RDF(S) needs annotations

RDF Next Steps W3C Workshop

Nuno Lopes Antoine Zimmermann Aidan Hogan Gergely Lukácsy **Axel Polleres** Umberto Straccia Stefan Decker

June 26, 2010

RDF is good...

...but triples alone are often not enough:
RDF statements s p o are true with respect to a certain **context**:

RDF is good...

...but triples alone are often not enough:
RDF statements s p o are true with respect to a certain **context**:

- Time

```
:axel :worksfor :DERI true ‘‘since 2007’’
```

RDF is good...

...but triples alone are often not enough:
RDF statements s p o are true with respect to a certain **context**:

- Time

```
:axel :worksfor :DERI true ‘‘since 2007’’
```

- Provenance

```
:axel f:knows :ivanherman true ‘‘in http://polleres.net/foaf.rdf’’
f:knows rdfs:domain f:Person true ‘‘in http://xmlns.com/foaf/0.1’’
```

RDF is good...

...but triples alone are often not enough:
RDF statements s p o are true with respect to a certain **context**:

- Time

```
:axel :worksfor :DERI true ‘‘since 2007’’
```

- Provenance

```
:axel f:knows :ivanherman true ‘‘in http://polleres.net/foaf.rdf’’
f:knows rdfs:domain f:Person true ‘‘in http://xmlns.com/foaf/0.1’’
```

- Trust/Certainty (fuzzy values):

```
:audiTT rdf:type :SportsCar true ‘‘to some extent, e.g.  0.8’’
```

- etc.

RDF needs annotations...

This need comes from several sides:

- Time

...seems to be a practical need... Data is NOT static! some suggestions in academia [Gutierrez+ 2005] [Tappolet&Bernstein, 2009]

- Provenance

...seems to be a practical need... (Linked) Data is NOT universal! Named Graphs [Carroll+ 2005], Quads (Authoritative reasoning) [Hogan+ 2009]

- Trust/Certainty (fuzzy values):

...NOT all data is certain/trusted explored in the W3C Uncertainty Reasoning for the Web XG

- not so new...e.g. modules in TRIPLE

Adding information to RDF triples

- Issues:
  - Representation of annotations
  - Semantics of annotations

Adding information to RDF triples

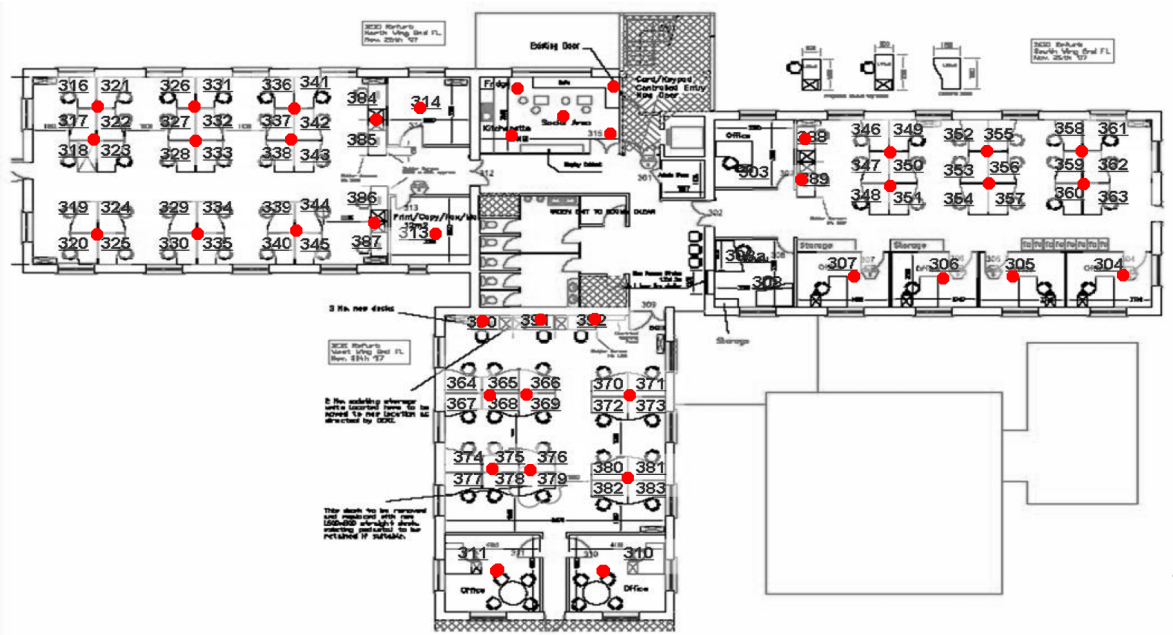
- Issues:
  - Representation of annotations
  - Semantics of annotations
- Our Claim:
  - RDF needs **agreement** on representation and semantics for the most important annotation domains.

Adding information to RDF triples

- Issues:
  - Representation of annotations
  - Semantics of annotations
- Our Claim:
  - RDF needs **agreement** on representation and semantics for the most important annotation domains.
- One Proposal:
  - Annotated RDFS

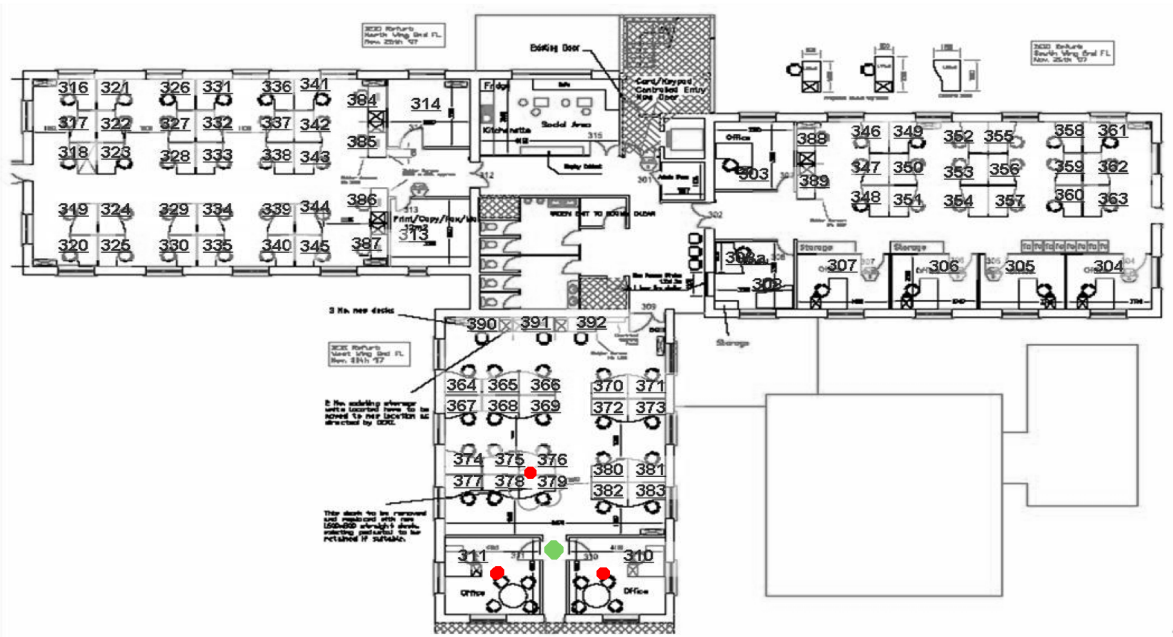
Example: Sensor data

Even combinations of several domains may be necessary:



Example: Sensor data

Even combinations of several domains may be necessary:

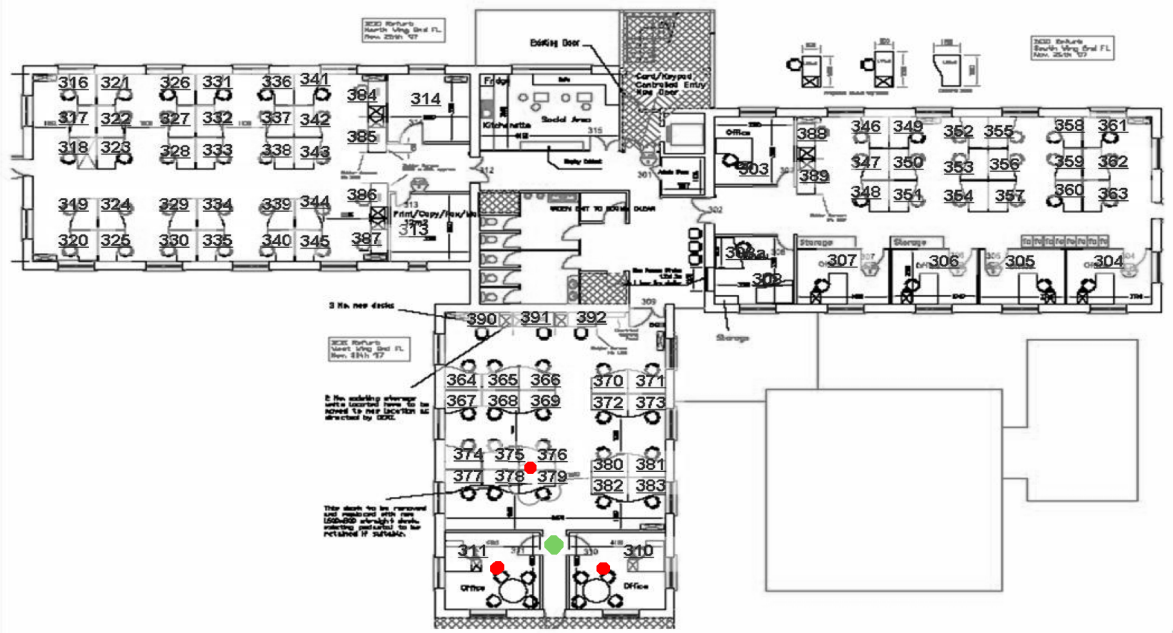


sensors readings output:

```
2010-06-26 14:57:51   10.254.2.15   4302   83
2010-06-26 14:57:51   10.254.3.1    4302   83
2010-06-26 14:57:51   10.254.2.6    4302   83
```

Example: Sensor data

Even combinations of several domains may be necessary:



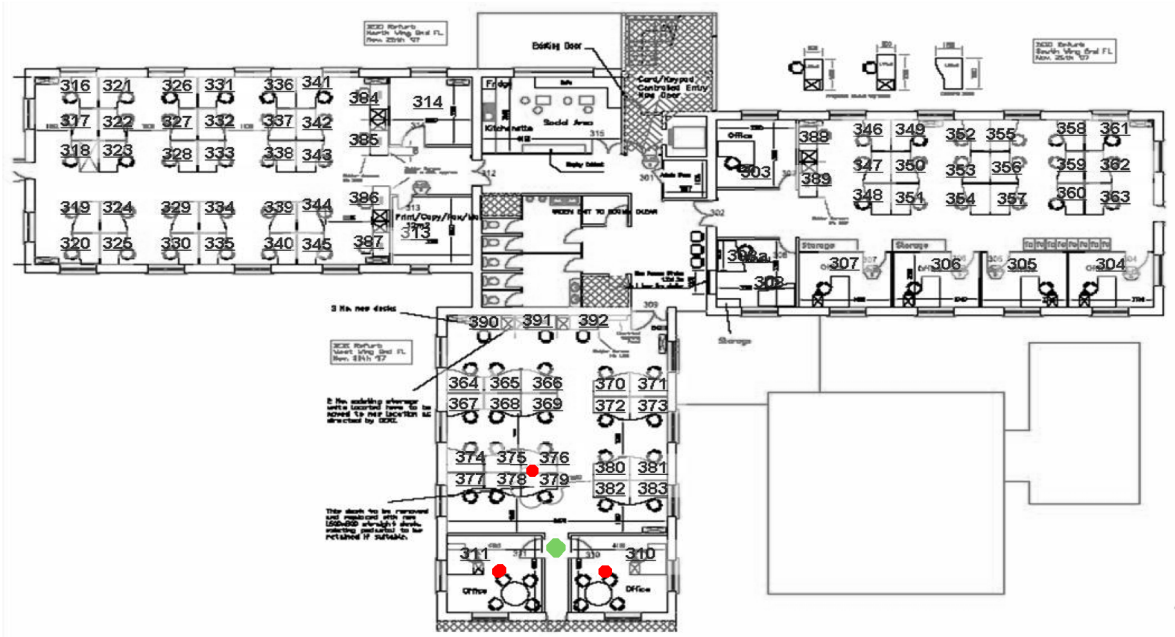
sensors readings output:

```
2010-06-26 14:57:51   10.254.2.15   4302   83
2010-06-26 14:57:51   10.254.3.1    4302   83
2010-06-26 14:57:51   10.254.2.6    4302   83
```

Convert to

Example: Sensor data

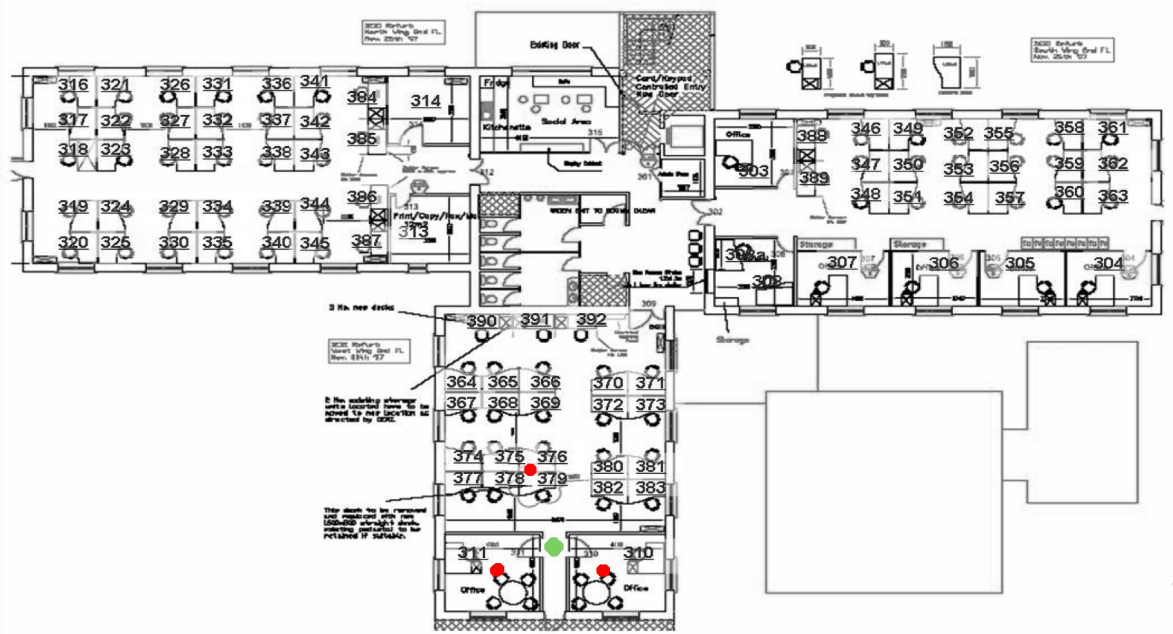
Even combinations of several domains may be necessary:



- location of a tag in a room (pure RDF)
- time of the sensor reading (temporal annotation)
- signal strength of the sensor reading (fuzzy annotation)

Example: Sensor data

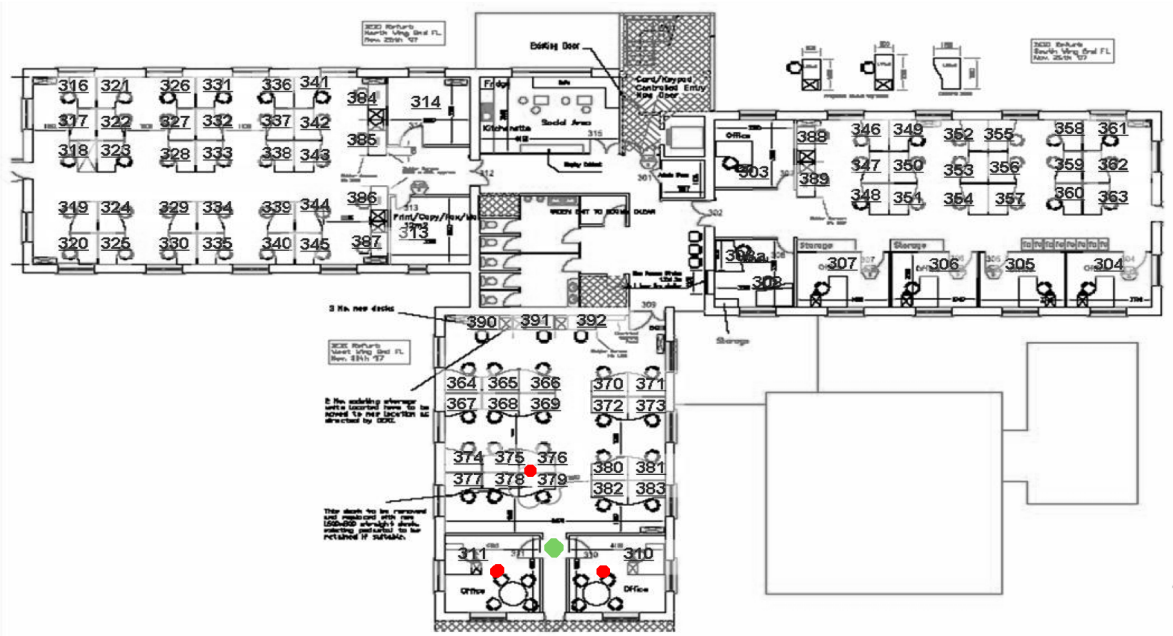
Even combinations of several domains may be necessary:



```
:tag4302 :locatedIn :room311 . [14:57, 15:01] [0.8]
:tag4302 :locatedIn :room310 . [15:02, 16:10] [0.7]
```

Example: Sensor data

Even combinations of several domains may be necessary:




```
:tag4302 :locatedIn :room311 . [14:57, 15:01] [0.8]
:tag4302 :locatedIn :room310 . [15:02, 16:10] [0.7]
```

This is not RDF

Issue 1: Representation

How to represent annotations?

```
:tag4302 :locatedIn :room311 . [14:57, 15:01]
```

Issue 1: Representation


```
:tag4302 :locatedIn :room311 . [14:57, 15:01]
```

Reification?

```
:record1 rdf:type rdf:Statement
         rdf:subject :tag4302;
         rdf:predicate :locatedIn ;
         rdf:object :room311 ;
         time:start "2010-06-26 14:57"^^xs:timeStamp;
         time:end "2010-06-26 15:01"^^xs:timeStamp .
```

Issue 1: Representation


```
:tag4302 :locatedIn :room311 . [14:57, 15:01]
```

Reification?

```
:record1 rdf:type rdf:Statement
         rdf:subject :tag4302;
         rdf:predicate :locatedIn ;
         rdf:object :room311 ;
         time:start "2010-06-26 14:57"^^xs:timeStamp;
         time:end "2010-06-26 15:01"^^xs:timeStamp .
```

- no semantics

Issue 1: Representation


```
:tag4302 :locatedIn :room311 . [14:57, 15:01]
```

Reification?

```
:record1 rdf:type rdf:Statement
         rdf:subject :tag4302;
         rdf:predicate :locatedIn ;
         rdf:object :room311 ;
         time:start "2010-06-26 14:57"^^xs:timeStamp;
         time:end "2010-06-26 15:01"^^xs:timeStamp .
```

- no semantics
- not really "popular" some people prior to this WS even claimed to drop reification alltogether

Issue 1: Representation


```
:tag4302 :locatedIn :room311 . [14:57, 15:01]
```

Other formats?

- N-Quads

```
:tag4302 :locatedIn :room311 _:c.
_:c  time:start "2010-06-26 14:57"^^xs:timeStamp ;
     time:end "2010-06-26 15:01"^^xs:timeStamp .
```

- alternatively TriG, TriX
- non-standard (yet)
- semantics of annotations still not clear

Issue 2: Semantics

- What do annotations mean for RDF(S) semantics?
- How to combine non-annotated and annotated RDF semantically?

Issue 2: Semantics

- What do annotations mean for RDF(S) semantics?
- How to combine non-annotated and annotated RDF semantically?

```
:axel f:knows :ivanherman .          [http://polleres.net/foaf.rdf]
f:knows rdfs:domain f:Person .          [http://xmlns.com/foaf/0.1]
```

Issue 2: Semantics

- What do annotations mean for RDF(S) semantics?
- How to combine non-annotated and annotated RDF semantically?

```
:axel f:knows :ivanherman .          [http://polleres.net/foaf.rdf]
f:knows rdfs:domain f:Person .          [http://xmlns.com/foaf/0.1]
-------------------------------------------------------------------
:axel rdf:type f:Person .                                    [???]
```

Issue 2: Semantics

- What do annotations mean for RDF(S) semantics?
- How to combine non-annotated and annotated RDF semantically?

```
:axel :worksfor :DERI .                    [2007,2010]
:worksFor rdfs:domain :Employee .
-------------------------------------------------------
:axel rdf:domain :Employee                 [???]
```

Our approach – Annotated RDF

[Straccia+, AAAI2010] Generic Framework to

1. describe annotation domains
2. give them a semantics
3. live side-by-side with non-annotated RDF

Annotation Domains

Temporal domain example:

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

Any **annotation domain** consists of a lattice:

- the *representation* of the annotations

Annotation Domains

Temporal domain example:

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

Any **annotation domain** consists of a lattice:

- 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]
```

Any **annotation domain** consists of a lattice:

- 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]
```

Any **annotation domain** consists of a lattice:

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

Annotation Domains

Temporal domain example:

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

Any **annotation domain** consists of a lattice:

- 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]
```

Any **annotation domain** consists of a lattice:

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

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

Annotation Domains

Temporal domain example:

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

Any **annotation domain** consists of a lattice:

- 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 ($\otimes$) is a so-called t-norm

Annotation Domains

Temporal domain example:

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

Any **annotation domain** consists of a lattice:

- 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 ($\otimes$) is a so-called t-norm : $\cap$

Annotation Domains

Temporal domain example:

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

Any **annotation domain** consists of a lattice:

- 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 ($\otimes$) is a so-called t-norm : $\cap$



`[09:25,11:49]` $\otimes$ `[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]
```

Any **annotation domain** consists of a lattice:

- 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 ($\otimes$) is a so-called t-norm : $\cap$
  operator ($\vee$) for combining annotations

Annotation Domains

Temporal domain example:

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

Any **annotation domain** consists of a lattice:

- 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 ($\otimes$) is a so-called t-norm : $\cap$
  operator ($\vee$) for combining annotations: $\cup$

Annotation Domains

Temporal domain example:

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

Any **annotation domain** consists of a lattice:

- 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 ($\otimes$) is a so-called t-norm : $\cap$
  operator ($\vee$) for combining annotations: $\cup$



`[09:25,11:49]` $\vee$ `[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]
```

Any **annotation domain** consists of a lattice:

- 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 ($\otimes$) is a so-called t-norm : $\cap$
  operator ($\vee$) for combining annotations: $\cup$



`[09:25,11:49]` $\vee$ `[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]}
```

Any **annotation domain** consists of a lattice:

- 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 ($\otimes$) is a so-called t-norm : $\cap$
  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

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

annotations: [0,1]

order: $\leq$

$\otimes$: *min* $\vee$: *max*

$\top = 1, \quad \bot = 0$

Other domains: Examples

Trust/Fuzzy

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

annotations: $[0,1]$

order: $\leq$

$\otimes$: *min* $\vee$: *max*

$\top = 1, \quad \bot = 0$

Provenance

```
:axel rdf:type Person .
 [xmlns.com/foaf/0.1/ ∧ polleres.net/foaf.rdf]
```

annotations: *prop.*

formulae in DNF over URIs

order: $\models$

$\otimes$: $\wedge$ $\vee$: $\vee$

$\top$ = *disj. of all URIs*,
$\bot$ = *conj. of all URIs*

Other domains: Examples

Trust/Fuzzy

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

annotations: $[0,1]$

order: $\leq$

$\otimes$: *min* $\vee$: *max*

$\top = 1, \quad \bot = 0$

Provenance

```
:axel rdf:type Person .
 [xmlns.com/foaf/0.1/ ∧ polleres.net/foaf.rdf]
```

annotations: *prop.*

formulae in DNF over URIs

order: $\models$

$\otimes$: $\wedge$ $\vee$: $\vee$

$\top$ = *disj. of all URIs*,
$\bot$ = *conj. of all URIs*

Other domains: Examples

Trust/Fuzzy

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

annotations: [0,1]

order: $\leq$

$\otimes$: *min* $\vee$: *max*

$\top = 1, \quad \bot = 0$

Provenance

```
:axel rdf:type Person .
 [xmlns.com/foaf/0.1/ ∧ polleres.net/foaf.rdf]
```

annotations: *prop.*

formulae in DNF over URIs

order: $\models$

$\otimes$: $\wedge$ $\vee$: $\vee$

$\top$ = *disj. of all URIs*,
$\bot$ = *conj. of all URIs*

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

```
: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

```
: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 $\top$ 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

```
: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

Annotated RDFS Inference rules

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

Annotated RDFS Inference rules

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

- RDFS "rdfs:domain" rule:

```
    ?p rdfs:domain ?c
    ?s ?p ?o
⇒   ?s rdf:type ?c
```

Annotated RDFS Inference rules

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

- RDFS "rdfs:domain" rule:

```
   ?p rdfs:domain ?c
   ?s ?p ?o
⇒  ?s rdf:type ?c
```

Example:

```
   :worksFor rdfs:domain :Employee
   :nuno :worksFor :DERI
⇒  :nuno rdf:type :Employee
```

Annotated RDFS Inference rules

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

- Annotated RDFS "rdfs:domain" rule:

	`?p rdfs:domain ?c`	$?v1$
	`?s ?p ?o`	$?v2$
$\Rightarrow$	`?s rdf:type ?c`	$?v1 \otimes ?v2$

Example:

```
   :worksFor rdfs:domain :Employee
   :nuno :worksFor :DERI
⇒  :nuno rdf:type :Employee
```

Annotated RDFS Inference rules

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

- Annotated RDFS "rdfs:domain" rule:

`?p rdfs:domain ?c`	$?v1$
`?s ?p ?o`	$?v2$
$\Rightarrow$ `?s rdf:type ?c`	$?v1 \otimes ?v2$

Example:

`:worksFor rdfs:domain :Employee`	$[-\infty, +\infty]$
`:nuno :worksFor :DERI`	["2009-01-01", "2010-06-26"]
$\Rightarrow$ `:nuno rdf:type :Employee`	["2009-01-01", "2010-06-26"]

Annotated RDFS Inference rules

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

- Annotated RDFS "rdfs:domain" rule:

```
   ?p rdfs:domain ?c     ?v1
   ?s ?p ?o              ?v2
⇒  ?s rdf:type ?c        ?v1 ⊗ ?v2
```

Example:

```
   :worksFor rdfs:domain :Employee     [−∞, +∞]
   :nuno :worksFor :DERI               ["2009-01-01", "2010-06-26"]
⇒  :nuno rdf:type :Employee            ["2009-01-01", "2010-06-26"]
```

- Extra rule to group annotations triples ($\vee$):

```
   ?s ?p ?o      λ1
   ?s ?p ?o      λ2
⇒  ?s ?p ?o      λ1 ∨ λ2
```

Annotated RDFS Inference rules

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

- Annotated RDFS "rdfs:domain" rule:

	`?p rdfs:domain ?c`	$?v1$
	`?s ?p ?o`	$?v2$
$\Rightarrow$	`?s rdf:type ?c`	$?v1 \otimes ?v2$

Example:

	`:worksFor rdfs:domain :Employee`	$[-\infty, +\infty]$
	`:nuno :worksFor :DERI`	`["2009-01-01", "2010-06-26"]`
$\Rightarrow$	`:nuno rdf:type :Employee`	`["2009-01-01", "2010-06-26"]`

- Extra rule to group annotations triples ($\vee$):

	`:nuno :worksFor :DERI`	`["2008-05-01", "2010-01-01"]`
	`:nuno :worksFor :DERI`	`["2009-01-01", "2010-06-26"]`
$\Rightarrow$	`:nuno :worksFor :DERI`	`["2008-05-01", "2010-06-26"]`

Summary

- Our Claim:
  - RDF needs **agreement** on representation and semantics for important annotation domains e.g. time, provenance, trust
- Representational Issues:
  - several options (reification, N-quads,TriG/X)
  - reification the only standards compliant thus far, sub-optimal
- Semantics of annotations:
  - Our proposal: Annotated RDFS
    - allows arbitrary ordered annotation domains
    - give them a semantics on top of RDFS
    - live side-by-side with non-annotated RDF
    - SPARQL(1.1) compatibe...
- TODO for us here?
  - At the least: Representation to add context to triples
  - Needs to be "upwards-compatible"
  - wish-list: tackle semantic vacuum on context for important domains (e.g., time, provenance, trust/fuzzy)

Annotated SPARQL:

Extend SPARQL to allow querying annotated RDF

- "Annotation aware" SPARQL

Annotated SPARQL:

Extend SPARQL to allow querying annotated RDF

- "Annotation aware" SPARQL

  "Where was Stefan between 14:30 and 15:00?"

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

Annotated SPARQL:

Extend SPARQL to allow querying annotated RDF

- "Annotation aware" SPARQL

  "Where was Stefan between 14:30 and 15:00?"

```
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

Annotated SPARQL

"When were Stefan and Axel in the same room?"

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

Annotated SPARQL

"When were Stefan and Axel in the same room?"

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

Answers:

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

(?Room, ?TimeInterval) = (:conferenceRoom, {["14:25", "14:57"]})
```