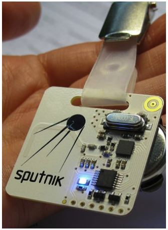
AnQL: SPARQLing Up Annotated RDFS

Nuno Lopes Axel Polleres Umberto Straccia
Antoine Zimmermann

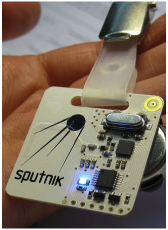
November 09, 2010

Usecase: Exposing Sensor data as RDF



sensor tags are assigned to people

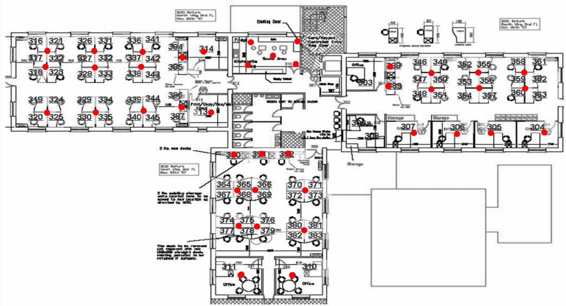
Usecase: Exposing Sensor data as RDF



sensor tags are assigned to people

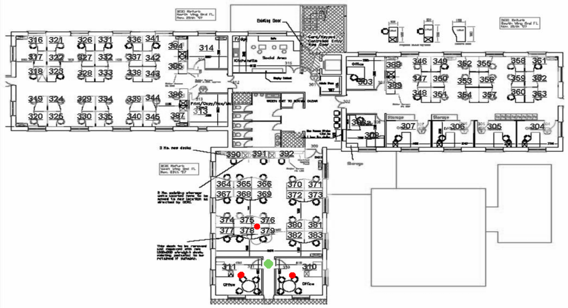
tag proximity is registered by *base stations*

Usecase: Exposing Sensor data as RDF



base stations are deployed throughout a building

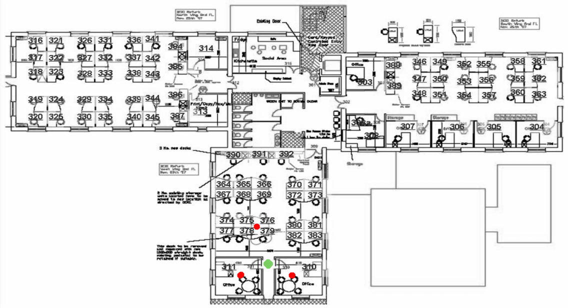
Usecase: Exposing Sensor data as RDF



sensor data readings:

timestamp	ip	tag	ssi
2010-11-09 14:57:51	10.254.2.15	4302	83
2010-11-09 14:57:51	10.254.3.1	4302	83
2010-11-09 14:57:51	10.254.2.6	4302	83

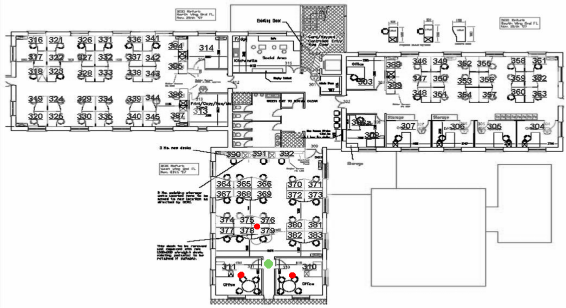
Usecase: Exposing Sensor data as RDF



sensor data readings:

timestamp	ip	tag	ssi
2010-11-09 14:57:51	10.254.2.15	4302	83
2010-11-09 14:57:51	10.254.3.1	4302	83
2010-11-09 14:57:51	10.254.2.6	4302	83

Usecase: Exposing Sensor data as RDF



sensor data readings:

timestamp	ip	tag	ssi
2010-11-09 14:57:51	10.254.2.15	4302	83
2010-11-09 14:57:51	10.254.3.1	4302	83
2010-11-09 14:57:51	10.254.2.6	4302	83

How to represent sensor data as RDF?

- RDF triples

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

How to represent sensor data as RDF?

- RDF triples

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

How to represent sensor data as RDF?

- RDF triples

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

How to represent sensor data as RDF?

- RDF triples

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

Not enough info!

How to 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;
         :timestamp "2010-11-09 14:57:51" .
```

How to 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 ;
         :timestamp "2010-11-09 14:57:51" .
```

How to 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 ;
         :timestamp "2010-11-09 14:57:51" .
```

No defined semantics!

Use Annotated RDF(S)!

Annotations refer to a specific **domain**

Use Annotated RDF(S)!

Annotations refer to a specific **domain**

Temporal:

```
:tag4302 :locatedIn :room311 . "2010-11-09 14:57:51"
```

Use Annotated RDF(S)!

Annotations refer to a specific **domain**

Temporal:

```
:tag4302 :locatedIn :room311 . "2010-11-09 14:57:51"
```

Fuzzy:

```
:tag4302 :locatedIn :room311 . "0.9"
```

Use Annotated RDF(S)!

Annotations refer to a specific **domain**

Temporal:

```
:tag4302 :locatedIn :room311 . "2010-11-09 14:57:51"
```

Fuzzy:

```
:tag4302 :locatedIn :room311 . "0.9"
```

Queries:

"When were two people in the same room?"

"Who is closer to room 311?"

Annotated RDF(S) (Straccia et al. [2010])

- Based on previous work on Annotated RDF (Udrea et al. [2010])
- Encompasses other proposals for domain-specific RDF: Temporal, Fuzzy, Trust, Provenance, ...
- Deductive system as extension of RDFS

Overview

Annotated RDF(S) (Straccia et al. [2010])

- Based on previous work on Annotated RDF (Udrea et al. [2010])
- Encompasses other proposals for domain-specific RDF: Temporal, Fuzzy, Trust, Provenance, . . .
- Deductive system as extension of RDFS

AnQL: Annotated SPARQL

- Annotation-aware SPARQL
- Extension of the semantics presented in Pérez et al. [2009]
- Includes features from SPARQL 1.1
  - subqueries, aggregates and assignment

Annotated RDF(S)

Annotation Domain Example

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 Domain Example

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: `["09:25", "11:49"]`

Annotation Domain Example

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: `["09:25", "11:49"]`
- an *order* between the elements:

Annotation Domain Example

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: `["09:25", "11:49"]`
- an *order* between the elements:

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

Annotation Domain Example

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: `["09:25", "11:49"]`
- an *order* between the elements:

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

Annotation Domain Example

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: `["09:25", "11:49"]`
- an *order* between the elements:

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

Annotation Domain Example

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: `["09:25", "11:49"]`
- an *order* between the elements:

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

  operator ($\otimes$) for conjunction of annotations: $\cap$

`["09:25","11:49"]` $\otimes$ `["10:35", "12:57"]` = `["10:35", "11:49"]`

Annotation Domain Example

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: `["09:25", "11:49"]`
- an *order* between the elements:

  universal ($\top$) and *empty* ($\bot$) annotations: $\top = [-\infty, +\infty]$ $\bot = []$
  operator ($\otimes$) for conjunction of annotations: $\cap$
  operator ($\oplus$) for combining annotations

Annotation Domain Example

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: `["09:25", "11:49"]`
- an *order* between the elements:

  universal ($\top$) and *empty* ($\bot$) annotations: $\top = [-\infty, +\infty]$ $\bot = []$
  operator ($\otimes$) for conjunction of annotations: $\cap$
  operator ($\oplus$) for combining annotations: $\cup$



`["09:25","11:49"]` $\oplus$ `["10:35", "12:57"]` = `["09:25", "12:57"]`

Annotation Domain Example

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: `["09:25", "11:49"]`
- an *order* between the elements:

  universal ($\top$) and *empty* ($\bot$) annotations: $\top = [-\infty, +\infty]$ $\bot = []$
  operator ($\otimes$) for conjunction of annotations: $\cap$
  operator ($\oplus$) for combining annotations: $\cup$

["09:25","11:49"] $\oplus$ ["14:35", "15:57"] = ["09:25","11:49"], ["14:35", "15:57"]

Annotation Domain Example

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: {["09:25", "11:49"]}
- an *order* between the elements:

  universal ($\top$) and *empty* ($\bot$) annotations: $\top = \{[-\infty, +\infty]\}$ $\bot = \{[]\}$
  operator ($\otimes$) for conjunction of annotations: $\cap$
  operator ($\oplus$) for combining annotations: $\cup$

["09:25","11:49"] $\oplus$ ["14:35", "15:57"] = {["09:25","11:49"], ["14:35", "15:57"]}

Annotation Domain

Consider a non-empty set L: annotation values

Annotation Domain

Consider a non-empty set L: annotation values

An annotation domain is an idempotent, commutative semi-ring

$$D = \langle L, \oplus, \otimes, \perp, \top \rangle$$

where $\oplus$ is $\top$-annihilating (Buneman and Kostylev [2010]).

Annotation Domain

Consider a non-empty set L: annotation values

An annotation domain is an idempotent, commutative semi-ring

$$D = \langle L, \oplus, \otimes, \bot, \top \rangle$$

where $\oplus$ is $\top$-annihilating (Buneman and Kostylev [2010]).

Any idempotent semi-ring defines a partial order $\preceq$ over L as:

$$\lambda_1 \preceq \lambda_2 \text{ iff } \lambda_1 \oplus \lambda_2 = \lambda_2$$

Annotation Domain

Consider a non-empty set L: annotation values

An annotation domain is an idempotent, commutative semi-ring

$$D = \langle L, \oplus, \otimes, \bot, \top \rangle$$

where $\oplus$ is $\top$-annihilating (Buneman and Kostylev [2010]).

Any idempotent semi-ring defines a partial order $\preceq$ over L as:

$$\lambda_1 \preceq \lambda_2 \quad \text{iff} \quad \lambda_1 \oplus \lambda_2 = \lambda_2$$

For $\lambda, \lambda_i \in L$

1. $\oplus$ and $\otimes$ are commutative and associative;
2. $\oplus$ is idempotent;
3. $\bot \oplus \lambda = \lambda$, $\top \oplus \lambda = \top$, $\top \otimes \lambda = \lambda$ and $\bot \otimes \lambda = \bot$;
4. $\otimes$ is distributive over $\oplus$, i.e. $\lambda_1 \otimes (\lambda_2 \oplus \lambda_3) = (\lambda_1 \otimes \lambda_2) \oplus (\lambda_1 \otimes \lambda_3)$;

Annotated RDF(S)

Consider the alphabets **U** (*URI references*), **B** (*blank nodes or variables*) and **L** (*Literals*).

Annotated triple and graph

An "extended" RDF triple is $\tau = (s, p, o) \in \mathbf{UBL} \times \mathbf{U} \times \mathbf{UBL}$.

Annotated RDF(S)

Consider the alphabets **U** (*URI references*), **B** (*blank nodes or variables*) and **L** (*Literals*).

Annotated triple and graph

An "extended" RDF triple is $\tau = (s, p, o) \in \mathbf{UBL} \times \mathbf{U} \times \mathbf{UBL}$.
An annotated triple is $\tau : \lambda$, τ a triple, $\lambda \in L$ an *annotation value*.

Annotated RDF(S)

Consider the alphabets **U** (*URI references*), **B** (*blank nodes or variables*) and **L** (*Literals*).

Annotated triple and graph

An "extended" RDF triple is $\tau = (s, p, o) \in \mathbf{UBL} \times \mathbf{U} \times \mathbf{UBL}$.

An annotated triple is $\tau : \lambda$, τ a triple, $\lambda \in L$ an *annotation value*.

An annotated graph G is a finite set of *annotated triples*.

Annotated RDF(S)

Consider the alphabets **U** (*URI references*), **B** (*blank nodes or variables*) and **L** (*Literals*).

Annotated triple and graph

An "extended" RDF triple is $\tau = (s, p, o) \in \mathbf{UBL} \times \mathbf{U} \times \mathbf{UBL}$.
An annotated triple is $\tau : \lambda$, τ a triple, $\lambda \in L$ an *annotation value*.
An annotated graph G is a finite set of *annotated triples*.

ρdf vocabulary

(p, sp, q)	property p is a subproperty of property q
(c, sc, d)	class c is a subclass of class d
(a, type, b)	a is of type b
(p, dom, c)	the domain of property p is c
(p, range, c)	the range of property p is c

Annotated RDF(S) Semantics

Interpretation

An interpretation $\mathcal{I}$ assigns to a triple τ an element $\lambda \in L$

Annotated RDF(S) Semantics

Interpretation

An interpretation $\mathcal{I}$ assigns to a triple τ an element $\lambda \in L$

Models

An interpretation $\mathcal{I}$ is a model of G if it assigns to the triples of G a value that is greater or equal (i.e., $\succeq$) to the annotation and satisfies the schema axioms constraints (sp, sc, type, dom, range).

Annotated RDF(S) Semantics

Interpretation

An interpretation $\mathcal{I}$ assigns to a triple τ an element $\lambda \in L$

Models

An interpretation $\mathcal{I}$ is a model of G if it assigns to the triples of G a value that is greater or equal (i.e., $\succeq$) to the annotation and satisfies the schema axioms constraints (sp, sc, type, dom, range).

Graph entailment

G entails H under ρdf ($G \models H$) iff every model under ρdf of G is also a model under ρdf of H.

Annotated RDF(S) Inference example

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

Annotated RDF(S) Inference example

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

RDFS "sp" rule:

```
   ?Prop1 sp ?Prop2 .
   ?x ?Prop1 ?y .
⇒ ?x ?Prop2 ?y .
```

```
   :locatedIn sp :basedNear .
   :tag4302 :locatedIn :room311 .
⇒ :tag4302 :basedNear :room311 .
```

Annotated RDF(S) Inference example

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

Annotated RDFS "sp" rule:

```
  ?Prop1 sp ?Prop2 .    ?v1
  ?x ?Prop1 ?y .        ?v2
⇒ ?x ?Prop2 ?y .        ?v1 ⊗ ?v2
```

```
  :locatedIn sp :basedNear .          [−∞, +∞]
  :tag4302 :locatedIn :room311 .      ["14:25", "14:57"]
⇒ :tag4302 :basedNear :room311 .      ["14:25", "14:57"]
```

Annotated RDF(S) Inference example

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

Annotated RDFS "sp" rule:

	?Prop1 sp ?Prop2 .	$?v1$
	?x ?Prop1 ?y .	$?v2$
$\Rightarrow$	?x ?Prop2 ?y .	$?v1 \otimes ?v2$

	:locatedIn sp :basedNear .	$[-\infty, +\infty]$
	:tag4302 :locatedIn :room311 .	["14:25", "14:57"]
$\Rightarrow$	:tag4302 :basedNear :room311 .	["14:25", "14:57"]

Extra rule to group annotations triples ($\oplus$):

:tag4302 :locatedIn :room311 .	["14:25", "14:57"]
:tag4302 :locatedIn :room311 .	["14:43", "15:20"]

Annotated RDF(S) Inference example

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

Annotated RDFS "sp" rule:

```
   ?Prop1 sp ?Prop2 .   ?v1
   ?x ?Prop1 ?y .       ?v2
⇒  ?x ?Prop2 ?y .       ?v1 ⊗ ?v2
```

```
   :locatedIn sp :basedNear .         [−∞, +∞]
   :tag4302 :locatedIn :room311 .     ["14:25", "14:57"]
⇒  :tag4302 :basedNear :room311 .     ["14:25", "14:57"]
```

Extra rule to group annotations triples (⊕):

```
   :tag4302 :locatedIn :room311 .     ["14:25", "14:57"]
   :tag4302 :locatedIn :room311 .     ["14:43", "15:20"]
⇒  :tag4302 :locatedIn :room311 .     ["14:25", "15:20"]
```

Annotated RDF(S) Inference example

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

Annotated RDFS "sp" rule:

```
   ?Prop1 sp ?Prop2 .   ?v1
   ?x ?Prop1 ?y .       ?v2
⇒  ?x ?Prop2 ?y .       ?v1 ⊗ ?v2
```

```
   :locatedIn sp :basedNear .           [−∞, +∞]
   :tag4302 :locatedIn :room311 .       ["14:25", "14:57"]
⇒  :tag4302 :basedNear :room311 .       ["14:25", "14:57"]
```

Extra rule to group annotations triples (⊕):

```
⇒  :tag4302 :locatedIn :room311 .       ["14:25", "15:20"]
```

Integration with RDF

Transparent integration of annotated and classical RDF

```
:nuno foaf:name "Nuno Lopes" .
:tag4302 :assignedTo :nuno .
:tag4302 :locatedIn  :room311 .      ["14:25", "14:57"]
```

Integration with RDF

Transparent integration of annotated and classical RDF

```
:nuno foaf:name "Nuno Lopes" .        [−∞, +∞]
:tag4302 :assignedTo :nuno .          [−∞, +∞]
:tag4302 :locatedIn  :room311 .       ["14:25", "14:57"]
```

Possible approaches:

- use $\top$ as annotation

Integration with RDF

Transparent integration of annotated and classical RDF

```
:nuno foaf:name "Nuno Lopes" .        [_:a, _:b]
:tag4302 :assignedTo :nuno .          [_: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

```
:nuno foaf:name "Nuno Lopes" .        [−∞, now]
:tag4302 :assignedTo :nuno .          [−∞, now]
: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
- triple is valid until "now" (Gutiérrez et al. [2005])
  - represents current time

Integration with RDF

Transparent integration of annotated and classical RDF

`:nuno foaf:name "Nuno Lopes" .`	$[-\infty, +\infty]$
`:tag4302 :assignedTo :nuno .`	$[-\infty, +\infty]$
`:tag4302 :locatedIn :room311 .`	`["14:25", "14:57"]`

Possible approaches:

- use $\top$ as annotation "compatible with classical RDF"
- triple is valid at a time interval common throughout the graph
  - requires blank nodes in annotations
- triple is valid until "now" (Gutiérrez et al. [2005])
  - represents current time

AnQL: Annotated SPARQL

AnQL: Annotated SPARQL

Consider the alphabets **U**, **B**, **L** as before

SPARQL

- $\tau = (s, p, o)$ where $s, o \in$ **UBL** and $p \in$ **UB** is a triple pattern.

triple pattern example

```
?tag :assignedTo :nuno .
```

AnQL: Annotated SPARQL

Consider the alphabets **U**, **B**, **L** as before

Annotated SPARQL

- $\tau = (s, p, o)$ where $s, o \in \mathbf{UBL}$ and $p \in \mathbf{UB}$ is a triple pattern.
- $\tau : \lambda$ is an annotated triple pattern if τ is a triple pattern and $\lambda \in L$ (annotation term)

annotated triple pattern example

?tag :assignedTo :nuno . $[-\infty, +\infty]$

AnQL: Annotated SPARQL

Consider the alphabets **U**, **B**, **L** as before, and **V** as the alphabet for *Annotation variables*.

Annotated SPARQL

- $\tau = (s, p, o)$ where $s, o \in$ **UBL** and $p \in$ **UB** is a triple pattern.
- $\tau : \lambda$ is an annotated triple pattern if τ is a triple pattern and $\lambda \in L$ (annotation term) or $\lambda \in$ **V** (annotation variable)

annotated triple pattern example

```
?tag :locatedIn ?room . ?l
```

AnQL: Annotated SPARQL

Consider the alphabets **U**, **B**, **L** as before, and **V** as the alphabet for *Annotation variables*.

Annotated SPARQL

- $\tau = (s, p, o)$ where $s, o \in \mathbf{UBL}$ and $p \in \mathbf{UB}$ is a triple pattern.
- $\tau : \lambda$ is an annotated triple pattern if τ is a triple pattern and $\lambda \in L$ (annotation term) or $\lambda \in \mathbf{V}$ (annotation variable)
- Basic Annotated Patterns (BAP) are sets of annotated triple patterns

BAP example

```
{ ?tag :assignedTo :nuno . [−∞, +∞]
  ?tag :locatedIn ?room . ?l }
```

AnQL definitions

An annotated graph pattern (AGP) is:

1. any BAP
2. P, P' are annotated graph patterns, R a filter expression:
   - (P AND P')
   - (P OPTIONAL P')
   - (P UNION P')
   - (P FILTER R)

AnQL definitions

An annotated graph pattern (AGP) is:

1. any BAP
2. P, P' are annotated graph patterns, R a filter expression:
   - $(P$ AND $P')$
   - $(P$ OPTIONAL $P')$
   - $(P$ UNION $P')$
   - $(P$ FILTER $R)$

Filter expressions

- any SPARQL filter $(=, \vee, \wedge, \mathrm{isBOUND}, \ldots)$
- Domain $\preceq$ for comparing annotations
- Domain specific built-in functions

Compatible substitutions

SPARQL

```
SELECT *
WHERE { ?tag :assignedTo :nuno .
        ?tag :locatedIn ?room . }
```

Substitutions

$$\begin{aligned}
\theta_1 &= \{?tag \to \text{:tag4302}\} \\
\theta_2 &= \{?tag \to \text{:tag4304}\} \\
\mu_1 &= \{?tag \to \text{:tag4302},\ ?room \to \text{:room311}\} \\
\mu_2 &= \{?tag \to \text{:tag4302},\ ?room \to \text{:room312}\}
\end{aligned}$$

Compatible substitutions

SPARQL

```
SELECT *
WHERE { ?tag :assignedTo :nuno .
        ?tag :locatedIn ?room . }
```

Substitutions

$$\begin{aligned}
\theta_1 &= \{?tag \to :\mathrm{tag4302}\}\\
\theta_2 &= \{?tag \to :\mathrm{tag4304}\}\\
\mu_1 &= \{?tag \to :\mathrm{tag4302},\ ?room \to :\mathrm{room311}\}\\
\mu_2 &= \{?tag \to :\mathrm{tag4302},\ ?room \to :\mathrm{room312}\}
\end{aligned}$$

Union of compatible substitutions

$$\begin{aligned}
\theta_1 \cup \mu_1 &= \{?tag \to :\mathrm{tag4302},\ ?room \to :\mathrm{room311}\}\\
\theta_1 \cup \mu_2 &= \{?tag \to :\mathrm{tag4302},\ ?room \to :\mathrm{room312}\}
\end{aligned}$$

Compatible substitutions

AnQL

```
SELECT *
WHERE { ?tag :assignedTo :nuno . ?l
        ?tag :locatedIn ?room . ?l }
```

Substitutions

$$\theta_1 = \{?tag \rightarrow :\mathrm{tag4302},\ ?l \rightarrow [\text{"13:00"}, \text{"15:00"}]\}$$
$$\theta_2 = \{?tag \rightarrow :\mathrm{tag4304},\ ?l \rightarrow [\text{"12:00"}, \text{"13:00"}]\}$$
$$\mu_1 = \{?tag \rightarrow :\mathrm{tag4302},\ ?room \rightarrow :\mathrm{room311},\ ?l \rightarrow [\text{"14:00"}, \text{"16:00"}]\}$$
$$\mu_2 = \{?tag \rightarrow :\mathrm{tag4302},\ ?room \rightarrow :\mathrm{room312},\ ?l \rightarrow [\text{"16:00"}, \text{"18:00"}]\}$$

Union of compatible substitutions

$$\theta_1 \cup \mu_1 = \{?tag \rightarrow :\mathrm{tag4302},\ ?room \rightarrow :\mathrm{room311},\ ?l \rightarrow [\text{"14:00"}, \text{"15:00"}]\}$$

AnQL query evaluation

- for a BAP P a substitution is a mapping $\theta : var(P) \rightarrow term(G)$.
- $\theta(P)$ represents the triples obtained by replacing the variables in P according to θ.
- $G \models \theta(P)$ denotes $\theta(P)$ is entailed by G.
- $[\![P]\!]_G = \{\theta \mid dom(\theta) = var(P) \text{ and } G \models \theta(P)\}$

AnQL query evaluation

- for a BAP P a substitution is a mapping $\theta : var(P) \to term(G)$.
- $\theta(P)$ represents the triples obtained by replacing the variables in P according to θ.
- $G \models \theta(P)$ denotes $\theta(P)$ is entailed by G.
- $[\![P]\!]_G = \{\theta \mid dom(\theta) = var(P) \text{ and } G \models \theta(P)\}$

Extension of the semantics presented in Pérez et al. [2009]

compatible substitutions (extension)

- Two substitutions θ_1, θ_2 are compatible if the value for all shared annotation variables v is not "disjoint": $\theta_1(v) \otimes \theta_2(v) \neq \bot$
- The union of compatible substitutions θ_1, θ_2, the value of a shared annotation variable v is: $\theta_1(v) \otimes \theta_2(v)$

OPTIONALs

"When was :nuno located in room 311 optionally with another person."

```
SELECT   ?l ?person
WHERE    { ?tag1 :assignedTo :nuno .
           ?tag1 :locatedIn :room311 . ?l
           OPTIONAL { ?tag2 :assignedTo ?person .
                      ?tag2 :locatedIn :room311 . ?l } }
```

Sample input:

:tag4302	:assignedTo	:nuno .	
:tag4302	:locatedIn	:room311 .	["13:48","14:34"]
:tag4304	:assignedTo	:axel	
:tag4304	:locatedIn	:room311 .	["14:26","15:17"]
:tag4301	:assignedTo	:antoine	
:tag4301	:locatedIn	:room311 .	["13:31","13:53"]

OPTIONALs

"When was :nuno located in room 311 optionally with another person."

```
SELECT   ?l ?person
WHERE    { ?tag1 :assignedTo :nuno .
           ?tag1 :locatedIn :room311 . ?l
           OPTIONAL { ?tag2 :assignedTo ?person .
                      ?tag2 :locatedIn :room311 . ?l } }
```

Sample input:

:tag4302	:assignedTo	:nuno .	
:tag4302	:locatedIn	:room311 .	["13:48", "14:34"]
:tag4304	:assignedTo	:axel	
:tag4304	:locatedIn	:room311 .	["14:26", "15:17"]
:tag4301	:assignedTo	:antoine	
:tag4301	:locatedIn	:room311 .	["13:31", "13:53"]

Answers:

$$\theta_1 = \{?l \rightarrow ["13{:}48", "14{:}34"]\}$$

OPTIONALs

"When was :nuno located in room 311 optionally with another person."

```
SELECT   ?l ?person
WHERE    { ?tag1 :assignedTo :nuno .
           ?tag1 :locatedIn :room311 . ?l
           OPTIONAL { ?tag2 :assignedTo ?person .
                      ?tag2 :locatedIn :room311 . ?l } }
```

Sample input:

:tag4302	:assignedTo	:nuno .	
:tag4302	:locatedIn	:room311 .	["13:48" , "14:34"]
:tag4304	:assignedTo	:axel	
:tag4304	:locatedIn	:room311 .	["14:26" , "15:17"]
:tag4301	:assignedTo	:antoine	
:tag4301	:locatedIn	:room311 .	["13:31" , "13:53"]

Answers:

$$\theta_1 = \{?l \rightarrow [\text{"13:48"}, \text{"14:34"}]\}$$
$$\theta_2 = \{?l \rightarrow [\text{"14:26"}, \text{"14:34"}],\ ?person \rightarrow :\mathrm{axel}\}$$

OPTIONALs

"When was :nuno located in room 311 optionally with another person."

```
SELECT   ?l ?person
WHERE    { ?tag1 :assignedTo :nuno .
           ?tag1 :locatedIn :room311 . ?l
           OPTIONAL { ?tag2 :assignedTo ?person .
                      ?tag2 :locatedIn :room311 . ?l } }
```

Sample input:

:tag4302	:assignedTo	:nuno .	
:tag4302	:locatedIn	:room311 .	["13:48", "14:34"]
:tag4304	:assignedTo	:axel	
:tag4304	:locatedIn	:room311 .	["14:26", "15:17"]
:tag4301	:assignedTo	:antoine	
:tag4301	:locatedIn	:room311 .	["13:31", "13:53"]

Answers:

$$\theta_1 = \{?l \to [\text{"13:48"}, \text{"14:34"}]\}$$
$$\theta_2 = \{?l \to [\text{"14:26"}, \text{"14:34"}],\ ?person \to \text{:axel}\}$$
$$\theta_3 = \{?l \to [\text{"13:48"}, \text{"13:53"}],\ ?person \to \text{:antoine}\}$$

OPTIONALs

"When was :nuno located in room 311 optionally with another person."

```
SELECT   ?l ?person
WHERE    { ?tag1 :assignedTo :nuno .
           ?tag1 :locatedIn :room311 . ?l
           OPTIONAL { ?tag2 :assignedTo ?person .
                      ?tag2 :locatedIn :room311 . ?l } }
```

Sample input:

:tag4302	:assignedTo	:nuno .	
:tag4302	:locatedIn	:room311 .	["13:48","14:34"]
:tag4304	:assignedTo	:axel	
:tag4304	:locatedIn	:room311 .	["14:26","15:17"]
:tag4301	:assignedTo	:antoine	
:tag4301	:locatedIn	:room311 .	["13:31","13:53"]

Answers:

$$\theta_1 = \{?l \rightarrow [\text{"13:48"},\text{"14:34"}]\}$$
$$\theta_2 = \{?l \rightarrow [\text{"14:26"},\text{"14:34"}],\ ?person \rightarrow \text{:axel}\}$$
$$\theta_3 = \{?l \rightarrow [\text{"13:48"},\text{"13:53"}],\ ?person \rightarrow \text{:antoine}\}$$

OPTIONAL provide more information maybe restricting annotation values

Further extensions

Features under discussion in SPARQL 1.1

Further extensions

Features under discussion in SPARQL 1.1

"During how long was a tag located in a room?"

Variable assignment & domain built-in functions

```
SELECT ?tag ?room ?dur
WHERE { ?tag :locatedIn ?room . ?l
        ASSIGN length(?l) AS ?dur }
```

Further extensions

Features under discussion in SPARQL 1.1

"During how long was a tag located in a room?"

Variable assignment & domain built-in functions

```
SELECT ?tag ?room ?dur
WHERE { ?tag :locatedIn ?room . ?l
        ASSIGN length(?l) AS ?dur }
```

"What was the average length a tag was located in a room?"

Aggregators

```
SELECT ?tag ?avgL
WHERE { ?tag :locatedIn ?room . ?l
        GROUPBY(?tag) AVG(length(?l)) AS ?avgL }
```

Uniform Evaluation of Queries

Uniform Evaluation of annotated and classical triple patterns

```
SELECT   ?l ?person
WHERE    { ?tag1 :assignedTo :nuno .
           ?tag1 :locatedIn :room311 . ?l
           OPTIONAL { ?tag2 :assignedTo ?person .
                      ?tag2 :locatedIn :room311 . ?l } }
```

Uniform Evaluation of Queries

Uniform Evaluation of annotated and classical triple patterns

```
SELECT   ?l ?person
WHERE    { ?tag1 :assignedTo :nuno . ?g1
           ?tag1 :locatedIn :room311 . ?l
           OPTIONAL { ?tag2 :assignedTo ?person . ?g1
                      ?tag2 :locatedIn :room311 . ?l } }
```

Possible approaches:

1. adding the same annotation variable for each non-annotated triple

Uniform Evaluation of Queries

Uniform Evaluation of annotated and classical triple patterns

```
SELECT   ?l ?person
WHERE    { ?tag1 :assignedTo :nuno . ?g1
           ?tag1 :locatedIn :room311 . ?l
           OPTIONAL { ?tag2 :assignedTo ?person . ?g2
                      ?tag2 :locatedIn :room311 . ?l } }
```

Possible approaches:

1. adding the same annotation variable for each non-annotated triple
2. adding a different annotation variable for each non-annotated triple

Uniform Evaluation of Queries

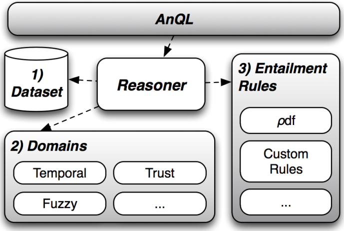
Uniform Evaluation of annotated and classical triple patterns

```
SELECT   ?l ?person
WHERE    { ?tag1 :assignedTo :nuno . [−∞, +∞]
           ?tag1 :locatedIn :room311 . ?l
           OPTIONAL { ?tag2 :assignedTo ?person . [−∞, +∞]
                      ?tag2 :locatedIn :room311 . ?l } }
```

Possible approaches:

1. adding the same annotation variable for each non-annotated triple
2. adding a different annotation variable for each non-annotated triple
3. adding the $\top$ element from the domain

Implementation



- Prototype implementation of Annotated RDF(S) and AnQL
- Based on SWI-Prolog's semweb library
- Modular system: can use different domains and rulesets

More info and downloads available at:
http://anql.deri.org

Conclusions

Presented

- General framework for annotating RDF triples.
- Deductive system as extension of RDFS
- SPARQL extension for Annotated RDF(S)
- Includes the most salient SPARQL 1.1 features

Conclusions

Presented

- General framework for annotating RDF triples.
- Deductive system as extension of RDFS
- SPARQL extension for Annotated RDF(S)
- Includes the most salient SPARQL 1.1 features

Future work

- Refine combination of domains
- Uniform evaluation of queries
- Define interchangeable format for representing annotations

Conclusions

Presented

- General framework for annotating RDF triples.
- Deductive system as extension of RDFS
- SPARQL extension for Annotated RDF(S)
- Includes the most salient SPARQL 1.1 features

Future work

- Refine combination of domains
- Uniform evaluation of queries
- Define interchangeable format for representing annotations

Thank you! Questions?

Extra Slides

AnQL query evaluation

Extension of the semantics presented in Pérez et al. [2009]

AnQL query evaluation

Extension of the semantics presented in Pérez et al. [2009]

- for a BAP P a substitution is a mapping $\theta : var(P) \rightarrow term(G)$.

  $\theta(P)$ represents the triples obtained by replacing the variables in P according to θ. $dom(\theta)$ are the variables for which θ is defined.

AnQL query evaluation

Extension of the semantics presented in Pérez et al. [2009]

- for a BAP P a substitution is a mapping $\theta : var(P) \to term(G)$.

  $\theta(P)$ represents the triples obtained by replacing the variables in P according to θ. $dom(\theta)$ are the variables for which θ is defined.

- $G \models \theta(P)$ denotes $\theta(P)$ is entailed by G.

AnQL query evaluation

Extension of the semantics presented in Pérez et al. [2009]

- for a BAP P a substitution is a mapping $\theta : var(P) \rightarrow term(G)$.

  $\theta(P)$ represents the triples obtained by replacing the variables in P according to θ. $dom(\theta)$ are the variables for which θ is defined.
- $G \models \theta(P)$ denotes $\theta(P)$ is entailed by G.
- two substitutions θ_1 and θ_2 are $\otimes$-*compatible* iff:
  1. The mappings agree on all non-annotated shared variables: $\theta_1(x) = \theta_2(x), \qquad x \text{ non-annot var} \in dom(\theta_1) \cap dom(\theta_2);$

AnQL query evaluation

Extension of the semantics presented in Pérez et al. [2009]

- for a BAP P a substitution is a mapping $\theta : \mathit{var}(P) \to \mathit{term}(G)$.

  $\theta(P)$ represents the triples obtained by replacing the variables in P according to θ. $\mathit{dom}(\theta)$ are the variables for which θ is defined.
- $G \models \theta(P)$ denotes $\theta(P)$ is entailed by G.
- two substitutions θ_1 and θ_2 are $\otimes$-*compatible* iff:
  1. The mappings agree on all non-annotated shared variables: $\theta_1(x) = \theta_2(x), \qquad x \text{ non-annot var} \in \mathit{dom}(\theta_1) \cap \mathit{dom}(\theta_2)$;
  2. All the shared annotation variables must not be "disjoint": $\theta_1(\lambda) \otimes \theta_2(\lambda) \neq \bot, \quad \lambda \text{ annot var} \in \mathit{dom}(\theta_1) \cap \mathit{dom}(\theta_2)$.

AnQL query evaluation

Extension of the semantics presented in Pérez et al. [2009]

- for a BAP P a substitution is a mapping $\theta : var(P) \rightarrow term(G)$.

  $\theta(P)$ represents the triples obtained by replacing the variables in P according to θ. $dom(\theta)$ are the variables for which θ is defined.
- $G \models \theta(P)$ denotes $\theta(P)$ is entailed by G.
- two substitutions θ_1 and θ_2 are $\otimes$-*compatible* iff:
  1. The mappings agree on all non-annotated shared variables: $\theta_1(x) = \theta_2(x), \quad x \text{ non-annot var} \in dom(\theta_1) \cap dom(\theta_2)$;
  2. All the shared annotation variables must not be "disjoint": $\theta_1(\lambda) \otimes \theta_2(\lambda) \neq \bot, \quad \lambda \text{ annot var} \in dom(\theta_1) \cap dom(\theta_2)$.
- θ_1, θ_2 $\otimes$-*compatible*, $\theta_1 \otimes \theta_2 = \theta_1 \cup \theta_2$, except any annotation variable $\lambda \in dom(\theta_1) \cap dom(\theta_2)$, $(\theta_1 \otimes \theta_2)(\lambda) = \theta_1(\lambda) \otimes \theta_2(\lambda)$.

AnQL query evaluation (cont.)

Let P be a BAP, P_i AGPs, G an annotated graph and R a filter expression:

- $[\![P]\!]_G = \{\theta \mid dom(\theta) = var(P) \text{ and } G \models \theta(P)\}$

Answers of a BAP P are the substitutions *entailed* by G

AnQL query evaluation (cont.)

Let P be a BAP, P_i AGPs, G an annotated graph and R a filter expression:

- $[\![P]\!]_G = \{\theta \mid dom(\theta) = var(P) \text{ and } G \models \theta(P)\}$
- $[\![P_1 \text{ AND } P_2]\!]_G = \{\theta_1 \otimes \theta_2 \mid \theta_1 \in [\![P_1]\!]_G, \theta_2 \in [\![P_2]\!]_G, \theta_1 \text{ and } \theta_2 \otimes\text{-compatible}\}$

Answers of two AGPs are the substitutions that are $\otimes$-*compatible*

AnQL query evaluation (cont.)

Let P be a BAP, P_i AGPs, G an annotated graph and R a filter expression:

- $[\![P]\!]_G = \{\theta \mid dom(\theta) = var(P) \text{ and } G \models \theta(P)\}$
- $[\![P_1 \text{ AND } P_2]\!]_G = \{\theta_1 \otimes \theta_2 \mid \theta_1 \in [\![P_1]\!]_G, \theta_2 \in [\![P_2]\!]_G, \theta_1 \text{ and } \theta_2 \otimes\text{-compatible}\}$
- $[\![P_1 \text{ UNION } P_2]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$

Answers for the UNION of two AGPs is the *union* of the substitutions

AnQL query evaluation (cont.)

Let P be a BAP, P_i AGPs, G an annotated graph and R a filter expression:

- $[\![P]\!]_G = \{\theta \mid dom(\theta) = var(P) \text{ and } G \models \theta(P)\}$
- $[\![P_1 \text{ AND } P_2]\!]_G = \{\theta_1 \otimes \theta_2 \mid \theta_1 \in [\![P_1]\!]_G, \theta_2 \in [\![P_2]\!]_G, \theta_1 \text{ and } \theta_2 \otimes\text{-compatible}\}$
- $[\![P_1 \text{ UNION } P_2]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$
- $[\![P_1 \text{ FILTER } R]\!]_G = \{\theta \mid \theta \in [\![P_1]\!]_G \text{ and } R\theta \text{ is true}\}$

$R\theta$ is *true* iff: ...

(9)	R	$=$	$(x \preceq y)$ with $x, y \in dom(\theta) \cup L$ and $\theta(x) \preceq \theta(y)$;
(10)	R	$=$	$p(\vec{z})$ with $p(\vec{z})\theta =$ true iff $p(\theta(\vec{z})) =$ true, p built-in predicate.

AnQL query evaluation (cont.)

Let P be a BAP, P_i AGPs, G an annotated graph and R a filter expression:

- $[\![P]\!]_G = \{\theta \mid dom(\theta) = var(P) \text{ and } G \models \theta(P)\}$
- $[\![P_1 \text{ AND } P_2]\!]_G = \{\theta_1 \otimes \theta_2 \mid \theta_1 \in [\![P_1]\!]_G, \theta_2 \in [\![P_2]\!]_G, \theta_1 \text{ and } \theta_2 \otimes\text{-compatible}\}$
- $[\![P_1 \text{ UNION } P_2]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$
- $[\![P_1 \text{ FILTER } R]\!]_G = \{\theta \mid \theta \in [\![P_1]\!]_G \text{ and } R\theta \text{ is true}\}$
- $[\![P_1 \text{ OPTIONAL } P_2[R]]\!]_G = \{\theta \mid$ and θ meets one of the following conditions:

$\}$

Answers for an OPTIONAL where the P_2 may contain a FILTER expression are:

AnQL query evaluation (cont.)

Let P be a BAP, P_i AGPs, G an annotated graph and R a filter expression:

- $[\![P]\!]_G = \{\theta \mid dom(\theta) = var(P) \text{ and } G \models \theta(P)\}$
- $[\![P_1 \text{ AND } P_2]\!]_G = \{\theta_1 \otimes \theta_2 \mid \theta_1 \in [\![P_1]\!]_G, \theta_2 \in [\![P_2]\!]_G, \theta_1 \text{ and } \theta_2 \ \otimes\text{-compatible}\}$
- $[\![P_1 \text{ UNION } P_2]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$
- $[\![P_1 \text{ FILTER } R]\!]_G = \{\theta \mid \theta \in [\![P_1]\!]_G \text{ and } R\theta \text{ is true}\}$
- $[\![P_1$ OPTIONAL $P_2[R]]\!]_G = \{\theta \mid$ and θ meets one of the following conditions:
    1. $\theta = \theta_1 \otimes \theta_2$, $\theta_1 \in [\![P_1]\!]_G, \theta_2 \in [\![P_2]\!]_G$ are $\otimes$-compatible, and $R\theta$ is true

  $\}$

Keep compatible substitutions that make the FILTER R true

AnQL query evaluation (cont.)

Let P be a BAP, P_i AGPs, G an annotated graph and R a filter expression:

- $[\![P]\!]_G = \{\theta \mid dom(\theta) = var(P) \text{ and } G \models \theta(P)\}$
- $[\![P_1 \text{ AND } P_2]\!]_G = \{\theta_1 \otimes \theta_2 \mid \theta_1 \in [\![P_1]\!]_G, \theta_2 \in [\![P_2]\!]_G, \theta_1 \text{ and } \theta_2 \otimes\text{-compatible}\}$
- $[\![P_1 \text{ UNION } P_2]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$
- $[\![P_1 \text{ FILTER } R]\!]_G = \{\theta \mid \theta \in [\![P_1]\!]_G \text{ and } R\theta \text{ is true}\}$
- $[\![P_1$ OPTIONAL $P_2[R]]\!]_G = \{\theta \mid$ and θ meets one of the following conditions:
    1. $\theta = \theta_1 \otimes \theta_2$, $\theta_1 \in [\![P_1]\!]_G, \theta_2 \in [\![P_2]\!]_G$ are $\otimes$-compatible, and $R\theta$ is true
    2. $\theta = \theta_1 \in [\![P_1]\!]_G$ and $\forall \theta_2 \in [\![P_2]\!]_G, \theta_1, \theta_2$ $\otimes$-compatible, $R(\theta_1 \otimes \theta_2)$ is true, and all annotation variables $\lambda \in dom(\theta_1) \cap dom(\theta_2)$ $\theta_2(\lambda) \prec \theta_1(\lambda)$

$\}$

Keep substitutions θ_1 if, for all θ_2 such that θ_1 and θ_2 are $\otimes$-compatible and for all the shared annotation variables between those substitutions, θ_1 has a "better" annotation

AnQL query evaluation (cont.)

Let P be a BAP, P_i AGPs, G an annotated graph and R a filter expression:

- $[\![P]\!]_G = \{\theta \mid dom(\theta) = var(P) \text{ and } G \models \theta(P)\}$
- $[\![P_1 \text{ AND } P_2]\!]_G = \{\theta_1 \otimes \theta_2 \mid \theta_1 \in [\![P_1]\!]_G, \theta_2 \in [\![P_2]\!]_G, \theta_1 \text{ and } \theta_2 \otimes\text{-compatible}\}$
- $[\![P_1 \text{ UNION } P_2]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$
- $[\![P_1 \text{ FILTER } R]\!]_G = \{\theta \mid \theta \in [\![P_1]\!]_G \text{ and } R\theta \text{ is true}\}$
- $[\![P_1$ OPTIONAL $P_2[R]]\!]_G = \{\theta \mid$ and θ meets one of the following conditions:
  1. $\theta = \theta_1 \otimes \theta_2$, $\theta_1 \in [\![P_1]\!]_G, \theta_2 \in [\![P_2]\!]_G$ are $\otimes$-compatible, and $R\theta$ is true
  2. $\theta = \theta_1 \in [\![P_1]\!]_G$ and $\forall \theta_2 \in [\![P_2]\!]_G, \theta_1, \theta_2$ $\otimes$-compatible, $R(\theta_1 \otimes \theta_2)$ is true, and all annotation variables $\lambda \in dom(\theta_1) \cap dom(\theta_2)$ $\theta_2(\lambda) \prec \theta_1(\lambda)$
  3. $\theta = \theta_1 \in [\![P_1]\!]_G$ and $\forall \theta_2 \in [\![P_2]\!]_G, \theta_1, \theta_2$ $\otimes$-compatible $R(\theta_1 \otimes \theta_2)$ is false

  $\}$

Keep substitutions θ_1 if, for all θ_2 such that θ_1 and θ_2 are $\otimes$-compatible, the FILTER expression is false

Bibliography I

Peter Buneman and Egor Kostylev. Annotation algebras for rdfs. In *The Second International Workshop on the role of Semantic Web in Provenance Management (SWPM-10)*. CEUR Workshop Proceedings, 2010.

Claudio Gutiérrez, Carlos A. Hurtado, and Alejandro A. Vaisman. Temporal RDF. In *Proc. of 2nd European Semantic Web Conference (ESWC'2005)*, pages 93–107, 2005.

Jorge Pérez, Marcelo Arenas, and Claudio Gutiérrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems*, 34(3), 2009.

Umberto Straccia, Nuno Lopes, Gergely Lukacsy, and Axel Polleres. A General Framework for Representing and Reasoning with Annotated Semantic Web Data. In *Proc. of 24th AAAI Conference on Artificial Intelligence (AAAI'2010)*, 2010.

Bibliography II

Octavian Udrea, Diego Reforgiato Recupero, and V. S. Subrahmanian. Annotated RDF. *ACM Trans. Comput. Logic*, 11 (2):1–41, 2010.