Annotated RDF

DERI Reading Group Presentation

Nuno Lopes

May 5, 2010

RDF

subject predicate object

RDF

subject predicate object

```
@prefix : <http://nunolopes.org/foaf.rdf#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:me foaf:name "Nuno Lopes" .
```

RDF

subject predicate object

```
@prefix : <http://nunolopes.org/foaf.rdf#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:me foaf:name "Nuno Lopes" .
```

```
:me foaf:workplaceHomepage <http://www.si.uevora.pt/> .
:me foaf:workplaceHomepage <http://www.deri.ie/> .
```

RDF

subject predicate object

Correct

```
@prefix : <http://nunolopes.org/foaf.rdf#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:me foaf:name "Nuno Lopes" .
```

Incorrect information!

```
:me foaf:workplaceHomepage <http://www.si.uevora.pt/> .
:me foaf:workplaceHomepage <http://www.deri.ie/> .
```

Annotated RDF

`subject predicate object annotation`

Annotated RDF

subject predicate object annotation

```
:me foaf:wpH <http://www.si.uevora.pt/> . [24-10-2005, 30-04-2008]
:me foaf:wpH <http://www.deri.ie/> .      [01-05-2008, now]
```

wpH = workplaceHomepage

Annotated RDF

subject predicate object annotation

```
:me foaf:wpH <http://www.si.uevora.pt/> . [24-10-2005, 30-04-2008]
:me foaf:wpH <http://www.deri.ie/> .      [01-05-2008, now]
```

wpH = workplaceHomepage

Annotations refer to a specific **domain**

- temporal
- trust (fuzzy)
- provenance
- ...

Domain Example - Provenance


```
"Mary"    :hasSupervisor   "William".  "Personal Webpage"
"Mary"    :hasSupervisor   "William".  "Faculty List"
"Max"     :hasAdvisor      "William".  "Faculty List"
"Max"     :hasSupervisor   "Stephen".  "Departmental Webpage"
"William" :hasSupervisor   "Stephen".  "Graduate School"
```

Domain Example - Provenance


```
"Mary"    :hasSupervisor   "William".  "Personal Webpage"
"Mary"    :hasSupervisor   "William".  "Faculty List"
"Max"     :hasAdvisor      "William".  "Faculty List"
"Max"     :hasSupervisor   "Stephen".  "Departmental Webpage"
"William" :hasSupervisor   "Stephen".  "Graduate School"
```

Partial order $\preceq$:

```
"Personal Webpage" ≼ "Departmental Webpage"
"Faculty List" ≼ "Graduate School"
```

Annotation **domain**: partially ordered set $(\mathcal{A}, \preceq)$

- $\mathcal{A}$ is the set of annotations
- $\preceq$ is the partial order (with a bottom element $\perp$)

Consistency

aRDF can introduce inconsistencies:

- e.g. the triple (*Mary*, *hasSupervisor*, *William*) in the previous example

Consistency

aRDF can introduce inconsistencies:

e.g. the triple (*Mary*, *hasSupervisor*, *William*) in the previous example

"Faculty List" is not comparable to "Personal Webpage"

```
"Mary"    :hasSupervisor   "William".  "Personal Webpage"
"Mary"    :hasSupervisor   "William".  "Faculty List"

"Personal Webpage" ⪯ "Departmental Webpage"
"Faculty List" ⪯ "Graduate School"
```

Consistency

aRDF can introduce inconsistencies:

e.g. the triple (*Mary*, *hasSupervisor*, *William*) in the previous example

"Faculty List" is not comparable to "Personal Webpage"

```
"Mary"    :hasSupervisor   "William".  "Personal Webpage"
"Mary"    :hasSupervisor   "William".  "Faculty List"

"Personal Webpage" ⪯ "Departmental Webpage"
"Faculty List" ⪯ "Graduate School"
```

If the partially ordered set $\mathcal{A}$ contains a top element $\top$ the aRDF is guaranteed to be consistent.

RDFS schema

Supported vocabulary:

- `rdfs:subClassOf`
- `rdf:type`
- `rdfs:subPropertyOf`
  - *transitive* and *non-transitive* properties

RDFS schema

Supported vocabulary:

- `rdfs:subClassOf`
- `rdf:type`
- `rdfs:subPropertyOf`
  - *transitive* and *non-transitive* properties

Mentions that other RDFS constructs are possible, but consider *rdfs:subPropertyOf* particularly important.

Query example


```
"Max"      :hasAdvisor      "Adam".     (0.9,  2004)
"Adam"     :hasSupervisor   "William".  (0.95, 2003)
"Mary"     :hasAdvisor      "William".  (0.7,  2003)

:hasAdvisor   rdfs:subPropertyOf   :hasSupervisor
```

- A query is a triple (with possible variables) $q = (Max, ?p, William) : (0.8, 2002)$

Query example


```
"Max"      :hasAdvisor       "Adam".      (0.9,   2004)
"Adam"     :hasSupervisor    "William".   (0.95,  2003)
"Mary"     :hasAdvisor       "William".   (0.7,   2003)

:hasAdvisor   rdfs:subPropertyOf   :hasSupervisor
```

- A query is a triple (with possible variables)
  $q = (Max, ?p, William) : (0.8, 2002)$
- Possible annotation answers: all $a \in \mathcal{A}$ where $(0.8, 2002) \preceq a$

$$A_{\mathcal{O}}(q) = \begin{array}{l}\{\ldots, (Max, hasSupervisor, William) : (0.8, 2002), \\ \ldots, (Max, hasSupervisor, William) : (0.9, 2003)\}\end{array}$$

Query example


```
"Max"       :hasAdvisor       "Adam".      (0.9,   2004)
"Adam"      :hasSupervisor    "William".   (0.95,  2003)
"Mary"      :hasAdvisor       "William".   (0.7,   2003)

:hasAdvisor    rdfs:subPropertyOf    :hasSupervisor
```

- A query is a triple (with possible variables) $q = (Max, ?p, William) : (0.8, 2002)$
- Possible annotation answers: all $a \in \mathcal{A}$ where $(0.8, 2002) \preceq a$
$$A_{\mathcal{O}}(q) = \begin{array}{l}\{\ldots, (Max, hasSupervisor, William) : (0.8, 2002), \\ \ldots, (Max, hasSupervisor, William) : (0.9, 2003)\}\end{array}$$
- *answer* is a set of triples, eliminating redundant annotations $Ans_{\mathcal{O}}(q) = \{(Max, hasSupervisor, William) : (0.9, 2003)\}$

Query answering algorithms

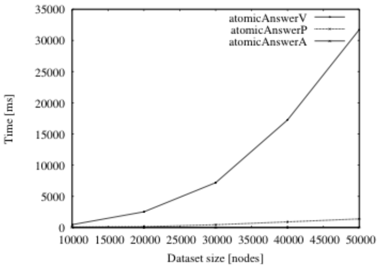
Algorithms for different types of queries:

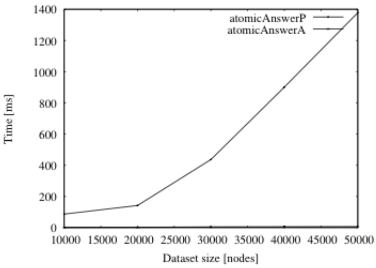
- *atomicAnswerV* - $(r, p, ?v) : a$
- *atomicAnswerP* - $(r, ?p, v) : a$
- *atomicAnswerA* - $(r, p, v) :?a$

Polynomial complexity for these algorithms. Conjunctive query answering yield exponential complexity.

Experimental Results

- Tested using generated aRDF dataset ranging from 10 000 to 100 000 triples





Conclusions

- Representation capable of encompassing several annotations
- Consistency results for annotation domains
- no proper support for RDF schema
- no SPARQL

Conclusions

- Representation capable of encompassing several annotations
- Consistency results for annotation domains
- no proper support for RDF schema
- no SPARQL

Friday talk presenting our extensions to this work

Annotated RDF triples graphs

Assuming fixed sets:

- $\mathcal{R}$ of resource names
- $\mathcal{P}$ of property names

  $dom(p)$ set of values associated with property p

  $(r, p, v) : a$ is an annotated triple if

  - r is a *resource* name
  - p is a *property* name
  - v is a value (may also be a *resource*)

- An annotated-RDF ontology $\mathcal{O}$ is a set of finite annotated triples

Ontology Graph

Ontology graphs

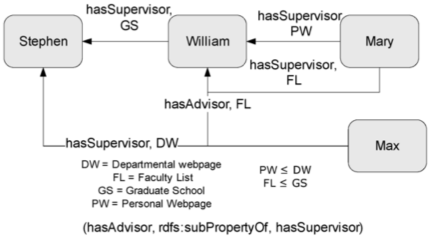
- $V = \mathcal{R} \cup \bigcup_{p \in \mathcal{P}} dom(p)$
- $E = \{(r, r') \mid (r, p, r') : a \in \mathcal{O}\}$
- $\lambda(r, r') = \{p : a \mid (r, p, r') : a \in \mathcal{O}\}$ (edge labelling function)

Ontology Graph

Ontology graphs

- $V = \mathcal{R} \cup \bigcup_{p \in \mathcal{P}} dom(p)$
- $E = \{(r, r') \mid (r, p, r') : a \in \mathcal{O}\}$
- $\lambda(r, r') = \{p : a \mid (r, p, r') : a \in \mathcal{O}\}$ (edge labelling function)

Ontology Graph of the example on Slide 3:



Property paths

- for a *transitive* property, a *p-path* between nodes r, r' are the triples $\{t_1 = (r, p_1, r_1) : a_1, \ldots, t_i = (r_{i-1}, p_i, r_i) : a_i, \ldots, t_k = (r_{k-1}, p_k, r') : a_k\}, \forall i \in [1, k](p_i, rdfs : subPropertyOf*, p)$

Semantics

Property paths

- for a *transitive* property, a *p-path* between nodes r, r' are the triples $\{t_1 = (r, p_1, r_1) : a_1, \ldots, t_i = (r_{i-1}, p_i, r_i) : a_i, \ldots, t_k = (r_{k-1}, p_k, r') : a_k\}, \forall i \in [1, k](p_i, rdfs : subPropertyOf*, p)$
- *Univ* denotes the set of all triples (r, p, v)
- An aRDF-interpretation *I is a mapping from Univ to* $\mathcal{A}$.

Semantics

Property paths

- for a *transitive* property, a *p-path* between nodes r, r' are the triples $\{t_1 = (r, p_1, r_1) : a_1, \ldots, t_i = (r_{i-1}, p_i, r_i) : a_i, \ldots, t_k = (r_{k-1}, p_k, r') : a_k\}, \forall i \in [1, k](p_i, rdfs : subPropertyOf*, p)$

- *Univ* denotes the set of all triples (r, p, v)
- An aRDF-interpretation *I is a mapping from Univ to* $\mathcal{A}$.

- An aRDF-interpretation I satisfies $(r, p, v) : a$ iff $a \preceq I(r, p, v)$.

Semantics

Property paths

- for a *transitive* property, a *p-path* between nodes r, r' are the triples $\{t_1 = (r, p_1, r_1) : a_1, \ldots, t_i = (r_{i-1}, p_i, r_i) : a_i, \ldots, t_k = (r_{k-1}, p_k, r') : a_k\}, \forall i \in [1, k](p_i, rdfs : subPropertyOf*, p)$
- *Univ* denotes the set of all triples (r, p, v)
- An aRDF-interpretation I *is a mapping from Univ to* $\mathcal{A}$.
- An aRDF-interpretation I satisfies $(r, p, v) : a$ iff $a \preceq I(r, p, v)$.
- I satisfies $\mathcal{O}$ iff:
  - I satisfies every $(r, p, v) : a \in \mathcal{O}$;
  - For all *transitive* properties $p \in \mathcal{P}$, for all *p-paths* $Q = \{t_1, \ldots, t_k\}$, $t_i = (r_i, p_i, r_{i+1}) : a_i$, for all $a \in \mathcal{A}$ such that $a \preceq a_i, 1 \leq i \leq k, a \preceq I(r_1, p, r_{k+1})$.

aRDF query answering

- Two triples $(r, p, v) : a$ and $(r', p', v') : a'$ are *semi-unifiable* if there exists a substitution θ such that $\theta(r) = \theta(r')$, $\theta(p) = \theta(p')$ and $\theta(v) = \theta(v')$.

aRDF query answering

- Two triples $(r, p, v) : a$ and $(r', p', v') : a'$ are *semi-unifiable* if there exists a substitution θ such that $\theta(r) = \theta(r')$, $\theta(p) = \theta(p')$ and $\theta(v) = \theta(v')$.
- Given a consistent ontology $\mathcal{O}$ and a query $q = (r_q, p_q, v_q) : a_q$, then $A_{\mathcal{O}}(q) = \{(r, p, v) : a\}$ s.t.
  - $(r, p, v) : a$ is semi-unifiable with q
  - $\mathcal{O} \models (r, p, v) : a$
  - $(a \textit{ is a variable}) \vee (a_q \preceq a)$

aRDF query answering

- Two triples $(r, p, v) : a$ and $(r', p', v') : a'$ are *semi-unifiable* if there exists a substitution θ such that $\theta(r) = \theta(r')$, $\theta(p) = \theta(p')$ and $\theta(v) = \theta(v')$.

- Given a consistent ontology $\mathcal{O}$ and a query $q = (r_q, p_q, v_q) : a_q$, then $A_{\mathcal{O}}(q) = \{(r, p, v) : a\}$ s.t.
  - $(r, p, v) : a$ is semi-unifiable with q
  - $\mathcal{O} \models (r, p, v) : a$
  - $(a \textit{ is a variable}) \vee (a_q \preceq a)$

- Eliminate redundant triples:

  An *answer* to q is $Ans_{\mathcal{O}}(q) = \{(r, p, v) : a\}$ s.t.:
  - $(r, p, v) : a \in A_{\mathcal{O}}(q)$
  - $\nexists S \subseteq Ans_{\mathcal{O}}(q) - \{(r, p, v) : a\}$ $s.t.$ $S \models (r, p, v) : a$