A Deductive System for Annotated RDFS

DERI Institute Meeting

Umberto Straccia  Nuno Lopes  Gergely Lukácsy
Antoine Zimmermann  Axel Polleres

Presented by: Nuno Lopes

May 28, 2010
Sensor data
Sensor data

sensors readings output:

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>IP Address</th>
<th>Reading</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010-05-28</td>
<td>14:57:51</td>
<td>10.254.2.15</td>
<td>4302</td>
<td>83</td>
</tr>
<tr>
<td>2010-05-28</td>
<td>14:57:51</td>
<td>10.254.3.1</td>
<td>4302</td>
<td>83</td>
</tr>
<tr>
<td>2010-05-28</td>
<td>14:57:51</td>
<td>10.254.2.6</td>
<td>4302</td>
<td>83</td>
</tr>
</tbody>
</table>
Sensor data

sensors readings output:

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>IP Address</th>
<th>Port</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010-05-28 14:57:51</td>
<td>10.254.2.15</td>
<td>4302</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>2010-05-28 14:57:51</td>
<td>10.254.3.1</td>
<td>4302</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>2010-05-28 14:57:51</td>
<td>10.254.2.6</td>
<td>4302</td>
<td>83</td>
<td></td>
</tr>
</tbody>
</table>
Sensor data

Queries over sensor data:

Q1: “Where was Stefan before this institute meeting?”
Queries over sensor data:

Q1: “Where was Stefan before this institute meeting?”
Q2: “When were Stefan and Manfred in the same room?”
Represent sensor data as RDF

- RDF triples

:tag4302 :locatedIn :room311 .
Represent sensor data as RDF

- RDF triples

```plaintext
:tag4302 :locatedIn :room311 .
:tag4302 :locatedIn :room311 .
```
Represent sensor data as RDF

- RDF triples

```
:tag4302 :locatedIn :room311 .
:tag4302 :locatedIn :room311 .
:tag4302 :locatedIn :room310 .
```
Represent sensor data as RDF

- RDF triples

```<code>
:tag4302 :locatedIn :room311 .
:tag4302 :locatedIn :room311 .
:tag4302 :locatedIn :room310 .
</code>
```

Not enough info!
Represent sensor data as RDF

- **RDF triples**

  ```
  :tag4302 :locatedIn :room311 .
  :tag4302 :locatedIn :room311 .
  :tag4302 :locatedIn :room310 .
  ```

  Not enough info!

- **Domain vocabulary/ontology**

  ```
  :record1 a :SensorRecord;
  :tag :tag4302;
  :locatedIn :room311;
  ```
Represent sensor data as RDF

- RDF triples

```
:tag4302 :locatedIn :room311 .
:tag4302 :locatedIn :room311 .
:tag4302 :locatedIn :room310 .
```

- Reification

```
:record1 rdf:type rdf:Statement
  rdf:subject :tag4302;
  rdf:predicate :locatedIn ;
  rdf:object :room311 ;
```

Not enough info!
Represent sensor data as RDF

- RDF triples

\[
\begin{align*}
\text{:tag4302} & \quad \text{:locatedIn} & \quad \text{:room311} \\
\text{:tag4302} & \quad \text{:locatedIn} & \quad \text{:room311} \\
\text{:tag4302} & \quad \text{:locatedIn} & \quad \text{:room310}
\end{align*}
\]

- Reification

\[
\begin{align*}
\text{:record1} & \quad \text{rdf:type} & \quad \text{rdf:Statement} \\
& \quad \text{rdf:subject} & \quad \text{:tag4302} \\
& \quad \text{rdf:predicate} & \quad \text{:locatedIn} \\
& \quad \text{rdf:object} & \quad \text{:room311} \\
& \quad \text{:timestamp} & \quad \text{"2010-05-28 14:57:51"}
\end{align*}
\]

Not enough info!

No defined semantics!
Represent sensor data as RDF

- **RDF triples**

\[
\begin{align*}
:tag4302 & :locatedIn :room311 . \\
:tag4302 & :locatedIn :room311 . \\
:tag4302 & :locatedIn :room310 . \\
\end{align*}
\]

- **Reification**

\[
\begin{align*}
:record1 & \text{rdf:type } \text{rdf:Statement} \\
& \text{rdf:subject } :tag4302 ; \\
& \text{rdf:predicate } :\text{locatedIn } ; \\
& \text{rdf:object } :\text{room311 } ; \\
& \text{:timestamp } "2010-05-28 14:57:51" .
\end{align*}
\]

- **Annotated RDF**

\[
\begin{align*}
:tag4302 & :\text{locatedIn } :\text{room311 . } "2010-05-28 14:57:51"
\end{align*}
\]
Represent sensor data as RDF

- **RDF triples**

  
  ```
  :tag4302 :locatedIn :room311 .
  :tag4302 :locatedIn :room311 .
  :tag4302 :locatedIn :room310 .
  ```

  Not enough info!

- **Reification**

  ```
  :record1 rdf:type rdf:Statement
  rdf:subject :tag4302;
  rdf:predicate :locatedIn ;
  rdf:object :room311 ;
  ```

  No defined semantics!

- **Annotated RDF**

  ```
  ```

  Annotations refer to a specific domain, like temporal.
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a **new domain** you need to specify:

- the *representation* of the annotations
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a new domain you need to specify:

- the representation of the annotations: ["14:35", "14:57"]
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a **new domain** you need to specify:

- the *representation* of the annotations: ["14:35", "14:57"]
- an *order* between the elements
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a new domain you need to specify:

- the representation of the annotations: ["14:35", "14:57"]
- an order between the elements: ⊆
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a new domain you need to specify:

- the *representation* of the annotations: ["14:35", "14:57"]
- an *order* between the elements: \( \subseteq \)
- *universal* (\( \top \)) and *empty* (\( \bot \)) annotations
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a **new domain** you need to specify:

- the *representation* of the annotations: ["14:35", "14:57"]
- an *order* between the elements: ⊆

  *universal* (⊤) and *empty* (⊥) annotations:  
  
  \( ⊤ = [−\infty, +\infty] \)
  \( ⊥ = [] \)
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a **new domain** you need to specify:

- the *representation* of the annotations: ["14:35", "14:57"]
- an *order* between the elements: $\subseteq$

*universal* ($\top$) and *empty* ($\bot$) annotations: $\top = [-\infty, +\infty]$ $\bot = []$
operator ($\land$) for *conjunction* of annotations
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a **new domain** you need to specify:

- the **representation** of the annotations: ["14:35", "14:57"]
- an **order** between the elements: \( \subset \)

*universal* \( (\top) \) and *empty* \( (\bot) \) annotations:

\( \top = [-\infty, +\infty] \)
\( \bot = [] \)

operator \( (\land) \) for **conjunction** of annotations:

\( \cap \)
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a **new domain** you need to specify:

- the *representation* of the annotations: ["14:35", "14:57"]
- an *order* between the elements: ⊆

*universal* (⊤) and *empty* (⊥) annotations:  

\[ \top = [-\infty, +\infty] \quad \bot = [] \]

operator (∧) for *conjunction* of annotations: \( \cap \)

\[ ["09:25","11:49"] \land ["10:35", "12:57"] = ["10:35", "11:49"] \]
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a **new domain** you need to specify:

- the *representation* of the annotations: ["14:35", "14:57"]
- an *order* between the elements: $\subseteq$
  
  *universal* ($\top$) and *empty* ($\bot$) annotations: $\top = [\neg\infty, +\infty]$ $\bot = []$

  operator ($\land$) for **conjunction** of annotations: $\cap$

  operator ($\lor$) for **combining** annotations
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ['09:25', '11:49']
:tag4302 :locatedIn :room311 . ['10:35', '12:57']

To define a new domain you need to specify:

- the *representation* of the annotations: ['14:35', '14:57']
- an *order* between the elements: ⊆

universal (⊤) and empty (⊥) annotations:

⊤ = [−∞, +∞] ⊥ = []

operator (∧) for conjunction of annotations: ∩
operator (∨) for combining annotations: ∪
Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a **new domain** you need to specify:

- the **representation** of the annotations: ["14:35", "14:57"]
- an **order** between the elements: \( \subseteq \)

*universal* (\( \top \)) and *empty* (\( \bot \)) annotations: \( \top = [-\infty, +\infty] \quad \bot = [] \)

operator (\( \land \)) for **conjunction** of annotations: \( \cap \)
operator (\( \lor \)) for **combining** annotations: \( \cup \)

["09:25", "11:49"] \( \lor \) ["10:35", "12:57"] = ["09:25", "12:57"]
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . ["09:25", "11:49"]
:tag4302 :locatedIn :room311 . ["10:35", "12:57"]

To define a **new domain** you need to specify:

- the *representation* of the annotations: ["14:35", "14:57"]
- an *order* between the elements: ⊆

*universal* (⊤) and *empty* (⊥) annotations:

\[ \top = [-\infty, +\infty] \quad \bot = [] \]

operator (\(\wedge\)) for **conjunction** of annotations: \(\cap\)

operator (\(\lor\)) for **combining** annotations: \(\cup\)

\[ ["09:25","11:49"] \lor ["14:35", "15:57"] = ["09:25","11:49"], ["14:35", "15:57"] \]
Annotation Domains

Temporal domain example:

:tag4302 :locatedIn :room311 . {"09:25", "11:49"]
:tag4302 :locatedIn :room311 . {"10:35", "12:57"]

To define a **new domain** you need to specify:

- the **representation** of the annotations: {"14:35", "14:57"]
- an **order** between the elements: \( \subseteq \)

*universal* (\( \top \)) and *empty* (\( \bot \)) annotations: \( \top = \{ -\infty, +\infty \} \) \( \bot = \{ \} \)

operator (\( \wedge \)) for **conjunction** of annotations: \( \cap \)
operator (\( \vee \)) for **combining** annotations: \( \cup \)

["09:25", "11:49"] \( \vee \) ["14:35", "15:57"] = \{ ["09:25", "11:49"] , ["14:35", "15:57"] \}
Other domains: Examples

Trust/Fuzzy

```prolog
:tag4302 :locatedIn :room311 . 0.9
:tag4302 :locatedIn :room310 . 0.2
```

Annotations: [0,1]
Order: \( \leq \)
\( \land \): min \( \lor \): max
\( \top \) = 1, \( \bot \) = 0
Other domains: Examples

Trust/Fuzzy

:tag4302 :locatedIn :room311 . 0.9
:tag4302 :locatedIn :room310 . 0.2

Provenance

foaf:Person subClassOf foaf:Agent .
{http://xmlns.com/foaf/0.1/}

annotations: [0,1]
order: ≤
∧: min  ∨: max
⊤ = 1,  ⊥ = 0

annotations: {set of URIs}
order: ⊆
∧: ∩  ∨: ∪
⊤ = {list of all URIs},
⊥ = {}
Other domains: Examples

Trust/Fuzzy

```
:tag4302 :locatedIn :room311 . 0.9
:tag4302 :locatedIn :room310 . 0.2
```

Provenance*

```
foaf:Person subClassOf foaf:Agent .
{http://xmlns.com/foaf/0.1/}
```

* this representation of provenance is a first draft

Annotations: \([0,1]\)

Order: \(\leq\)

\(\wedge: \text{min} \quad \vee: \text{max}\)

\(\top = 1, \quad \bot = 0\)

Annotations: \(\{\text{set of URIs}\}\)

Order: \(\subseteq\)

\(\wedge: \cap \quad \vee: \cup\)

\(\top = \{\text{list of all URIs}\},\quad \bot = \{\}\)
Other domains: Examples

Trust/Fuzzy

```sparql
:tag4302 :locatedIn :room311 . 0.9
:tag4302 :locatedIn :room310 . 0.2
```

Provenance*

```
foaf:Person subClassOf foaf:Agent .
{http://xmlns.com/foaf/0.1/}
```

* this representation of provenance is a first draft

Our generic semantics allows to combine domains:

```
:tag4302 :locatedIn :room311 . (["14:25", "14:57"], 0.8)
```
Integration with RDF

Transparent integration of annotated and classical RDF

```reasoning
:stefan foaf:name "Stefan Decker" .
:tag4302 :assignedTo :stefan .
:tag4302 :locatedIn :room311 .
["14:25", "14:57"]
```
Integration with RDF

Transparent integration of annotated and classical RDF

```
:stefan foaf:name "Stefan Decker" . \([-∞, +∞]\)
:tag4302 :assignedTo :stefan . \([-∞, +∞]\)
:tag4302 :locatedIn :room311 . \["14:25", "14:57"]
```

Possible approaches:

- use $\top$ as annotation
Integration with RDF

Transparent integration of annotated and classical RDF

```r
:stefan foaf:name "Stefan Decker" . [_:a, _:b]
:tag4302 :assignedTo :stefan . [_:a, _:b]
:tag4302 :locatedIn :room311 . ["14:25", "14:57"]
```

Possible approaches:

- use $\top$ as annotation
- triple is valid at a time interval common throughout the graph
  requires blank nodes in annotations
Integration with RDF

Transparent integration of annotated and classical RDF

```
:stefan foaf:name "Stefan Decker" . [−∞, now]
:tag4302 :assignedTo :stefan . [−∞, now]
:tag4302 :locatedIn :room311 . ["14:25", "14:57"]
```

Possible approaches:
- use ⊤ as annotation
- triple is valid at a time interval common throughout the graph
  requires blank nodes in annotations
- triple is valid until “now” ([Temporal RDF, Gutierrez et al, 2005])
  represents current time
Integration with RDF

Transparent integration of annotated and classical RDF

```html
:stefan foaf:name "Stefan Decker" . \([−∞, +∞]\)
:tag4302 :assignedTo :stefan . \([-∞, +∞]\)
:tag4302 :locatedIn :room311 . \("14:25", "14:57"\)
```

Possible approaches:

- use \(\top\) as annotation “upwards compatible”
- triple is valid at a time interval common throughout the graph requires blank nodes in annotations
- triple is valid until “now” ([Temporal RDF, Gutierrez et al, 2005]) represents current time
Inference rules are **independent** of the annotation domain
Inference rules are independent of the annotation domain.

RDFS “rdfs:domain” rule:

\[
?\text{SomeProp} \, \text{rdfs:domain} \, ?\text{SomeClass} \\
\text{x} \, ?\text{SomeProp} \, ?y \\
\Rightarrow \text{x} \, \text{rdf:type} \, ?\text{SomeClass}
\]
Inference rules are **independent** of the annotation domain

RDFS “rdfs:domain” rule:

\[
\text{?SomeProp} \ rdfs:domain \ ?SomeClass \\
\text{?x} \ ?SomeProp \ ?y \\
\Rightarrow \ ?x \ rdf:type \ ?SomeClass
\]

Example:

\[
\text{:worksFor} \ rdfs:domain \ :Employee \\
\text{:nuno} \ :worksFor \ :DERI \\
\Rightarrow \ \text{:nuno} \ rdf:type \ :Employee
\]
Annotated RDFS Inference rules

Inference rules are **independent** of the annotation domain

**Annotated RDFS “rdfs:domain” rule:**

\[
\begin{align*}
?\text{SomeProp} & \text{rdfs:domain} \ ?\text{SomeClass} \ ?v1 \\
?x \ ?\text{SomeProp} \ ?y & \ ?v2 \\
\Rightarrow \ ?x \ \text{rdf:type} \ ?\text{SomeClass} \ ?v1 \land ?v2
\end{align*}
\]

**Example:**

```
:worksFor rdfs:domain :Employee
:nuno :worksFor :DERI
\Rightarrow :nuno rdf:type :Employee
```
Inference rules are **independent** of the annotation domain

Annotated RDFS “rdfs:domain” rule:

\[
?\text{SomeProp} \text{ rdfs:domain } ?\text{SomeClass} \quad ?v1  
\text{x } ?\text{SomeProp } ?y \quad ?v2  
\Rightarrow \text{x } \text{rdf:type } ?\text{SomeClass} \quad ?v1 \land ?v2
\]

Example:

\[
\text{:worksFor rdfs:domain :Employee } \quad [-\infty, +\infty]  
\text{:nuno :worksFor :DERI } \quad ["2009-01-01", "2010-05-28"]  
\Rightarrow \text{:nuno rdf:type :Employee } \quad ["2009-01-01", "2010-05-28"]
\]
Inference rules are **independent** of the annotation domain

Annotated RDFS “rdfs:domain” rule:

\[
\text{?SomeProp rdfs:domain ?SomeClass } \quad ?v1 \\
?x \text{ ?SomeProp } ?y \quad ?v2 \\
\Rightarrow ?x \text{ rdf:type ?SomeClass } \quad ?v1 \land ?v2
\]

Example:

\[
{:\text{worksFor rdfs:domain :Employee } [-\infty, +\infty]} \\
{:\text{nuno :worksFor :DERI } ["2009-01-01", "2010-05-28"]} \\
\Rightarrow {:\text{nuno rdf:type :Employee } ["2009-01-01", "2010-05-28"]}
\]

- Extra rule to group annotations triples (\(\lor\)):

\[
{:\text{nuno :worksFor :DERI } ["2008-05-01", "2010-01-01"]} \\
{:\text{nuno :worksFor :DERI } ["2009-01-01", "2010-05-28"]}
\]
Inference rules are **independent** of the annotation domain.

Annotated RDFS “rdfs:domain” rule:

\[
\text{?SomeProp rdfs:domain ?SomeClass } \quad ?v1 \\
\text{?x ?SomeProp ?y } \quad ?v2 \\
\Rightarrow \text{?x rdf:type ?SomeClass } \quad ?v1 \land ?v2
\]

Example:

\[
\text{:worksFor rdfs:domain :Employee } \quad [-\infty, +\infty] \\
\text{:nuno :worksFor :DERI } \quad ["2009-01-01", "2010-05-28"] \\
\Rightarrow \text{:nuno rdf:type :Employee } \quad ["2009-01-01", "2010-05-28"]
\]

Extra rule to group annotations triples (∨):

\[
\text{:nuno :worksFor :DERI } \quad ["2008-05-01", "2010-01-01"] \\
\text{:nuno :worksFor :DERI } \quad ["2009-01-01", "2010-05-28"] \\
\Rightarrow \text{:nuno :worksFor :DERI } \quad ["2008-05-01", "2010-05-28"]
\]
Annotated RDFS Inference rules

Inference rules are **independent** of the annotation domain

**Annotated RDFS “rdfs:domain” rule:**

\[
\begin{align*}
?\text{SomeProp} & \text{rdfs:domain} ?\text{SomeClass} & ?v1 \\
?x & ?\text{SomeProp} ?y & ?v2 \\
\Rightarrow & ?x \text{ rdf:type } ?\text{SomeClass} & ?v1 \land ?v2
\end{align*}
\]

**Example:**

:worksFor rdfs:domain :Employee \([-\infty, +\infty]\]

**Extra rule to group annotations triples (\(\lor\)):**

:nuno :worksFor :DERI \["2008-05-01", "2010-01-01"]
:nuno :worksFor :DERI \["2009-01-01", "2010-05-28"]
\Rightarrow :nuno :worksFor :DERI \["2008-05-01", "2010-05-28"]
Prototype implementation that computes the closure of an annotated RDF graph

Modular system: can use different domains and rulesets

More info and downloads available at: http://sprinql.deri.ie
Sensor data
Each sensor record is composed of timestamp, ip, tag and ssi.

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>IP Address</th>
<th>Port</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010-05-28</td>
<td>14:57:51</td>
<td>10.254.2.15</td>
<td>4302</td>
<td>83</td>
</tr>
<tr>
<td>2010-05-28</td>
<td>14:57:51</td>
<td>10.254.3.1</td>
<td>4302</td>
<td>83</td>
</tr>
<tr>
<td>2010-05-28</td>
<td>14:57:51</td>
<td>10.254.2.6</td>
<td>4302</td>
<td>83</td>
</tr>
</tbody>
</table>
Sensor data

Time = 14:34

:tag4302 :locatedIn ":room311" (["13:58", "14:34"], 0.9)
Sensor data

Time = 14:40

:tag4302 :locatedIn ":room311" ("13:58", "14:34"],0.9)
:tag4302 :locatedIn ":room311, :room310" ("14:35", "14:52"],0.6)
Sensor data

Time = 14:56

:tag4302 :locatedIn ":room311" ("13:58","14:34"], 0.9)
:tag4302 :locatedIn ":room311, :room310" ("14:35","14:52"], 0.6)
:tag4302 :locatedIn ":room311, :room310" ("14:53","14:56"], 0.6)
:tag4145 :locatedIn ":room310" ("14:40","14:56"], 0.9)
Data processing

- Sensor data for 1 hour approx. 434,000 records.
Data processing

- Sensor data for 1 hour approx. 434,000 records.

1. Group all the ips (with the lowest ssi) for a given timestamp and tag;
Data processing

- Sensor data for 1 hour approx. 434,000 records.

1. Group all the ips (with the lowest ssi) for a given timestamp and tag;

2. Merge all records that have consecutive timestamp and equal tag and ip into a single interval;
Data processing

- Sensor data for 1 hour approx. 434,000 records.

1. Group all the ips (with the lowest ssi) for a given timestamp and tag;

2. Merge all records that have consecutive timestamp and equal tag and ip into a single interval;

3. Compute the trust value of each merged record based on the average of the ssi;
Data processing

- Sensor data for 1 hour approx. 434,000 records.

1. Group all the ips (with the lowest ssi) for a given timestamp and tag;

2. Merge all records that have consecutive timestamp and equal tag and ip into a single interval;

3. Compute the trust value of each merged record based on the average of the ssi;

4. Based on the trust, we can discard the triples below a certain threshold:
   - threshold of 0.1 results in approx. 70,000 triples
   - threshold of 0.4 results in approx. 53,000 triples
Extend SPARQL to allow querying annotated RDF

- “Annotation aware” SPARQL

*SPaRql Inspired aNnotation Query Language*
Annotated SPARQL : SPRINQL*

Extend SPARQL to allow querying annotated RDF

- “Annotation aware” SPARQL

  “Where was Stefan before this institute meeting?”

```
SELECT ?Room WHERE {
  ?Tag :assignedTo :stefan ;
  :locatedIn ?Room . ["14:30", "15:00"]
}
```

* SPaRql Inspired aNnotation Query Language
Extend SPARQL to allow querying annotated RDF

- “Annotation aware” SPARQL

  “Where was Stefan before this institute meeting?”

```
SELECT ?Room WHERE {
  ?Tag :assignedTo :stefan ;
  :locatedIn ?Room . ["14:30", "15:00"]
}
```

- Evaluation based on an extension of the SPARQL relational algebra to support annotations

* SPaRql Inspired aNnotation Query Language
“When were Stefan and Manfred in the same room?”

```
SELECT ?Room ?TimeInterval WHERE {
  ?Tag1 :assignedTo :stefan ;
  :locatedIn ?Room . ?TimeInterval
  ?Tag2 :assignedTo :manfred ;
  :locatedIn ?Room . ?TimeInterval
}
```
“When were Stefan and Manfred in the same room?”

SELECT ?Room ?TimeInterval WHERE {
    ?Tag1 :assignedTo :stefan ;
    :locatedIn ?Room . ?TimeInterval
    ?Tag2 :assignedTo :manfred ;
    :locatedIn ?Room . ?TimeInterval
}

Answers:

(?
Room, ?TimeInterval) = (:room311, {["09:13", "10:35"],
        ["11:23", "12:47"]})

(?
Conclusions

- Concise way to attach information to triples
- Inference support over the annotations

Future work:
- Working on Annotated SPARQL
- Define other annotation domains (spatial, ...)
- Annotated DBPedia
Conclusions

- Concise way to attach information to triples
- Inference support over the annotations

Future work:
- Working on Annotated SPARQL
- Define other annotation domains (spatial, ...)
- Annotated DBPedia

- “A general framework for representing and reasoning with annotated semantic web data”. AAAI 2010
- Position paper accepted for presentation at W3C RDF Next Steps Workshop
Conclusions

- Concise way to attach information to triples
- Inference support over the annotations

Future work:
- Working on Annotated SPARQL
- Define other annotation domains (spatial, ...)
- Annotated DBPedia

“A general framework for representing and reasoning with annotated semantic web data”. AAAI 2010

- Position paper accepted for presentation at W3C RDF Next Steps Workshop

Other possible uses?