Representation of Multi-Dimensional Quantities, Time-Series, and More*

Maxime Lefrançois

École Nationale Supérieure des Mines, FAYOL-ENSMSE, Laboratoire Hubert Curien, F-42023 Saint-Étienne, France
maxime.lefrancois@emse.fr

Abstract. There are so many domains that need ontologies of quantities, units of measure, and to represent multi-dimensional quantities. Currently, no formal RDF model has been proposed that enable to describe systems that are multiple quantified, and quantities having values that evolve on multi-dimensional domains. Quantities may be probabilstic distributions, points, or intervals.

Keywords: literals, datatypes, RDF, linked data

1 Introduction

This paper introduces the ontology of quantities that has been developed for the ITEA 2 SEAS project. With respect to other ontologies that describe quantities, this ontology shows several important contributions:

- it enables to represent quantities defined on algebraic rings: booleans, reals, or even complex quantities, which is of utmost importance for scientific domains.
- it uses representation of Units as custom datatypes that can be recognized on the fly by RDF processors.
- its semantics is well defined, and grounded on probability theory;
- it enables to represent time-series, or actually a generalization for series of any quantity;
- it enables to represent aggregation of quantities;
- it enables to represent growth rate, derivatives, sum, and jumps of quantities with respect to other quantities;
- the part of the ontology that represent specific dimensions and quantities needed for a domain (Smart Grids in the case of the SEAS project), is automatically generated by a JSON configuration file. This enables to ease the understanding and the maintenance of this ontology by domain experts;
- the web platform where it is published enables any project to post configuration files, and generates a new corresponding part of the ontology on the fly. This makes this ontology extensible for representing new units, or quantities needed for other domains.

Note 1. The SEAS project imposes that dimensions with the rings over the following sets be possible: $\{true, false\}$, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{C}$.

* This work has been supported by ITEA2 project SEAS 12004.

2 Preliminaries

2.1 Examples

We use the Turtle syntax in RDF snippets, with prologue:

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix lindt: <http://purl.org/NET/lindt#>.
@base <http://w3id.org/fs/concepts/> .
```

2.2 RDF abstract syntax

Let $\mathbf{I}$, $\mathbf{B}$, and $\mathbf{L}$, be the disjoint sets of *IRIs*, *blank nodes* (later called *bnodes*), and *literals*, respectively. The set of *RDF terms* is $\mathbf{T} = \mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$. Let $\mathbf{V}$ be the set of *variables*, disjoint from all the other sets. A *generalized triple pattern* (later simply called an *triple pattern*) is an element of $(\mathbf{T} \cup \mathbf{V})^3$ and the set of all triple patterns is $\mathcal{T}$. A (generalized) *Basic Graph Pattern (BGP)* is a finite set of triple patterns.

A triple pattern (resp., BGP) that does not contain any variable is called a *generalized RDF triple* (resp., *generalized RDF graph*). An instance of a BGP G is an BGP G' where all the variables of G have been replaced by terms in $\mathbf{T}$. The set of instances of a BGP G is written $\mathsf{inst}(G)$. By extension, $\mathsf{inst}(x)$ is defined when x is a triple, or an element of $(\mathbf{T} \cup \mathbf{V})$. The set of all generalized RDF graphs is $\mathcal{G}$.

An RDF triple (resp., RDF graph) that does not contain any bnode is called a *ground triple* (resp., *ground graph*). A grounding of an RDF graph G is an RDF graph G' where all the bnodes of G have been replaced by terms in $\mathbf{I} \cup \mathbf{L}$. The set of groundings of a graph G is written $\mathsf{grnd}(G)$. By extension, $\mathsf{grnd}(x)$ is defined when x is a triple.

2.3 RDF semantics

RDF 1.1 Semantics [**?**] defines several formal semantics for RDF graphs, each of which offers a different expressive power. The most generic of them is the formal semantics that defines *simple entailment*:

Definition 1 (Simple interpretation). *A* simple interpretation *is a tuple* $\langle \Delta_R, \Delta_P, [\![\cdot]\!], \cdot^{\mathcal{I}} \rangle$ *such that:*

- Δ_R *is a nonempty set, called the* domain *of* $\mathcal{I}$*, and whose elements are called* resources*;*
- Δ_P *is a nonempty set whose elements are called* properties*;*
- $[\![\cdot]\!]$ *is a mapping from* Δ_P *to* $2^{\Delta_R \times \Delta_R}$*;*
- $\cdot^{\mathcal{I}}$ *is a partial mapping from* $\mathbf{I} \cup \mathbf{L}$ *to* Δ_R *such that the restriction to* $\mathbf{I}$ *is total.*

Definition 2 (Simple model of RDF graphs). *A simple interpretation* $\mathcal{I} = \langle \Delta_R, \Delta_P, [\![\cdot]\!], \cdot^{\mathcal{I}} \rangle$ *is a* simple model *of a ground triple* $t = \langle s, p, o \rangle$*, noted* $\mathcal{I} \models_{\mathsf{simple}} t$*, iff* $\langle s^{\mathcal{I}}, o^{\mathcal{I}} \rangle \in [\![p^{\mathcal{I}}]\!]$*.* $\mathcal{I}$ *is a simple model of a ground graph* G *(also noted* $\mathcal{I} \models_{\mathsf{simple}} G$*) iff it is a simple model of all triples in* G*. Finally,* $\mathcal{I}$ *is a simple model of an RDF graph* H *iff there exists a grounding* H' *of* H *such that* $\mathcal{I} \models_{\mathsf{simple}} H'$ *(again, this is noted* $\mathcal{I} \models_{\mathsf{simple}} H'$*).*

Definition 3 (Simple entailment). *An RDF graph G is said to* simply entail *another RDF graph H if all simple models of G are simple models of H.*

Also we will introduce the D-entailment regime, and the D-RDFS-entailment regime. (TODO)

- Δ_C is a nonempty set whose elements are called *classes*;
- $[\![\cdot]\!]_C$, the *class extension*, is a mapping from Δ_C to 2^{Δ_R}.

3 RDF-based Semantics

3.1 Dimensions and units

Definition 4 (Dimensions). *The set of* dimensions *is noted $\mathcal{D}$. Every dimension $D \in \mathcal{D}$ is associated with a set* $\mathrm{set}(D)$ *of values.*

Example of dimensions include time, length, temperature.
The RDF representation of the set of dimensions is the class $\texttt{<Dimension>}^{\mathcal{I}} \in \Delta_C$:

$$[\![\texttt{<Dimension>}^{\mathcal{I}}]\!]_C = \mathcal{D} \tag{1}$$

Property $\texttt{<unit>}^{\mathcal{I}}$ links a dimension to a datatype, whose *value space* is a subset of $\mathrm{set}(D)$. The RDF Graph below represents that any of the datatypes $\texttt{<joule>}^{\mathcal{I}}$, $\texttt{<wattHour>}^{\mathcal{I}}$, or $\texttt{<electronVolt>}^{\mathcal{I}}$ may be used to describe a value of the energy dimension.

```
<EnergyDimension> a <Dimension> ;
  <unit> <joule>, <wattHour>, <electronVolt> .
```

When appropriate, datatypes should be one of the XML Schema datatypes. Else, it should be a LINDT[1] custom datatype, which may be recognized on the fly by RDF processors and SPARQL engines. A LINDT datatype is a datatype at the URL of which one may retrieve a RDF Document that describes it, and a Javascript script that can be used to process literals with this datatype. Example of resources in the LINDT vocabulary include:

- class $\texttt{lindt:TotallyOrdered}^{\mathcal{I}}$ specifies that the Javascript script implements function **compare**, to compare two literals with this datatype;
- class $\texttt{lindt:Group}^{\mathcal{I}}$, specifies that the Javascript script implements functions **add** and **substract** to add and substract two literals with this datatype, and **additiveIdentity** to retrieve the additive identity element;
- class $\texttt{lindt:Ring}^{\mathcal{I}}$, specifies that the Javascript script also implements functions **multiply** and **divide** to multiply and divide two literals with this datatype, and **multiplicativeIdentity** to retrieve the multiplicative identity element.

We define important subclasses of $\texttt{<Dimension>}^{\mathcal{I}}$:

[1] LINDT - Linked Datatypes - `http://purl.org/NET/lindt` - work in progress.

– <GroupDimension>$^{\mathcal{I}}$ is the class of dimensions that are also groups, i.e., unit datatypes of the dimension should be group, and coherent (todo: define coherence). Property <additiveIdentityValue>$^{\mathcal{I}}$ links a group dimension to an element that must be the additive identity element of any of its unit datatypes. For some group dimensions, the datatype to describe values is different from the datatype to describe the delta between these values. In such cases, we use property <deltaUnit>$^{\mathcal{I}}$ for the latter. The canonical example is dimension <TimeDimension>$^{\mathcal{I}}$, that uses xsd:dateTime for values and xsd:duration for deltas:

```
<TimeDimension> <unit> xsd:dateTime ; <deltaUnit> xsd:duration .
```

Also, property <additiveIdentityValue>$^{\mathcal{I}}$ is canonically an instance of the <deltaUnit>$^{\mathcal{I}}$.

– <TotallyOrderedGroupDimension>$^{\mathcal{I}}$ is the class of dimensions that are also totally ordered groups, i.e., unit datatypes of the dimension should be totally ordered, and coherent (todo: define coherence). Property <nullValue>$^{\mathcal{I}}$ links a totally ordered group dimension to an element that must be the null element of any of its unit datatype. For example, even though <TemperatureDimension>$^{\mathcal{I}}$ may be defined as being an instance of <TotallyOrderedGroupDimension>$^{\mathcal{I}}$, it does not have a null value as this value is different for <degreeCelsius>, <degreeKelvin>, and <degreeFarenheit>.

We also anticipate cross dimension operations. Property <derivativeOfProperty>$^{\mathcal{I}} \in \Delta_P$ links a dimension to a property. The semantics of RDF Graph below is that dimension <PowerDimension>$^{\mathcal{I}}$ is the derivative of <EnergyDimension>$^{\mathcal{I}}$ with respect to <TimeDimension>$^{\mathcal{I}}$.

```
<TimeDimension> <derivativeOfProperty> <timeDerivativeOf> .
<PowerDimension> <timeDerivativeOf> <EnergyDimension> .
```

Conversely, we define property <primitiveOfProperty>$^{\mathcal{I}} \in \Delta_P$, that links a dimension to a property. The semantics of RDF Graph below is that dimension <EnergyDimension>$^{\mathcal{I}}$ is the primitive of <PowerDimension>$^{\mathcal{I}}$ with respect to <TimeDimension>$^{\mathcal{I}}$.

```
<TimeDimension> <primitiveOfProperty> <timePrimitiveOf> .
<EnergyDimension> <timePrimitiveOf> <PowerDimension> .
```

We extend the LINDT vocabulary to represent this at the level of custom datatypes:

```
<watt> lindt:multiplyArgType xsd:duration ; lindt:multiplyReturnType <wattHour> .
```

This RDF Graph suggests that the Javascript object retrievable from Javascript script at URL <watt> implements the **multiply** function, which takes two lexical forms as first and second parameters, accepts xsd:duration as third parameter (i.e., the datatype of the second lexical form), and accepts <wattHour> as fourth parameter (i.e., the datatype of the returned lexical form). Similarly, RDF Graph:

```
<wattHour> lindt:divideArgType xsd:duration ; lindt:divideReturnType <watt> .
```

suggests that the Javascript object retrievable from Javascript script at URL <wattHour> implements the **divideBy** function, which takes two lexical forms as first and second parameters, accepts xsd:duration as third parameter (i.e., the datatype of the second lexical form), and accepts <watt> as fourth parameter (i.e., the datatype of the returned lexical form).

3.2 Distributions

Definition 5 (Distributions on a dimension). *The set of* distributions *on a dimension* $D \in \mathcal{D}$*, noted* $\mathrm{distr}(D) = 2^{\mathrm{set}(D)}$*, is the set of subsets of* $\mathrm{set}(D)$.

Let be a dimension $\texttt{?dim}^{\mathcal{I}} = D \in \mathcal{D}$. Then the following RDF graph defines that class $\texttt{?dDistributionClass}^{\mathcal{I}} \in \Delta_C$ is the class of distributions over D. Then for any distribution $\texttt{?d}^{\mathcal{I}} = \delta \in \mathrm{distr}(D)$ of $\texttt{?dDistributionClass}^{\mathcal{I}} \in \Delta_C$, property $\texttt{<dimension>}^{\mathcal{I}} \in \Delta_P$ links $\texttt{?d}^{\mathcal{I}}$ to $\texttt{?dim}^{\mathcal{I}}$.

```
?dim <distributionsClass> ?dDistributionClass .
?d a ?dDistributionClass ; <dimension> ?dim .
```

Let be a value on dimension D, $\texttt{?v}^{\mathcal{I}} = v \in \mathrm{set}(D)$. Then `{?d <contains> ?v}` (resp. `{?d <excludes> ?v}`) represents $v \in \delta$ (resp. $v \notin \delta$). For conciseness, we note:

$$\texttt{\{?d <contains> ?v\}} \Leftrightarrow v \in \delta \tag{2}$$

$$\texttt{\{?d <excludes> ?v\}} \Leftrightarrow v \notin \delta \tag{3}$$

One could fully characterize distribution δ with an infinite number of triples. Now we introduce a set of classes and properties that enable to describe the most classical distributions in a limited number of triples, and ease the writing of RDF snippets by domain experts in the most simple use cases. These resources were chosen during requirement capturing sessions organized in December 2015 for the SEAS project[2].

$$\texttt{\{?d a <EmptyDistribution>\}} \Leftrightarrow \delta = \varnothing \tag{4}$$

$$\texttt{\{?d a <FullDistribution>\}} \Leftrightarrow \delta = \mathrm{set}(D) \tag{5}$$

$$\texttt{\{?d <value> ?v\}} \Leftrightarrow \delta = \{v\} \tag{6}$$

$$\texttt{\{?d1 <subDistributionOf> :d2\}} \Leftrightarrow \delta_1 \subseteq \delta_2 \tag{7}$$

$$\texttt{\{?d1 <equivalentDistribution> :d2\}} \Leftrightarrow \delta_1 = \delta_2 \tag{8}$$

$$\texttt{?d <intersectionOfDistributions> (?d1 ... ?dn) .} \Leftrightarrow \delta = \delta_1 \cap \ldots \cap \delta_n \tag{9}$$

$$\texttt{?d <unionOfDistributions> (?d1 ... ?dn) .} \Leftrightarrow \delta = \delta_1 \cup \ldots \cup \delta_n \tag{10}$$

For dimension D that are totally ordered by $\leqslant$, let functions $\min$ represent the potential greatest lower bound, $\max$ represent the potential least upper bound, and rng represent the delta between the $\max$ and the $\min$, if both exist. These functions are coupled: $\forall \delta \in \mathrm{distr}(D), \min(\delta) + \mathrm{rng}(\delta) = \max(\delta)$.

Let $\texttt{?d1}^{\mathcal{I}} = \delta_1 \in \mathrm{distr}(D), \ldots \texttt{?dn}^{\mathcal{I}} = \delta_n \in \mathrm{distr}(D)$:

[2] SEAS Requirement capturing Workshop - `http://data.the-smart-energy.com/workshop/2015/12/`

$$\texttt{\{?d <lowerBound> ?v\}} \Leftrightarrow (\forall x)\big[x < v \Rightarrow x \notin \delta\big] \tag{11}$$

$$\texttt{\{?d <upperBound> ?v\}} \Leftrightarrow (\forall x)\big[x > v \Rightarrow x \notin \delta\big] \tag{12}$$

$$\texttt{\{?d <minimum> ?v\}} \Leftrightarrow \min(\delta) = v \tag{13}$$

$$\texttt{\{?d <maximum> ?v\}} \Leftrightarrow \max(\delta) = v \tag{14}$$

$$\texttt{\{?d <range> ?v\}} \Leftrightarrow \mathrm{rng}(\delta) = v \tag{15}$$

$$\texttt{?d a <Interval>; <minimum> ?v1; <maximum> ?v2.} \Leftrightarrow \begin{cases} \texttt{?d <lowerBound> ?v1; <upperBound> ?v2.} \\ (\forall x)\big[v_1 < x < v_2 \Rightarrow x \in \delta\big] \end{cases} \tag{16}$$

$$\texttt{\{?d a <minimumIncluded> ?v\}} \Leftrightarrow \texttt{?d <minimum> ?v; <includes> ?v} \tag{17}$$

$$\texttt{\{?d a <minimumExcluded> ?v\}} \Leftrightarrow \texttt{?d <minimum> ?v; <excludes> ?v} \tag{18}$$

$$\texttt{\{?d a <maximumIncluded> ?v\}} \Leftrightarrow \texttt{?d <maximum> ?v; <includes> ?v.} \tag{19}$$

$$\texttt{\{?d a <maximumExcluded> ?v\}} \Leftrightarrow \texttt{?d <maximum> ?v; <excludes> ?v} \tag{20}$$

3.3 Facets, F-points, and F-spaces

Definition 6 (Facets). *The set of* facets *is noted $\mathcal{F}$. Any facet $f \in \mathcal{F}$ is assigned a dimension by mapping* $\dim$.

The set of facets is represented in RDF by class $\texttt{<Facet>}^{\mathcal{I}} \in \Delta_C$, and the mapping $\dim$ is represented by property $\texttt{<facetDimension>}^{\mathcal{I}} \in \Delta_P$:

$$[\![\texttt{<Facet>}^{\mathcal{I}}]\!]_{\mathcal{C}} = \mathcal{F} \tag{21}$$

$$[\![\texttt{<facetDimension>}^{\mathcal{I}}]\!] = \dim \tag{22}$$

The following RDF Graph illustrates the difference between dimensions and facets:

```
<PowerDimension> a <Dimension> .
<ConsumptionPowerFacet> a <Facet> ; <facetDimension> <PowerDimension> .
<StoringPowerFacet> a <Facet> ; <facetDimension> <PowerDimension> .
<ProductionPowerFacet> a <Facet> ; <facetDimension> <PowerDimension> .
```

Definition 7 (Faceted Point (F-point), and its Facets). *The set of* Faceted Points*, or* F-points*, Ω, is equal to the Cartesian product of the dimension sets of each facet in $\mathcal{F}$, augmented with a special "nothing" element $\bullet$:*

$$\Omega = \bigtimes_{f \in \mathcal{F}} \Big(\mathrm{set}(\dim(f)) \cup \{\bullet\}\Big)$$

Element f of a F-point $\boldsymbol{v} \in \Omega$ is named facet f*, and is noted: $\pi_f \boldsymbol{v}$. If $\pi_f \boldsymbol{v} = \bullet$, then there is conceptually no facet f of $\boldsymbol{v}$.*

Element $\bullet$ helps to cope with the RDF open world assumption. Without further knowledge, any F-point has on any facet either a value of $\mathrm{set}(\dim(f))$, or no value at all. For instance, some MD-Value may have facet $\texttt{<ConsumptionPowerFacet>}^{\mathcal{I}}$ $\texttt{"3.4"^^<watt>}^{\mathcal{I}}$, and facet $\texttt{<TimeFacet>}^{\mathcal{I}}$ $\texttt{"2016-02-25T12:00:00Z"^^xsd:dateTime}^{\mathcal{I}}$, but no facet $\texttt{<WaterDepthFacet>}^{\mathcal{I}}$. We do not introduce direct RDF representation of F-points.

Definition 8 (Faceted Space (F-spaces)). *The set of* Faceted Spaces*, or* F-spaces*,* $\Sigma = 2^{\Omega}$*, is the set of subsets of* Ω*. A F-space* $\tau \in \Sigma$ *is characterized by the set of F-points it* contains*.*

The set of F-spaces is represented by RDF class $\texttt{<FacetedSpace>}^{\mathcal{I}} \in \Delta_C$:

$$[\![\texttt{<FacetedSpace>}^{\mathcal{I}}]\!]_{\mathcal{C}} = \Sigma \tag{23}$$

Then we introduce the following properties, defined for any set of F-spaces $\texttt{?t}^{\mathcal{I}} = \tau, \texttt{?t1}^{\mathcal{I}} = \tau_1, \ldots \texttt{?tn}^{\mathcal{I}} = \tau_n \in \Sigma$:

$$\texttt{\{?t1 <subSpaceOf> ?t2\}} \Leftrightarrow \tau_1 \subseteq \tau_2 \tag{24}$$

$$\texttt{\{?t1 <equivalentSpace> ?t2\}} \Leftrightarrow \tau_1 = \tau_2 \tag{25}$$

$$\texttt{\{?t <intersectionOfSpaces> (?t1 ... ?tn)\}} \Leftrightarrow \tau = \tau_1 \cap \ldots \cap \tau_n \tag{26}$$

$$\texttt{\{?t <unionOfSpaces> (?t1 ... ?tn)\}} \Leftrightarrow \tau = \tau_1 \cup \ldots \cup \tau_n \tag{27}$$

Definition 9 (Projection of a F-space on a facet). *The* projection *of a F-space* $\tau \in \Sigma$ *on facet* $f \in \mathcal{F}$ *is noted* $\pi_f \tau$*:*

$$\pi_f \tau = \{\pi_f \boldsymbol{v} | \boldsymbol{v} \in \tau\} \backslash \bullet \in \mathrm{distr}(\dim(f)) \tag{28}$$

For any facet $\texttt{?f}^{\mathcal{I}} = f \in \mathcal{F}$, triple pattern `{?f <projectionProperty> ?fProjection}` defines $\texttt{?fProjection}^{\mathcal{I}}$ as the property that links a F-space $\texttt{?t}^{\mathcal{I}} = \tau \in \Sigma$ to its projection $\texttt{?d}^{\mathcal{I}} = \delta \in \mathrm{distr}(\dim(f))$ on facet f:

$$\texttt{\{?t ?fProjection ?d\}} \Leftrightarrow \pi_f \tau = \delta \tag{29}$$

For sake of conciseness, we also enable the definition of other properties and classes as shortcuts to describe the projections on facet f directly at the level of the F-space:

```
?f <projectionSubDistributionOfProperty> ?fSubDistributionOf ;
   <projectionContainsProperty> ?fContains ;
   <projectionExcludesProperty> ?fExcludes ;
   <projectionEmptyClass> ?fEmpty ;
   <projectionFullClass> ?fFull ;
   <projectionValueProperty> ?fValue ;
   <projectionLowerBoundProperty> ?fLowerBound ;
   <projectionUpperBoundProperty> ?fUpperBound ;
   <projectionIntervalClass> ?fInterval ;
   <projectionMinimumProperty> ?fMinimum ;
   <projectionMaximumProperty> ?fMaximum ;
   <projectionRangeProperty> ?fRange ;
   <projectionMinimumIncludedProperty> ?fMinimumIncluded ;
   <projectionMinimumExcludedProperty> ?fMinimumExcluded ;
   <projectionMaximumIncludedProperty> ?fMaximumIncluded ;
   <projectionMaximumExcludedProperty> ?fMaximumExcluded .
```

Then for any $\texttt{?v}^{\mathcal{I}} = v \in \text{set}(\dim(f))$ and $\texttt{?t}^{\mathcal{I}} = \tau, \texttt{?t1}^{\mathcal{I}} = \tau_1, \texttt{?t2}^{\mathcal{I}} = \tau_2 \in \Sigma$:

$$\texttt{\{?t1 ?fSubDistributionOf ?t2\}} \quad \Leftrightarrow \quad \pi_f\tau_1 \subseteq \pi_f\tau_2 \tag{30}$$

$$\texttt{\{?t ?fContains ?v\}} \quad \Leftrightarrow \quad v \in \pi_f\tau \tag{31}$$

$$\texttt{\{?t ?fExcludes ?v\}} \quad \Leftrightarrow \quad v \notin \pi_f\tau \tag{32}$$

$$\texttt{\{?t a ?fEmpty\}} \quad \Leftrightarrow \quad \pi_f\tau = \varnothing \tag{33}$$

$$\texttt{\{?t a ?fFull\}} \quad \Leftrightarrow \quad \pi_f\tau = \text{set}(\dim(f)) \tag{34}$$

$$\texttt{\{?t ?fValue ?v\}} \quad \Leftrightarrow \quad \pi_f\tau = \{v\} \tag{35}$$

$$\texttt{\{?t ?fLowerBound ?v\}} \quad \Leftrightarrow \quad \forall x \in \dim(f), x < v \Rightarrow x \notin \pi_f\tau \tag{36}$$

$$\texttt{\{?t ?fUpperBound ?v\}} \quad \Leftrightarrow \quad \forall x \in \dim(f), x > v \Rightarrow x \notin \pi_f\tau \tag{37}$$

$$\texttt{?t a ?fInterval; ?fMinimum ?v1; ?fMaximum ?v2; ?fRange :z.} \quad \Leftrightarrow \quad \begin{cases} \texttt{\{?t ?fLowerBound ?v1; ?fUpperBound ?v2.\}} \\ \forall x \in \dim(f), v_1 < x < v_2 \Rightarrow x \in \pi_f\tau \end{cases} \tag{38}$$

$$\texttt{\{?t ?fMinimumIncluded ?v\}} \quad \Leftrightarrow \quad \texttt{?t piMinimum ?v; ?fContains ?v} \tag{39}$$

$$\texttt{\{?t ?fMinimumExcluded ?v\}} \quad \Leftrightarrow \quad \texttt{?t piMinimum ?v; ?fExcludes ?v} \tag{40}$$

$$\texttt{\{?t ?fMaximumIncluded ?v\}} \quad \Leftrightarrow \quad \texttt{?t ?fMaximum ?v; ?fContains ?v.} \tag{41}$$

$$\texttt{\{?t ?fMaximumExcluded ?v\}} \quad \Leftrightarrow \quad \texttt{?t ?fMaximum ?v; ?fExcludes ?v} \tag{42}$$

3.4 MD-Space series

Definition 10 (Restriction of a F-space with respect to a distribution along a facet). *The restriction of a F-space τ with respect to a distribution $\delta \in \text{distr}(\dim(f))$ along facet f is noted $\sigma_f^\delta\tau$, and is defined as follows:*

$$\sigma_f^\delta\tau = \{\boldsymbol{v} | \boldsymbol{v} \in \tau \textit{ and } \pi_f\boldsymbol{v} \in \delta\} \tag{43}$$

Let be two F-spaces $\texttt{?t1}^{\mathcal{I}} = \tau_1, \texttt{?t2}^{\mathcal{I}} = \tau_2 \in \Sigma$, a facet $\texttt{?f}^{\mathcal{I}} = f \in \mathcal{F}$, and a distribution on the dimension of that facet $\texttt{?d}^{\mathcal{I}} = \delta \in \text{distr}(\dim(f))$. Then,

$$\begin{array}{l}\texttt{?f <restrictionProperty> ?fRestriction ;} \\ \texttt{?t2 a <Restriction> ;} \\ \quad \texttt{<base> ?t1 ;} \\ \quad \texttt{?fRestriction ?d ;}\end{array} \quad \Leftrightarrow \quad \sigma_f^\delta\tau_1 = \tau_2 \tag{44}$$

Multiple restrictions can be associated with a $\texttt{Restriction}^{\mathcal{I}}$. This is slicing. We define series of F-spaces, that enable to decompose the description of a F-space in a series of its restrictions along a common facet.

Let be a facet $\texttt{?f}^{\mathcal{I}} = f \in \mathcal{F}$, and a F-space $\texttt{?t}^{\mathcal{I}} = \tau \in \Sigma$. Then the following RDF graph defines:

- property $\texttt{?fSeries}^{\mathcal{I}} \in \Delta_P$ links F-spaces to series of their restrictions along facet f;
- $\texttt{?ts}^{\mathcal{I}} \in \Delta_P$ is a series of restrictions of F-space τ along facet f;
- F-spaces $\tau_1, \ldots \tau_n \in \Sigma$ are the individual F-spaces in the series.

```
?f <seriesProperty> ?fSeries .
?t ?fSeries ?ts .
?ts <series> ( ?t1 ... ?tn ) .
```

Every τ_k is the restriction of F-space τ with respect to $\pi_f\tau_k$ along facet f:

$$\tau_k = \sigma_f^{\pi_f\tau_k}\tau \tag{45}$$

The description of individual projections $\pi_f\tau_k$ may be constrained at the level of the series using the following properties and classes:

$$\texttt{\{?ts <firstStart> ?v\}} \Leftrightarrow \min(\pi_f\tau_1) = v \tag{46}$$

$$\texttt{\{?ts <firstRange> ?z\}} \Leftrightarrow \mathrm{rng}(\pi_f\tau_1) = z \tag{47}$$

$$\texttt{\{?ts <firstEnd> ?v\}} \Leftrightarrow \max(\pi_f\tau_1) = v \tag{48}$$

$$\texttt{\{?ts a <SameStartSeries>\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n-1)[\min(\pi_f\tau_{k+1}) = \min(\pi_f\tau_k)] \tag{49}$$

$$\texttt{\{?ts a <SameRangeSeries>\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n-1)[\mathrm{rng}(\pi_f\tau_{k+1}) = \mathrm{rng}(\pi_f\tau_k)] \tag{50}$$

$$\texttt{\{?ts a <SameEndSeries>\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n-1)[\max(\pi_f\tau_{k+1}) = \max(\pi_f\tau_k)] \tag{51}$$

$$\texttt{\{?ts <consecutiveStartOffset> ?z\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n-1)[\min(\pi_f\tau_{k+1}) = \min(\pi_f\tau_k) + z] \tag{52}$$

$$\texttt{\{?ts <consecutiveRangeOffset> ?z\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n-1)[\mathrm{rng}(\pi_f\tau_{k+1}) = \mathrm{rng}(\pi_f\tau_k) + z] \tag{53}$$

$$\texttt{\{?ts <consecutiveEndOffset> ?z\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n-1)[\max(\pi_f\tau_{k+1}) = \max(\pi_f\tau_k) + z] \tag{54}$$

$$\texttt{\{?ts a <PointedSeries>\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n)[\min(\pi_f\tau_k) = \max(\pi_f\tau_k)] \tag{55}$$

$$\texttt{\{?ts a <JoinedSeries>\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n-1)[\max(\pi_f\tau_k) = \min(\pi_f\tau_{k+1})] \tag{56}$$

$$\texttt{\{?ts a <StartIncludedSeries>\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n)[\min(\pi_f\tau_k) \in \pi_f\tau_k] \tag{57}$$

$$\texttt{\{?ts a <StartExcludedSeries>\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n)[\min(\pi_f\tau_k) \notin \pi_f\tau_k] \tag{58}$$

$$\texttt{\{?ts a <EndIncludedSeries>\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n)[\max(\pi_f\tau_k) \in \pi_f\tau_k] \tag{59}$$

$$\texttt{\{?ts a <EndExcludedSeries>\}} \Leftrightarrow (\forall 1 \leqslant k \leqslant n)[\max(\pi_f\tau_k) \notin \pi_f\tau_k] \tag{60}$$