Datalog revisited for reasoning in Linked Data

Marie-Christine Rousset

Université de Grenoble-Alpes & Institut Universitaire de France

Joint work with M. Al Bakri, M. Atencia, J. David, F. Jouanot, S.Lalande, O.Palombi and F. Ulliana
Linked Data: the Semantic Web published in RDF

Datalog revisited for reasoning in Linked Data

September 2 2016 2 / 29
An RDF dataset: a set of triples (called an RDF graph)

@prefix rdf: ⟨http://www.w3.org/1999/02/22-rdf-syntax-ns#⟩.
@prefix rdfs: ⟨http://www.w3.org/2000/01/rdf-schema#⟩.
@prefix owl: ⟨http://www.w3.org/2002/07/owl#⟩.

wikipedia:Marie_Curie hasName "Marie Curie".
wikipedia:Marie_Curie rdf:type Chemist.
wikipedia:Marie_Curie hasWonPrize NobelPrize.
wikipedia:Marie_Curie bornIn Europe.
wikipedia:Albert_Einstein hasName "Albert Einstein".
wikipedia:Albert_Einstein rdf:type Physicist.
wikipedia:Albert_Einstein hasWonPrize NobelPrize.
wikipedia:Albert_Einstein birthPlace Ulm.
Ulm locatedIn Germany.
Germany partOf Europe.
Chemist rdfs:subClassOf Scientist.
Physicist rdfs:subClassOf Scientist.

owl:ObjectPropertyChain (birthPlace Located partOf) rdfs:subPropertyOf bornIn.
An RDF dataset: a set of triples (called an RDF graph)

@prefix rdf:  ⟨http://www.w3.org/1999/02/22-rdf-syntax-ns#⟩.
@prefix rdfs: ⟨http://www.w3.org/2000/01/rdf-schema#⟩.
@prefix owl: ⟨http://www.w3.org/2002/07/owl#⟩.
@prefix wikipedia: ⟨https://fr.wikipedia.org/wiki/⟩.

wikipedia:Marie_Curie hasName "Marie Curie".
wikipedia:Marie_Curie rdf:type Chemist.
wikipedia:Marie_Curie hasWonPrize NobelPrize.
wikipedia:Marie_Curie bornIn Europe.
wikipedia:Albert_Einstein hasName "Albert Einstein".
wikipedia:Albert_Einstein rdf:type Physicist.
wikipedia:Albert_Einstein hasWonPrize NobelPrize.
wikipedia:Albert_Einstein birthPlace Ulm.
Ulm locatedIn Germany.
Germany partOf Europe.
Chemist rdfs:subClassOf Scientist.
Physicist rdfs:subClassOf Scientist.

owl:ObjectPropertyChain (birthPlace Located partOf) rdfs:subPropertyOf bornIn.
The core query language of SPARQL is: Basic Graph Pattern (BGP) queries, i.e. conjunctive or SELECT-PROJECT-JOIN queries.

Example of a SPARQL conjunctive query

Return the names of scientists born in Europe who received a Nobel Prize

- SELECT ?n WHERE { ?p rdf:type Scientist . ?p hasWon NobelPrize . ?p bornIn Europe . ?p hasName ?n . }

A SPARQL query can search over the data and the schema

Return the properties having Europe as value

- q(?prop):- ?s ?prop Europe.
SPARQL evaluation over an RDF graph (by example)

$q(\theta(prop))$ is an answer for each substitution $\theta$ of the query variables by constants that maps every query conjunct to a fact.

**RDF graph $G$**

```
wikipedia:Marie_Curie hasName "Marie Curie" .
wikipedia:Marie_Curie rdf:type Chemist .
wikipedia:Marie_Curie hasWonPrize NobelPrize.
wikipedia:Marie_Curie bornIn Europe.
wikipedia:Albert_Einstein hasName "Albert Einstein".
wikipedia:Albert_Einstein rdf:type Physicist.
wikipedia:Albert_Einstein hasWonPrize NobelPrize.
wikipedia:Albert_Einstein birthPlace Ulm .
Ulm locatedIn Germany. Germany partOf Europe.
Chemist rdfs:subClassOf Scientist . Physicist rdfs:subClassOf Scientist .
owl:ObjectPropertyChain (birthPlace Located partOf) rdfs:subPropertyOf bornIn.
```  

```
q(\theta(prop)):- \theta(s \ \theta(prop) \ \text{Europe}).
```

**Result of SPARQL evaluation over $G$**

$q(G)= \{ \text{bornIn} , \text{partOf} \}$
SPARQL evaluation over an RDF graph (by example)

**RDF graph G**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>[wikipedia:Marie_Curie]</td>
<td>hasName</td>
<td>&quot;Marie Curie&quot;</td>
</tr>
<tr>
<td>[wikipedia:Marie_Curie]</td>
<td>rdf:type</td>
<td>Chemist</td>
</tr>
<tr>
<td>[wikipedia:Marie_Curie]</td>
<td>hasWonPrize</td>
<td>NobelPrize</td>
</tr>
<tr>
<td>[wikipedia:Marie_Curie]</td>
<td>bornIn</td>
<td>Europe</td>
</tr>
<tr>
<td>[wikipedia:Albert_Einstein]</td>
<td>hasName</td>
<td>&quot;Albert Einstein&quot;</td>
</tr>
<tr>
<td>[wikipedia:Albert_Einstein]</td>
<td>rdf:type</td>
<td>Physicist</td>
</tr>
<tr>
<td>[wikipedia:Albert_Einstein]</td>
<td>hasWonPrize</td>
<td>NobelPrize</td>
</tr>
<tr>
<td>[wikipedia:Albert_Einstein]</td>
<td>birthPlace</td>
<td>Ulm</td>
</tr>
<tr>
<td>Ulm</td>
<td>locatedIn</td>
<td>Germany</td>
</tr>
<tr>
<td>Germany</td>
<td>partOf</td>
<td>Europe</td>
</tr>
</tbody>
</table>

Chemist rdfs:subClassOf Scientist . Physicist rdfs:subClassOf Scientist .


**Result of standard SPARQL evaluation over G**

q(G) = ∅
Query answering over RDF graphs requires reasoning

\[ G_{rdfs} : \text{RDF facts} + \text{inferred facts by RDFS entailment} \]

\[
\text{wikipedia}:\text{Marie_Curie} \ \text{hasName} \ "\text{Marie Curie}" . \\
\text{wikipedia}:\text{Marie_Curie} \ \text{rdf:type} \ \text{Chemist}. \\
\text{wikipedia}:\text{Marie_Curie} \ \text{hasWonPrize} \ \text{NobelPrize}. \\
\text{wikipedia}:\text{Marie_Curie} \ \text{bornIn} \ \text{Europe}. \\
\text{wikipedia}:\text{Albert_Einstein} \ \text{hasName} \ "\text{Albert Einstein}" . \\
\text{wikipedia}:\text{Albert_Einstein} \ \text{rdf:type} \ \text{Physicist}. \\
\text{wikipedia}:\text{Albert_Einstein} \ \text{hasWonPrize} \ \text{NobelPrize}. \\
\text{wikipedia}:\text{Albert_Einstein} \ \text{birthPlace} \ \text{Ulm} . \\
\text{Ulm} \ \text{locatedIn} \ \text{Germany}. \quad \text{Germany} \ \text{partOf} \ \text{Europe}. \\
\text{Chemist} \ \text{rdfs:subClassOf} \ \text{Scientist}. \quad \text{Physicist} \ \text{rdfs:subClassOf} \ \text{Scientist}. \\
\text{wikipedia}:\text{Marie_Curie} \ \text{rdf:type} \ \text{Scientist}. \\
\text{wikipedia}:\text{Albert_Einstein} \ \text{rdf:type} \ \text{Scientist}. \\
\text{owl:ObjectPropertChain} \ (\text{birthPlace} \ \text{Located} \ \text{partOf}) \ \text{rdfs:subPropertyOf} \ \text{bornIn}. \\
\]

\[ q(\text{?n}):\text{?p} \ \text{rdf:type} \ \text{Scientist}, \text{?p} \ \text{hasWon} \ \text{NobelPrize}, \text{?p} \ \text{bornIn} \ \text{Europe}, \text{?p} \ \text{hasName} \ \text{?n} \]

**Query answer**

\[ q(G_{rdfs}^\infty) = \{"\text{Marie Curie}\} \]

Marie-Christine Rousset (Université de Grenoble) | Datalog revisited for reasoning in Linked Data | September 2 2016 | 8 / 29
Complete query answering may require full reasoning

\( G^\infty: \text{RDF facts} + \text{inferred facts by RDFS entailment} + \text{owl rules} \)

\[
\text{wikipedia:Marie_Curie hasName "Marie Curie" .}
\]

\[
\text{...}
\]

\[
\text{wikipedia:Albert_Einstein hasName "Albert Einstein".}
\]

\[
\text{wikipedia:Albert_Einstein rdf:type Physicist.}
\]

\[
\text{wikipedia:Albert_Einstein hasWonPrize NobelPrize.}
\]

\[
\text{wikipedia:Albert_Einstein birthPlace Ulm .}
\]

\[
\text{Ulm locatedIn Germany. Germany partOf Europe.}
\]

\[
\text{Physicist rdfs:subClassOf Scientist .}
\]

\[
\text{wikipedia:Albert_Einstein rdf:type Scientist.}
\]

\[
\text{owl:ObjectPropertChain (birthPlace Located partOf) rdfs:subPropertyOf bornIn}
\]

\[
\text{wikipedia:Albert_Einstein bornIn Europe}
\]

\[
q(?n):?p rdf:type Scientist,?p hasWon NobelPrize,?p bornIn Europe,?p hasName ?n
\]

Query answer

\[
q(G^\infty) = \{"Marie Curie", "Albert Einstein"\}
\]
Challenges raised by query answering in Linked Data

**Scalability**

- Linked Data cloud today: 9960 datasets, almost 150 billions triples (according to stats.lod2.eu)
- Almost no support for reasoning and thus very incomplete answers

⇒ Need for efficient query answering techniques involving some reasoning

**Data quality**

- Incomplete data (missing links, missing type information)
- Noisy data (some hub datasets like DBpedia or Yago are automatically generated)

⇒ Need for robust query answering and information discovery techniques

**Remaining of the talk**

A (partial) survey of recent works that have (partially) addressed some of these challenges using deductive RDF triplestores.
Deductive RDF triplestore: RDF dataset + a set of rules

Simple formalism for capturing several types of knowledge

- **RDFS entailment**
  
  \[(?i \text{ rdf:type } ?s), (\text{?s rdfs:subClassOf } ?o) \rightarrow (?i \text{ rdf:type } ?o)\]

- **(Most of) OWL constraints**
  
  \[(?p \text{ birthPlace } ?b), (\text{?b Located } ?c), (\text{?c partOf } ?d) \rightarrow (?p \text{ bornIn } ?d)\]

- **Beyond FOL constraints**
  
  \[(?p \text{ rdf:type owl:SymmetricProperty}), (?p \text{ rdfs:domain } ?c) \rightarrow (?p \text{ rdfs:range } ?c)\]

- **(Complex) mappings**
  
  \[(?p1 \text{ ina:presenter } ?v), (?v \text{ ina:title } ?t), (?p2 \text{ db:presenter } ?t) \rightarrow (?p1 \text{ owl:sameAs } ?p2)\]

- **Domain-specific rules (human embryo development)**
  
  \[(?x \text{ mycf:absence_implies } ?y), (?x \text{ mycf:depends_on } ?z) \rightarrow (?z \text{ mycf:absence_implies } ?x)\]

A Datalog operational semantics to compute \(G^\infty = \text{SAT}(D,R)\)

- **Direct correspondence with a deductive DB using a single relation** \(T\)
  
  \[(s p o) \leftrightarrow T(s p o)\]
Several instances of this generic framework

My Corporis Fabrica: an ontology-based suite of tools for combining complex anatomical models

- Rule-based interoperability between anatomical entities, human body functions and 3D graphic models
  ▶ with O. Palombi et al,
  - “My Corporis Fabrica: an ontology-based tool for reasoning and querying on complex anatomical models.”, Journal of Biomedical Semantics 2014

Module extraction from Semantic Web datasets

- Extraction of bounded-size RDF data modules enriched with rules
  ▶ with F. Ulliana, “Extracting Bounded-level Modules from Deductive RDF Triplestores.”, AAAI 2015

Rule-based Data Linkage

- Automatic discovery of same-As and DifferentFrom facts
My Corporis Fabrica and MyCF Embryo

Rule-based interoperability between anatomical entities, human body functions and 3D graphic models

⇒ a declarative approach assisting interactive simulation and visualization

Datalog revisited for reasoning in Linked Data

September 2 2016

Marie-Christine Rousset (Université de Grenoble-Alpes & Institut Universitaire de France)
Joint work with M. Al Bakri, M. Atencia, J. David, F. Jouanot, S. Lalande, O. Palombi and F. Ulliana
Module extraction from Semantic Web datasets

Reuse of relevant extracts of big reference Web knowledge bases
⇒ a coherent and modular development of the Semantic Web

Existing works

- Well studied for Description Logics
  - not applicable to RDF datasets (e.g., DBpedia, Yago)
  - generally untractable, tractable approximations
  - may output large modules: the whole Tbox in the worst case

- Little work for RDF databases
  - RDF subgraph extraction, traversal views
  - reasoning not considered
Our contribution

A novel semantics of modules adapted to deductive RDF datasets

- Module signature \((p_1, \ldots, p_n)^k[a]\) involving properties, and individual and a bound \(k\) for property paths rooted in the specified individual.
- \(\langle D_M, R_M \rangle\) is a bounded-level module of \(\langle D, R \rangle\) iff \(D_M\) and \(R_M\) are conform to the signature, \(\langle D, R \rangle \vdash \langle D_M, R_M \rangle\), and:

  \[
  D, R^{\text{NonRec}} \vdash \pi(a,b) \iff D_M, R_M \vdash \pi(a,b) \quad (1)
  \]

  \[
  D_M, R \vdash \pi(a,b) \iff D_M, R_M \vdash \pi(a,b) \quad (2)
  \]

  for every path of atoms \(\pi(a,b)\) of bounded length in the signature.

Non-recursive rules distinguished from recursive ones to avoid to waste \(k\)-parametricity.

Algorithms for module extraction

- Module data extraction expressed as a non-recursive Datalog program
- Construction of the \(R_M\) module rules by rule unfolding with a breadth-first strategy
Illustrative example

- Non recursive rules are needed to compute $D_M$
- Recursive rules must be delegated to $R_M$ (if they are conform to the signature)
Module succinctness: experiments

1. Comparison on MyCF with Traversal Views (applied to the saturated RDF dataset) and Locality-based extractor (applied to the corresponding DL ontology)

2. Impact of the properties in the signature: their number, their involvement and their interaction in (recursive) rules
Rule-based data linkage

Within a local dataset or across different datasets

⇒ Our contributions:

- **Import-by-Query**, a backward-chaining algorithm combining local reasoning and external querying to bypass local data incompleteness
- **ProbFR**, a forward-chaining algorithm for reasoning with uncertain data and rules.

▷ joint work with M. Al Bakri, M. Atencia, J. David and Steffen Lalande (from INA)
▷ AAAI2015, ECAI2016
Reasoning with local data may not be enough

<table>
<thead>
<tr>
<th></th>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>![IF formula](?p1 name ?name ?p1 birthdate ?d ?p2 name ?name ?p2 birthdate ?d)</td>
<td>![THEN formula](?p1 same_as ?p2)</td>
</tr>
<tr>
<td>R3</td>
<td>![IF formula](?p1 birthdate ?d1 ?p2 birthdate ?d2 ?d1 &lt;&gt; ?d2)</td>
<td>![THEN formula](?p1 differentFrom ?p2)</td>
</tr>
<tr>
<td>R4</td>
<td>![IF formula](?x1 same_as ?x2 ?x2 same_as ?x3)</td>
<td>![THEN formula](?x1 same_as ?x3)</td>
</tr>
<tr>
<td>R5</td>
<td>![IF formula](?x1 same_as ?x2 ?x2 differentFrom ?x3)</td>
<td>![THEN formula](?x1 differentFrom ?x3)</td>
</tr>
</tbody>
</table>

**BUT** ![inverted image](ina:per1, same_as, ina:per2) ? STILL UNKNOWN
Import-by-Query

- Build **on demand** queries to some entry points of Linked Data
- The queries should be **as instantiated as possible**.
- Alternates steps of **query rewriting** and of **distant query evaluation**.
Query rewriting by adapting Query-SubQuery

A backward-chaining algorithm developed for answering queries in Datalog

Correctness
Completeness
Optimization
Implementation

QESQ

External Rewritings of Q

DBpedia
Query rewriting (by example)

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
<th></th>
</tr>
</thead>
</table>
| r1: `<?p1 name ?name>, <?p1 birthdate ?d>,
| r2: `<?p1 name ?name>, <?p1 ina:presenter ?v1 >,
|                                         |                                         | `<ina:per2 birthdate “22/06/1933”>` |
Query rewriting (ctd)

Local facts

No match with local facts
No possible match with external facts

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
</table>

Local facts

<ina:per1 name "Jacques Martin">
<ina:per1 presenter ina:v1 >
<ina:v1 title "Le Petit Rapporteur"
<ina:per2 name "Jacques Martin">
<ina:per2 birthdate "22/06/1933"
Query rewriting (ctd)

\[
\begin{align*}
&\text{Local facts} \quad \text{Local facts} \quad \text{Local facts} \\
\text{IF} & \quad \text{THEN} \\
\begin{array}{l}
\text{r1:} \quad (?\text{p1 name ?name}, \quad ?\text{p1 birthdate ?d}, \\
\text{\quad } \quad \quad \text{?\text{p2 name ?name} \quad ?\text{p2 birthdate ?d}}) \\
\text{r2:} \quad (?\text{p1 name ?name}, \quad ?\text{p1 ina:presenter ?v1}, \\
\text{\quad } \quad \quad \text{?\text{v1 title ?t}, \quad ?\text{p2 name ?name} \quad ?\text{p2 db:presenter ?t}}) \\
\text{r3:} \quad (?\text{p3 same_as ?p2}, \quad ?\text{p3 same_as ?p1}) \\
\quad \quad \quad 9/23/2015
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{l}
<\text{ina:per1 name ?name}> \\
<\text{ina:per1 birthdate ?d}> \\
<\text{ina:per2 name ?name}> \\
<\text{ina:per2 birthdate ?d}>
\end{array} \\
<\text{r1}>
\]

\[
\begin{align*}
<\text{ina:per1 name ?name}> \\
<\text{ina:per1 presenter ?v1}> \\
<\text{?v1 title ?t}> \\
<\text{ina:per2 name ?name}> \\
<\text{ina:per2 birthdate ?d}>
\end{align*} \\
<\text{r2}>
\]

\[
\begin{align*}
<\text{r1}> \\
<\text{?p3 same_as ina:per1}> \\
<\text{?p3 same_as ina:per2}>
\end{align*}
\]

\[
\begin{align*}
<\text{r3}> \\
<\text{?p3 name ?name}> \\
<\text{?p3 birthdate ?d}>
\end{align*}
\]

\[
\begin{align*}
<\text{<p3 name “Jacques Martin”>}> \\
<\text{<p3 birthdate “22/06/1933”>}> \\
<\text{<p3 same_as ina:per1>}> \\
\end{align*}
\]

\[
\begin{align*}
<\text{ina:per1 name “Jacques Martin”}> \\
<\text{ina:per1 presenter ina:v1}> \\
<\text{ina:v1 title “Le Petit Rapporteur”}> \\
<\text{<p3 name “Jacques Martin”}> \\
<\text{<p3 birthdate “22/06/1933”}>}
\end{align*}
\]
Experiments

Conducted on a deductive RDF triplestore built with INA

- one million RDF facts (provided by INA) : RDF export and extraction of metadata from the INA catalog
- 35 rules (built with the help of INA experts)

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>r7</td>
<td>(\langle x_1, \text{foaf:name}, \text{?name1} \rangle, \langle x_2, \text{skos:altLabel}, \text{?name2} \rangle, \text{Similar(?name1, ?name2, 0.99)}</td>
</tr>
<tr>
<td>r8</td>
<td>(\langle x_1, \text{foaf:name}, \text{?name1} \rangle, \langle x_2, \text{skos:prefLabel}, \text{?name2} \rangle, \text{Similar(?name1, ?name2, 0.99)}</td>
</tr>
<tr>
<td>r9</td>
<td>(\langle x_1, \text{rdfs:label}, \text{?name1} \rangle, \langle x_2, \text{skos:prefLabel}, \text{?name2} \rangle, \text{Similar(?name1, ?name2, 0.99)}</td>
</tr>
<tr>
<td>r10</td>
<td>(\langle x_1, \text{rdfs:label}, \text{?name1} \rangle, \langle x_2, \text{skos:altLabel}, \text{?name2} \rangle, \text{Similar(?name1, ?name2, 0.99)}</td>
</tr>
<tr>
<td>r11</td>
<td>(\langle x_1, \text{prop-fr:nom}, \text{?name1} \rangle, \langle x_2, \text{skos:prefLabel}, \text{?name2} \rangle, \text{Similar(?name1, ?name2, 0.99)}</td>
</tr>
<tr>
<td>r12</td>
<td>(\langle x_1, \text{prop-fr:nom}, \text{?name1} \rangle, \langle x_2, \text{skos:altLabel}, \text{?name2} \rangle, \text{Similar(?name1, ?name2, 0.99)}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>r13</td>
<td>(\langle x_1, \text{ina:sameNameDbp}, x_2 \rangle, \langle x_1, \text{dbpedia:birthYear}, Y_1 \rangle, \langle x_2, \text{ina:birthYear}, Y_2 \rangle) \langle x_1, \text{ina:sameAs}, x_2 \rangle</td>
</tr>
<tr>
<td>r14</td>
<td>(\langle x_1, \text{ina:sameNameDbp}, x_2 \rangle, \langle x_1, \text{dbpedia:birthYear}, Y_1 \rangle, \langle x_2, \text{ina:birthYear}, Y_2 \rangle, \text{notEqual(Y_1, Y_2)})</td>
</tr>
<tr>
<td>r15</td>
<td>(\langle x_1, \text{ina:sameNameDbp}, x_2 \rangle, \langle x_1, \text{dbpedia:deathYear}, Y_1 \rangle, \langle x_2, \text{ina:deathYear}, Y_2 \rangle, \text{notEqual(Y_1, Y_2)})</td>
</tr>
</tbody>
</table>
Results

- External information in Linked Data is useful for disambiguation

<table>
<thead>
<tr>
<th></th>
<th>sameAs</th>
<th>DifferentFrom</th>
</tr>
</thead>
<tbody>
<tr>
<td>35 rules ina</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>35 rules ina Dp</td>
<td>4884</td>
<td>9764</td>
</tr>
</tbody>
</table>

- Full reasoning on (recursive) rules is useful
  - Comparison between Silk and a forward reasoner applied to our rules
    Silk only discovered 3% of the sameAs links discovered by our approach
  - 100% precision by construction (if the rules and the facts are correct)
    - checked in practice on a sample of 500 links

- Import-by-Query brings a drastic reduction of the imported facts

<table>
<thead>
<tr>
<th>Number of Imported Facts for a sample of 500 Boolean queries</th>
<th>Import By Query</th>
<th>Forward Reasoner</th>
</tr>
</thead>
<tbody>
<tr>
<td>6,417 facts (13 per Boolean query)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500,000 facts</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Import-by-Query requires 3 iterations of rewritings on average

| Time to answer a boolean query after fact propagation | 7 seconds |
| Time to answer a boolean query without fact propagation | 186 seconds |
| Time to propagate facts (done once for all queries) | 191 seconds |
| Gain of doing fact propagation beforehand for answering the 500 reference queries using import-by-query | 96% |
ProbFR: Probabilistic Forward Reasoner

Unifying modeling of any kind of uncertainty as probabilities
- noisy data (e.g., due to automatic data extraction from WikiPedia)
- pseudo-keys, constraints with exceptions
- weighted mappings between vocabularies across datasets

Operational semantics of Probabilistic Datalog
- extension of probabilistic databases
- each input fact and rule is associated with a symbolic event
- an event expression is computed for each inferred fact, that encapsulates its provenance
- the probabilities are computed from the event expressions

ProbFR implemented on top of JENA RETE
Linkage between MusicBrainz and DBpedia using ProbFR

- MusicBrainz: 112 millions triples (12 GB)
- DBpedia (extract on songs, bands and persons): 73 millions triples
- 20 certain rules, 36 uncertain rules (probabilities from 0.3 to 0.9)
  - Runtime performance: less than 2 hours in total (including the loading time and the use of SOLR to compute some built-in predicates)
  - Impact of using uncertain information:
    - Precision and recall based on certain rules only
    - Precision and recall based on all the rules
    - Precision and recall after filtering the inferred facts with a probability over a threshold

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>100%</td>
<td>8%</td>
</tr>
<tr>
<td>Band</td>
<td>100%</td>
<td>12%</td>
</tr>
<tr>
<td>Song</td>
<td>94%</td>
<td>74%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>100%</td>
<td>80%</td>
</tr>
<tr>
<td>Band</td>
<td>94%</td>
<td>84%</td>
</tr>
<tr>
<td>Song</td>
<td>94%</td>
<td>74%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Band $\geq 0.9$</td>
<td>100%</td>
<td>80%</td>
</tr>
<tr>
<td>Song $\geq 0.9$</td>
<td>100%</td>
<td>44%</td>
</tr>
</tbody>
</table>
Conclusion

Semantic Web standards, data and applications are there

Linked Data is flourishing due to the simplicity and flexibility of the RDF data model.

However many challenges remain

- Efficient Semantic Web data and knowledge management is still challenging.
- Novel problems arise to handle at large scale the incomplete and uncertain nature of Web data

Our message:

(Extensions of) Datalog on top of RDF datasets is an interesting angle of attack for many of these challenges