Shapes as property restrictions and property-based similarity

Silvia LIKAVEC [1]
Dipartimento di Informatica, Unviersità di Torino, Italy

Abstract. In this work we look into the details of modeling shapes in an ontology as property restrictions on classes. In this way shapes do not have to be categorized in an exhaustive hierarchy and there is no need to take immediate decisions on how to group the objects, it is rather possible to individualize some important characteristics of shapes and use them as the basis for their categorization and comparison. This approach also makes it possible to use the adequate similarity measure based on properties which helps find similar shapes in different contexts, depending on the relevance of the properties in the particular situation.

Keywords. ontology, OWL, shapes, properties, restrictions, similarity

Introduction

In many different areas, from virtual reality to architectural design, from biology to medicine, from mathematics to computer science, the notion of shape plays a crucial role. Finding patterns and forms in objects surrounding us, describing them and understanding their interaction is essential to human nature. Hence, varied approaches to the categorization of shapes and forms, as well as their mutual similarity and connectedness, are of great importance for the development of many scientific fields.

In various domains, there is a raising tendency to use ontologies as powerful formalisms for knowledge representation with associated reasoning mechanisms (inheritance, subsumption, classification etc.). Ontologies provide explicit specifications of domain concepts and relationships that exist between them [1]. They guarantee exact semantics for each statement and avoid semantic ambiguities. Their usage allows for extensibility and re-usability, since they are expressed with standard formats and technologies.

In the domain of shape, form and structure representation, there were some attempts at modeling shapes ontologically, as an exhaustive hierarchy. In [9] the authors develop a limited graphics ontology for natural language interfaces, covering the concepts like "Shape", "Action" and various features which describe shapes (size, color, position etc.). In [5] two first-order ontologies for representing 2D and 3D shapes like surfaces and boxes are introduced. They use only the notions such as part-of and connectedness, rather than Euclidean geometric relations, such as alignment and length of segments, or the

[1]Corresponding Author: Silvia Likavec, Dipartimento di Informatica, Corso Svizzera 185, 10149 Torino, Italy; E-mail: likavec@di.unito.it.

notions of curvature or surface area. In the domain of architectural design, [8] presents a conceptual "building shape ontology" which sorts building shapes and captures their meaning and semantics.

In the realm of spatial design and reasoning, authors in [2] explore the role ontological formalization plays in modeling of high-level conceptual requirement constraints. They concentrate on ontological modeling of structural forms from different perspectives. As for architectural design, information that is being used often originates from various sources. In [6], the authors take a step towards integration of various aspects of architectural domain (spatial constraints, relations among objects, abstract conceptualizations) designing modular ontologies based on the theory of ε-connections. [13] describes a method for the retrieval of 3-dimensional shapes (in this case furniture models) based on a mapping between low level features described by the shape descriptor and ontology concepts. This furniture ontology is used in annotation and key word based retrieval of furniture models.

As far as similarity among ontological concepts is concerned, three main approaches can be distinguished. The first one [12] is based on information content of a class in an IS-A taxonomy, given by the negative logarithm of the probability of occurrence of the class in a text corpus. The second approach [11] uses the ontology graph structure, by measuring directly the distance between nodes (usually, the number of edges or the number of nodes that need to be traversed in order to reach one node from the other). Finally, the third approach combines the information content approach with edge counting based approach (see for example [7]). Different notions of similarity and the relationships among them are tackled in [3]. Starting form Leibnizian relative identity as the only local form of similarity, they show that more sophisticated notions can be obtained by applying transformations across heterogeneous logics. They also distinguish between ontological and epistemic similarities. While ontological similarities stem from the structure of the world itself, epistemic similarities are used to connect entities in different worlds.

The rest of the paper is organized as follows. Section 1 provides a summary of the treatment of properties in OWL and the definition of property restrictions on classes. In Section 2 we discuss some issues concerning modeling of shapes as property restrictions on classes, followed by some examples of shape definitions. Details of the approach to calculating similarity of shapes based on properties can be found in Section 3. Section 4 concludes.

1. Properties and property restrictions in OWL

1.1. Properties in OWL

In different contexts, domain knowledge can be represented semantically using ontologies expressed in OWL[2]. In an ontology, domain concepts are organized hierarchically and have their features defined as properties. Two kinds of properties can be distinguished in OWL: (i) *object properties* relating individuals among themselves and (ii) *data type properties* relating individuals to data type values.

Characteristics of a property are defined with a property axiom, most commonly defining its domain and range. rdfs:domain links a property to a class description,

[2] http://www.w3.org/TR/owl-ref

whereas rdfs:range links a property to either a class description or a data range. For example:

```
<owl:ObjectProperty rdf:ID="has_curvature">
  <rdfs:domain rdf:resource="#Shape"/>
  <rdfs:range rdf:resource="#Curvature "/>
</owl:ObjectProperty>
```

defines the property *has_curvature* which ties the elements of Shape class to the elements of Curvature class.

Equivalent properties are defined with owl:equivalentProperty.

In the following section we will see how properties are used in OWL to define classes with property restrictions.

1.2. Defining classes with property restrictions

Properties are used in OWL for defining classes with property restrictions, by means of *local anonymous classes*, which are collections of individuals all satisfying certain restrictions on certain properties. Two kinds of property restrictions exist: *value constraints* and *cardinality constraints*. A value constraint concerns constraints on the range of the property when applied to a particular class description. A cardinality constraint imposes constraints on the number of values a property can take, in the context of a particular class description.

We start with the brief description of value constraints. There are three ways of defining value constraints:

- owl:allValuesFrom defines a class for which all the values of the given property are either members of the specified class or data values within the specified data range. It is possible not to have any values for the given property. For example:

  ```
  <owl:Restriction>
    <owl:onProperty rdf:resource="#has_angle" />
    <owl:allValuesFrom rdf:resource="#RightAngle"  />
  </owl:Restriction>
  ```

  describes an anonymous OWL class of all individuals for which the has_angle property only has values of the class RightAngle (for example square or rectangle). In predicate logic, the counterpart of owl:allValuesFrom constraint is the universal quantifier, i.e. for each instance of the class defined with the restriction, every value for the property must fulfill the constraint and the constraint is trivially satisfied for an instance that has no value for the specified property.
- owl:someValuesFrom specifies a class for which at least one of the values of the given property is either a member of the specified class or a data value within the specified data range (at least one must exist). There might be other values for the given property. For example:

  ```
  <owl:Restriction>
    <owl:onProperty rdf:resource="#has_angle" />
    <owl:someValuesFrom rdf:resource="#RightAngle" />
  </owl:Restriction>
  ```

describes an anonymous OWL class of all individuals which have at least one right angle (for example right-angled triangle). In predicate logic, the counterpart of owl:someValuesFrom constraint is the existential quantifier, i.e. for each instance of the class defined with the restriction, there exists at least one value for the property that fulfills the constraint.

- owl:hasValue defines a class for which the specified property has at least one value semantically equal to the specified value, which can be either an individual or a data value[3]. For example:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#contains" />
  <owl:hasValue rdf:resource="#Circle100" />
</owl:Restriction>
```

describes an anonymous OWL class which contains a specific circle.

On the other hand, cardinality constraints can be expressed by using one of the following three constraints:

- owl:maxCardinality describes a class with at most *max* semantically distinct values for the specified property (*max* being the value of the cardinality constraint). For example

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#has_number_of_edges" />
  <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">5
  </owl:maxCardinality>
</owl:Restriction>
```

describes an anonymous OWL class of individuals that have at most five edges (for example polygons with 3 or 4 or 5 edges),.

- owl:minCardinality is defined analogously to maxCardinality, where the class has at least *min* semantically distinct values.
- owl:cardinality is defined analogously to maxCardinality, where the class has exactly *m* semantically distinct values. It is actually a redundant concept, since it can be defined with the combination of maxCardinality and minCardinality.

From the above we can see that we can consider each of the concepts in our ontology, to have certain properties defined for it. These properties further describe the concepts in the ontology and can be used to categorize them and to calculate their mutual similarity.

1.3. Instances and their properties

An instance in the ontology is defined with individual axioms called “facts” which describe its class membership, property values and individual identity. An instance is related to the class it belongs to directly with the rdfs:type relation and basically inherits

[3]For datatypes “semantically equal” means that the lexical representation of the literals maps to the same value. For individuals it means that they either have the same URI reference or are defined as being the same individual with owl:sameAs.

the properties of the class it belongs to. Hence, in OWL the properties of the instances are defined by associating to each property its specific value. For example, the following describes a red circle with the radius equal to 3 and a dotted outline.

```
<Circle rdf:ID="Circle100">
  <has_radius rdf:datatype="&xsd;positiveInteger">3</has_radius>
  <has_outline rdf:resource="#Dotted"/>
  <has_color rdf:resource="#Red"/>
</Circle>
```

2. Shapes defined as class restrictions

Categorization of the world around us is inherent to human perception and reasoning. Many objects are internalized easier if they are reduced to simpler forms and shapes that we are familiar with and that we can easily group with other similar objects.

We give some directions on how to model two-dimensional shapes, since they are the easiest to comprehend and understand.[4] The most common categorization of two-dimensional shapes is according to the kind of edges the shapes contain to curved shapes and straight line composed shapes (polygons). But another categorization could start from convex and concave shapes. Or we might want to categorize the shapes based on the number of edges they have. The possibilities are many.

So instead of forcing this somehow artificial categorization upon the shape world, we would do the shapes more justice by defining them as property restrictions on classes. We can start by defining many different properties which would help us precisely describe the shapes we need. For example, the property *has_edge_kind* could be used to define as property restrictions the classes `CurvedShape` and `Polygon` (shapes composed from straight lines), whereas the property *has_curvature* would be used to distinguish `ConvexShape` class from `ConcaveShape` class (again defining them as restrictions). In this way, there is no need to a-priori decide which categorization should happen higher up in the hierarchy, they can peacefully co-exist together (and not be the only ones). Once the first level is modeled, we can proceed to model their subclasses. At this point we can have direct subclasses with additional properties or additional restrictions. So we can include properties like *number_of_edges*, *number_of_equal_edges*, *number_of_parallel_edges* etc. This would also help us compare the shapes having all these properties defined explicitly for them.

In this light, let us have a look at two shape definitions, namely rhombus and rectangle. A rhombus can be defined as a simple (non-self-intersecting) quadrilateral with four equal edges, whereas a rectangle can be defined as quadrilateral with four right angles (and they are both convex). So if we define the `Quadrilateral` class as a subclass of `Polygon` class which has the property *has_number_of_sides* restricted to 4, we can define rhombus and rectangle as follows:

```
<owl:Class rdf:ID="Rectangle">
  <rdfs:subClassOf rdf:resource="#Quadrilateral"/>
  <rdfs:subClassOf rdf:resource="#ConvexShape"/>
```

[4] Three-dimensional and n-dimensional objects are treated similarly.

```
  <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty rdf:resource="#has_angle" />
     <owl:allValuesFrom rdf:resource="#RightAngle"  />
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has_number_of_angles"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">4
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:ID="Rhombus">
  <rdfs:subClassOf rdf:resource="#Quadrilateral"/>
  <rdfs:subClassOf rdf:resource="#Convex"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#has_number_of_equal_edges"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">4
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Obviously, these are not the only ways to define these two, or any other shape. The process is versatile and applicable in many different contexts. Above all, it enables very natural comparison of shapes and establishes their similarity based on properties, as we will see in the following section.

3. Property-based similarity of shapes

If we define shapes as property restrictions (on values and cardinality), we can find similar shapes by comparing their properties. The property-based similarity of two shapes S_1 and S_2, can be calculated by starting from Tversky's feature-based model of similarity [14], where similarity between objects is a function of both their common and distinctive features:

$$\text{SIM}_T(S_1, S_2) = \frac{\alpha(\psi(S_1) \cap \psi(S_2))}{\beta(\psi(S_1) \setminus \psi(S_2)) + \gamma(\psi(S_2) \setminus \psi(S_1)) + \alpha(\psi(S_1) \cap \psi(S_2))}. \tag{1}$$

Here $\psi(S)$ is the function which describes all the relevant features of S, and $\alpha, \beta, \gamma \in \mathbb{R}$ are parameters which permit us to treat differently the various components. For $\alpha = 1$ maximal importance is assigned to the common features of the two shapes and for $\beta = \gamma$ non-directional similarity measure is achieved. We will use $\alpha = \beta = \gamma = 1$.

Hence, to be able to use Tversky's model we need to calculate the following:

- *common features of* S_1 *and* S_2: $\text{CF}(S_1, S_2) = \psi(S_1) \cap \psi(S_2)$,
- *distinctive features of* S_1: $\text{DF}(S_1) = \psi(S_1) \setminus \psi(S_2)$ and
- *distinctive features of* S_2: $\text{DF}(S_2) = \psi(S_2) \setminus \psi(S_1)$.

Putting these values into the formula (1) and taking $\alpha = \beta = \gamma = 1$ we obtain:

$$\text{SIM}_T(S_1, S_2) = \frac{\text{CF}(S_1, S_2)}{\text{DF}(S_1) + \text{DF}(S_2) + \text{CF}(S_1, S_2)}. \tag{2}$$

In order to calculate common and distinctive features for S_1 and S_2, for each property p, we calculate CF_p, DF_p^1 and DF_p^2, which denote how much the property p contributes to common features of S_1 and S_2, distinctive features of S_1 and distinctive features of S_2, respectively. In what follows we would see how different ways of defining properties in OWL influence the calculation of these values. We consider equal the properties defined with owl:EquivalentProperty.

We start from three kinds of value restriction declarations: (i) owl:allValuesFrom; (ii) owl:someValuesFrom; (iii) owl:hasValue. Based on how the restrictions on properties are defined for S_1 and S_2, we can distinguish the following six cases:

1. The property p is defined with owl:allValuesFrom for both S_1 and S_2. Let the property p be defined for S_1 with
   ⟨owl:allValuesFrom rdf:resource="#A_1"⟩
   and for S_2 with
   ⟨owl:allValuesFrom rdf:resource="#A_2"⟩.
   Let a_1 (resp. a_2) be the number of sub-classes of A_1 (resp. A_2). If A_1 and A_2 are equal classes or declared equivalent with owl:equivalentClass or A_1 is a subclass of A_2, then $\text{CF}_p = \frac{1}{(a_1+1)}$. Otherwise $\text{DF}_p^1 = \frac{1}{a_1+1}$ and $\text{DF}_p^2 = \frac{1}{a_2+1}$.

2. The property q is defined with owl:someValuesFrom both for S_1 and S_2. Let the property q be defined for S_1 with
   ⟨owl:someValuesFrom rdf:resource="#B_1"⟩
   and for S_2 with
   ⟨owl:someValuesFrom rdf:resource="#B_2"⟩.
   Let b_1 (resp. b_2) be the number of sub-classes of B_1 (resp. B_2) and w be the number of classes in the whole domain. If B_1 and B_2 are equal classes or declared equivalent with owl:equivalentClass or B_1 is a subclass of B_2, then $\text{CF}_q = \frac{1}{(b_1+1)w}$. Otherwise $\text{DF}_q^1 = \frac{1}{(b_1+1)w}$ and $\text{DF}_q^2 = \frac{1}{(b_2+1)w}$.

3. Let property r be defined for S_1 with
   ⟨owl:hasValue rdf:resource="#V_1"⟩
   and for S_2 with
   ⟨owl:hasValue rdf:resource="#V_2"⟩
   If V_1 and V_2 are the same values or declared same with owl:sameAs, then $\text{CF}_q = 1$. Otherwise $\text{DF}_q^1 = 1$ and $\text{DF}_q^2 = 1$.

4. If the property t is defined for S_1 with
   ⟨owl:hasValue rdf:resource="#V_3"⟩
   and for S_2 with

⟨owl:allValuesFrom rdf:resource="#A_3"⟩,

and if V_3 is an instance of A_3 or one of its subclasses, then $\text{CF}_t = 1$. Otherwise, if a_3 is the number of sub-classes of A_3, then $\text{DF}_t^1 = 1$ and $\text{DF}_t^2 = \frac{1}{a_3 + 1}$.

5. If the property x is defined for S_1 with

   ⟨owl:hasValue rdf:resource="#V_4"⟩

   and for S_2 with

   ⟨owl:someValuesFrom rdf:resource="#B_3"⟩

   and if V_4 is an instance of B_3 or one of its subclasses, then $\text{CF}_x = 1$. Otherwise, if b_3 is the number of sub-classes of B_3 and if w is the number of classes in the whole domain, then $\text{DF}_x^1 = 1$ and $\text{DF}_x^2 = \frac{1}{(b_3 + 1)w}$.

6. If the property y is defined for S_1 with

   ⟨owl:allValuesFrom rdf:resource="#A_4""⟩

   and for S_2 with

   ⟨owl:someValuesFrom rdf:resource="#B_4""⟩

   and if a_4 (resp. b_4) is the number of sub-classes of A_4 (resp. B_4) and w is the number of classes in the whole domain, then $\text{CF}_y = \frac{1}{(a_4 + 1)(b_4 + 1)w}$, $\text{DF}_y^1 = \frac{1}{a_4 + 1}$ and $\text{DF}_y^2 = \frac{1}{(b_4 + 1)w}$.

Next we consider three kinds of cardinality restriction declarations: (i) minCardinality; (ii) maxCardinality; (iii) cardinality. We can distinguish the following cases:

1. If the property f is defined with owl:maxCardinality for both S_1 and S_2, and it has value m in S_1 and value n in S_2, where $m \leq n$, then $\text{CF}_f = \frac{1}{n-1}$, $\text{DF}_f^1 = 0$ and $\text{DF}_f^2 = n - m$. The case when $m \geq n$ is analogous.
2. If the property g is defined with owl:minCardinality for both S_1 and S_2, the values for this restriction would not contribute to common and distinctive features, since each of these restrictions can have infinitely many values. It only contributes to similarity calculation if it is declared together with owl:maxCardinality restriction, which is then the following case.
3. If the property h is defined with owl:cardinality for both S_1 and S_2, and it has value m in S_1 and value n in S_2, then if $m = n$ $\text{CF}_f = 1$. Otherwise, if $m < n$ then $\text{DF}_f^1 = 0$ and $\text{DF}_f^2 = n - m$. The case when $m > n$ is analogous.

Of course, the subclass relation should be taken into account, hence providing each class with the property definitions inherited from parent classes.

Finally, to calculate all common and distinctive features of S_1 and S_2 we repeat the above process for each property defined for S_1 and S_2, obtaining:

$$\begin{aligned}\mathrm{CF}(S_1,S_2) &= \Sigma_{i_p=1}^{n_p}\mathrm{CF}_{p_{i_p}} + \Sigma_{i_q=1}^{n_q}\mathrm{CF}_{q_{i_q}} + \Sigma_{i_r=1}^{n_r}\mathrm{CF}_{r_{i_r}} + \Sigma_{i_t=1}^{n_t}\mathrm{CF}_{t_{i_t}} + \Sigma_{i_x=1}^{n_x}\mathrm{CF}_{x_{i_x}} + \Sigma_{i_y=1}^{n_y}\mathrm{CF}_{y_{i_y}} \\ &+ \Sigma_{i_f=1}^{n_f}\mathrm{CF}_{f_{i_f}} + \Sigma_{i_g=1}^{n_g}\mathrm{CF}_{g_{i_g}} + \Sigma_{i_h=1}^{n_h}\mathrm{CF}_{h_{i_h}}\end{aligned}$$

$$\begin{aligned}\mathrm{DF}(S_1) &= \Sigma_{i_p=1}^{n_p}\mathrm{DF}^1_{p_{i_p}} + \Sigma_{i_q=1}^{n_q}\mathrm{DF}^1_{q_{i_q}} + \Sigma_{i_r=1}^{n_r}\mathrm{DF}^1_{r_{i_r}} + \Sigma_{i_t=1}^{n_t}\mathrm{DF}^1_{t_{i_t}} + \Sigma_{i_x=1}^{n_x}\mathrm{DF}^1_{x_{i_x}} \\ &+ \Sigma_{i_y=1}^{n_y}\mathrm{DF}^1_{y_{i_y}} + \Sigma_{i_f=1}^{n_f}\mathrm{DF}^1_{f_{i_f}} + \Sigma_{i_g=1}^{n_g}\mathrm{DF}^1_{g_{i_g}} + \Sigma_{i_h=1}^{n_h}\mathrm{DF}^1_{h_{i_h}}\end{aligned}$$

$$\begin{aligned}\mathrm{DF}(S_2) &= \Sigma_{i_p=1}^{n_p}\mathrm{DF}^2_{p_{i_p}} + \Sigma_{i_q=1}^{n_q}\mathrm{DF}^2_{q_{i_q}} + \Sigma_{i_r=1}^{n_r}\mathrm{DF}^2_{r_{i_r}} + \Sigma_{i_t=1}^{n_t}\mathrm{DF}^2_{t_{i_t}} + \Sigma_{i_x=1}^{n_x}\mathrm{DF}^2_{x_{i_x}} \\ &+ \Sigma_{i_y=1}^{n_y}\mathrm{DF}^2_{y_{i_y}} + \Sigma_{i_f=1}^{n_f}\mathrm{DF}^2_{f_{i_f}} + \Sigma_{i_g=1}^{n_g}\mathrm{DF}^2_{g_{i_g}} + \Sigma_{i_h=1}^{n_h}\mathrm{DF}^2_{h_{i_h}}\end{aligned}$$

where n_p (resp. $n_q, n_r, n_t, n_x, n_y, n_f, n_g, n_h$) is the number of properties defined in each of the possible ways explained above. Finally, we calculate the similarity between two entities S_1 and S_2 defined with restrictions using the formula (2):

$$\mathrm{SIM}(S_1,S_2) = \frac{\mathrm{CF}(S_1,S_2)}{\mathrm{DF}(S_1)+\mathrm{DF}(S_2)+\mathrm{CF}(S_1,S_2)}.$$

Another feature we want to take into account is the presence of equivalent classes, even though they are not defined as restrictions. We assume that two classes declared equivalent with owl:equivalentClass have similarity based on properties equal to 1.

As far as individuals are concerned (instances of the classes) we simply compare the property-value pairs for each instance. If the property p has h' different values in S_1 and h'' different values in S_2, and we denote by k the number of times S_1 and S_2 have the same value for p, then $\mathrm{CF}_p = \dfrac{k^2}{h'h''}, \mathrm{DF}^1_p = \dfrac{h'-k}{h'}$ and $\mathrm{DF}^2_p = \dfrac{h''-k}{h''}$. We repeat this for every property $p_1,\ldots,p_k$ used to describe the given instance. Finally, for instances of classes we obtain:

$$\begin{aligned}\mathrm{CF}(S_1,S_2) &= \Sigma_{i_{p_1}=1}^{n_{p_1}}\mathrm{CF}_{p_{i_{p_1}}} + \ldots + \Sigma_{i_{p_k}=1}^{n_{p_k}}\mathrm{CF}_{p_{i_{p_k}}} \\ \mathrm{DF}^1(S_1,S_2) &= \Sigma_{i_{p_1}=1}^{n_{p_1}}\mathrm{DF}^1_{p_{i_{p_1}}} + \ldots + \Sigma_{i_{p_k}=1}^{n_{p_k}}\mathrm{DF}^1_{p_{i_{p_k}}} \\ \mathrm{DF}^1(S_1,S_2) &= \Sigma_{i_{p_1}=1}^{n_{p_1}}\mathrm{DF}^2_{p_{i_{p_1}}} + \ldots + \Sigma_{i_{p_k}=1}^{n_{p_k}}\mathrm{DF}^2_{p_{i_{p_k}}}.\end{aligned}$$

3.1. Relevance of properties

When defining a certain shape, not all the properties have the same importance in different contexts. For example, in one context two shapes would be regarded similar if they have similar number of angles and edges, in another one if they are of similar size or if they are both concave or convex. In the above presented approach, it is possible to account for relevance of properties by providing the *relevance factors* $R^p_{i_p}, i_p = 1,\ldots,n_p$, for each property p. Relevance factors can be either given as a-priori expert values or gathered as user preferences. In this way, some properties become more important than the others and the formula for calculating the mutual similarity between shapes S_1 and S_2 becomes:

$$\mathrm{SIM}^r(S_1,S_2) = \frac{\mathrm{CF}^r(S_1,S_2)}{\mathrm{DF}^r(S_1)+\mathrm{DF}^r(S_2)+\mathrm{CF}^r(S_1,S_2)}$$

where $\text{CF}^r(S_1, S_2) = \Sigma_{i_p=1}^{n_p} R_{i_p} \text{CF}_{p_{i_p}} + \ldots + \Sigma_{i_h=1}^{n_h} R_{i_h} \text{CF}_{h_{i_h}}$ and similarly for $\text{DF}^r(S_1)$ and $\text{DF}^r(S_2)$.

4. Conclusions

When using ontologies to represent domain knowledge, not always it is convenient to represent shapes in an exhaustive hierarchy. It might be desirable to single our certain properties of shapes and then categorize them having these properties in mind. This is possible if shapes are defined as property restrictions on classes, both on value and cardinality. Representing shapes as property restrictions makes it possible to introduce a very natural similarity measure based on properties. This measure changes depending on the context in which it is being used, making it possible to give more relevance to certain properties in different situations. Apart from modeling shapes as property restrictions on classes, this approach would bring new insights into modeling forms and patterns as well, as it avoids strict categorizations, providing a flexible environment for expressing various features of complex forms.

The presented technique for calculating property-based similarity was first used in [4], for propagation of user interests in ontology based user model. It was evaluated in the context of PIEMONTE project [10][5] which developed a framework based on intelligent objects composed from a real and a virtual part coexisting at the same time, in the context of gastronomy. Although this initial approach did not include cardinality restrictions it showed satisfying performance w.r.t. to actual reasoning and computation of similarity and helped improve the recommendation process. A future implementation of this method would include the cardinality restrictions and show how it handles them, providing feedback for any necessary adjustments.

References

[1] G. Antoniou and F. van Harmelen, *A Semantic Web Primer*, second edition. The MIT Press, 2008.

[2] M. Bhatt and J. Hois and O. Kutz and F. Dylla, *Modelling functional requirements in spatial design*. in *Proc. 29th international conference on Conceptual modeling, ER '10*, pp. 464–470, Springer-Verlag, 2010.

[3] S. Borgo and O. Kutz, *A General Framework for Shape Similarity*. in *Proc. SHAPES 1.0 - The Shape of Things. Workshop at CONTEXT-11*, CEUR-WS, Vol. 812, 2011.

[4] F. Cena and S. Likavec and F. Osborne, *Property-based interest propagation in ontology-based user model*. in *Proc. 20th Conference on User Modeling, Adaptation, and Personalization UMAP 2012*, Lecture Notes in Computer Science 7379, pp. 38–50, Springer-Verlag, 2012.

[5] M. Grüninger and S. Bouafoud, *Thinking Outside (and Inside) the Box*. in *Proc. SHAPES 1.0 - The Shape of Things. Workshop at CONTEXT-11*, CEUR-WS, Vol. 812, 2011.

[6] J. Hois and M. Bhatt and O. Kutz, *Modular Ontologies for Architectural Design*. in *Proc. 2009 Conference on Formal Ontologies Meet Industry*, pp. 66–77, IOS Press, 2009.

[7] J. Jiang and D. Conrath, *Semantic similarity based on corpus statistics and lexical taxonomy*. in *Proc. International Conference on Research in Computational Linguistics*, pp. 19–33, 1997.

[8] P. Jurewicz. *Building Shape Ontology, Organising, Visualising and Presenting Building Shape with Digital Tools*, Diplomarbeit, Technischen Universität Wien, 2005.

[5] PIEMONTE project: People Interaction with Enhanced Multimodal Object for a New Territory Experience, financed by Regione Piemonte, in the context of Converging Technologies.

[9] M. Niknam and C. Kemke, *Modeling Shapes and Graphics Concepts in an Ontology*. in *Proc. SHAPES 1.0 - The Shape of Things. Workshop at CONTEXT-11*, CEUR-WS, Vol. 812, 2011.

[10] L. Console and F. Antonelli and G. Biamino and F. Carmagnola and F. Cena and E. Chiabrando and V. Cuciti and M. Demichelis and F. Fassio and F. Franceschi and R. Furnari and C. Gena and M. Geymonat and P. Grimaldi and P. Grillo and E. Guercio and S. Likavec and I. Lombardi and D. Mana and A. Marcengo and M. Mioli and M. Mirabelli and M. Perrero and C. Picardi and F. Protti and A. Rapp and R. Sandon and R. Simeoni and D. Theseider Dupré and I. Torre and A. Toso and F. Torta and F. Vernero, *WantEat: interacting with social netnetworks of intelligent things and people in the world of gastronomy*, *ACM Transactions on Interactive Intelligent Systems*, Accepted for publication, 2013.

[11] R. Rada and H. Mili and E. Bicknell and M. Blettner, *Development and application of a metric on semantic nets*, *IEEE Transactions on Systems Management and Cybernetics*, 19(1):1730, 1989.

[12] P. Resnik, *Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language*, *Journal of Artificial Intelligence Research*, 11:95130, 1998.

[13] O. Symonova and M. S. Dao and G. Ucelli and R. De Amicis, *Ontology Based Shape Annotation and Retrieval*. in *Proc. ECAI06 International Workshop on Contexts and Ontologies: Theory, Practice and Applications*, 2006.

[14] A. Tversky, *Features of similarity*, *Psychological Review*, 84(4):327–352, 1977.