Conceptual Space Markup Language (CSML): Towards the Cognitive Semantic Web

Benjamin Adams
Dept. of Computer Science
University of California, Santa Barbara
Santa Barbara, CA, USA
Email: badams@cs.ucsb.edu

Martin Raubal
Dept. of Geography
University of California, Santa Barbara
Santa Barbara, CA, USA
Email: raubal@geog.ucsb.edu

***Abstract*—CSML is a semantic markup language created for the publishing and sharing of conceptual spaces, which are geometric structures that represent semantics at the conceptual level. CSML can be used to describe semantics that are not captured well by the ontology languages commonly used in the Semantic Web. Measurement of the semantic similarity of concepts as well as the combination of concepts without shared properties are common human cognitive tasks. However, these operations present sources of difficulty for tools reliant upon set-theoretic and syllogistic reasoning on symbolic ontologies. In contrast, these operations can be modeled naturally using conceptual spaces. This paper describes the design decisions behind CSML, introduces the key component elements of a CSML document, and presents examples of its usage.**

***Keywords*-representation languages; ontology design; ontology languages; semantic web; xml; conceptual spaces**

I. Introduction

CSML is an XML-based semantic computing language designed to represent the semantic content of information using geometric and topological structures called conceptual spaces [1], [2]. CSML differs from existing languages for the Semantic Web and other semantic computing applications, because 1) it is based on a theory derived from the paradigm of cognitive semantics and 2) inferences about the information stored in CSML are made using linear algebra and computational geometry operations rather than set theoretic operations and syllogistic reasoning [3], [4]. These inferencing operations include semantic similarity measurement and the learning of new concept combinations, which will support semantic interoperability between web services.

A fundamental goal of the Semantic Web is the interchange of data encoded in a machine understandable format by web services. The computational foundation of the Semantic Web is the formal representation of concepts and their relationships using the Resource Description Framework (RDF) and the Web Ontology Language (OWL) variants [5], [6]. These families of languages allow one to describe semantic relationships between concepts and ontologies, which can be queried using a first-order logic reasoner. An underlying assumption to these methods is the realist approach to semantics, which states that the meanings of concepts are in the real world. That is, there is a direct mapping between language terms and the world but there is no consideration of how individuals understand concepts differently. In contrast, cognitive semantics states that the meanings of terms are cognitive structures in people's minds. This approach is of central importance for the Semantic Web, because web services interface with human users.

In addition, these symbolic languages cannot well represent semantic similarity and complex concept combinations in a manner accordant with their set theoretic foundations, even though both of these operations are essential for understanding the meanings of concepts. Semantic similarity measurement is fundamental to the categorization of concepts, since concepts are classified with other concepts with which they are most similar [7], [8]. Using ontology languages that represent classes as sets of objects, the combination of concepts is measured as the intersection of the properties of the two classes. However, this method breaks down when combining concepts without shared properties. Take for example Truman Capote's original classification of his book *In Cold Blood* as a non-fiction novel, a previously unknown genre. *Non-fiction novel* combines the concepts *non-fiction* and *novel*. The *novel* concept has no intersecting properties with the *non-fiction* concept, since a novel has the property of being a work of fiction. However, when Capote first coined it people had no trouble making sense of this concept combination even though they had never heard it before. Likewise, ad hoc categories (e.g., *things to take to the beach*) are even harder to describe using ontology languages because the objects that compose the group often do not share any common properties [9].

As an alternative to the symbolic knowledge representation schemes listed above, Gärdenfors has proposed using geometric structures called conceptual spaces to represent concepts for the Semantic Web [10]. The semantic similarity of concepts represented in a conceptual space is measured naturally as an inverse function of their distance in the space. In addition, well defined operations can be applied to conceptual spaces to create new concept combinations, even from concepts without shared properties [11]. In this paper we present CSML, an XML-based interchange format for conceptual spaces. This language is intended to be an exchange format for sharing conceptual spaces, so that they

can be created and used by various semantic software tools.

Following a review of the related research in Section 2, the structure of the paper is as follows. In Section 3, we present a motivating example of a conceptual space and in section 4 follow with a discussion of the general structure of a CSML document. In section 5 we describe the syntax of the elements in a CSML file, and conclude in section 6 with directions for future work.

II. Background

A. Related Work

The representation of human knowledge in schemas and ontologies has a long research history spanning diverse fields such as databases, artificial intelligence, and cognitive science [12], [13]. A common approach is to express units of knowledge in a formal language, which is used to infer new knowledge through logical deduction. More recently the notion of the Semantic Web has been proposed, which applies these techniques to knowledge stored at resources on the web [3]. The primary emphasis in Semantic Web research has been on the development of description logics. However, other techniques are necessary to fully represent semantic content, because description logics were chosen for theoretical reasons (i.e., they are complete and decidable) that come at the expense of expressibility of similarity and uncertainty [10], [14].

The ability to integrate semantically heterogeneous information from different sources is an essential goal towards building a fully functional Semantic Web. Towards that goal there has been a great deal of interest in developing methods for mapping (or otherwise aligning, merging, etc.) one ontology or schema to another [15]. Research on the integration of heterogeneous information predates the Semantic Web [16], but it has become an increasingly important problem given the extremely heterogeneous and unstructured nature of information on the web. A particular area of interest is orchestrating the interoperation of heterogeneous web services [17].

A number of tools for automating ontology matching have been developed [18]. An important method is to impose a metric for comparing the similarity of classes in the different ontologies [19]. The application of probabilistic machine learning algorithms to learn these similarities based on feature sets has been put forward (see e.g., GLUE) [20].

B. Conceptual Space Theory

Conceptual space theory was conceived as a knowledge representation scheme to support reasoning at the conceptual level [2]. It stands in contrast to the symbolic representations that have dominated ontology research for the Semantic Web as well as earlier work on semantic databases. Conceptual spaces are designed to be mid-level representations that complement both high-level symbolic representations and low-level connectionist representations. A formalization of conceptual spaces that demonstrates how they can be connected to high-level representations can be found in [21].

Conceptual spaces represent concepts as sets of convex regions within multidimensional metric spaces called domains. Each domain is made up of a set of integral quality dimensions, which are separable from dimensions in other domains. An example domain is *color*, which is made up of three quality dimensions: *hue*, *value*, and *saturation*. The quality dimensions present a means for measuring and ordering the objects within a domain based on their quality values. Domains can be scientific or phenomenal. The dimensions of a scientific domain are determined by scientific theories such as Newtonian mechanics, whereas phenomenal domains can be highly abstract. The decision of which domains to use is determined by a particular application's needs. Because domains are dynamic geometric structures, mappings can be made between different domain representations of the same concept using geometric transformation and projection operations.

Concepts are represented as sets of convex regions spanning one or more domains. A property is a special case of a concept in one domain. For example, the property *red* is represented as a convex region in the *color* domain. An advantage of conceptual spaces is that taxonomies and classifications are exposed by the topological relationships of regions and there is no need to explicitly state these relationships. For example, the convex regions that represent *penguin* are contained within the regions that represent *bird*.

Properties and concepts can roughly be equated to adjectives and nouns, respectively, in a natural language representation. The convexity of conceptual regions allows one to describe points in the regions as having degrees of centrality, which aligns this representational framework with prototype theory [8]. Analogous to concepts, instances are sets of points spanning one or more domains that represent individual objects. One use of instances is as training data to learn new conceptual regions. Another is to define a prototype for a given concept. For example, the prototype of a *bird* concept might be an instance of a *robin*.

Conceptual space theory describes query operations that can be applied to the concepts represented in a conceptual space, including semantic similarity and concept combination. The programming of conceptual spaces leverages their geometric structure so that these operations can be reduced to vector operations and problems of computational geometry. The data model for CSML is based on a formal conceptual space algebra that defines a conceptual space as a six-tuple, $S = \langle \Delta, \Gamma, \breve{I}, \blacklozenge, K, c \rangle$ [22].

- Δ is a finite set of *domains*, where a *domain* $\delta \in \Delta$.
- Γ is a finite set of *concepts*, where a *concept* $\gamma \in \Gamma$.
- $\breve{I}$ is a finite set of *instances*, where an *instance* $\breve{\imath} \in \breve{I}$.
- $\blacklozenge$ is a finite set of *contrast classes*, where a *contrast class* $\blacklozenge \in \blacklozenge$.
- K is a finite set of *contexts*, where a *context* $k \in K$.

- c is a constant *similarity sensitivity* parameter.

A concept is defined formally as a set of convex polytopes [23]. A polytope can be described as a bounded set of linear inequalities. In addition, a set of points can be defined that make up the prototype instance for a concept.

In this algebra, the semantic similarity operation is defined as an inverse exponential function of the distance between two concepts (or their prototype instances): $sim(d) = e^{-cd}$, where d is the result of a distance function and c is the similarity sensitivity parameter[1]. The context of a similarity measurement is specified with a salience weighting on the quality dimensions and domains when calculating distances.

Three types of concept combination operations are also specified in the algebra: property-concept, concept-concept, and contrast class-concept combinations. Property-concept combinations are used to represent natural language adjective-noun combinations such as *red book*. Concept-concept combinations are complex noun-noun combinations such as *iron gate*. In this case the regions of the modifying concept may either intersect or override the modified concept in one or more of the domains depending on the context.

A contrast class is a special case of property that is used to describe subregions of conceptual regions and are akin to secondary predicates in natural languages [24]. An example contrast class is *large* in the physical space domain. Formally, a contrast class is a pair of parallel linear inequalities representing the bounds of a minimum and a maximum hyperplane intersecting a unit hypercube. The contrast class-concept combination operation applies this range to another region in the domain to produce a new region. Using this technique one can, for example, represent *chihuahua* as a subclass (i.e., subregions) of *dog*, but also represent *large chihuahua* as a subclass of *small dog*, not *large dog*. Such relationships are difficult to do using hierarchical symbolic representation schemes designed for syllogistic reasoning. We direct readers to review [22] for further details on these similarity and concept combination operations.

Categorical reasoning can be performed on concepts in a conceptual space without an explicit representation of links between concepts such as those found in semantic networks [25]. Instead, the topological relationships of the regions that compose the concepts can be used to infer semantic relationships using a Region Connection Calculus [26]. For example, the is-a relationship can be tested by checking if a region is contained within another region. The Voronoi tessellation of a domain from a given set of prototype points can be used to generate a set of category regions. Alternately, the convex hull of a set of exemplar points can be used to learn concepts. Furthermore, nonmonotonic reasoning can be modeled through the application of context weights on the dimensions of the domain, thereby altering the boundaries of the regions and potentially reclassifying individual instances.

[1]This parameter is used to fine-tune the rate of similarity decay.

III. Example for the Geospatial Semantic Web

The following sections describe the structure and components of CSML. To illustrate these components we use an example of a conceptual space model created for a geospatial semantic web service [27]. Although conceptual space theory has generated interest in the geospatial information theory community due to its spatialization approach to knowledge representation, CSML (like OWL) is designed to be universally applicable to any type of semantic computing application.

A geographic information system (GIS) database will often describe information about various geographic entities, such as mountains, hills, streams, and lakes. These features are common concepts, though their semantics may vary from database to database as well as from user to user. For instance, an object classified as a mountain by one GIS might be classified as a hill by another. This is, in fact, a standard ontology integration problem [28]. By representing different instances of landscape objects (e.g., specific mountains) using conceptual spaces, it becomes possible to measure their semantic similarity regardless of their classification [29]. This is because the objects are represented at the conceptual level, which exposes their quality dimension values. Furthermore, since concepts are defined in terms of geometric regions, two different conceptualizations of *mountain*, for example, can be reconciled by calculating functional mappings from one convex region to another [30]. We will use a basic representation of a *mountain* concept and some related domains, contrast classes, and contexts to exemplify usage of CSML elements.

IV. CSML Document

A Conceptual Space Markup Language (CSML) document represents a hierarchy of conceptual space data structure elements in XML format. Figure 1 shows the top levels of a sample CSML document. Figure 2 shows the hierarchy of CSML tags. A CSML document is always rooted with a `csml:CSML` tag and denotes the beginning and end of a *conceptual space model file*. The top level elements in a *conceptual space model file* are *domains*, *concepts*, *contrast classes*, *instances*, and *contexts*. In the following section we will define the syntax of each of these elements and their constituent child elements from a top-down orientation, beginning with the top-level tags and moving our way down to the deepest and most primitive levels of the CSML tree.

A. CSML Schema and Extensibility

The syntax of CSML conforms to XML syntax rules. For instance, all opening tags must have a corresponding closing tag. Furthermore, the language-specific set of rules that a CSML file must conform to are defined in the CSML Schema, which is written in XML Schema [31]. The CSML Schema can be used to validate CSML files and can be found at: http://geocog.geog.ucsb.edu/csml/csml.xsd.

```
<?xml version="1.0"?>
<csml:CSML xml:base="http://www.example.org/example.csml"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
           xmlns:csml="http://geocog.geog.ucsb.edu/csml/csml-ns#">
   <csml:Domain csml:ID="#Size">
      ...
   </csml:Domain>
   <csml:Concept csml:ID="#Mountain">
      ...
   </csml:Concept>
   <csml:ContrastClass csml:ID="#Tall" csml:domainID="#Size">
      ...
   </csml:ContrastClass>
   <csml:Instance csml:ID="#Matterhorn">
      ...
   </csml:Instance>
   <csml:Context csml:ID="#LocationImportant">
      ...
   </csml:Context>
</csml:CSML>
```

Figure 1: Sample CSML file top levels.

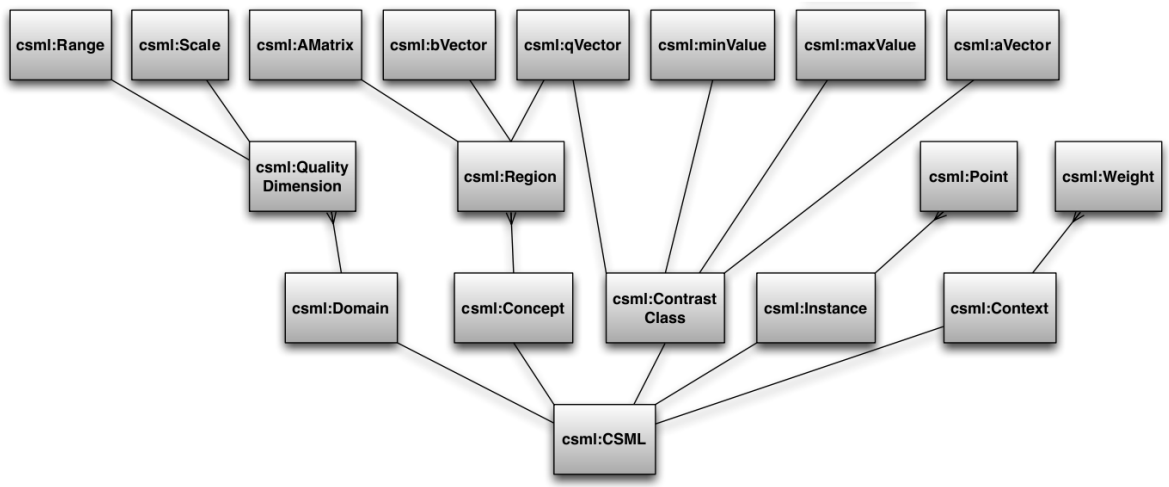

Figure 2: CSML tag hierarchy.

A fundamental design principle behind CSML is extensibility. The formalization of conceptual space theory is an active research area and there are and no doubt will be differences among various formalizations that must be reconciled. Therefore, the structure and syntax of CSML must be open to revision and extension. Since CSML is XML-based, this can be accomplished by creating an extended schema that includes the base CSML Schema file.

B. CSML URIs

Conceptual space knowledge bases written in CSML are designed to be stored as web documents in much the same manner as OWL ontologies. Entities within a CSML document file have unique identifiers in the form of Uniform Resource Identifiers (URI). These identifiers are used to connect conceptual space elements to each other. For example, a prototype for a concept can be identified by the prototype instance's URI.

The identification of conceptual space elements using URIs means that they are unique resources that can be referred to no matter where they are located, which allows one to create knowledge bases that span multiple CSML documents. The use of URIs has a further advantage in that it facilitates interoperability between knowledge bases, because uniquely identified quality dimensions can be mapped to one another using transformation operations. Thus, even if two data sources using different quality dimensions classify the same real world entity as exemplars of two different concepts, the equivalence of the exemplars can still be determined by applying these mappings [30]. Furthermore,

concepts can be translated from one set of domains to another, which affords the ability to quantitatively measure the similarity of concepts described by different data sources.

V. CSML Elements

In this section we describe in detail the elements of a CSML file. We show the tag for each element type along with its description. We note here that we do not attempt a formal model of the semantics of CSML using model theory. Such a task requires work beyond the scope of this paper; however, it is an important future goal, since a formal model is required to have unambiguous interpretation of CSML syntax.

There are two types of elements that can be child nodes for any other element, the label of the element and a longer description. The tags for these elements are `<csml:Label>` and `<csml:Description>`. The label is distinct from the element's URI, and two different elements in a CSML file can have the same label. For example, this situation may occur if a domain has only one quality dimension (e.g., the age domain). Both of these elements function as comments and can be useful for communicating the intended use of the element to users. For brevity we do not include the label and description tags when describing the structure of other elements.

In the following subsections, we begin by defining CSML data elements that are used to specify the geometric structure of a conceptual space, namely the quality dimensions and their classification into mutually exclusive domains. Following that we show how to describe concepts, contrast classes, instances, and contexts in CSML.

A. csml:Domain

The description of domain structures is the foundation of using conceptual spaces as a representational framework for semantic computing. Once domains are defined they can be referred to by different users to define concepts and contexts for different uses. In CSML a domain element will have one or more quality dimension child elements. These quality dimensions are integral quality dimensions that are separable from dimensions defined in other domains. The following is an example of the size domain defined with two quality dimensions, height and width:

```
<csml:Domain csml:ID="#Size">
   <csml:QualityDimension csml:ID="#Height">
      <csml:Scale>ratio</csml:Scale>
      <csml:Units>meters</csml:Units>
   </csml:QualityDimension>
   <csml:QualityDimension csml:ID="#Width">
      <csml:Scale>ratio</csml:Scale>
      <csml:Units>meters</csml:Units>
   </csml:QualityDimension>
</csml:Domain>
```

In CSML syntax a quality dimension is identified by the `<csml:QualityDimension>` tag. The `csml:Scale` tag indicates the measurement level of the dimension: ordinal, interval, or ratio [32]. Nominal dimensions are not directly supported in CSML. However, one solution for including nominal data is to represent the nominal values as property regions in a domain [33]. The `csml:Units` element indicates the scale's units as a string value. For scientific dimensions with well-defined conversion mechanisms the units value can be used to convert to other scales (e.g. from inches to meters).

A quality dimension may also have a range of possible values. For example, the longitude dimension in the geographical coordinate domain has a minimum value of -180.0 degrees and a maximum of 180.0 degrees. The range is specified with a `<csml:Range>` tag. The quality dimension tag also has an optional attribute called `csml:Circular`, which indicates that distances in the spaces are modulo the range magnitude divided by two.

```
<csml:QualityDimension
      csml:ID="#Longitude"
      csml:circular="true">
   <csml:Scale>interval</csml:Scale>
   <csml:Range>
      <csml:Min>-180.0</csml:Min>
      <csml:Max>180.0</csml:Max>
   </csml:Range>
   <csml:Units>degrees</csml:Units>
</csml:QualityDimension>
```

B. csml:Concept

A concept is represented as a set of regions in one or more domains and is denoted by a `<csml:Concept>` tag. Properties, which are concepts in only one domain, are also identified by this tag. The following is an abbreviated example of how the concept *mountain* can be specified in CSML (see figure 3).

```
<csml:Concept csml:ID="#Mountain"
      csml:prototypeID="#MountainPrototype">
   <csml:Region
         csml:ID="#MountainSizeRegion"
         csml:domainID="#Size">
        <csml:AMatrix> ... </csml:AMatrix>
        <csml:qVector> ... </csml:qVector>
        <csml:bVector> ... </csml:bVector>
   </csml:Region>
   ...
</csml:Concept>
```

A concept may have an associated prototype that is specified with the `<csml:prototypeID>` tag. The prototype refers to the URI for another CSML element that is defined elsewhere. In many cases the prototype URI will point to a `<csml:Instance>` element, though it can also be another `<csml:Concept>` element. One might want to define the prototype in terms of prototypical *regions* rather than prototypical points. The regions that represent a concept are child elements of the concept tag identified by `<csml:Region>`. The example above shows only one region (for the size domain), but complex concepts such as

mountain will have many more regions defined for other domains, such as coordinates, color, climate, age, amount of development, etc.

Each region is a multidimensional convex polytope within a domain. A convex polytope can be mathematically defined in two ways: 1) as the intersection of a set of half-spaces, called an H-polytope, or 2) as the convex hull of a set of points, called a V-polytope [23]. CSML currently supports the H-polytope method for defining a region, but future versions will allow V-polytopes as well, since it may be more convenient to describe a region using point values in some circumstances. A subset of the MathML markup language is used by CSML to define mathematical primitives, i.e. the matrices and vectors [34].

An H-Polytope is described as a system of linear inequalities, represented in matrix form in CSML: $Aq \leq b$, where A is an $m \times n$ matrix, q is a column vector of n variables, and b is a column vector of m constant values. Each q variable corresponds to one of the n dimensions that make up the domain and m is the number of facets ($n-1$ dimensional faces) of the polytope.

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{n1} \\ a_{21} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{nm} \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix} \leq \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

In CSML A, q, and b are defined using the `<csml:AMatrix>`, `<csml:qVector>`, and `<csml:bVector>` tags respectively. The A matrix uses the MathML syntax for matrices; each row is defined with a `<matrixrow>` tag and the matrix values are found in `<cn>` tags, e.g.:

```
<csml:AMatrix>
   <matrixrow>
      <cn> -1.0 </cn>
      <cn> 0.0 </cn>
   </matrixrow>
   <matrixrow>
      <cn> 0.6 </cn>
      <cn> -1.0 </cn>
   </matrixrow>
   ...
</csml:AMatrix>
```

Since the quality dimensions are not ordered in a domain structure, the q vector definition is used to identify which columns in A correspond to which dimensions. The q vector uses the MathML vector notation where the variable in each `<ci>` tag is a quality dimension URI.

```
<csml:qVector>
   <ci> #Width </ci>
   <ci> #Height </ci>
</csml:qVector>
```

The b column vector is defined similarly using `<cn>` tags.

```
<csml:bVector>
   <cn> -0.25 </cn>
   <cn> -0.3 </cn>
   <cn> 0.8 </cn>
</csml:bVector>
```

We do not intend that users of CSML will hand-code H-polytope representations of concepts very often. Software tools will generate the appropriate CSML as the output of learning operations such the application of the convex hull algorithm on a set of training exemplars.

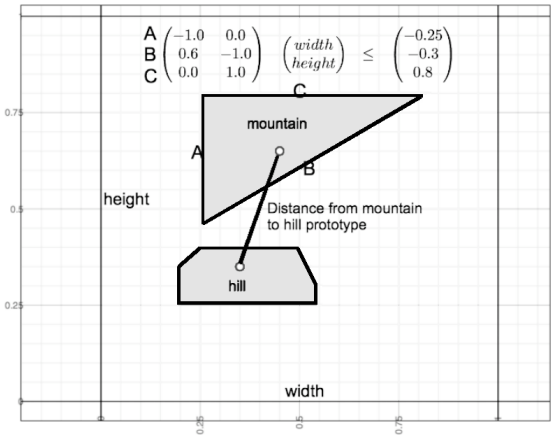

Figure 3: Concepts in normalized [0,1] *size* domain.

C. csml:ContrastClass

A contrast class is represented by a region bounded by one or two parallel hyperplanes that intersect a unit hypercube. The two hyperplanes are labeled min and max and take the following form:

$$min : -a_1q_1 - a_2q_2 - \cdots - a_nq_n \leq -b_1$$
$$max : a_1q_1 + a_2q_2 + \cdots + a_nq_n \leq b_2$$

In CSML these equations are written as a row vector a, a column vector q, and two values `csml:ccMin` and `csml:ccMax`, which correspond to b_1 and b_2 in the equation above. The a vector is defined by a `<csml:aVector>` tag and the q vector is the same as in the region tag above.

```
<csml:ContrastClass csml:ID="#Tall"
      csml:domainID="#Size">
   <csml:aVector>
      <cn>-1.0</cn>
      <cn>0.0</cn>
   </csml:aVector>
   <csml:qVector>
      <ci>#Height</ci>
      <ci>#Width</ci>
   </csml:qVector>
   <csml:ccMin> -0.7 </csml:ccMin>
   <csml:ccMax> 1.0 </csml:ccMax>
</csml:ContrastClass>
```

Here the *tall* contrast class is defined as the top 30% of a unit square along the height dimension. This contrast class can be combined with the *mountain* concept by stretching the unit square (and corresponding contrast class region) to

the shape of *mountain* in the size domain, centering it over the prototype, and then finding the intersection of the two to represent *tall mountain* (see figure 4). A formalization of this operation is detailed in [22].

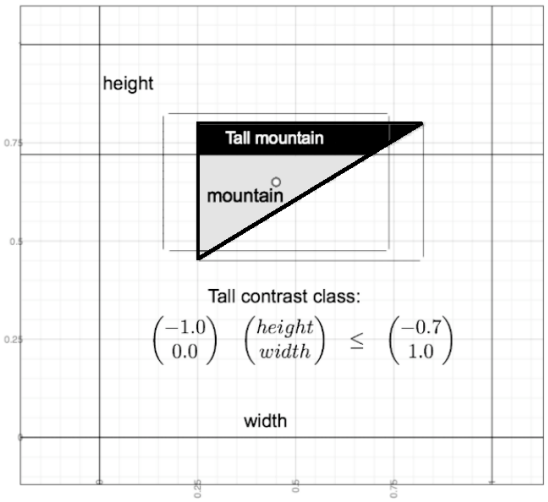

Figure 4: Tall contrast class + mountain concept in normalized [0,1] *size* domain.

D. csml:Instance

An instance is a set of points in one or more domains. In theory one can think of an instance as a concept composed of convex regions of size zero. However, a unique representation for instances is needed, since the H-polytope representation used for concepts does not work for instances.

```
<csml:Instance csml:ID="#Matterhorn">
   <csml:Point csml:ID="#MatterhornCoord"
               csml:domainID="#Coordinates">
      <csml:PointValues>
         <cn>45.98</cn>
         <cn>7.66</cn>
      </csml:PointValues>
      <csml:qVector>
         <ci>#Latitude</ci>
         <ci>#Longitude</ci>
      </csml:qVector>
   </csml:Point>
   ...
</csml:Instance>
```

E. csml:Context

Contexts in a conceptual space are defined generically, as salience weights on conceptual space components. The components that are weighted in a context must all be of the same type (e.g., domains or quality dimensions) and sum to 1.0. These weights are used by operations such as semantic similarity, so that the results can be personalized for a specific context [35]. For example, when calculating the similarity of two mountains, in one context the geographical locations of the mountains will be more salient and in another context the sizes of the mountains will be more salient.

```
<csml:Context csml:ID="#SizeImportant"
              csml:type="domain">
   <csml:Weight csml:cID="#Coordinates"> 0.2
   </csml:Weight>
   <csml:Weight csml:cID="#Size"> 0.8
   </csml:Weight>
</csml:Context>
```

VI. Conclusions and Future Work

Conceptual space theory presents a novel approach to concept representation for semantic computing. The geometric structure of conceptual spaces allows for a richer representation of semantics than is possible with symbolic ontology languages, which affords new methods for measuring semantic similarity and analyzing concept combinations. In this paper we have introduced CSML, a standard interchange format for semantic computing applications to describe conceptual spaces. The representation of concepts using CSML will facilitate the creation and sharing of descriptions of domain structures and concept regions, which will form the basis of a *cognitive* semantic web.

Knowledge bases written in CSML are not necessarily intended to supplant symbolic forms of representation such as OWL ontologies but rather to augment the existing representation schemes, given the already broad investment in those other technologies. Therefore, an important future work direction will be to identify translations between topological concept relations in conceptual spaces and existing methods of taxonomic classification using ontology languages. In addition, CSML does not currently support the description of fuzzy concepts, correlations between regions in different domains, or representations of incomplete information [36]. Such additions would greatly enrich the expressibility of semantic content using CSML.

References

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C Recommendation, 2008.

[2] P. Gärdenfors, *Conceptual Spaces: The Geometry of Thought*. Cambridge: MIT Press, 2000.

[3] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, pp. 34–43, 2001.

[4] G. Lakoff, "Cognitive Semantics," in *Meaning and Mental Representations*, U. Eco, M. Santambrogio, and P. Violi, Eds. Bloomington: Indiana University Press, pp. 119–154, 1988.

[5] G. Klyne and J. Carroll, Eds., *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation, 2004.

[6] D. McGuinness and F. van Harmelen, Eds., *OWL Web Ontology Language Overview*. W3C Recommendation, 2004.

[7] R. Goldstone, "The Role of Similarity in Categorization: Providing a Groundwork," *Cognition* vol. 52, pp. 125–157, 1994.

[8] E. Rosch, "Cognitive representations of semantic categories," *Journal of Experimental Psychology: General*, vol. 104, pp. 192–233, 1975.

[9] L. Barsalou, "Situated simulation in the human conceptual system," *Language and Cognitive Processes*, vol. 5, pp. 513–562, 2003.

[10] P. Gärdenfors, "How to Make the Semantic Web More Semantic," in *Formal Ontology in Information Systems, Proceedings of the Third International Conference (FOIS 2004). Frontiers in Articial Intelligence and Applications*, A. Varzi and L. Vieu, Eds., vol. 114, pp. 153–164. Amsterdam: IOS Press, 2004.

[11] P. Gärdenfors, "Conceptual Spaces as a Framework for Knowledge Representation," *Mind and Matter*, vol. 2, pp. 9–27, 2004.

[12] R. Hull, and R. King, "Semantic Database Modeling: Survey, Applications, and Research Issues," *ACM Computing Surveys*, vol. 19, pp. 201–260, 1988.

[13] J. Sowa, *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, CA: Brooks Cole Publishing Co., 2000.

[14] A. Sheth, C. Ramakrishnan, and C. Thomas, "Semantics for the Semantic Web: The Implicit, the Formal and the Powerful," *International Journal on Semantic Web and Information Systems*, vol. 1, pp. 1–18, 2005.

[15] N. Noy, "Semantic Integration: A Survey Of Ontology-Based Approaches," *SIGMOD Record Special Issue on Semantic Integration*, 2004.

[16] Y. Arens, C. Chee, C. Hsu, and C. Knoblock, "Retrieving and Integrating Data from Multiple Information Sources," *International Journal of Intelligent and Cooperative Information Systems*, vol. 2, pp. 127–158, 1993.

[17] M. Burstein and D. McDermott, "Ontology Translation for Interoperability Among Semantic Web Services," *AI Magazine*, vol. 26, pp. 71–82, 2005.

[18] E. Rahm and P. Bernstein, "A survey of approaches to automatic schema matching," *The VLDB Journal*, vol. 10, pp. 334–350, 2001.

[19] J. Euzenat and P. Valtchev, "Similarity-based ontology alignment in OWL-Lite," in *Proc. 15th European Conference on Artificial Intelligence*, Valencia (ES), pp. 333–337, 2004.

[20] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, "Ontology Matching: A Machine Learning Approach," *Handbook on Ontologies in Information Systems*, S. Staab and R. Studer (Eds.), Springer-Verlag, pp. 397–416, 2004.

[21] J. Aisbett and G. Gibbon, "A general formulation of conceptual spaces as a meso level representation," *Artificial Intelligence*, vol. 133, pp. 189–232, 2001.

[22] B. Adams and M. Raubal, A Metric Conceptual Space Algebra. in *Spatial Information Theory - 9th International Conference, COSIT 2009, Aber Wrac'h, France. Lecture Notes in Computer Science*, K. Stewart Hornsby, G. Ligozat, C. Claramunt, and M. Denis, Eds., pp. 51-68, Berlin: Springer, 2009.

[23] P. Grünbaum, *Convex Polytopes*, 2nd ed. V. Kaibel, V. Klee, and G. Ziegler, Eds. New York: Springer, 2003.

[24] P. Gärdenfors, "Some Tenets of Cognitive Semantics," in *Cognitive Semantics: Meaning and Cognition*, J. Allwood and P. Gärdenfors, Eds., Amsterdam: John Benjamins, pp. 19–36, 1990.

[25] P. Gärdenfors and M. Williams, "Reasoning About Categories in Conceptual Spaces," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* Morgan Kaufmann, pp. 385–392, 2001.

[26] A. Cohn, B. Bennett, J. Gooday, and N. Gotts, "Qualitative Spatial Representation and Reasoning with the Region Connection Calculus," *GeoInformatica*, vol. 1, pp. 275–316, 1997.

[27] M. Egenhofer, "Toward the Semantic Geospatial Web," in *GIS '02: Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pp. 1–4, 2002.

[28] J. Heflin and J. Hendler, "Semantic Interoperability on the Web," in *Extreme Markup Languages 2000*.

[29] A. Schwering and M. Raubal, "Measuring Semantic Similarity between Geospatial Conceptual Regions," in *GeoSpatial Semantics - First International Conference, GeoS 2005, Mexico City, Mexico*, A. Rodriguez, I. Cruz, M. Egenhofer, and S. Levashkin, Eds., vol. 3799, pp. 90–106. Berlin: Springer, 2005.

[30] M. Raubal, "Mappings for Cognitive Semantic Interoperability," in *AGILE 2005 - 8th Conference on Geographic Information Science*, F. Toppen and M. Painho, Eds. pp. 291-296, Instituto Geografico Portugues (IGP), Lisboa, Portugal, 2005.

[31] D. C. Fallside and P. Walmsley, *XML Schema Part 0: Primer Second Edition* W3C Recommendation, 2004.

[32] S. Stevens, "On the Theory of Scales of Measurement," *Science*, vol. 103, pp. 677–680, 1946.

[33] O. Ahlqvist and H. Ban, "Categorical Measurement Semantics: A New Second Space for Geography," *Geography Compass*, vol. 1, pp. 536–555, 2007.

[34] D. Carlisle, P. Ion, R. Miner, and N. Poppelier, *Mathematical Markup Language (MathML) Version 2.0*. W3C Recommendation, 2001.

[35] A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness," in *CHI 2000 Workshop on the What, Who, Where, When, Why and How of Context-Awareness*. The Hague, The Netherlands, 2000.

[36] O. Ahlqvist, "A Parameterized Representation of Uncertain Conceptual Spaces," *Transactions in GIS*, vol. 8, pp. 492–514, 2004.