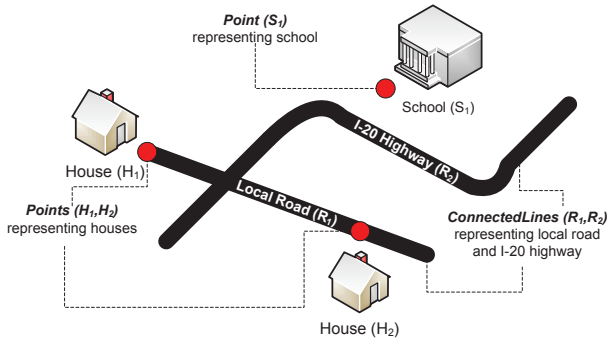


Figure 8. An example of how entities are modeled in spatial dimension

An example is shown in Figure 8. Suppose we have a yellow page ontology which contains contact information of individuals, businesses, parks, train rails, roads and highways, etc. We would like to add a spatial dimension to it. As a result, we will add house, park, train rail, road and highway classes to be subclasses of SProposition2D class as shown in Figure 7 so that those classes will have spatial features of one of the spatial data types. In this example, houses and schools are modeled as points, roads and highways are modeled as connected lines, and parks are modeled as polygons. The choice of geometry type depends on the application scenarios.

In Protégé, we define a point by two functional datatype properties, *hasX* and *hasY* of type float. A point is used to model an entity which does not have area. A single line is defined by exactly two distinct points. *hasP1* and *hasP2* are functional object properties of a single line. A connected line is defined as a list of three or more ordered points. *hasList* is a functional datatype property of a connected line, of type string, which has a following string pattern.

$$\{n, p_1, p_2, ..., p_n\}$$

where n is number of points, $n \geq 3$ and p_i is point i.

These string patterns will be eventually parsed by Protégé built-ins into a number of point instances.

3.2 Developing Spatial Built-ins in Protégé

According to Figure 3, for spatial reasoning, inferencing, and querying in Protégé, we develop the spatial operations as Protégé built-ins based on our formalization of spatial operations. We implemented 33 major spatial operations along wth additional 6 utility built-ins in Protégé as shown in Table 1 and 2.

Spatial built-ins can be divided into six categories depending on the combinations of geometry data types: point versus point, point versus single line, point versus connected line, single line versus single line, single line versus connected line, and connected line versus connected line. At this point, polygon data type is left for future work.

In our notation, 1 represents point, 2 represents single line, and 3 represents connected line (non-ring and ring), which are appended to the operation (built-in) name. When defining a spatial rule in Protégé, the built-ins have to be specified as the following pattern:

```
spatial:<BuiltinName>XY(<GeoTypeX>,<GeoTypeY>)
```

where X and Y are ordered number corresponding to geometry data types.

Table 1. 33 major spatial built-ins

Spatial Built-ins	Point(1)	Single Line(2)	Connected Line(3) [Non-Ring & Ring]
Point(1)	Equal	Endpoint EndpointNe EndpointSw Ontheline	Endpoint* EndpointNe* EndpointSw* Ontheline InteriorPoint
Single Line(2)		Equal Meet Cross TCross Overlap Within: - Complete - SharedEndpoint Intersect	Meet* Cross TCross Overlap Within: - Complete - SharedEndpoint* Intersect
Connected Line(3) [Non-Ring & Ring]			Equal Meet* Cross TCross Overlap Within: - Complete - SharedEndpoint* Intersect

*Operations apply only on non-ring connected lines.

Table 2. 6 additional utility built-ins

Additional Built-ins	Slope, IntersectPoint, Between, ProperBetween, Distance (in km.), CreateSingleLine

For example, to define an *Endpoint* built-in of point and single line, we specify:

```
spatial:endpoint12(<point>,<single line>)
```

This is because different combinations of geometry data types have different ways of implementing the same operation. As a result, we need to indicate the exact operation we are using.

3.3 Spatial Reasoning, Inferencing and Querying

Once we have a spatial dimension added to an ontology and have spatial built-ins ready in Protégé, we can do spatial reasoning, inferencing and querying.

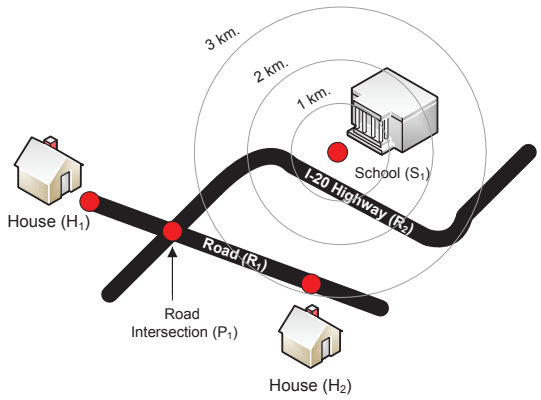


Figure 9. An example of how spatial operations can be used in reasoning, inferencing and querying

We enable spatial reasoning by defining rules in SWRL Tab [30] and we use the Jess® [27] rule engine to perform inferencing on those rules. Spatial built-ins that we developed can be combined with other built-in libraries such as SWRLB [16], TEMPORAL [22], and SQWRL [21], to create complicated rules or queries [21].

The following example shows how spatial built-ins can be used in reasoning and inferencing. Suppose we have 2 houses, 1 school and 2 roads as illustrated in Figure 9. We would like to define a rule regarding a school zone. Suppose, we define a location in a school zone as a point within 3 kilometers radius centered at a school. This definition can be implemented in Protégé by the following rule:

$$Point(?p) \wedge hasX(?p,?px) \wedge hasY(?p,?py) \wedge School(?s) \wedge hasGeoShape(?s,?ps) \wedge hasX(?ps,?sx) \wedge hasY(?ps,?sy) \wedge spatial{:}\,distance(?d,?px,?py,?sx,?sy) \wedge swrlb{:}\,lessthan(?d,3) \rightarrow SchoolZonePoints(?p)$$

For the next example, we would like to define highway-connected local roads as local roads that intersect with any highway at some point. The corresponding rule is:

$$Roads(?r) \wedge hasGeoShape(?r,?clr) \wedge hasList(?clr,?rl) \wedge Highways(?h) \wedge hasGeoShape(?h,?clh) \wedge hasList(?clh,?hl) \wedge spatial{:}\,Intersect33(?rl,?hl) \rightarrow HighwayConnectedLocalRoads(?r)$$

When we use Jess®, the above rules will classify points *?p* and roads *?r* as SchoolZonePoints and HighwayConnectedLocalRoads respectively if they satisfy the rules above.

4. CONCLUSION AND FUTURE WORK

4.1 Conclusion

In conclusion, this paper presents a spatial ontology formalization and a method to add spatial dimension to existing ontologies. Then, we develop spatial built-ins to be used for spatial reasoning in Protégé. To process reasoning and inferencing we use SWRL Tab and Jess® rule engine. To do querying, we use built-ins from SQWRL library [21] along with SQWRL Query Tab [30]. With all components presented in this paper, they allow us to add spatial dimension to ontologies through the Protégé framework and make them able to capture and reason about spatial concepts.

4.2 Future work

This paper is an initial process which does not include the polygon data type. In the future, we will add a spatial polygon data type and also develop spatial built-ins for it in Protégé. In addition, we will continue to define formalizations of distance and area measurements. Finally, we also would like to develop a technique by which the spatial ontology formalization can play a role as a medium to convert spatial concepts from one system to another.

5. ACKNOWLEDGEMENTS

The authors would like to thank Martin O'Connor for his help with Protégé issues. This work was conducted using the Protégé resource, which is supported by grant LM007885 from the United States National Library of Medicine. The authors also would like to thank Sandia National Laboratories for providing Jess® [27] for this work. (Jess is not available for Open Source licensing under any GPL license. To license and download Jess, please visit the Jess website at: www.jessrules.com).

6. REFERENCES

[1] Abdelmoty, A.I., Smart P.D., El-Geresy, B.A. and Jones, C.B. 2009. Supporting Frameworks for the Geospatial Semantic Web. In *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*. SSTD'09. LNCS 5644, Springer, 355-372.

[2] Allen, J.F. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM 26*, 11 , 832-843.

[3] Allen, J.F. and Kautz, H.A. 1985. A Model of Naïve Temporal Resoning. In *Formal Theories of the Commonsense World*. Ablex Pub, 251-268.

[4] Andronikos, T., Michalis, S. and Ioannis, P. 2009. Adding Temporal Dimension to Ontologies via OWL Reification. In *Proceedings of the 13th Panhellenic Conference on Informatics*. PCI'09. IEEE Computer Society Wachington, DC, 19-22.

[5] Baglioni, M.,Masserotti, M.V., Renso,C. and Spinsanti, L. 2007. Building Geospatial Ontologies from Geographical Databases. In *Proceedings of the 2nd International Conference on GeoSpatial Semantics*. GeoS'07. LNCS 4853, Springer, 195-209.

[6] Baratis, E., Petrakis, E.G.M., Batsakis, S., Maris, N. and Papadakis, N. 2009. TOQL: Temporal Ontology Querying Language. In *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*. SSTD'09. LNCS 5644, Springer, 338-354.

[7] Bechhofer, S., Horrocks, I. and Patel-Schneider, P.F. 2003. Tutorial on OWL. Retrieved March 25, 2011, from: http://www.cs.man.ac.uk/~horrocks/ISWC2003/Tutorial/.

[8] Bennacer, N., Aufaure, M., Cullot, N., Sotnykova, A. and Vangenot, C. 2004. Representing and Reasoning for Spatiotemporal Ontology Integration. In *LNCS 3292*. Springer, 30-31.

[9] Bittner, T., Donnelly, M. and Smith, B. 2006. A Spatio-Temporal Ontology for Geographic Information Integration. *International Journal of Geographical Information Science0(0)*, 1-29.

[10] Gruber, T. 2008. Ontology. In *Encyclopedia of Database System*, L. Liu and M.T. Özsu, Eds. Springer.

[11] Grutter, R. and Bauer-Messmer, B. 2007. Combining OWL with RCC for spatioterminological reasoning on environmental data. In *Proceedings of the 3rd International Workshop OWL: Experiences and Directions*. OWLED'07. CEUR-WS.org.

[12] Hess, G.N., Iochpe, C. and Castano, S. 2006. An Algorithm and Implementation for GeoOntologies Integration. In *Proceedings of the 8thBrazillian Symposium on GeoInformatics*. Adv. in Geoinformatics, Springer, 129-140.

[13] Hobbs, J.R. and Pan, F. 2004. An Ontology of Time for the Semantic Web. In *ACM Transactions on Asian Language Information Processing (TALIP) 3(1)*, 66-85.

[14] Hogenboom, F., Borgman, B., Frasincar, F. and Kaymak, U. 2010. Spatial Knowledge Representation on the Semantic Web. In *Proceedings of the 4th IEEE International Conference on Semantic Computing*. ICSC'10, 252-259.

[15] Horridge, M., Knublauch, H., Rector, A., Stevens, R. and Wroe, C. 2004. A Practical Guide to Building OWL Ontologies Using the Protégé-OWL Plugin and CO-ODE Tools Edition 1.0. University of Manchester.

[16] Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B. and Dean, M. 2004. SWRL: A Semantic Web Rule Language. Retrieved May 19, 2011, from World Wide Web Consortium (W3C): http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

[17] Jitkajornwanich, K., Elmasri, R., Li, C. and McEnery, J. 2011. Formalization of 2-D Spatial Ontology and OWL/Protégé Realization. Technical Report. CSE-2011-6, University of Texas at Arlington.

[18] Mark, D., Egenhofer, M., Hirtle, S. and Smith, B. 2000. Ontological Foundations for Geographic Information Science. University Consortium for Geographic Information Science (UCGIS)

[19] Milea, V., Fransincar, F. and Kaymak, U. 2009. A Temporal Web Ontology Language. Erasmus Research Institute of Management (ERIM).

[20] Noy, N.F. and McGuinness, D.L. 2001. Ontology Development 101: A Guide to Creating Your First Ontology. Study Guide. Stanford University.

[21] O'Connor, M.J. and Das, A.K. 2009. SQWRL: A Query Language for OWL. In *Proceeding of OWL: Experiences and Directions (OWLED'09)*. CEUR Workshop Proceedings 529.

[22] O'Connor, M.J. and Das, A.K. 2010. A Method for Representing and Querying Temporal Information in OWL. *Communications in Computer and Information Science 127*, 97-110.

[23] Open GIS Consortium. 1999. OpenGIS Simple Features Specification For SQL. Open Geospatial Consortium, Inc.

[24] Parent, C., Spaccapietra, S. and Zimanyi, E. 1999. Spatio-Temporal Conceptual Models: Data Structures + Space + Time. In*Proceedings of the 7th ACM international symposium on Advances in geographic information systems*. ACM DL, 26 - 33.

[25] Parent, C., Spaccapietra, S. and Zimanyi, E. 2006. *Conceptual Modeling for Traditional and Spatio-Temporal Applications: The MADS Approach*. Springer.

[26] Randell, D.A., Cui, Z. and Cohn, A.G. 1992. A Spatial Logic Based on Regions and Connection. In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning*. KR'92. Morgan Kaufmann, 165-176.

[27] Sandia National Laboratories. Jess®: The Rule Engine for the Java™ Platform. Retrieved Jan 1, 2011, from: http://www.jessrules.com/jess/index.shtml.

[28] Shekhar, S. and Chawla, S. 2003. *Spatial Databases: A Tour*. Pearson Education, New Jersey.

[29] Spaccapietra, S., Cullot, N., Parent, C. and Vangenot, C. 2004. On Spatial Ontologies. Database Laboratory, Swiss Federal Institute of Technology, Lausanne, Switzerland

[30] Stanford Center for Biomedical Informatics Research. 2011. Protégé Project. Retrieved May 20, 2011, from Protégé: http://protege.stanford.edu.

[31] Wang, X., Zhou, X. and Lu, S. 2000. Spatiotemporal Data Modeling and Management: Survey. In *Proceedings of the 36th International Conference on Technology of Object-Oriented Languages and Systems*. IEEE Xplore, 202-211.