Modeling and Querying Metadata in the Semantic Sensor Web: the Model stRDF and the Query Language stSPARQL [⋆]

Manolis Koubarakis and Kostis Kyzirakos

Dept. of Informatics and Telecommunications
National and Kapodistrian University of Athens
Greece
{koubarak,kkyzir}@di.uoa.gr

Abstract. RDF will often be the metadata model of choice in the Semantic Sensor Web. However, RDF can only represent thematic metadata and needs to be extended if we want to model spatial and temporal information. For this purpose, we develop the data model stRDF and the query language stSPARQL. stRDF is a constraint data model that extends RDF with the ability to represent spatial and temporal data. stSPARQL extends SPARQL for querying stRDF data. In our extension to RDF, we follow the main ideas of constraint databases and represent spatial and temporal objects as quantifier-free formulas in a first-order logic of linear constraints. Thus an important contribution of stRDF is to bring to the RDF world the benefits of constraint databases and constraint-based reasoning so that spatial and temporal data can be represented in RDF using constraints.

1 Introduction

Sensors are rapidly becoming ubiquitous and may be found everywhere, ranging from consumer products to the medical, automotive and industrial markets. Collections of spatially distributed sensors with communication infrastructure form sensor networks. Semantically enriched sensor networks should allow the discovery and analysis of sensor data based on spatial, temporal and thematic information. In this paper we extend RDF, which will often be the metadata model of choice in the Semantic Sensor Web, with a uniform way to represent spatial and temporal characteristics of sensors and sensor networks such as the spatial location of a sensor, the trajectory of a moving sensor, the coverage of a sensor network, the valid time of sensor acquisitions etc.

Up to now, little attention has been paid to the problem of extending RDF to represent spatial and/or temporal information. The most important work that deal with representing temporal information in RDF is [1]. More recently, [2,3,4] proposed to represent spatial data in RDF(S) using spatial ontologies

[⋆] This work was supported in part by the European Commission project Semsor-Grid4Env (http://www.semsorgrid4env.eu/)

e.g., ontologies based on the GeoRSS GML specification [5]. [3] also compares various ways to use SPARQL to query such spatial data, while [4] proposes a useful extension to SPARQL, called SPARQL-ST, to query data expressed in a spatial and temporal extension of RDF. The temporal extension of RDF in [4], uses the model of [1] to represent the valid time of triples.

The work presented in this paper has the same goal with the papers cited above: to enrich the Semantic Web with spatial and temporal data by extending RDF and SPARQL. To achieve this we diverge significantly from the papers cited above and follow the main ideas of *spatial and temporal constraint databases* [6,7,8,9]. We represent spatial geometries by *semi-linear* point sets in the n-dimensional space $\mathbb{Q}^n$ i.e., sets that can be defined by quantifier-free formulas in the first-order logic of linear equations and inequalities over $\mathbb{Q}^n$. Semi-linear sets can capture a great variety of spatial geometries, e.g., points, lines, line segments, polygons, k-dimensional unions of convex polygons possibly with holes, thus they give us a lot of expressive power [10]. Similarly, we represent the *valid times* of triples using *temporal constraints* (a very restricted class of linear constraints).

The main contributions of this paper are the following: Following the approach of Dédale [11] and CSQL [12], we develop a constraint-based extension of RDF, called stRDF, that can be used to represent thematic and spatial data that might change over time. The main contribution of stRDF is to bring to the RDF world the benefits of constraint databases and constraint-based reasoning so that spatial and temporal data can be represented in RDF using constraints. In this way, application areas with a rich spatial and temporal component such as the Semantic Sensor Web [13] and the Geospatial Semantic Web [14] can be tackled using Semantic Web technologies. The model stRDF and the query language stSPARQL are currently implemented (by extending the Sesame RDF store) in the context of the project SemsorGrid4Env where they serve as the data model and query language for the *semantic sensor registry.*

We also present an extension of SPARQL, called stSPARQL, to query spatial and temporal data expressed in stRDF, in a declarative way. We introduce stSPARQL by example and present a detailed semantics using the algebraic approach pioneered for SPARQL in [15]. Technically, stSPARQL follows closely the ideas in [12] and to a lesser extent the ideas in [11]; this allows us to have a useful language for expressing spatial and temporal queries while maintaining closure (i.e., staying within the realm of semi-linear point sets).

The organization of this paper is the following. In Section 2 we present the data model stRDF and in Section 3 we present the query language stSPARQL by means of examples. In Section 4 we give a formal definition of stSPARQL and define its semantics by following an algebraic approach. Comparison with related work is presented in Section 5 and in Section 6 we present our conclusions and discuss future work.

2 Data Model

To develop stRDF, we follow closely the ideas of constraint databases [6,8] and especially the work on CSQL [12]. First, we define the formulae that we allow as constraints. Then, we develop stRDF in two steps. The first step is to define the model sRDF which extends RDF with the ability to represent spatial data. Then, we extend sRDF to stRDF so that thematic and spatial data with a temporal dimension can be represented.

2.1 Linear constraints

Constraints will be expressed in the first-order language $\mathcal{L} = \{\leq, +\} \cup \mathbb{Q}$ over the structure $\mathcal{Q} = \langle \mathbb{Q}, \leq, +, (q)_{q\in\mathbb{Q}} \rangle$ of the linearly ordered, dense and unbounded set of the rational numbers, denoted by $\mathbb{Q}$, with rational constants and addition. The atomic formulae of this language are *linear equations* and *inequalities* of the form: $\sum_{i=1}^{p} a_i x_i \Theta a_0$, where Θ is a predicate among $=$, or $\leq$, the x_i's denote variables and the a_i's are integer constants. Note that rational constants can always be avoided in linear equations and inequalities. The multiplication symbol is used as an abbreviation i.e., $a_i x_i$ stands for $x_i + \cdots + x_i$ (a_i times).

We now define semi-linear subsets of $\mathbb{Q}^k$, where k is a positive integer.

Definition 1. *Let S be a subset of $\mathbb{Q}^k$. S is called* semi-linear *if there is a quantifier-free formula $\phi(x_1, \ldots, x_k)$ of $\mathcal{L}$ where $x_1, \ldots, x_k$ are variables such that $(a_1, \ldots, a_k) \in S$ iff $\phi(a_1, \ldots, a_k)$ is true in the structure $\mathcal{Q}$.*

We will use $\varnothing$ to denote the empty subset of $\mathbb{Q}^k$ represented by any inconsistent formula of $\mathcal{L}$.

2.2 The sRDF data model

We now define sRDF. As in theoretical treatments of RDF [15], we assume the existence of pairwise-disjoint countably infinite sets I, B and L that contain IRIs, blank nodes and literals respectively. In sRDF, we also assume the existence of an infinite sequence of sets $C_1, C_2, \ldots$ that are pairwise-disjoint with I,B and L. The elements of each $C_k, k = 1, 2, \ldots$ are the quantifier-free formulae of the first-order language $\mathcal{L}$ with k free variables. We denote with C the infinite union $C_1 \cup C_2 \cup \cdots$.

Definition 2. *An* sRDF triple *is an element of the set $(I \cup B) \times I \times (I \cup B \cup L \cup C)$. If (s, p, o) is an sRDF triple, s will be called the subject, p the predicate and o the object of the triple. An* sRDF graph *is a set of sRDF triples.*

In the above definition, the standard RDF notion of a triple is extended, so that the object of a triple can be a quantifier-free formula with linear constraints. According to Definition 1 such a quantifier-free formula with k free variables is a finite representation of a (possibly infinite) semi-linear subset of $\mathbb{Q}^k$. Semi-linear subsets of $\mathbb{Q}^k$ can capture a great variety of spatial geometries, e.g., points, lines, line segments, polygons, k-dimensional unions of convex polygons possibly with holes, thus they give us a lot of expressive power. However, they cannot be used to represent other geometries that need higher-degree polynomials e.g., circles .

Example 1. The following are sRDF triples :

```
ex:s1 rdf:type, ex:Sensor .
ex:s1 ex:has_location "x=10 and y=20"^^strdf:SemiLinearPointSet
```

The above triples define a sensor and its location using a conjunction of linear constraints. The last triple is not a standard RDF triple since its object is an element of set C.

In terms of the W3C specification of RDF, sRDF can be realized as an extension of RDF with a new kind of *typed literals*: quantifier-free formulae with linear constraints. The datatype of these literals is e.g., `strdf:SemiLinearPointSet` (see Example 1 above) and can be defined using XML Schema. Alternatively, linear constraints can be expressed in RDF using MathML[1] and serialized as `rdf:XMLLiterals` as in [16]. [16] specifies a syntax and semantics for incorporating linear equations in OWL 2. We now move on to define stRDF.

2.3 The stRDF Data Model

We will now extend sRDF with time. Database researchers have differentiated among user-defined time, valid time and transaction time. RDF (and therefore sRDF) supports user-defined time since triples are allowed to have as objects literals of the following XML Schema datatypes: texttt`xsd:dateTime`, `xsd:time`, `xsd:date`, `xsd:gYearMonth`, `xsd: gYear`, `xsd:gMonthDay`, `xsd:gDay`, `xsd:gMonth`.

stRDF extends sRDF with the ability to represent the *valid time* of a triple (i.e., the time that the triple was valid in reality) using the approach of Gutierrez et al. [1] where the a fourth component is added to each sRDF triple.

The *time structure* that we assume in stRDF is the set of rational numbers $\mathbb{Q}$ (i.e., time is assumed to be linear, dense and unbounded). Temporal constraints are expressed by quantifier-free formulas of the language $\mathcal{L}$ defined earlier, but their syntax is limited to elements of the set C_1. *Atomic* temporal constraints are formulas of $\mathcal{L}$ of the following form: $x \sim c$, where x is a variable, c is a rational number and $\sim$ is $<, \leq, \geq, >, =$ or $\neq$. *Temporal constraints* are Boolean combinations of atomic temporal constraints using a *single* variable.

The following definition extends the concepts of triple and graph of sRDF so that thematic and spatial data with a temporal dimension can be represented.

Definition 3. *An* stRDF quad *is an sRDF triple (a, b, c) with a fourth component τ which is a temporal constraint. For quads, we will use the notation (a, b, c, τ), where the temporal constraint τ defines the set of time points that the fact represented by the triple (a, b, c) is valid in the real world. An* stRDF graph *is a set of sRDF triples and stRDF quads.*

[1] `http://www.w3.org/Math/`, last accessed February 20, 2010.

3 Query Language

We present the syntax of stSPARQL by means of examples involving sensor networks. The semantics of the language is presented in Section 4. More examples of stSPARQL from a GIS perspective are given in [17]. We will consider a dataset that describe static and moving sensors and use the CSIRO/SSN Ontology [18] to describe them . The main classes of interest in the SSN ontology is the class *Feature* that describes the observed domain, the class *Sensor* that describes the sensor, the class *SensorGrounding* that describes the physical characteristics and the location of the sensor and the class *Location* that is self explained. We extend the aforementioned ontology with the properties `strdf:hasGeometry` and `strdf:hasTrajectory` with range `strdf:SemiLinearPointSet`.

The stRDF description of a static sensor that measures temperature and has a certain location is the following (`ssn` is the namespace of the CSIRO/SSN ontology and `ex` an example ontology):

```
ex:sensor1 rdf:type ssn:Sensor .
ex:sensor1 ssn:measures ex:temperature .
ex:temperature ssn:type ssn:PhysicalQuality .
ex:sensor1 ssn:supports ex:grounding1 .
ex:grounding1 rdf:type ssn:SensorGrounding .
ex:grounding1 ssn:hasLocation ex:location1 .
ex:location1 rdf:type ssn:Location .
ex:location1 strdf:hasGeometry
               "x=10 and y=10"^^strdf:SemiLinearPointSet .
```

We choose to use the O&M-OWL ontology [19] to represent sensor observations. However, since we use stRDF to model space and time, we choose not to use the classes *Time*, *TimeInterval* and *TimeInstant* that come from OWL-Time and the classes *Geometry* and *Point* that come from GML. So our modeling is similar to the modeling in [19] but instead of relying on OWL-Time and GML we rely on the stRDF constructs. The stRDF representation of the sensor's observations is the following (`om` is the namespace of the O&M-OWL ontology):

```
ex:sensor1 rdf:type ex:TemperatureSensor .
ex:TemperatureSensor rdf:subClassOf om:Sensor .
ex:obs1 rdf:type om:Observation .
ex:obs1 om:procedure ex:sensor1 .
ex:obs1 om:observedProperty ex:temperature .
ex:temperature rdf:type om:Property .
ex:obs1 om:observationLocation ex:obslocation1 .
ex:obslocation1 rdf:type om:Location .
ex:obslocation1 strdf:hasGeometry
                  "x=10 and y=10"^^strdf:SemiLinearPointSet .
ex:obs1 om:result ex:obs1Result .
ex:obs1Result rdf:type om:ResultData .
ex:obs1Result om:uom ex:Celcius .
```

```
ex:obs1Result om:value "27"
                  "(10 <= t <= 11)"^^strdf:SemiLinearPointSet .
```

Notice the last quad that capture the spatiotemporal information.

Let us now present an example of modeling *moving sensors* in stRDF. Note that trajectories of moving sensors are easily represented in stRDF.

```
ex:sensor2 rdf:type ssn:Sensor .
ex:sensor2 ssn:measures ex:temperature .
ex:sensor2 ssn:supports ex:grounding2 .
ex:grounding2 rdf:type ssn:SensorGrounding .
ex:grounding2 ssn:hasLocation ex:location2 .
ex:location2 rdf:type ssn:Location .
ex:location2 strdf:hasTrajectory
          "(x=10t and y=5t and 0<=t<=5) or
           (x=10t and y=25 and 5<=t<=10)"^^strdf:SemiLinearPointSet.
```

Finally, we assume that we have the stRDF descriptions of some rural area where the sensors are deployed. The stRDF description of such an area called Brovallen is the following:

```
ex:area1 rdf:type ex:RuralArea .
ex:area1 ex:hasName "Brovallen" .
ex:area1 strdf:hasGeometry
           "(-10x+13y<=-50 and y<=79 and y>=13 and
              x<=133) or (y<=13 and x<=133 and
              x+2y>=129)"^^strdf:SemiLinearPointSet .
```

Example 2. Spatial selection. Find the URIs of the static sensors that are inside the rectangle R(0,0,100,100)?

```
select ?S
where {?S rdf:type ssn:Sensor . ?G rdf:type ssn:SensorGrounding .
       ?L rdf:type ssn:Location . ?S ssn:supports ?G .
       ?G ssn:haslocation ?L . ?L strdf:hasGeometry ?GEO .
       filter(?GEO inside "0<=x<=100 and 0<=y<=100")}
```

Let us now explain the new features of stSPARQL by referring to the above example. stSPARQL has a new kind of variables called . Spatial variables can be used in basic graph patterns to refer to spatial literals denoting semi-linear point sets. They can also be used in *spatial filters*, a new kind of filter expressions introduced by stSPARQL that is used to compare *spatial terms* using spatial predicates. Spatial terms include spatial constants (finite representations of semi-linear sets e.g., `"0<=x<=10 and 0<=y<=10"`), spatial variables and complex spatial terms (e.g., `?GEO INTER "x=10 and y=10"` which denotes the intersection of the value of spatial variable `?GEO` and the semi-linear set `"x=10 and y=10"`). There are several types of spatial predicates such as topological, distance, directional, etc. that one could introduce in a user-friendly spatial query language. In the current version of stSPARQL only the topological relations of [20] can be used as predicates in a spatial filter expression e.g., `filter(?GEO1 inside ?GEO2)`.

Example 3. Temporal selection. Find the values of all observations that were valid at time 11 and the rural area they refer to.

```
select ?V ?RA
where {?OBS rdf:type om:Observation . ?LOC rdf:type om:Location .
       ?R rdf:type om:ResultData . ?RA rdf:type ex:RuralArea .
       ?OBS om:observationLocation ?LOC . ?OBS om:result ?R .
       ?R om:value ?V ?T . ?LOC strdf:hasGeometry ?OBSLOC .
       ?RA strdf:hasGeometry ?RAGEO .
       filter(?T contains (t = 11) && ?RAGEO contains ?OBSLOC)}
```

The above query demonstrates the features of stSPARQL that are used to query the valid times of triples. stSPARQL offers one more new kind of variables in addition to spatial ones: *temporal variables* . Temporal variables can be used as the last term in a new kind of basic graph pattern called *quad pattern* to refer to the valid time of a triple. Temporal variables can also appear in temporal filters, a new kind of filter that can be used in stSPARQL to constrain the valid time of triples.

The expressions that make up temporal filters are Boolean combinations of *interval predicates* that are used to compare temporal terms. A *temporal term* in stRDF is a temporal variable or a temporal constant (i.e., an element of the set C_1 e.g., `"(t>=0 and t<=2) or (t>=5 and t<=7)"`). We allow any of the thirteen interval relations identified by Allen in [21] to be used as the interval predicates e.g, `contains` in the above example .

Example 4. Intersection of an area with a trajectory. Which areas of Brovallen were sensed by a moving sensor and when?

```
select (?TR[1,2] INTER ?GEO) as ?SENSEDAREA  ?GEO[3] as ?T1
where {?SN rdf:type ssn:Sensor . ?RA rdf:type ex:RuralArea.
       ?X rdf:type ssn:SensorGrounding . ?Y rdf:type ssn:Location.
       ?SN ssn:supports ?X . ?X ssn:hasLocation ?Y.
       ?Y strdf:hasTrajectory ?TR . ?RA ex:hasName "Brovallen".
       ?RA strdf:hasGeometry ?GEO . filter(?TR[1,2] overlap ?GEO)}
```

The above query demonstrates the projection of spatial terms. Projections of spatial terms (e.g., `?TR[1,2]`) denote the projections of the corresponding point sets on the appropriate dimensions, and are written using the notation `Variable "[" Dimension1 "," ... "," DimensionN "]"`.

Example 5. Projection and spatial function application. Find the URIs of the sensors that are north of Brovallen.

```
select ?SN
where {?SN rdf:type ssn:Sensor  . ?X rdf:type ssn:SensorGrounding .
       ?Y rdf:type ssn:Location . ?RA rdf:type ex:RuralArea .
       ?RA ex:hasName "Brovallen" . ?RA strdf:hasGeometry ?GEO .
       ?SN ssn:supports ?X . ?X ssn:hasLocation ?Y .
       ?Y strdf:hasGeometry ?SN_LOC .
       filter(MAX(?GEO[2])<MIN(?SN_LOC[2]))}
```

The above query demonstrates the projection of spatial terms and the application of metric spatial functions to spatial terms. We allow expressions like `MAX(?GEO[2])` that return the maximum value of the unary term `?GEO[2]`. The metric functions allowed in stSPARQL will be defined in detail in Section 4.

4 Formalization and Semantics of stSPARQL

In this section, we give a formal definition of stSPARQL and define its semantics by following an algebraic approach like the one originally pioneered in [15]. We only cover the spatial features of stSPARQL in detail and their interactions with existing SPARQL concepts. The temporal features of stSPARQL (quad patterns and temporal filters) can be formalized similarly and are omitted.

Let us recall from Section 2.2 the definitions of sets I, B, L, $C_1, C_2, \ldots$ and C. We define $ILC = I \cup L \cup C$ and $T = I \cup B \cup L \cup C \cup \mathbb{R}$. We need to include the set of real numbers $\mathbb{R}$ in the set T since as we will see below (Definition 8) the application of certain metric functions such as $AREA$ etc. can result in real numbers as answers to stSPARQL queries.

We also assume the existence of the following disjoint sets of *variables*: (i) the set of non-spatial variables V_{ns}, (ii) an infinite sequence $V_s^1, V_s^2, \ldots$ of sets of variables that will be used to denote elements of the sets $C_1, C_2, \ldots$ and (iii) the set of real variables V_r. We use V_s to denote the infinite union $V_s^1 \cup V_s^2 \cup \ldots$ and V to denote the union $V_{ns} \cup V_s \cup V_r$. The set V is assumed to be disjoint from the set T.

Let us now define a concept of mapping appropriate for stSPARQL by modifying the definition of [15]. A *mapping* μ from V to T is a partial function $\mu : V \rightarrow T$ such that $\mu(x) \in I \cup B \cup L$ if $x \in V_{ns}$, $\mu(x) \in C_i$ if $x \in V_s^i$ for all $i = 1, 2, \ldots$ and $\mu(x) \in \mathbb{R}$ if $x \in V_r$.

Example 6. The following is a mapping:

$$\{?S \rightarrow s_1,\ ?O \rightarrow John,\ ?GEO \rightarrow \text{“}x \geq 1 \wedge y \geq 0 \wedge y \leq 5\text{”}\}$$

The notions of domain and compatibility of mappings is as in [15]. The *domain* of a mapping μ, denoted by $dom(\mu)$, is the subset of V where the mapping is defined. Two mappings μ_1 and μ_2 are *compatible* if for all $x \in dom(\mu_1) \cap dom(\mu_2)$ we have $\mu_1(x) = \mu_2(x)$. For two sets of mappings Ω_1 and Ω_2, the operations of join, union, difference and left outer-join are also defined exactly as in [15]:

$\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2$ are compatible mappings$\}$

$\Omega_1 \cup \Omega_2 = \{\mu \mid \mu \in \Omega_1$ or $\mu \in \Omega_2\}$

$\Omega_1 \setminus \Omega_2 = \{\mu \in \Omega_1 \mid \text{for all } \mu' \in \Omega_2, \mu \text{ and } \mu' \text{ are not compatible}\}$
$\Omega_1 \text{⟕} \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2)$

Using an algebraic syntax for stSPARQL graph patterns which extends the one introduced for SPARQL in [15], we now define the result of evaluating a graph pattern over an stRDF graph.

Definition 4. *Let G be an stRDF graph over T, p a triple pattern and P_1, P_2 graph patterns. Evaluating a graph pattern P over a graph G is denoted by $[[P]]_G$ and is defined as follows [15]:*

1. *$[[p]]_G = \{\mu \mid dom(\mu) = var(p) \text{ and } \mu(p) \in G\}$, where $var(p)$ is the set of variables occurring in p.*
2. *$[[(P_1\ AND\ P_2)]]_G = [[P_1]]_G \bowtie [[P_2]]_G$*
3. *$[[(P_1\ OPT\ P_2)]]_G = [[P_1]]_G \text{⟕} [[P_2]]_G$*
4. *$[[(P_1\ UNION\ P_2)]]_G = [[P_1]]_G \cup [[P_2]]_G$*

The semantics of $FILTER$ expressions in stSPARQL are defined as in [15] for filters that do not involve spatial predicates. To define the semantics of spatial filters formally, we first need the following definitions.

Definition 5. *A k-*ary spatial term *is an expression of the following form:*

(i) a quantifier-free formula of $\mathcal{L}$ from the set C_k (in quotes).
(ii) a spatial variable from the set V_s^k.
(iii) $t \cap t'$ (intersection), $t \cup t'$ (union), $t \setminus t'$ (difference), $BD(t)$ (boundary), $MBB(t)$ (minimum bounding box), $BF(t, a)$ (buffer) where t and t' are k-ary spatial terms and a is a rational number.
(iv) the projection *$t[i_1, \ldots, i_{k'}]$ of a k-ary spatial term t where $i_1, \ldots, i_{k'}$ are positive integers less than or equal to k.*

Example 7. The following are examples of binary spatial terms:

$$\text{“}(x \geq 1 \wedge x = y) \vee y = 7\text{”}$$
$$?GEO \cap \text{“}(x \geq 1 \wedge x = y) \vee y = 7\text{”}$$
$$BD(?GEO \cap \text{“}(x \geq 1 \wedge x = y) \vee y = 7\text{”})$$
$$\text{“}(x \geq 1 \wedge x \leq 10 \wedge y \geq 0 \wedge x = y)\text{”}[1, 2] \cap \text{“}(z \geq 0 \wedge z \leq 10)\text{”}$$

Definition 6. *A* metric spatial term *is an expression of the form $f(t)$ where f is one of the metric functions VOL (volume), $AREA$ (area or surface), LEN (length), MAX (maximal value) or MIN (minimal value) and t is a k-ary spatial term. In the case of $AREA$ we require $k \geq 2$. In the case of LEN, MAX and MIN, we require $k = 1$.*

Example 8. The following are examples of metric spatial terms:

$$AREA(\text{“}(x \geq 1 \wedge x \leq 10 \wedge y \geq 0 \wedge x = y)\text{”})$$
$$MIN(\text{“}(x \geq 1 \wedge x \leq 10 \wedge y \geq 0 \wedge x = y)\text{”}[1])$$

Note that Definition 6 is not recursive like Definition 5 i.e., f can only be applied once to a k-ary spatial term. The result of the application of f is a real number and the definition of mapping has already catered for this possibility.

Definition 7. *A* spatial term *is a k-ary spatial term or a metric spatial term.*

We will be interested in the value of a k-ary spatial term t for a given mapping μ such that the variables of t are all among the spatial variables of μ. This is captured by the following definition.

Definition 8. *Let t be a spatial term. Let μ be a mapping such that all the spatial variables of t are elements of $dom(\mu)$. The* value *of t for μ is denoted by $\mu(t)$ and is defined as follows:*

(i) *If t is an element of C_k then $\mu(t) = t$.*
(ii) *If t is a spatial variable x then $\mu(t) = \mu(x)$.*
(iii) *If t is a projection expression of the form $t'[i_1, \ldots, i_{k'}]$ then $\mu(t)$ is a quantifier-free formula ϕ of $\mathcal{L}$ which is obtained after eliminating from $\mu(t')$ the variables corresponding to all the other dimensions except $i_1, \ldots, i_{k'}$.*
(iv) *If t is the* intersection *$t' \cap t''$ of two k-ary spatial terms then $\mu(t) = \mu(t' \cap t'') = \mu(t') \wedge \mu(t'')$.*[2]
(v) *If t is the* union *$t' \cup t''$ of two k-ary spatial terms then $\mu(t) = \mu(t' \cup t'') = \mu(t') \vee \mu(t'')$.*
(vi) *If t is the* difference *$t' \setminus t''$ of two k-ary spatial terms then $\mu(t) = \mu(t' \setminus t'') = \mu(t') \wedge \neg\mu(t'')$.*
(vii) *If t is $MBB(t')$ where t' is a k-ary spatial term, then $\mu(t)$ is a quantifier-free formula of the language $\mathcal{L}$ that represents the minimum bounding box of $\mu(t')$.*
(viii) *If t is $BD(t')$ where t' is a k-ary spatial term, then $\mu(t)$ is a quantifier-free formula of the language $\mathcal{L}$ that represents the boundary of $\mu(t')$.*
(ix) *If t is $BF(t', a)$ where t' is a k-ary spatial term and a is a rational number, then $\mu(t)$ is a quantifier-free formula of the language $\mathcal{L}$ that represents the buffer of $\mu(t')$ within distance a. The buffer of t contains t and a zone of width a around t.*
(x) *If t is $VOL(t')$, $AREA(t')$ or $LEN(t')$ where t' is a k-ary spatial term, then $\mu(t)$ is a real number that represents the volume, surface (or area) or length of $\mu(t')$.*
(xi) *If t is $MIN(t')$, $MAX(t')$ where t' is a unary spatial term, then $\mu(t)$ is a real number that represents the minimum or the maximum value of $\mu(t')$.*

To guarantee closure of stSPARQL, it is important to point out that the value $\mu(t)$ in the above definition is a well-defined formula of $\mathcal{L}$ in the cases *(i)*-*(ix)* and a real number in the cases of *(x)* and *(xi)*. This is easy to see for cases *(i)*-*(vi)*. For the case $t = MBB(t')$, $\mu(t)$ is $\bigwedge_{i=1}^{k} (l_i \leq x_i \wedge x_i \leq u_i)$ where l_i, u_i are the minimum and maximum values of x_i for which the formula $\mu(t')$ holds in the structure $\mathcal{Q}$. For the case $t = BD(t')$, the formula $\mu(t)$ can be constructed by performing quantifier elimination in the quantified formula defining the boundary given in

[2] In this and subsequent definitions, we assume that standardization of variables takes place before forming the conjunction, disjunction of formulas etc.

Proposition 3.1 of [10]. For the case $t = BF(t', a)$ and the standard definition of buffer that uses the Euclidean distance , the formula $\mu(t)$ is not general an element of $\mathcal{L}$ (e.g., `BF("x=0 and y=0",1)` is the unit circle with center $(0, 0)$). There are two alternative non-standard definitions of BF that allow us to stay in the realm of linear constraints. In the first case, BF can be defined using the *Manhattan distance* which measures the distance between two points along axes at right angles. For example, in the case of two dimensions, the formula $\mu(t)$ would now be the formula that remains if we eliminate variables x', y' from the formula:

$$\begin{gathered}(\phi(x', y') \wedge 0 \leq x - x' \leq a \wedge 0 \leq y - y' \leq a)\vee \\ (\phi(x', y') \wedge 0 \leq x - x' \leq a \wedge 0 \leq y' - y \leq a)\vee \\ (\phi(x', y') \wedge 0 \leq x' - x \leq a \wedge 0 \leq y - y' \leq a)\vee \\ (\phi(x', y') \wedge 0 \leq x' - x \leq a \wedge 0 \leq y' - y \leq a)\end{gathered}$$

where $\phi(x', y')$ is the formula $\mu(t')$. If using Manhattan distance seems like a crude alternative to the standard definition then more detailed alternatives are possible. For example, if t defines a polygon then $BF(t, a)$ is a new polygon that contains t and the zone of width a around the polygon (however, "circular" curves are approximated by polylines). Note that the same approach is followed by vector data models e.g. the computational geometry library CGAL [3]. The cases *(x)* and *(xi)* are easy to see as well.

Definition 9. *An atomic* spatial condition *is an expression in any of the following forms:*

(i) $t_1 \ R \ t_2$ *where* t_1 *and* t_2 *are k-ary spatial terms and* R *is one of the topological relationships DISJOINT, TOUCH, EQUALS, INSIDE, COVEREDBY, CONTAINS, COVERS, OVERLAPBDDISJOINT (overlap with disjoint boundaries) or OVERLAPBDINTER (overlap with intersecting boundaries).*

(ii) a linear equation or inequality of $\mathcal{L}$ *with metric spatial terms in the place of variables.*

Note that the form (ii) does not destroy closure of our language since these equations/inequalities allows linear equations or inequalities with terms that evaluate to real numbers and they will only be checked for satisfaction (see Definition 11), not used as constraints i.e., as elements of sets C_k.

Example 9. The following are atomic spatial selection conditions:

$$?GEO1 \ INSIDE \ \text{``}x \geq 1 \wedge x \leq 5 \wedge y \geq 0 \wedge y \leq 5\text{''}$$
$$AREA(?GEO1) \geq 2 \cdot AREA(?GEO2)$$

Definition 10. *A* spatial condition *is a Boolean combination of atomic spatial conditions.*

Definition 11. *A mapping* μ satisfies *a spatial condition* R *(denoted* $\mu \models R$*) if*

[3] CGAL, `http://www.cgal.org/`, last accessed February 20, 2010

1. R is atomic and the spatial condition that results from substituting every spatial variable x of R with $\mu(x)$ holds for semi-linear sets in $\mathbb{Q}^n$.
2. R is $(\neg R_1)$, R_1 is a spatial condition, and it is not the case that $\mu \models R_1$.
3. R is $(R_1 \vee R_2)$, R_1 and R_2 are spatial conditions, and $\mu \models R_1$ or $\mu \models R_2$.
4. R is $(R_1 \wedge R_2)$, R_1 and R_2 are spatial conditions, and $\mu \models R_1$ and $\mu \models R_2$.

The semantics of spatial filters can now be defined as follows.

Definition 12. *Given an stRDF graph G over T, a graph pattern P and a spatial condition R, we have: $[[P\ FILTER\ R]]_G = \{\mu \in [[P]]_G \mid \mu \models R\}$.*

Now we can define the semantics of the `SELECT` clause of an stSPARQL expression where variables (spatial or non-spatial) are selected and new spatial terms are computed. To capture the peculiarities of the `SELECT` clause of stSPARQL, we first need the following definitions.

Definition 13. *Let t be a spatial (resp. metric spatial) term and z a spatial (resp. real) variable that does not appear in t. Then, $t\ AS\ z$ is called an* extended spatial term *with* target variable *z.*

Example 10. $(BD(?GEO) \cap \text{“}x = 1\text{”})\ AS\ ?L$ is an extended spatial term.

Definition 14. *A* projection specification *is a set consisting of non-spatial variables, spatial variables and extended spatial terms such that all the target variables of the extended spatial terms are different from each other and different from each spatial variable.*

Definition 15. *Let μ be a mapping and W a projection specification with spatial and non-spatial variables $x_1, \ldots, x_l$ and extended spatial terms $t_1\ AS\ z_1, \ldots, t_m\ AS\ z_m$. Then, $\pi_W(\mu)$ is a new mapping such that*

(i) $dom(\pi_W(\mu)) = \{x_1, \ldots, x_l, z_1, \ldots, z_m\}$.
(ii) $\pi_W(\mu)(x_i) = \mu(x_i)$ for $1 \leq i \leq l$ and $\pi_W(\mu)(z_j) = \mu(t_j)$ for $1 \leq j \leq m$.

Example 11. Let μ be the mapping $\{?S \rightarrow s_1,\ ?O \rightarrow John,\ ?GEO \rightarrow \text{“}x \geq 1 \wedge x \leq 5 \wedge\ y \geq 0 \wedge\ y \leq 5\text{”}\}$ and W the projection specification $\{?O, (BD(?GEO) \cap \text{“}x = 1\text{”})\ AS\ ?L\}$ then $\pi_W(\mu)$ is the following mapping:

$$\{?O \rightarrow John, ?L \rightarrow \text{“}x = 1 \wedge y \geq 0 \wedge y \leq 5\text{”}\}.$$

The next definition gives the semantics of an arbitrary stSPARQL query.

Definition 16. *An stSPARQL query is a pair (W, P) where W is a projection specification and P is a graph pattern. The* answer *to an stSPARQL query (W, P) over a graph G is the set of mappings $\{\pi_W(\mu) \mid \mu \in [[P]]\}$.*

Example 12. Let G be the following stRDF graph:
$\{(s_1, geom, \text{“}x \geq 1 \wedge x \leq 5 \wedge y \geq 0 \wedge y \leq 5\text{”}), (s_1, owner, John)\}$
and consider the query with $W = \{?O, BD(?GEO) \cap \text{“}x = 1\text{”}\ AS\ ?L\}$ and $P = (?S, owner, ?O)\ AND\ (?S, geom, ?GEO)$. Then, the answer to (W, P) over G is the set which consists of the mapping: $\{?O \rightarrow John, ?L \rightarrow \text{“}x = 1 \wedge y \leq 5 \wedge y \geq 0\text{”}\}$

5 Related Work

Let us now compare stRDF and stSPARQL with relevant proposals in the literature. The closest language to stSPARQL is SPARQL-ST presented in Perry's Ph.D. thesis [4]. SPARQL-ST adopts the model of temporal RDF graphs of [1] to represent the valid time of a triple. Similarly, stSPARQL offers support for valid time of a triple but uses a temporal constraint language to define a valid time. As a result, the notion of valid time in stSPARQL is more expressive (thus it requires a more sophisticated implementation). The spatial part of stSPARQL and SPARQL-ST are significantly different. SPARQL-ST assumes a particular upper ontology expressed in RDFS for modeling theme, space and time [4]. The spatial part of this upper ontology uses the class `geo:SpatialRegion` and its subclasses (e.g., `geo:Polygon`) defined in GeoRSS in order to model spatial geometries (e.g., polygons). Thematic data (e.g., a city) can then be connected to their spatial geometry (e.g., a polygon) using the property `stt:located_at`. Spatial geometries in SPARQL-ST are specified by sets of RDF triples that give various details of the geometry depending on its type (e.g., for a 2-dimensional polygonal area, they give the coordinates of its boundary and the relevant coordinate reference system). SPARQL-ST provides a set of built-in spatial conditions that can be used in SPATIAL FILTER clauses to constrain the geometries that are returned as answers to queries. Although a semantics for SPARQL-ST is presented in [4], the treatment of spatial conditions in these semantics is unsatisfactory in our opinion. The notion of "when a spatial condition evaluates to true" that is used to give semantics to built-in spatial conditions (page 99 of [4]) is not defined formally but is left to the intuition of the reader. When this definition is given explicitly, it will have to rely on the different types of geometries (e.g., `geo:Polygon`) allowed by the spatial ontology of [4], properties of these geometries (e.g., `geo:lrPosList`) and relevant co-ordinate systems (e.g., `geo:CRS_NAD83`). Currently, these semantics are hardwired in the implementation of SPARQL-ST presented in [4]. This means that if someone wants to use a different spatial ontology (e.g., an ontology based on the Open GIS SQL geometry types), this cannot be done unless the semantics of SPARQL-ST and its implementation are modified appropriately. Since geometries in stRDF and stSPARQL are based on the mathematical concept of semi-linear subsets of $\mathbb{Q}^k$, stSPARQL (as opposed to SPARQL-ST) can be given an elegant semantics based on well-understood mathematical machinery from constraint databases as we showed in Section 4 of this paper.

In addition, the new literal datatype `strdf:SemiLinearPointSet` of stRDF alluded to in Section 2.2 can be used together with spatial ontologies expressed in RDFS to give the same kind of class-based modeling capabilities offered by SPARQL-ST. Thus, stRDF and stSPARQL impose very minimal requirements to Semantic Web developers that want to use our approach: all they have to do is utilize a new literal datatype such as `strdf:SemiLinearPointSet`.

Two other papers related to our work on stSPARQL are [2,3] by Kolas and colleagues. Compared with our work on stRDF and stSPARQL, the system SPAUK presented in [2] has problems similar to the ones we pointed out for

SPARQL-ST. First, no semantics for query evaluation are given. Secondly, even if these semantics are given in great detail, they will rely on the spatial ontologies assumed by SPAUK. Thus, any query processor that implements these semantics will need to be extended if users of the system decide to use different spatial ontologies (this is said explicitly in [2]).

To summarize, the constraint extension to RDF and SPARQL that we have advocated in this paper gives us the following benefits. First, our extension is general and its only primitive (semi-linear point set) does not depend on the application at hand. This is in contrast to approaches based on spatial/temporal ontologies which need to choose an ontology with classes (e.g., Point, Interval, Polygon etc.) appropriate for the application at hand. This is reminiscent of spatial DBMSs based on data types (e.g., PostGIS and Oracle) that offer their own, often incompatible, spatial data type systems. Second, the elaboration of a query in a constraint-based query language such as stSPARQL does not depend on the spatial data type (or class) of the objects queried or the results that will be returned. Thus, the programmer does not need to think about composition of operators, closure etc. In approaches based on classes or data types, the programmer needs constantly to be thinking about the classes/data types of the objects queried (this point has been also made in [3]). Third, our extensions can be given a natural and intuitive semantics by extending the standard algebraic semantics of SPARQL [15]. Finally, as we showed in Section 3, our extensions to RDF and SPARQL can easily be integrated with current work on sensor network ontologies to realize the vision of the Semantic Sensor Web.

6 Conclusions and Future Work

In this paper we studied the problem of designing a data model and a query language that can be used in the Semantic Sensor Web for representing and querying spatial and temporal data. We proposed the data model stRDF and the query language stSPARQL. We gave a formal definition of stRDF, introduced stSPARQL by examples and presented a detailed semantics of stSPARQL using the algebraic approach pioneered for SPARQL in [15]. Finally, we compared our approach with related work. Our future work concentrates on studying the complexity of stSPARQL query processing theoretically, and carrying out an implementation of the language for the cases of 2 and 3 dimensions that are the most interesting ones in practice. In these cases, we would like to apply the lessons learned from the implementation of relational constraint databases [11] and demonstrate that our proposal can be implemented efficiently in comparison with competitive approaches.

Acknowledgements

We thank Jianwen Su and Gabi Kuper who were always available to answer questions regarding CSQL. We also thank Peter Revesz for interesting discussions on constraint databases.

References

1. Gutierrez, C., Hurtado, C., Vaisman, A.: Introducing Time into RDF. IEEE TKDE (2007)
2. Kolas, D., Self, T.: Spatially Augmented Knowledgebase. In: ISWC/ASWC2007. (2007)
3. Kolas, D.: Supporting Spatial Semantics with SPARQL. In: Terra Cognita Workshop. (2008)
4. Perry, M.: A Framework to Support Spatial, Temporal and Thematic Analytics over Semantic Web Data. PhD thesis, Wright State University (2008)
5. Singh, R., Turner, A., Maron, M., Doyle, A.: GeoRSS: Geographically Encoded Objects for RSS Feeds. `http://georss.org/gml`, last accessed February 20, 2010 (2008)
6. Kanellakis, P., Kuper, G., Revesz, P.: Constraint Query Languages. In: PODS. (1990)
7. Rigaux, P., Scholl, M., Voisard, A.: Introduction to Spatial Databases: Applications to GIS. Morgan Kaufmann (2000)
8. Revesz, P.Z.: Introduction to Constraint Databases. Springer (2002)
9. Koubarakis, M.: Database Models for Infinite and Indefinite Temporal Information. Information Systems **19** (1994) 141–173
10. Vandeurzen, L., Gyssens, M., Gucht, D.V.: On the expressiveness of linear-constraint query languages for spatial databases. Theoretical Computer Science (2001)
11. Rigaux, P., Scholl, M., Segoufin, L., Grumbach, S.: Building a constraint-based spatial database system: model, languages, and implementation. Information Systems **28**(6) (2003) 563–595
12. Kuper, G., Ramaswamy, S., Shim, K., Su, J.: A Constraint-based Spatial Extension to SQL. In: Proceedings of the 6th International Symposium on Advances in Geographic Information Systems. (1998)
13. Sheth, A., Henson, C., Sahoo, S.S.: Semantic Sensor Web. Internet Computing, IEEE **12**(4) (2008) 78–83
14. Egenhofer, M.J.: Toward the Semantic Geospatial Web. In: ACM-GIS, New York, NY, USA (2002)
15. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. In: ISWC. (2006) 30–43
16. Parsia, B., Sattler, U.: OWL 2 Web Ontology Language, Data Range Extension: Linear Equations. W3C Working Group Note (October 2009) , `http://www.w3.org/TR/2009/NOTE-owl2-dr-linear-20091027/`, last accessed February 20, 2010.
17. Kyzirakos, K., Koubarakis, M., Kaoudi, Z.: Data models and languages for registries in SemsorGrid4Env. Deliverable D3.1, SemSorGrid4Env (2009)
18. Neuhaus, H., Compton, M.: The Semantic Sensor Network Ontology: A Generic Language to Describe Sensor Assets. In: AGILE 2009 Pre-Conference Workshop Challenges in Geospatial Data Harmonisation. (2009)
19. Henson, C., Pschorr, J., Sheth, A., Thirunarayan, K.: SemSOS: Semantic Sensor Observation Service. In: CTS. (2009)
20. Cui, Z., Cohn, A.G., Randell, D.A.: Qualitative and Topological Relationships in Spatial Databases. In: Advances in Spatial Databases. (1993)
21. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. CACM **26**(11) (1983)