# An Ontology Design Pattern for Cartographic Map Scaling *

David Carral[1], Simon Scheider[2], Krzysztof Janowicz[3], Charles Vardeman[4],
Adila Krisnadhi[1], and Pascal Hitzler[1]

[1] Kno.e.sis Center, Wright State University, USA
[2] Institute for Geoinformatics, University of Münster, Münster, Germany
[3] Department of Geography University of California, Santa Barbara, CA, USA
[4] Center for Research Computing, University of Notre Dame, Notre Dame, IN, USA

**Abstract.** The concepts of *scale* is at the core of cartographic abstraction and mapping. It defines which geographic phenomena should be displayed, which type of geometry and map symbol to use, which measures can be taken, as well as the degree to which features need to be exaggerated or spatially displaced. In this work, we present an ontology design pattern for map scaling using the Web Ontology Language (OWL) within a particular extension of the OWL RL profile. We explain how it can be used to describe scaling applications, to reason over scale levels, and geometric representations. We propose an axiomatization that allows us to impose meaningful constraints on the pattern, and, thus, to go beyond simple surface semantics. Interestingly, this includes several functional constraints currently not expressible in any of the OWL profiles. We show that for this specific scenario, the addition of such constraints does not increase the reasoning complexity which remains tractable.

## 1 Introduction and Motivation

The notion of scale is at the very core of cartography and essential for the visualization of geo-information in maps [15]. However, scale also plays a key role for knowledge representation and measurement [8]. In its simplest form, scale can be expressed as a representative fraction that specifies the relation between the distance measured on a map to the corresponding distance in the physical world [17]. For example, a *large* map scale of 1:25000 indicates that one unit of measure on this map corresponds to 25000 units of the same measure on the ground. In turn, a *small* scale map of 1:100000 covers a larger region.[5]

Dealing with scale-dependent representations of phenomena in maps in a seamless manner is called *map scaling*. It involves a lot of specialized knowledge. For instance, highway symbols may have to be exaggerated and displaced before they can be rendered on a small scale map. Thus, trying to read the street widths or pathway from a small scale map will yield meaningless results. Similarly, geographic features such as creeks, lakes, and ponds will have to be fused, simplified, or omitted. Buildings may be represented as polygons or point-like features at some scale, but may be fused to blocks and neighborhoods at a smaller scale. Generalizing further will collapse these blocks into a representation of a whole city. Finally, on a global map, only major capitals will be left while all other cities may disappear.

[*] An extended technical report with the appendix can be found at `http://knoesis.wright.edu/pascal/resources/publications/odp-carto-scaling-TR.pdf`

[5] The reason for this is that the terms *small* and *large* refer to the representative fraction, with 1/25000 being the larger fraction. Note that this usage of small and large differs from how other domains refer to scale, e.g., as in 'large scale study'.

Map scaling requires choosing a certain data representation as well as a mode of display for every geographic phenomenon type at each scale level inside a map extent. Cartographic abstraction is, to a large degree, a sophisticated craft. It involves semantic as well as cartographic knowledge, including knowledge about the involved type of features, map generalization rules, and appropriate symbolism for layout as well as symbol placings [15].

However, currently, the knowledge about scale dependency of digital representations remains inaccessible. This makes the integration of digital information across scales and across applications challenging. While there is a rich body of work on how to address scale in cartography, most of the knowledge involved is not specified formally or is hidden in application source code. This contradicts with one of the major paradigm shifts underlying Semantic Web research, namely to enable the creation of smarter data instead of smarter applications. Rather than engineering increasingly complex software, the so-called business logic should be transferred to the level of (meta) data. The rationale behind this is that smarter data will enable more usable and flexible applications, while smarter applications alone fail to improve data along the same dimensions. So far, the notion of scale has barely been given any attention in the Semantic Web, even though most digital resources have an intrinsic scale level. In particular, we do not know of any published ontology patterns on scale.

In this paper, we propose a scale ontology design pattern (in the sense of [6]) which can be used to document and publish knowledge about map scaling applications on the web. It describes the scale dependent representation of geographic phenomena in such applications, and makes the underlying scaling decisions explicit and accessible on the Web. Hence, our work may be integrated with provenance ontologies such as Prov-O[6]. With respect to (semantic) Web services, our pattern can be used to link and track geo-features across scale levels. For example, one could query for a map service that serves base data in the scale required to visualize features from another service. Furthermore, the map scaling pattern allows to *reason on scaled geographic information*. For example, one can check whether two phenomena can be displayed together at a single scale level across scaling applications. One may also gather information about a certain geographic phenomenon at a high level of geographic detail across the Web. And one can check scaling applications for consistency of scaling and representation.

From the viewpoint of semantic technology, we will address two challenges for such a pattern: First, from a conceptual viewpoint, the pattern has to ensure that geographic features are traceable across scale levels, and that the basic logical constraints inherent in (rendering) applications are formally captured in the pattern. Second, from a computational point of view, reasoning with the pattern needs to be tractable. For this purpose, the application logic needs to captured in a tractable subset of first order logic (FOL). However, as we will show later on, current OWL fragments are not flexible enough to capture the required functional constraints. We will show that these constraints can be captured by a certain logical fragment that remains in polynomial order of complexity.

In the following, we will first discuss map scaling in order to motivate and help understand our axiomatization. Then, we will discuss a formal axiomatization on the level of a functional pattern as well as on the level of a DL fragment which allows tractable reasoning. We will evaluate the pattern by showing its use in an existing application that studies Malaria, before we discuss and conclude the paper.

[6] http://www.w3.org/TR/prov-o/

## 2 Map Scaling in a Nutshell

Practical solutions of the map scaling problem draw on a number of core issues in Geographic Information Science (GIScience) as well as Computer Cartography [15,14]. In this section, we will give a very brief overview of key concepts and related work. We will also suggest a conceptual view on them, which helps put our ontology pattern into context.

### 2.1 Map Scale

With the notion *scale* we mean *cartographic scale*, which refers to the ratio of the depicted size of a feature on a cartographic map relative to its actual size [17]. There are other notions of scale. For example, the *scale of analysis* and the *scale of observation* are scales induced by analyzing or observing a phenomenon. The *scale of a phenomenon* is the scale at which a phenomenon appears or can best be studied [21]. The latter kinds of scale are not explicitly addressed in this paper, however, they may decide about whether certain kinds of entities *appear* at certain scales or not [17]. In this context it will be important to keep in mind that a unique map scale exists only for a *map image*, i.e., a map displayed as an image, and not for map data, which may be displayed at several different scales. Furthermore, (digital) zooming should not be confused with scaling, as no new information is added or removed while zooming. So far, we do not know of any published work addressing the issue of scale in the Semantic Web context[7]. However, the topic is central for GIScience [5].

### 2.2 Geographic Phenomena

Geographic phenomena can be represented in a cartographic map. They come in different feature/object types, such as rivers, cities, roads, buildings, people, land-parcels, or the earth's surface. They may also consist of conventionally established regions in a spatial reference system, such as the borders of Germany. Furthermore, they may consist of qualities, such as temperature or windspeed or building height. Geographic phenomena can be represented by geo-ontologies, ranging from *top-level ontologies* such as [2] to domain ontologies such as NASA's Sweet[8]. Most importantly, however, geographic phenomena can be measured in terms of *reference systems*. This means one can unambiguously *observe their extent* in at least some *spatial reference system* that allows to refer to geographic locations, such as WGS84. This makes them amenable to cartographic mapping. Externally, geo-ontologies can be aligned with the phenomenon class of our pattern to differentiate among these types of phenomena and subclass them further.

### 2.3 Map Data

Map data is any set of data which represents geographic phenomena and which can be cartographically mapped (i.e., displayed in a map). For this purpose, it needs to contain a *spatial geometry*, i.e., a type of data which specifies a subset of points in some spatial reference system. This subset may be a single point, a line, or some region. A frequent data structure are *geodata records* used in Geographic Information Systems (GIS), i.e., records of a spatial geometry and non-spatial attribute values. The latter can represent measurable qualities, e.g., temperature, as well as cartographic symbol types, e.g., a color symbol, which may be used to display the

[7] Preliminary unpublished work is avaible here: `http://vocamp.org/wiki/Scale-vocab`.

[8] `http://sweet.jpl.nasa.gov/2.2/sweetAll.owl`.

map data at a certain scale level. Depending on the kind of geometry, one can distinguish two kinds of map data: one is *raster data*, where the geometry forms a regular tessellation (a topological cover with regular polygons) of a subset of the reference space. An example is a satellite image. Another one is *vector data*, where geometries can be irregular and need not form a tesselation. Map data often comes in collections representing phenomena of similar type called *layers*. The different kinds of map data form the context but are not part of our ontology pattern.

### 2.4 Resolution

Resolution is a central notion for map scaling, however, its semantic specification is challenging [4]. In this section, we clarify our use of the term in an informal way, following ideas in [4]. Resolution can be regarded as a *property of map data* which allows to measure its *level of detail*. Note that map data has a resolution but not a map scale in our sense, since it is not necessarily displayed, and not necessarily at a single scale level. There are different proxy-measures for resolution, depending on the purpose. One proxy measure is based on *grain size*, i.e., the extent of an atomic mapping entity in a spatial reference system. An example is the *instantaneaous field of view* of a satellite, i.e., the area on the ground surface that corresponds to a single remote detector element, or the *minimal mapping unit* [8]. Our pattern allows grain size resolution levels of map images interchangeably with their map scale levels in order to restrict potential display of map data to a scale level. However, we do not explicitly model scale and resolution of map data, which may be done in the future.

### 2.5 Map Image

A map image is the result of displaying map data in a map display, i.e., a medium (e.g., paper) used to visualize the map, according to a map scale. Therefore, a map image has a grain size resolution as well as a map scale, the latter because it is projected into a map display in which each pixel has a measurable size. A map scale can be computed from the image resolution by multiplying the latter with the pixel size. In a map scaling application, a new map image is generated every time one zooms in or out. Note that a *map image file* is a different beast. In contrast to the former, the latter is a form of map data in raster format. While the latter has a grain (pixel) size resolution, it does not have a map scale (since its pixels do not have fixed display sizes).

### 2.6 Scaling and Map Generalization

Scaling [21] refers to the seamless transfer of information between different scale levels. Even though a digital map display in principle allows zooming in and out regardless of map data or phenomena, a *visually graspable image* as well as a *semantically adequate and computationally efficient* data representation depends on the resolution as well as the design of the map data. For example, from a visual standpoint, representations can become congested, coalesced or imperceptible at inappropriate scales [15]. There is web technology available which allows to specify and serve scale dependent maps in the web[9]. The problem of map generalization has been addressed by Computer Cartography with operations that modify map data geometry and symbolization accordingly. These include simplification, smoothing, aggregation, amalgamation, merging, collapse, refinement, exaggeration, enhancement and

[9] For example, the Feature Portrayal Service (FPS) and Styled Layer Descriptor (SLD) standards of the Open Geospatial Consortium (OGC) `http://www.opengeospatial.org`.

displacement [15]. Since existing algorithms are not able to automatize generalization to a degree which corresponds in quality to manual cartographic techniques [16], map generalization in practice is still done manually or semi-automatically [16].

Furthermore, generalization has an aspect of *(semantic) modeling* [16], which relates to observation and intrinsic phenomenon scales mentioned above [21]. As a consequence of scaling, not only the visual representation of a phenomenon, but also geometry, semantic classes, properties and relations are affected. Furthermore, even the occurrence of individuals is not invariant across scales. Existing formalisms for scaling and generalization, such as *stratified map spaces* [20], simplify the problem to lifting or generalizing classes and spatial regions, not objects. However, as a matter of fact, individual geographic objects, such as cities and countries, appear at smaller scales, while streets, places and buildings only appear at larger scales. From an ontological viewpoint, generalization therefore affects the level of detail of an ontology and thus may change an ontological theory. This can be treated as a mereological problem on the side of phenomena, such as the problem to individuate road junctions from a road segment network [18]. For our pattern, we will therefore assume a generalization relation between map data sets, which reflects diverse ways of generalizing data sets on different scale levels.

## 3 Preliminaries

To define and implement the pattern in a tractable way, we introduce an extension of the existing description logics fragment DLP [9], the logic fragment underlying the OWL RL profile. We denote the extension as DLP$_\exists$ and show that, under certain appropriate syntactic restrictions, reasoning over this DL fragment remains tractable. Throughout the paper we will show how this extension allows us to express some useful constraints enhancing the usability of the map scaling design pattern.

We will make use of the DL notation along the paper, as we think it improves readability and understanding of the ideas presented. Furthermore, some of the new features included in the extended fragment DLP$_\exists$ are not even part of the OWL language, such as role conjunction, and therefore cannot be expressed in any of the existing OWL syntaxes. DL syntax allows us to express these constructors without having to introduce major changes in the existing notation. Henceforth, we only implement the part of the pattern that can be done making use of the current available constructors in OWL. If the reader is not familiar with the DL notation see [13] for a quick introduction and [1] for a lengthier one.

The set $\mathcal{C}$ of allowed concepts in DLP$_\exists$ is the set of all concepts that can be constructed making use of all constructors in Table 1. Like in other DL fragments, we can divide the axioms in a DLP$_\exists$ knowledge base into *ABox* $\mathcal{A}$, *TBox* $\mathcal{B}$, and *RBox* $\mathcal{R}$ statements.[10]

A DLP$_\exists$ *TBox* [*RBox*] is a finite set of general concept inclusions (GCIs) [role inclusion axioms (RIAs)] as described in Table 1. An *ABox* is a finite set of concept and role assertions also as described in Table 1. Furthermore, DLP$_\exists$ restricts the at-most one cardinal restriction constructor ($\leq 1R.C$) to only appear in the right hand side of GCIs. Note that DLP$_\exists$ allows for the use of unrestricted role conjunctions.

We do not impose any kind of role regularity restrictions in DLP$_\exists$ as defined for $\mathcal{SROIQ}$ [10], which is the logic fragment underlying the OWL DL profile. The regularity restrictions are applied to the tractable OWL profiles in order to define OWL DL as a superset of these. Otherwise some of the tractable fragments would allow some expressivity not available in OWL DL.

[10] Assertional, terminological, and role boxes respectively

**Table 1.** DLP$_\exists$ Constructors. $C, D \in \mathcal{C}$ are concepts, $R$ and $S$ are roles, and $\{t\}$ is a nominal.

| **Name** | **Syntax** | **Semantics** |
|---|---|---|
| Concept Assertion | $C(a)$ | $a^\mathcal{I} \in C^\mathcal{I}$ |
| Role Assertion | $R(a, b)$ | $\langle a, b\rangle \in R^\mathcal{I}$ |
| GCI | $C \sqsubseteq D$ | $C^\mathcal{I} \subseteq D^\mathcal{I}$ |
| Existential Restriction | $\exists R.C$ | $\{\delta \mid$ there is $\epsilon$ with $\langle \delta, \epsilon\rangle \in R^\mathcal{I}$ and $\epsilon \in C^\mathcal{I}\}$ |
| $\leq$ 1 Card. Restriction | $\leq 1R.C$ | $\{\delta \mid \sharp\{\langle \delta, \epsilon\rangle\} \in R^\mathcal{I} \mid \epsilon \in C^\mathcal{I}\} \leq 1$ |
| Concept Conjunction | $C \sqcap D$ | $C^\mathcal{I} \cap D^\mathcal{I}$ |
| Top concept | $\top$ | $\Delta^\mathcal{I}$ |
| Bottom concept | $\bot$ | $\emptyset$ |
| RIA | $R \sqsubseteq S$ | $R^\mathcal{I} \subseteq S^\mathcal{I}$ |
| Role Inverse | $R^-$ | $\{\langle \delta, \epsilon\rangle \mid \langle \epsilon, \delta\rangle \in V^\mathcal{I}\}$ |
| Role Chain (RIA) | $R_1 \circ \ldots \circ R_n$ | $R_1^\mathcal{I} \circ \ldots \circ R_n^\mathcal{I}$ |
| Role Conjunction | $R \sqcap S$ | $R^\mathcal{I} \cap S^\mathcal{I}$ |

To preserve tractability of DLP$_\exists$ we need to impose restrictions in the use of the existential constructor on the right hand side of GCIs, otherwise the fragment becomes undecidable as shown in [11]. The definition of necessary or at least sufficient conditions under which DLP$_\exists$ still retains tractability are out of the scope of this paper. Nevertheless as we show in the Appendix that reasoning in DLP$_\exists$ is not only decidable but tractable for the defined pattern and leave the definition of these restrictions as further work. We also elaborate on how simply we could devise a reasoning algorithm based on some of the existing reasoning procedures for DLP.

It may give the impression that the set of constructors represented in Table 1 do not cover the complete expressivity of the OWL RL profile. Indeed, we do not explicitly include some of the constructors that are part of the specifications of the RL profile. As shown Table 2, all of the original RL constructors can be constructed from the set of constructors in Table 1. A smaller set of constructors allows for more succinct definitions and theorems in further sections.

**Table 2.** DLP$_\exists$ Syntactic Sugar. $C$, $D$, and $E$ are concepts, and $R$ is a role.

| RL Axiom | Equivalent Axioms |
|---|---|
| $C \sqsubseteq \forall R.D$ | $\exists R^-.C \sqsubseteq D$ |
| $C \sqcup D \sqsubseteq E$ | $C \sqsubseteq E$<br>$D \sqsubseteq E$ |
| $C \sqsubseteq \neg D$ | $C \sqcap D \sqsubseteq \bot$ |
| $C \sqsubseteq \leq 0R.D$ | $C \sqcap \exists R.D \sqsubseteq \bot$ |

Note that we have also included the $\bot$ concept in our definition in Table 2, which does not appear as part of the RL constructors. This special concept is easy to simulate in RL adding axioms $C_\bot \sqcap D_\bot \sqsubseteq \bot$ and $C_\bot \sqsubseteq D_\top$ where $C_\bot$ and $D_\bot$ are fresh concepts. We have that if $K$ contains these axioms then $K \models C_\bot \equiv \bot$.

## 4 Formal Description of the Pattern

In Section 2, we introduced map scaling as seamless transfer of information in maps from one level of detail to another. This idea can be understood in terms of a *binary scaling function*. In this section, we will describe this idea first as a simple functional pattern (see Figure 1), illustrate it with examples, and then formalize it in DL such that it can published as ontology vocabulary (see Figure 2).

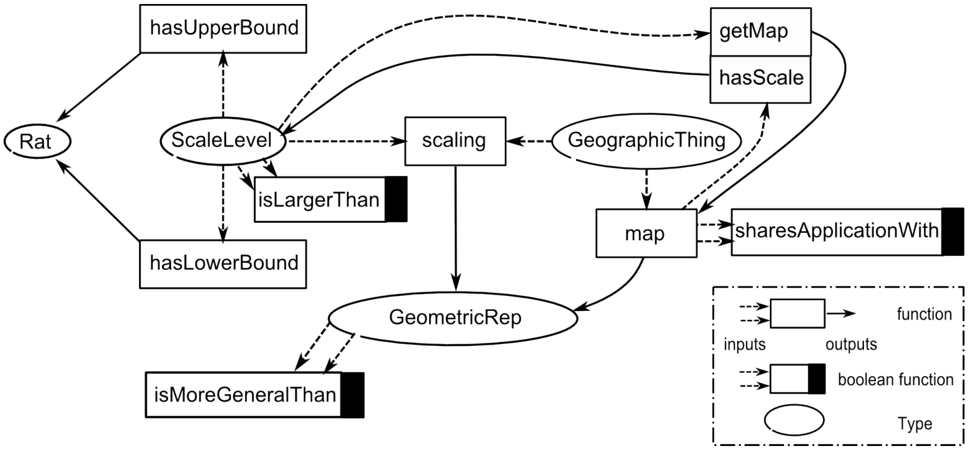

**Fig. 1.** The map scaling pattern in functional notation. Boxes denote functions, ellipses denote types of entities. Dotted arrows indicate input types and full arrows output types of functions. Relations are boolean functions.

The functional pattern in Figure 1 allows to quickly grasp the formal constraints in terms of functions (denoted as boxes) that map various input types to a single output (types denoted as ellipses). Along with the explanation of this pattern, the challenge addressed here is to translate these general functions into the previously described DL language DLP$_\exists$, i.e., to translate Figure 1 into Figure 2[11].

Simply put, map scaling applications provide *more or less generalized geometric representations* of *geographic phenomena* for different *scale levels*. That is, we need to deal with the primitive types of things listed in Table 3. The formal relationship between these types, which is manifest in map scaling applications, can be captured by a *scaling function* and *two ordering relations*. The former can be understood as a binary function from geographic phenomena and scale levels into geometric representations. The latter account for scale dependent orderings of scale levels and data representations. The numeric scale/resolution boundaries which correspond to a single scale level are given by *hasUpperBound* and *hasLowerBound*, respectively. The scaling function is specific for a certain scaling application. Other relations (such as *hasScale*, *getMap* and *sharesApplicationWith*) and types (e.g. *MapData*) can be based on this formal apparatus. For example, in this pattern, a *map* simply corresponds to a the projection of a scaling function to a fixed scale level.

The corresponding DL ontology pattern is described in Figure 2. Since DL does not support the specification of arbitrary functions, *scaling functions in our pattern are indirectly represented as subgraphs consisting of reified 3-tuples.* A *scaled representation* is a reified tuple of geographic thing, scale level and geometric representation. This work around requires the introduction of a new class *ScaledRep*, as well as its outgoing properties *representsObject*, *isScaled* and *isPresentedAs*, denoting the three slots in each 3-tuple. Maps need to be introduced as a primitive class, and we need to explicitly assert that they are constituted of scaled representations and have a single scale level. Furthermore, maps are assigned a single scale level which can only be shared within the same application. Our pattern also involves further formal constraints which will be discussed along the next paragraphs in terms of axioms.

Our pattern does not include concept hierarchies, because they are not primarily relevant for scaling. Scale/generalization orders and concept subsumption, even

[11] *General higher order functions*, which are used here as in functional programming, e.g. http://www.cl.cam.ac.uk/research/hvg/Isabelle/, are not formalizable in DL.

**Table 3.** Primitive types of entities involved in the functional pattern.

| **Name** | **Formal Type** | **Explanation** |
|---|---|---|
| Geographic phenomenon | *GeographicThing* | Type of mappable phenomenon, may be categorized by geo-ontologies and may be aggregated to *layers* |
| Geometric representation | *GeometricRep* | Type of data representation of a phenomenon which involves a geometric part (raster or vector representation) as well as a (map-) symbolization part |
| Scale level | *ScaleLevel* | Type of discrete level of detail |
| *Scaling* | *ScaleLevel* ⇒ *GeographicThing* ⇒ *GeometricRep* | Type of scaling function |

though related, are two separate things. However, the latter may be introduced through geo-ontologies of phenomena.

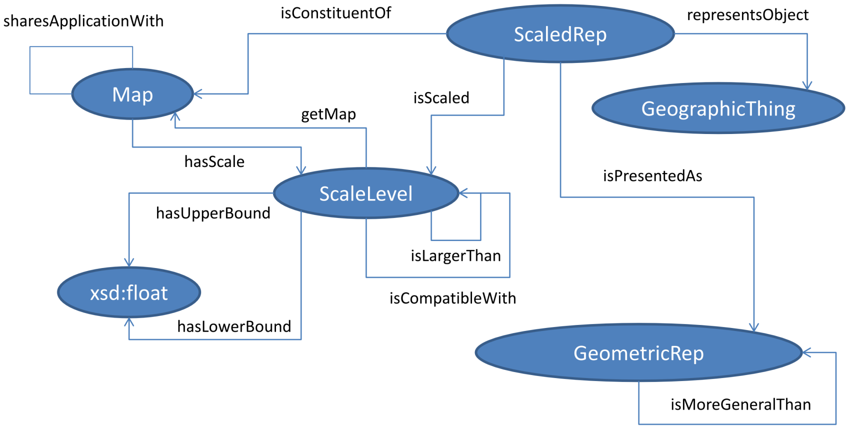

**Fig. 2.** The DL ontology design pattern corresponding to the functional map scale pattern. Ellipses stand for DL classes. Arrows represent DL properties that relate classes.

**Maps and Applications.** As mentioned above, we define maps as being constituted of scaled representations of phenomena at a fixed scale. This notion of a map captures the idea that maps are actually *semiotic signs*, i.e., they supply a unique cartographic representation of phenomena as referents.[12] In the ontology designed to fit the pattern a map can be seen as a set of scaled representation individuals.

The maps that stem from a certain scaling function (the former being projections of the latter to fixed scale levels) are considered part of the same scaling application. This leaves open the possibility that there may be different scaling applications for the same set of geographic phenomena but for different purposes. We express this relation making use of the property *sharesApplicationWith* which relates those maps that are part of the same application based on whether they derive from the same

[12] However, there are also other ways to capture the notion of cartographic maps. For example, from a GIS viewpoint, a map may be defined as a particular collection of map data.Or, from a cartographic point of view, maps may be seen as a kind of cartographic visualization or image.

scaling function. Property *sharesApplicationWith* is declared to be transitive (1), symmetric (2), and reflexive (3) with respect of the individuals within the class *Map*:

$$\mathit{sharesApplicationWith} \circ \mathit{sharesApplicationWith} \sqsubseteq \mathit{sharesApplicationWith} \tag{1}$$

$$\mathit{sharesApplicationWith}^{-} \sqsubseteq \mathit{sharesApplicationWith} \tag{2}$$

$$\mathit{Map} \sqsubseteq \exists \mathit{sharesApplicationWith}.\mathit{Self} \tag{3}$$

We define property *hasScale* as the function which delivers the unique scale level of a given map. This property is defined to be functional, since every map is associated with a single scale. We also define *getMap* as the inverse property of *hasScale* which allows us to retrieve the map associated with a given scale. These constraints are enforced using axioms (4) and (5).

$$\top \sqsubseteq \leq 1 \mathit{hasScale}.\top \tag{4}$$

$$\mathit{hasScale}^{-} \sqsubseteq \mathit{getMap} \tag{5}$$

We impose a less restrictive form of functionality over property *getMap*. We have that this property is functional over the set of maps that belong to the same application, which defined in our pattern as the set of maps that are connected through the *sharesApplicationWith*. Note that this property is declared to be both symmetric and transitive. I.e. a given scale $s_1$ cannot be shared by maps $m_1$ and $m_2$ if we have that *sharesApplicationWith*$(m_1, m_2)$ is entailed by the ontology. We enforce this constraint with axiom 6 which automatically collapses into one single individual all maps associated to the same scale that are within the same application.

$$\top \sqsubseteq \leq 1 (getMap \circ sharesApplicationWith).\top \tag{6}$$

Due to these constraints we have that for a given application, there is only one map at an specific scale level. The rationality behind this constraint is to eliminate all ambiguity at the time of representing map data at a given scale over the same application.

We have defined our ontology to allow for an easy retrieval of all the information pertaining to a single map. We can make use of the property connexion *isConstituentOf* to retrieve and query about all the existing scaled representations associated with a given map. Property *isScaled*, which links every scaled representation with the scale associated to the map this one belongs to, is automatically generated due to axiom (7).

$$\mathit{isConstituentOf} \circ \mathit{hasScale} \sqsubseteq \mathit{isScaled} \tag{7}$$

After imposing these restrictions over applications, maps and scales, we elaborate about *scale levels* and *geographic representations*, which are ordered in a chain-like manner.

**Orders on Scale Levels and Geometric Representations.** We enforce a strict partial order over properties *isLargerThan* and *isMoreGeneralThan* which respectively connect (and order) individuals over the classes *Scale* and *GeometricRep*.

$$isLargerThan \circ isLargerThan \sqsubseteq isLargerThan \tag{8}$$

$$\exists (isLargerThan \sqcap isLargerThan^{-}).\top \sqsubseteq \bot \tag{9}$$

$$isMoreGeneralThan \circ isMoreGeneralThan \sqsubseteq isMoreGeneralThan \tag{10}$$

$$\exists (isMoreGeneralThan \sqcap isMoreGeneralThan^{-}).\top \sqsubseteq \bot \tag{11}$$

As usual, a strict partial order is a binary relation that is irreflexive and transitive, and therefore antisymmetric. We enforce transitivity of properties *isLargerThan* and *isMoreGeneralThan* with axioms 8 and 10 respectively. Axioms 9 and 11 enforce irreflexivity of both properties[13] also enforcing antisymmetry.

Furthermore, we add a *similarity relation* among scale levels (i.e., one that is symmetric and reflexive), which allows us to connect compatible scales across different applications. Similarity among scale levels allows us to merge data from different applications, each having its separate scale level chain. Note that the computation of this similarity relation may be done in various ways based on their numerical scale boundaries, and we deliberately leave open in our pattern how this may be done. The similarity relation is represented in the pattern by the *isCompatibleWith* relation which is defined to be symmetric (12) and reflexive (13) connecting all individuals in class *ScaleLevel* with themselves.

$$isCompatibleWith^{-} \sqsubseteq isCompatibleWith \tag{12}$$

$$ScaleLevel \sqsubseteq \exists isCompatibleWith.Self \tag{13}$$

We assume that each scale level has one *upper and lower bound* in terms of numeric map scales or pixel resolutions. The latter are simply rational numbers. Every scale has at most one upper bound and one lower bound and therefore both properties *hasLowerBound* and *hasUpperBound* are declared functional. The defined constraint is enforced with axioms:

$$ScaleLevel \sqsubseteq \exists hasLowerBound.xsd{:}float \tag{14}$$

$$ScaleLevel \sqsubseteq \exists hasUpperBound.xsd{:}float \tag{15}$$

$$\top \sqsubseteq \leq 1 hasLowerBound.\top \tag{16}$$

$$\top \sqsubseteq \leq 1 hasUpperBound.\top \tag{17}$$

We skip the constraint that their order needs to comply with (or even defines) the scale level order. To improve the understanding of the publication we have not included datatypes within DLP$_\exists$, necessary to deal with algebraic operations over the *xsd:float* class. Once done it would not be difficult to verify than the existing *isLargerThan* relationships are valid, and to automatize the creation of this relationship between the existing scales.

Now we come to the most essential part of the pattern, namely representing the *scaling function*, compare Table 3 and Figure 1. The scaling function allows to switch to a new data representation for all phenomena by changing the scale level. This is done in a *monotonic* manner, i.e., such that the ordering of scale levels is preserved in generalization levels. Put differently, scaling to a larger scale level excludes the possibility that representations become more general. We include the possibility that a phenomenon is not represented at all at certain scale levels by creating an empty *GeometricRep* individual, as well as the possibility that it may be represented in a constant manner.

**Monotonicity of Scaling.** To enforce this constraint using OWL we make use of the class *ScaledRep*,and we add a logical equivalent to the following first order logic

[13] We have that $(\top \sqsubseteq \neg\exists R_1, Self) \equiv \top \sqsubseteq \exists(R_1 \sqcap R_1^{-}).\bot$ for any property $R_1$.

rule to the ontology:

$$\begin{aligned}&\mathit{sharesApplicationWith}(m_x, m_y) \land \mathit{hasScale}(s_y, m_y) \land \mathit{hasScale}(s_x, m_x)\\ &\land \mathit{isLargerThan}(s_x, s_y)\land\\ &\mathit{isConstituentOf}(m_x, sr_x) \land \mathit{isConstituentOf}(m_y, sr_y)\land\\ &\mathit{representsObject}(sr_x, g) \land \mathit{representsObject}(sr_y, g)\land\\ &\mathit{isPresentedAs}(sr_x, grr_x) \land \mathit{isPresentedAs}(sr_y, grr_y)\land\\ &\mathit{isMoreGeneralThan}(grr_x, grr_y) \rightarrow \bot(m_x)\end{aligned}$$

This rule enforces that the ontology becomes inconsistent if

- there exist maps $m_1$ and $m_2$ belonging to the same application with scales $s_1$ and $s_2$,
- scale $s_1$ is larger than scale $s_2$,
- maps $m_1$ and $m_2$ contain scaled representations $sr_1$ and $sr_2$ that represent the same geographic thing $g$, and
- the geographic representation record $grr_1$ for $sr_1$ is more general than the one for $sr_2$, namely $grr_2$.

Although this rule may look quite complex it is indeed expressible in OWL. An automatized way of performing this transformation is presented in [3], as well as a procedure to check if a given rule is indeed expressible in OWL. A possible set of DL axioms equivalent to the previous rule is:

$$\begin{aligned}\mathit{hasScale}^{-} \circ \mathit{sharesApplicationWith} \circ \mathit{hasScale} &\sqsubseteq R_1\\ R_1 \sqcap \mathit{isLargerThan} &\sqsubseteq R_2\\ \mathit{isScaled} \circ R_2 \circ \mathit{isScaled} &\sqsubseteq R_3\\ \mathit{isPresentedAs} \circ \mathit{isMoreGeneralThan}^{-} \circ \mathit{isPresentedAs}^{-} &\sqsubseteq R_4\\ \mathit{representsObject} \circ \mathit{representsObject}^{-} &\sqsubseteq R_5\\ R_3 \sqcap R_4 \sqcap R_5 &\sqsubseteq R_\bot\\ \exists R_\bot.\top &\sqsubseteq \bot\end{aligned}$$

where all $R_i$ are freshly introduced roles that do not appear previously in the ontology.

**Functionality of Scaling.** Next, we enforce *functionality* constraints on the subgraph of scaled representations which denotes a scaling function, i.e., a singular scaling application.

First, we make use of simple OWL axioms to enforce functionality for the properties *isPresentedAs*, *isScaled*, and *representsObject*, which respectively connect a scaled representation with the geometric representation, the scale, and the geographic thing it is associated with.

$$\top \sqsubseteq \leq 1 \mathit{isPresentedAs}.\top \tag{18}$$

$$\top \sqsubseteq \leq 1 \mathit{isScaled}.\top \tag{19}$$

$$\top \sqsubseteq \leq 1 \mathit{representsObject}.\top \tag{20}$$

Second, a scaled representation is enforced to have a geometric representation, a scale, and a geographic thing associated to it. This is enforced using OWL axioms:

$$\mathit{ScaledRep} \sqsubseteq \exists \mathit{isPresentedAs}.\mathit{GeometricRep} \tag{21}$$

$$\mathit{ScaledRep} \sqsubseteq \exists \mathit{isScaled}.\mathit{ScaleLevel} \tag{22}$$

$$\mathit{ScaledRep} \sqsubseteq \exists \mathit{representsObject}.\mathit{GeographicThing} \tag{23}$$

And third, since the relation expressed by scaled representations stands for a scaling function, it needs to be restricted to be *functional* with respect to a scale level and a geographic phenomenon represented. Furthermore, since a scaling function corresponds to a particular scaling application, we need to restrict functionality to only those scaled representations that are part of a single scaling application.

Since every scale is only associated to one map within the same application we only need to verify that there only exists one scaled representation for each geographic phenomenon. This constraint is enforced with axioms (24) and (25).

$$isConstituentOf^{-} \circ representsObject \circ representsObject^{-} \sqsubseteq R_{aux} \tag{24}$$

$$\top \sqsubseteq \leq 1(R_{aux} \sqcap isConstituentOf^{-}).\top \tag{25}$$

Due to axioms (24) and (25) we have that two different scaled representations are collapsed into a single one if they are constituents of a given map and represent the same object. Therefore we are guaranteed that, within the same map there only exists one scaled representation representing the same geographic thing. Given that functionality only needs to be enforced across the maps within the same application, and that by previous restrictions we have that there are no two maps with the same scale within the same application we have that the mapping to a geometric representation is functional depending on the specific geographic thin represented and the scale for any given scale representation.

We show a translation of the axioms presented in this section in Appendix, which is part of the technical report of the paper. In the translation we also establish a set of domain and range restrictions based on the relations between properties and roles in Figure 2. As further constraints, we also declare all classes defined in the pattern to be disjoint in order to avoid possible mistakes in the declaration of individuals.

## 5 Application Scenario

When interpreting and comparing maps such as the global kernel density map (raster data)[14] in Fig. 3, where the color ramp of each pixel represents the *fraction of malaria transmitting mosquitos* [7], it is crucial to take into account the effect of scale. At its original resolution [15], each single pixel covers a width of approximately 17 kilometers at the equator. Assuming that the malaria data is displayed at a screen pixel size of 0,25 mm, the map image has an appropriate maximal scale of 1:68000000. Thus, for instance, using such a map to determine whether a particular village (many of which could be contained in a single pixel) is affected to a higher degree than others is difficult, since villages are represented on much larger scale levels.

Similarly, as mosquitos require water for reproduction, one may be tempted to combine such a map with a river network layer. As in the case before, this particular map is too coarse to support a meaningful comparison, since river width is a fraction of the size of a pixel. Instead, scholars have to go back to the source data and generate raster data at a scale which is appropriate for the scale of river networks. While this is a simple operation for Geographic Information Systems (GIS), and may be automatized in terms of a map scaling service, the scale increase cannot be chanced ad libitum but has also a lower bound. While the upper bound is constrained by the process of cartographic abstraction mentioned before, the lower bound is limited by the measurement procedure. In this specific case, data was coded at the village level during the field study, and thus representing the data at a larger scale of, say, 1:2500, would create a misleading impression of accuracy.

[14] ©2010 Malaria Atlas Project, available under the Creative Commons Attribution 3.0 Unported License

[15] Keeping in mind what we said about zooming and scale in digital images.

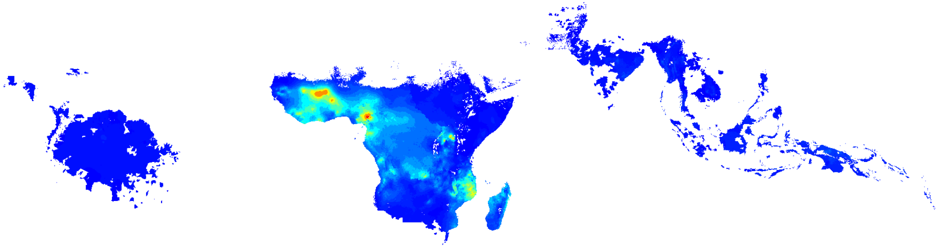

**Fig. 3.** Global kernel density map of the fraction of malaria infected mosquitos. The spectrum blue-yellow-orange-red denotes an increase of this fraction.

The question that we address here is whether the decision about appropriateness of scaled representations of phenomena can be automated in the Semantic Web, independently from and without (manual) interaction with particular scaling applications. In the following, we demonstrate how such a decision may be computed based on our pattern.

Suppose we have an *ABox* which describes map data from different scaling applications of the kind discussed above together with its scale level:

$$\begin{array}{cc}
\mathit{ScaleLevel}(\mathit{s1}) & \mathit{GeographicThing}(\mathit{malaria}) \\
\mathit{GeometricRep}(\mathit{raster}) & \mathit{ScaledRep}(\mathit{sr1}) \\
\mathit{representsObject}(\mathit{sr1}, \mathit{malaria}) & \mathit{isScaled}(\mathit{sr1}, \mathit{s1}) \\
\mathit{isPresentedAs}(\mathit{sr1}, \mathit{raster}) & \\
\mathit{ScaleLevel}(\mathit{s2}) & \mathit{Village} \sqsubseteq \mathit{GeographicThing} \\
\mathit{GeometricRep}(\mathit{polygon}) & \mathit{Village}(\mathit{village1}) \\
\mathit{representsObject}(\mathit{sr2}, \mathit{village1}) & \mathit{ScaledRep}(\mathit{sr2}) \\
\mathit{isPresentedAs}(\mathit{sr2}, \mathit{polygon}) & \mathit{isScaled}(\mathit{sr2}, \mathit{s2}) \\
\mathit{Map}(\mathit{m1}) & \mathit{Map}(\mathit{m2}) \\
\mathit{isConstituentOf}(\mathit{sr1}, \mathit{m1}) & \mathit{isConstituentOf}(\mathit{sr2}, \mathit{m2}) \\
\mathit{isCompatibleWith}(\mathit{s2}, \mathit{s1}) &
\end{array}$$

The data provided may come from different users that uploaded data from different applications. We assume there is one user that wants to merge existing information about the malaria and existing villages. We define a new subclass *Village* of the general class *GeographicThing*. The user can now query for all existing villages that are represented on a compatible scale together with *sr2*, which is the scaled representation of malaria.

$$(x?) : \exists \mathit{representObject}.\mathit{Village} \sqcap \exists(\mathit{isScaled} \circ \mathit{isCompatible} \circ \mathit{isScaled}^{-}).\{\mathit{sr2}\}$$

The query will retrieve all scaled representations that represent a *Village* type individual and have a compatible scale with *sr2*. The user can then select the most appropriate for his visualization of the data. Making use of the *xsd:float* values associated with a scale allows also to retrieve the scaled representations within a certain range.

## 6 Discussion and Conclusion

In this paper, we suggested a formal ontology design pattern that describes cartographic map scaling on a semantic level in terms of a functional relationship among geometric representations, phenomena and scale levels. Map scaling applications are used on the web to represent and display phenomena at different scale levels. In the Semantic Web, the notions of *scale* and *resolution* have, to the best of our knowledge, not been introduced so far, even though they are of central importance to deal with information at different levels of granularity. Scale, granularity and resolution are central notions of cartography [15] and GIScience [14], however, formal approaches to describe map scaling are mostly focused on mathematical models of generalization and granularity change [20], not on making publicly available the application logic of actual scaling systems.

In the Web, granularity levels are needed to improve performance of querying, reasoning, as well as in order to display information meaningfully on a map. The challenge lies in preventing mashups of data at inappropriate resolutions, or visual clutter across scaling applications, as well as in enabling the tracing of geographic phenomena across different levels of detail and across different applications. Opening up existing scaling implementation logic for the Semantic Web not only allows cross-linking web map services based on geographic phenomena, it also has the potential to make the Semantic Web itself scale across different levels of detail. This is because it adds the crucial information about whether certain information can be used on certain scale levels or not. This information today seems to be missing from the Semantic Web. For example, the geographic reference of DBpedia or Linked Geodata [19] is a scale-free coordinate point.

In the paper, we proposed formal constraints to the pattern in a tractable fragment of DL, which can be used to compute inferences on *ABox* descriptions of actual scaling applications. For example, we showed that it is possible to check automatically whether data representations from scaling applications are compatible with respect to their scale levels, and thus, can be meaningfully displayed in a single map. The constraints also allow to check consistency of a single scaling application, e.g., with respect to monotonicity and functionality of scaling. Future research may enrich the axiomatization based on a full functional specification in HOL, which could only be sketched in Figure 1. It may also address scalable reasoners for $\mathrm{DLP}_\exists$, which would allow testing the pattern on a set of scaling applications described by the pattern. Even without computational reasoning, the pattern can be directly used to annotate and query existing scaling applications based on RDF. Furthermore, the pattern may be specialized by complementary patterns describing geometric data formats as known in GIScience, different geographic ontologies, the relation of map displays and scale, as well as different notions of scale and resolution.

**Acknowledgements.** This work is a collaborative outcome of the GeoVoCampDayton2012[16]. Some of the authors credit funding from European Commission (ICT-FP7-249120 ENVISION project), as well as the German Research Foundation (Research fellowship grant DFG SCHE 1796/1-1). Authors from Wright State University acknowledge funding from by the National Science Foundation under award 1017225 III: Small: TROn – Tractable Reasoning with Ontologies.

## References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
2. Bittner, T., Donnelly, M., Smith, B.: A spatio-temporal ontology for geographic information integration. Int. J. Geogr. Inf. Sci. 23(6), 765–798 (Jun 2009)

[16] `http://vocamp.org/wiki/GeoVoCampDayton2012`

3. Carral-Martínez, D., Hitzler, P.: Extending description logic rules. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, Ó., Presutti, V. (eds.) ESWC. Lecture Notes in Computer Science, vol. 7295, pp. 345–359. Springer (2012)
4. Degbelo, A., Kuhn, W.: A Conceptual Analysis of Resolution. In: XIII GEOINFO, Brazilian Symposium on Geoinformatics, Campos do Jord ao. pp. 11–22. Sao Paolo, Brazil (2012)
5. Frank, A.: Scale is introduced in spatial datasets by observation processes. In: Spatial Data Quality From Process to Decision(6th ISSDQ 2009). pp. 17–29. CRC Presss (2009)
6. Gangemi, A., Presutti, V.: Towards a pattern science for the semantic web. Semantic Web 1(1-2), 61–68 (2010)
7. Gething, P., Patil, A., Smith, D., Guerra, C., Elyazar, I., Johnston, G., Tatem, A., Hay, S.: A new world malaria map: Plasmodium falciparum endemicity in 2010. Malaria Journal 10(1), 378 (2011)
8. Goodchild, M.F., Proctor, J.: Scale in a digital geographic world. Geographical and Environmental Modelling pp. 5–23 (1997)
9. Grosof, B., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proc. 12th Int. Conf. on World Wide Web (WWW'03). pp. 48–57. ACM (2003)
10. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006). pp. 57–67. AAAI Press (2006)
11. Kazakov, Y.: Saturation-Based Decision Procedures for Extensions of the Guarded Fragment. Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany (March 2006)
12. Krisnadhi, A., Maier, F., Hitzler, P.: OWL and rules. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web. Semantic Technologies for the Web of Data. 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures. Lecture Notes in Computer Science, vol. 6848, pp. 382–415. Springer, Heidelberg (2011)
13. Krötzsch, M., Simancik, F., Horrocks, I.: A description logic primer. CoRR abs/1201.4089 (2012)
14. Kuhn, W.: Core Concepts of Spatial Information: A First Selection. In: XII GEOINFO, November 27-29, 2011, Campos do Jordão, Brazil. pp. 13–26 (2011)
15. Mcmaster, R.B., Shea, K.S.: Cartographic Generalization in a Digital Environment: a Framework for implementation in a GIS. In: GIS/LIS'88. pp. 240–249. San Antonio, Texas, USA (1988)
16. M̈ller, J., Lagrange, J., Weibel, R.: GIS and Generalization: Methodology and Practice. Taylor and Francis (1989)
17. Montello, D.: Scale in Geography. In: International Encyclopedia of the Social and Behavioral Sciences, pp. 13501–13504 (2001)
18. Scheider, S., Kuhn, W.: Affordance-based categorization of road network data using a grounded theory of channel networks. International Journal of Geographical Information Science 24(8), 1249–1267 (2010)
19. Stadler, C., Lehmann, J., Höffner, K., Auer, S.: LinkedGeoData: A core for a web of spatial open data. Semantic Web (2012)
20. Stell, J., Worboys, M.: Stratified map spaces: A formal basis for multi-resolution spatial databases. In: SDH'98 Proceedings 8th International Symposium on Spatial Data Handling. pp. 180–189 (1998)
21. Wu, J., Li, H.: Concepts of scale and scaling, pp. 3–15. Springer (2006)

## A Ontology Pattern in Functional Syntax

$$\begin{aligned}&\mathit{SubObjectPropertyOf}\,(\\&\quad \mathit{ObjectPropertyChain}\ (\ \mathit{sharesApplicationWith}\ \mathit{sharesApplicationWith}\ )\\&\quad \mathit{sharesApplicationWith})\end{aligned}\tag{26}$$

$$\begin{aligned}&\mathit{SubObjectPropertyOf}\,(\\&\quad \mathit{ObjectPropertyChain}\ (\ \mathit{isConstituentOf}\ \ \mathit{hasScale}\ \ )\\&\quad \mathit{isScaled})\end{aligned}\tag{27}$$

$$\mathit{SubObjectPropertyOf}\,(\mathit{ObjectInverseOf}\ (\ \mathit{hasScale}\ )\ \mathit{getMap}\ )\tag{28}$$

$$\mathit{SymmetricObjectProperty}\ (\ \mathit{sharesApplicationWith}\ )\tag{29}$$

$$\mathit{SymmetricObjectProperty}\ (\ \mathit{isCompatibleWith}\ )\tag{30}$$

$$\mathit{SubClassOf}\ (\ \mathit{Map}\ \mathit{ObjectHasSelf}\ (\ \mathit{sharesApplicationWith}\ )\ )\tag{31}$$

$$\begin{aligned}&\mathit{SubClassOf}\ (\ \mathit{ScaledRep}\\&\qquad\qquad \mathit{ObjectSomeValuesFrom}\ (\ \mathit{isPresentedAs}\ \mathit{GeographicRecord}\ )\ )\end{aligned}\tag{32}$$

$$\begin{aligned}&\mathit{SubClassOf}\ (\ \mathit{ScaledRep}\\&\qquad\qquad \mathit{ObjectSomeValuesFrom}\ (\ \mathit{isScaled}\ \mathit{Scale}\ )\ )\end{aligned}\tag{33}$$

$$\begin{aligned}&\mathit{SubClassOf}\ (\ \mathit{ScaledRep}\\&\qquad\qquad \mathit{ObjectSomeValuesFrom}\ (\ \mathit{representsObject}\ \mathit{GeographicThing}\ ))\end{aligned}\tag{34}$$

$$\mathit{SubClassOf}\ (\ \mathit{Scale}\ \mathit{DataSomeValuesFrom}\ (\ \mathit{hasLowerBound}\ \mathit{xsd{:}float}\ )\ )\tag{35}$$

$$\mathit{SubClassOf}\ (\ \mathit{Scale}\ \mathit{DataSomeValuesFrom}\ (\ \mathit{hasUpperBound}\ \mathit{xsd{:}float}\ )\ )\tag{36}$$

$$\begin{aligned}&\mathit{FunctionalObjectProperty}\ (\ \mathit{ObjectPropertyChain}\\&\qquad\qquad (\mathit{getMap}\ \mathit{sharesApplicationWith}\ )\ )\end{aligned}\tag{37}$$

$$\mathit{FunctionalObjectProperty}\ (\ \mathit{hasScale}\ )\tag{38}$$

$$\mathit{FunctionalObjectProperty}\ (\ \mathit{isPresentedAs}\ )\tag{39}$$

$$\mathit{FunctionalObjectProperty}\ (\ \mathit{isScaled}\ )\tag{40}$$

$$\mathit{FunctionalObjectProperty}\ (\ \mathit{representsObject}\ )\tag{41}$$

$$\mathit{FunctionalDataProperty}\ (\ \mathit{hasUpperRatio}\ )\tag{42}$$

$$\mathit{FunctionalDataProperty}\ (\ \mathit{hasLowerRatio}\ )\tag{43}$$

$$\mathit{TransitiveObjectProperty}\ (\ isScaled\ )\tag{44}$$

$$\mathit{TransitiveObjectProperty}\ (\ isLargerThan\ )\tag{45}$$

$$\mathit{TransitiveObjectProperty}\ (\ isMoreGeneralThan\ )\tag{46}$$

## B Tractability of reasoning for the Scaling pattern

We consider atomic query problem as the basic reasoning task for the Scaling pattern. Let $\mathcal{T}_0$ be the TBox containing precisely all axioms for the Scaling pattern as defined in Section 4. Furthermore, let $\mathcal{K}$ be a knowledge base where $\mathcal{T}_0 \subseteq \mathcal{K}$. We call $\mathcal{K}$ an *applied scaling pattern* which contains not only the pattern, but also additional axioms added when the user uses the pattern for some application. These additional axioms include data (i.e., ABox axioms) as well as schema knowledge (i.e., TBox axioms) obtained through introducing fresh class/property names as a subclass/subproperty of some class/property names from $\mathcal{T}_0$, as demonstrated in Section 5. Denote by $Sig(\mathcal{K})$ the signature of $\mathcal{K}$ and $Ind(\mathcal{K})$ the set of named individuals occurring in $\mathcal{K}$. Using $\mathcal{T}_0$ as pattern means that the class and property names occurring in $\mathcal{T}_0$ must not be redefined by the user. In addition, as mentioned in Section 3, the use of existential restriction needs to be restricted to ensure tractability. Hence, the following restriction(*) is imposed on any applied scaling pattern $\mathcal{K}$: for every class inclusion axiom $\alpha \sqsubseteq \beta \in \mathcal{K} \setminus \mathcal{T}_0$ (i.e., those added by the user of the pattern), $Sig(\alpha) \cap (\{\top\} \cup Sig(\mathcal{T}_0)) = \emptyset$ and $\beta$ contains no existential restriction $\exists R.C$ for any class $C$ and property $R$.

An *atomic query* $q$ over a knowledge base $\mathcal{K}$ is an expression of the form $C(x)$ or $R(x, y)$ where $x, y$ are variables individuals and $C$ (resp. $R$) is a class (resp. property) names from $Sig(\mathcal{K})$. Denote by $Var(q)$ the set of variables occurring in $q$. Let $\mu\colon Var(q) \to Ind(\mathcal{K})$ be a total mapping and, by slight abuse of notation, $\mu(q)$ be an expression obtained from $q$ by substituting every occurrence of variables $x \in Var(q)$ with $\mu(x)$. An *answer* to $q$ is such a mapping $\mu$ such that $\mathcal{K}$ entails $\mu(q)$, written $\mathcal{K} \models \mu(q)$, i.e., every model of $\mathcal{K}$ is a model of $\mu(q)$. The *atomic query problem* for the Scaling pattern is the problem of finding all answers to a given query $q$ over $\mathcal{K}$ where $\mathcal{K}$ is an applied scaling pattern.

Let $\mathcal{K}$ be partitioned into $\mathcal{K}_{rl}$ and $\mathcal{K}_\exists$ where $\mathcal{K}_\exists$ contains only the axioms below:

$$\mathit{ScaleLevel} \sqsubseteq \exists \mathit{hasLowerBound.xsd{:}float} \tag{47}$$

$$\mathit{ScaleLevel} \sqsubseteq \exists \mathit{hasUpperBound.xsd{:}float} \tag{48}$$

$$\mathit{ScaledRep} \sqsubseteq \exists \mathit{isPresentedAs.GeometricRep} \tag{49}$$

$$\mathit{ScaledRep} \sqsubseteq \exists \mathit{isScaled.ScaleLevel} \tag{50}$$

$$\mathit{ScaledRep} \sqsubseteq \exists \mathit{representsObject.GeographicThing} \tag{51}$$

Observe that, unlike $\mathcal{K}_\exists$, $\mathcal{K}_{rl}$ can be rewritten into a set of rules in a Datalog language with equality where the Datalog language uses the class and property names as predicate symbols. The rewriting also applies to axioms which involve role conjunction because any role conjunction $R_1 \sqcap R_2$ can be rewritten as a conjunction $R_1(x, y) \wedge R_2(x, y)$. (Examples of such a rewriting can be found in [9,12,3]).

Let $P$ be a sound and complete decision procedure for query answering in Datalog with equality and $q$ be an atomic query over $\mathcal{K}$. If $Sig(q) \cap (Sig(\mathcal{K}_\exists) \setminus \{\mathit{ScaledRep}\}) = \emptyset$, then an answer to $q$ can be found by simply translating $\mathcal{K}_{rl}$ and $q$ into Datalog and then compute an answer to $q$ in Datalog framework using $P$. Tractability of answering $q$ for this case then follows from the (data-)tractability of query answering in Datalog.

For the case where $Sig(q) \cap (Sig(\mathcal{K}_\exists) \setminus \{\mathit{ScaledRep}\}) \neq \emptyset$, using the step as in the earlier case is not complete since there can be additional answers that can be found by also considering $\mathcal{K}_\exists$. We then use the following observation(**) on $\mathcal{K}$ which can easily be verified (and holds also due to restriction (*) given in the first paragraph of this section): there is no chain of existential restrictions $A \sqsubseteq \exists R_1.A_1, A_1 \sqsubseteq \exists R_2.A_2, \ldots, A_n \sqsubseteq \exists R_{n+1}.A$ such that they are all entailed by $\mathcal{K}$ where $A \in Sig(\mathcal{K}_\exists)$ and all $A_i, R_i \in Sig(\mathcal{K})$.This observation implies that the number of new individuals that must be created because of implied existential restriction is finite. In particular, because the only axioms involving existential restriction are those in $\mathcal{K}_\exists$, additional answers to $q$ can be found using the procedure $P$ by considering the following (assuming $Ind(\mathcal{K}) = \{a_1, \ldots, a_n\}$):

(i) For each named individuals $a_i \in Ind(\mathcal{K})$, $i = 1, \ldots, n$, create five fresh named individuals: $l_i, u_i, s_i, r_i, t_i$.

(ii) Find answers to the query $\hat{q} = \mathit{ScaledRep}(z)$ for a fresh variable $z$. Then, $\mathcal{K}'$ is obtained from $\mathcal{K}$ by adding the following ABox axioms to $\mathcal{K}$ for every $j$ such that $\mu_j$ is an answer $\hat{q}$ with let $a_j = \mu_j(x)$:

$$\begin{aligned}
&ScaleLevel(s_j), GeometricRep(r_j), GeometricThing(t_j),\\
&xsd{:}float(l_j), xsd{:}float(u_j),\\
&hasLowerBound(s_j, l_j), hasUpperBound(s_j, u_j)\\
&isScaled(a_j, s_j), isPresentedAs(a_j, r_j), representsObject(a_j, t_j)
\end{aligned}$$

Hence, answers to $q$ over $\mathcal{K}$ are precisely answers to $q$ over $\mathcal{K}'$.

The above steps ensure the soundness and completeness for solving the atomic query problem. Moreover, it is not hard to see that the complexity of reasoning still does not increase compared to the complexity of $P$ due to the fixed-size blow up of the knowledge base, i.e., tractability of answering $q$ for this case also follows from the (data-)tractability of query answering in Datalog. It is also important to note that the above steps can either be applied for a bottom-up version or a top-down version of $P$.

Finally, for a complex query expression such as the one presented in Section 5, it suffices to add one additional step, namely rewriting the query expression into a conjunctive query in Datalog, i.e., a conjunction of atomic query. The rewriting procedure is the same as the rewriting for $\mathcal{K}$. Complexity consideration is also the same as for the atomic case.