Linear Constraint Databases for the Semantic Web:
The Model stRDF and the Query Language stSPARQL

Manolis Koubarakis
Kostis Kyzirakos
Manos Karpathiotakis

Dept. of Informatics and Telecommunications
National and Kapodistrian University of Athens
Talk Outline

• Motivation
• The Data Model stRDF
• The Query Language stSPARQL
• Implementation
• Future Work
Motivation

How do we represent spatial and temporal metadata in the Semantic Web?

Example:

• The vision of the **Semantic Sensor Web**: annotate sensor data and services to enable discovery, integration, interoperability etc. (Sheth et al. 2008, SensorGrid4Env)

• Sensor annotations involve **thematic**, **spatial** and **temporal metadata**. Examples:
  - The sensor measures temperature. (thematic)
  - The sensor is located in the location represented by point \((A, B)\). (spatial)
  - The sensor measured \(-3^0\) Celsius on 26/01/2010 at 03:00pm. (temporal)
Example:

- The vision of the **Geospatial Semantic Web**: Provide a formal semantic specification to enable the discovery, query and consumption of geospatial content.

How about using RDF?

Good idea. But **RDF can represent only thematic metadata** properly. What can we do about spatial and temporal metadata?
Previous Work

- Time in RDF (Gutierrez and colleagues, 2005).
- SPARQL-ST (Perry, 2008).
- SPAUK (Kollas, 2007)
- ...

OGC Technical and Planning Committee Meeting 2010 - Geosemantics DWG 9 March 2010
Our Approach

• Use ideas from constraint databases (Kanellakis, Kuper and Revesz, 1991).
  
  **Slogan:** What’s in a tuple? Constraints.

• Extend RDF to a constraint database model.
  
  **Slogan:** What’s in a triple? Constraints.

• Extend SPARQL to a constraint query language.

• Follow exactly the approach of CSQL (Kuper et al., 1998).
  
  – Nested relational model with **one level of nesting to represent point sets.**
  
  – Use **linear constraints** to encode these point sets (that are used to represent spatial and temporal objects).
Spatial Metadata Using Constraints

- We start with a FO language $\mathcal{L} = \{\leq, +\} \cup \mathbb{Q}$ over the structure $\mathcal{Q} = \langle \mathbb{Q}, \leq, +, (q)_{q \in \mathbb{Q}} \rangle$.

- **Atomic formulae:** linear equations and inequalities of the form

  $$\left(\sum_{i=1}^{p} a_i x_i\right) \Theta a_0$$

  where $\Theta$ is one of $=, \leq$ or $<$.

- **Semi-linear point sets:** sets that can be defined by quantifier-free formulas of $\mathcal{L}$. 
Example

\[ x \leq 8 \land y \geq 2 \land y \leq x \]
Example

\[ 8 \leq x \leq 14 \land 2 \leq y \leq 8 \]
Example

\[(x \leq 8 \land y \geq 2 \land y \leq x) \lor (8 \leq x \leq 14 \land 2 \leq y \leq 8)\]
The sRDF data model

- Let $I$, $B$ and $L$ be the sets of IRIs, blank nodes and literals.
- Let $C_k$ be the set of quantifier-free formulae of $\mathcal{L}$ with $k$ free variables ($k = 1, 2, \ldots$).
- Let $C$ be the infinite union $C_1 \cup C_2 \cup \cdots$.
- An sRDF triple is an element of the set $(I \cup B) \times I \times (I \cup B \cup L \cup C)$.
- An sRDF graph is a set of sRDF triples.
- 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 `strdf:SemiLinearPointSet`. 
Example of sRDF graph

ex:s1 rdf:type ex:Sensor .
ex:s1 ex:has_location "x=40 and y=15"^^strdf:SemiLinearPointSet .
Temporal Metadata Using Constraints

- **Time structure**: 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**: 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**: Boolean combinations of atomic temporal constraints using a single variable.
Example

\[ t = 1 \lor (t \geq 5 \land t \leq 10) \lor (t \geq 12 \land t \leq 15) \]
The stRDF data model

stRDF extends sRDF with the ability to represent the valid time of a triple:

- An **stRDF quad** \((a, b, c, \tau)\) is an sRDF triple \((a, b, c)\) with a fourth component \(\tau\) which is a temporal constraint.

- An **stRDF graph** is a set of sRDF triples and stRDF quads.
Example of stRDF graph

ex:obs1Result om:uom ex:Celsius .
ex:obs1Result om:value "27"

"10 <= t <= 11"^^strdf:SemiLinearPointSet .
The Query Language stSPARQL

- Syntactic extension of SPARQL
- Formal semantics
  - Closure
- Efficient implementation
Example - Dataset I

Sensor metadata using the SSN ontology:

```
ex:sensor1 rdf:type ssn:Sensor .
ex:sensor1 ssn:measures ex:temperature .
ex:sensor1 ssn:supports ex:grounding1 .
ex:grounding1 ssn:hasLocation ex:location1 .
ex:location1 rdf:type ssn:Location .
ex:location1 strdf:hasGeometry
  "x=40 and y=15"^^strdf:SemiLinearPointSet .
```
Spatial selection. Find the URIs of the sensors that are inside the rectangle R(0, 0, 100, 100)?

select ?S
    ?L rdf:type ssn:Location .
    ?L strdf:hasGeometry ?GEO .
    filter(?GEO inside "0<=x<=100 and 0<=y<=100") }

OGC Technical and Planning Committee Meeting 2010 - Geosemantics DWG 9 March 2010
Answer

?q

ex:sensor1
**Queries - Dataset I**

**Spatial selection with OPTIONAL.** Find the URIs of the sensors that optionally has a grounding that has a location which is inside the rectangle R(0, 0, 100, 100)?

```
select ?S ?GEO
    optional {
      ?L rdf:type ssn:Location .
      ?L strdf:hasGeometry ?GEO .
      filter(?GEO inside "0<=x<=100 and 0<=y<=100") }
}
```
<table>
<thead>
<tr>
<th>S</th>
<th>GEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>ex:sensor1</td>
<td>&quot;x=40 and y=15&quot;^^strdf:SemiLinearPointSet</td>
</tr>
<tr>
<td>ex:sensor2</td>
<td></td>
</tr>
</tbody>
</table>
Example - Dataset II

Sensor observation metadata using the O&M ontology:

ex:sensor1 rdf:type ex:TemperatureSensor .
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=40 and y=15"^^strdf:SemiLinearPointSet .
ex:obs11 om:result ex:obs1Result .
ex:obs1Result rdf:type om:ResultData .
ex:obs1Result om:uom ex:Celsius .
ex:obs1Result om:value "27"
  "(10 <= t <= 11)"^^strdf:SemiLinearPointSet .
Example - Dataset II (cont’d)

Metadata about geographical areas:

```
ex:area1 rdf:type ex:RuralArea .
ex:area1 ex:hasName "Brovallen" .
ex:area1 strdf:hasGeometry
    "(-x+1.363636y<-5.272576 and y<79 and -y<-13 and
     x-0.090909y<131.818133) or (y<13 and x<133 and
     -x-1.5y<-128.5)"^^strdf:SemiLinearPointSet .
```
Spatial and 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 .
  ?RA rdf:type ex:RuralArea .
  ?LOC strdf:hasGeometry ?OBSLOC .
  filter(?T contains "t = 11" && ?RAGEO contains ?OBSLOC) }
### Answer

<table>
<thead>
<tr>
<th>?V</th>
<th>?RA</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;27&quot;</td>
<td>ex:area1</td>
</tr>
</tbody>
</table>
Example - Dataset III

Dataset from a typical GIS application:

```text
ex:s1 rdf:type ex:State .
ex:s1 ex:has_name "New York" .
ex:s1 ex:has_geometry "(-66x+90y<-8748 and 26y<12974
    and 110y>47630 and -66x+6y>-52380) or (24y<10392
    and -6x+15y>1497 and 6x+9y>8751) or (-6x+15y <1497
    and 18x<14994 and 12x+15y>16221)"^^strdf:SemiLinearPointSet .

ex:lp13 ex:land_use "forest" .
ex:lp13 ex:has_geometry
    "x+0.2y<=798 and x-2.5y<=-286 and -x-0.156864y<=-772
    and -x+11.6y<=4078"^^strdf:SemiLinearPointSet .

ex:fs1 rdf:type ex:FireStation .
ex:fs1 ex:has_location "x=796 and y=437"^^strdf:SemiLinearPointSet .
```
Example - Dataset III (cont’d)
Queries - Dataset III

Query with intersection of areas and spatial function application. Find all land parcels that are forests and intersect the state of New York; compute the area of this intersection.

```
select ?LP, AREA(?GEO1 INTER ?GEO2) AS ?LPAREA
where {?
  ?LP ex:has_use "forest" .
  ?LP ex:has_geometry ?GEO1
  ?S ex:has_name "New York".
  ?S ex:has_geometry ?GEO2 .
  filter(?GEO1 overlap ?GEO2) }
```
## Answer

<table>
<thead>
<tr>
<th>?LP</th>
<th>?LPAREA</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ex:lp13&quot;</td>
<td>88.625366</td>
</tr>
<tr>
<td>&quot;ex:lp15&quot;</td>
<td>644.926392</td>
</tr>
</tbody>
</table>
**Queries - Dataset III**

**Projection and spatial function application.** Find the URIs of the fire stations that are north of the state of Pennsylvania.

```sparql
select ?FS
where {?FS rdf:type ex:FireStation .
  ?FS ex:has_location ?FS_LOC .
  ?S ex:has_name "Pennsylvania".
  ?S ex:has_geometry ?GEO .
  filter(MAX(?GEO[2]) < MIN(?FS_LOC[2]))) }
```
Answer

?FS
"ex:fs1"
"ex:fs4"
"ex:fs5"
"ex:fs6"
What Else is There in stSPARQL Syntax?

- **k-ary spatial terms**
  - quantifier-free formulas
  - spatial variables
  - intersection, union, difference, boundary, buffer, minimum bounding box of $k$-ary spatial terms
  - projections of $k$-ary spatial terms

- **Metric spatial terms**
  - VOL, AREA, LEN, MAX, MIN

- **Selection predicates**
  - Spatial predicates (topological): DISJOINT, TOUCH, EQUALS, INSIDE, COVEREDBY, CONTAINS, COVERS, OVERLAPBDDISJOINT or OVERLAPBDINTER
  - Temporal predicates: BEFORE, EQUAL, MEETS, OVERLAPS, DURING, STARTS, FINISHES
  - a linear equation or inequality of $\mathcal{L}$ with metric spatial terms in the place of variables

- **Construction of new spatial terms**
  - intersection/union/difference/projection of spatial terms
stSPARQL Semantics

- Extension of the SPARQL semantics of (Perez et al., 2006).
  - Extend the concept of mapping.
  - The semantics of AND, OPT, UNION remain the same.
  - We need to define carefully the evaluation of spatial terms and the semantics of spatial and temporal filters.
The System Strabon

- Query Engine
  - Parser
  - Optimizer
  - Evaluator
  - Transaction Manager

- Constraint Engine
  - Constraint Solver
  - Representation Translator

- Storage Manager
  - Repository
  - SAIL
  - RDBMS

PostGIS

OGC Technical and Planning Committee Meeting 2010 - Geosemantics DWG 9 March 2010
Future Work

• Complete the implementation (see demo for what works now).

• Incomplete information (use the blank nodes for a bit more than what they are currently used).

• OWL 2 and constraints?

• Show that similar extension to RDF and SPARQL could be done for OGC datatypes (instead of constraints).
Thank you!
Backup slides
What if you don’t like constraints?

• Introduce datatypes like Point, Line, Polygon etc. and specify them using XML Schema.

• Introduce the appropriate operators in SPARQL e.g., introduce an operator @ to check whether a geometry is completely contained by another geometry.

• Semantics?
What if you don’t like constraints? (cont’d)

Then one could describe a sensor with the following triples:

ex:located_at rdfs:range georss:Point .
ex:sensor1 rdf:type ex:Sensor .
ex:sensor1 ex:located_at "5 5"^^georss:Point .

Then one could write a window query that asks for all the sensors that are located inside the rectangle R(0, 0, 10, 10) as follows:

select ?sensor
where { ?sensor rdf:type ex:Sensor .
  ?sensor ex:located_at ?sensor_loc .
  filter(?sensor_loc @ "0 0 10 10"^^georss:Box) }

SeRQL query

SELECT airport, city
FROM {airport} rdf:type {c5:C5CapableAirport};
    filter:satisfiesFilter {filter},
    {filter} rdf:type {filter:DWithin};
    filter:referenceGeometry {city};
    filter:radius {50000"^^xsd:int},
    {city} rdf:type {cities:City};
    filter:satisfiesFilter {cityFilter},
    {cityFilter} rdf:type {filter:PropertyIsLike};
    filter:property{"NAME"};
    filter:literal{"Saint Louis"}
stSPARQL query

SELECT ?airport ?city
WHERE {
    ?airport rdf:type c5:C5CapableAirport;
    ?airport ex:hasGeometry ?airportGEO .
    ?city rdf:type cities:City;
    ex:hasName ?cityName;
    ex:hasGeometry ?cityGEO .
    FILTER (?cityName LIKE "Saint Louis" &&
    BUFFER(?cityGEO, 5000) contains ?airportGEO) }

OGC Technical and Planning Committee Meeting 2010 - Geosemantics DWG 9 March 2010
Example - Dataset III

Moving sensor metadata using the SSN ontology:

ex:sensor2 ssn:measures ex:temperature .
ex:grounding2 ssn:hasLocation ex:location2 .
ex:location2 rdf:type ssn:Location .
Example - Dataset III (cont’d)

```
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.
```
Example - Dataset III (cont’d)

Metadata about geographical areas:

ex:area1 rdf:type ex:RuralArea .
ex:area1 ex:hasName "Brovallen" .
ex:area1 strdf:hasGeometry
   "(-x+1.363636y<-5.272576 and y<79 and -y<-13 and
   x-0.090909y<131.818133) or (y<13 and x<133 and
   -x-1.5y<-128.5)"^^strdf:SemiLinearPointSet .
Queries - Dataset III

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

```
    ?Y rdf:type ssn:Location .
    ?RA rdf:type ex:RuralArea .
    ?X ssn:hasLocation ?Y .
    ?Y strdf:hasTrajectory ?TR .
    ?RA ex:hasName "Brovallen" .
    ?RA strdf:hasGeometry ?GEO .
    filter(?TR[1,2] overlap ?GEO) }
```
**Answer**

<table>
<thead>
<tr>
<th>(?\text{SENSEDAREA})</th>
<th>(?\text{T1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;((y=0.5x and 0&lt;=x&lt;=50) or (x=50 and 25&lt;=y&lt;=50)) and ((y&lt;79 and -y&lt;-13 and -x+1.363636y&lt;-5.272576 and x-0.090909y&lt;131.818133) or (y&lt;13 and x&lt;133 and -x-1.5y&lt;-128.5))&quot; (\text{\textasciitilde} \text{strdf:SemiLinearPointSet})</td>
<td>&quot;0 &lt;= t &lt;= 10&quot; (\text{\textasciitilde} \text{strdf:SemiLinearPointSet}).</td>
</tr>
</tbody>
</table>
**One More Query**

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

```
select ?SN
  ?RA rdf:type ex:RuralArea .
  ?X ssn:hasLocation ?Y .
  ?RA ex:hasName "Brovallen".
  ?RA strdf:hasGeometry ?GEO .
  filter(MAX(?GEO[2])<MIN(?SN_LOC[2]))}
```
Implementation Details: Store

Constraints (at most 2 vars) → orbital → DNF → cddlib → Vector space → PostGIS
Implementation Details: Query

- stSPARQL query
- Parser
- Optimizer
- Query Processor
- Vector to Constraints
- PostGIS
- results
Strabon in SensorGrid4Env

Semantic Registration and Discovery Service

Service

Registration

Query

Strabon

Query Engine

Constraint Engine

Storage Manager

PostGIS

Metadata provider

Metadata consumer

OGC Technical and Planning Committee Meeting 2010 - Geosemantics DWG 9 March 2010