How to know how much we know

Towards a completeness-aware Semantic Web

Luis Galárraga

November 16th, 2017
Data Science Seminar @ LTCI
Outline

• Completeness in RDF knowledge bases
• State of the art on completeness
• Completeness oracles
• Vision on Completeness-aware Semantic Web
  – Representations for completeness oracles
  – Reasoning with completeness oracles
  – Enabling completeness in SPARQL
• Summary & conclusions
Outline

- Completeness in RDF knowledge bases
- State of the art on completeness
- Completeness oracles
- Vision on Completeness-aware Semantic Web
  - Representations for completeness oracles
  - Reasoning with completeness oracles
  - Enabling completeness in SPARQL
- Summary & conclusions
Outline

- Completeness in RDF knowledge bases
- State of the art on completeness
- Completeness oracles
- Vision on Completeness-aware Semantic Web
  - Representations for completeness oracles
  - Reasoning with completeness oracles
  - Enabling completeness in SPARQL
- Summary & conclusions
RDF Knowledge Bases (KBs)

Collection of structured knowledge

- Romance family
- Français
- Italiano
- Switzerland
- citizenOf
- Leonhard Euler

officialLanguage
RDF Knowledge Bases (KBs)

RDF KBs can be queried using SPARQL

```sparql
SELECT ?x WHERE {
  Switzerland officialLanguage ?x .
  ?x family Romance .
}
```
RDF Knowledge Bases (KBs)

RDF KBs can be queried using SPARQL

```sparql
SELECT ?x WHERE {
  Switzerland officialLanguage ?x .
  ?x family Romance .
}
```
RDF Knowledge Bases (KBs)

RDF KBs can be queried using SPARQL

```
SELECT ?x WHERE {
  Switzerland officialLanguage ?x .
  ?x family Romance .
}
```
RDF Knowledge Bases (KBs)

RDF KBs can be queried using SPARQL

SELECT ?x WHERE {
    Switzerland officialLanguage ?x .
    ?x family Romance .
}

Leonhard Euler
Plenty of KBs out there!
Plenty of KBs out there!
KBs in action
Outline

• Completeness in RDF knowledge bases
• State of the art on completeness
• Completeness oracles
• Vision on Completeness-aware Semantic Web
  – Representations for completeness oracles
  – Reasoning with completeness oracles
  – Enabling completeness in SPARQL
• Summary & conclusions
Completeness in RDF KBs

- KBs are highly incomplete
  - 1% of people have a citizenship in YAGO
Completeness in RDF KBs

- KBs are highly incomplete
  - 1% of people have a citizenship in YAGO
- We do not know where the incompleteness lies
Completeness in RDF KBs

• KBs are highly incomplete
  – 1% of people have a citizenship in YAGO

• We do not know where the incompleteness lies
  – A single person in the KB could be actually single or the KB may be incomplete
Completeness in RDF KBs

• KBs are highly incomplete
  – 1% of people have a citizenship in YAGO

• We do not know where the incompleteness lies
  – A single person in the KB could be actually single or the
    KB may be incomplete

• Problems for data producers and consumers
Completeness in RDF KBs

• KBs are highly incomplete
  – 1% of people have a citizenship in YAGO

• We do not know where the incompleteness lies
  – A single person in the KB could be actually single or the KB may be incomplete

• Problems for data producers and consumers
  – Consumers: no completeness guarantees for queries.
  – Producers: which parts of the KB need to be populated?
Completeness in RDF KBs
Completeness in RDF KBs

This list of results is complete!
Completeness

- Defined with respect to a query $q$ via a complete hypothetical KB $K^*$. 
Completeness

- Defined with respect to a query $q$ via a complete hypothetical KB $K^*$. 
  - A query $q$ is complete in $K$, iff $q(K^*) \subseteq q(K)$. 

Completeness

- Defined with respect to a query $q$ via a complete hypothetical KB $K^*$. 
  - A query $q$ is complete in $K$, iff $q(K^*) \subseteq q(K)$.

SELECT ?x WHERE { Switzerland officialLanguage ?x }
Completeness

- Defined with respect to a query $q$ via a complete hypothetical KB $K^*$.
- A query $q$ is complete in $K$, iff $q(K^*) \subseteq q(K)$.

SELECT ?x WHERE { Switzerland officialLanguage ?x }
Completeness

- Defined with respect to a **query q** via a complete hypothetical KB $K^*$.
  - A query $q$ is complete in $K$, iff $q(K^*) \subseteq q(K)$.

```
SELECT ?x WHERE { Switzerland officialLanguage ?x }
```

Are these all the official languages of Switzerland?

[Incomplete query]
Outline

- Completeness in RDF knowledge bases
- State of the art on completeness
- Completeness oracles
- Vision on Completeness-aware Semantic Web
  - Representations for completeness oracles
  - Reasoning with completeness oracles
  - Enabling completeness in SPARQL
- Summary & conclusions
Completeness in RDF data

Wikidata keeps lists of subject-relation pairs with missing values.

George of Trebizond

placeOfBirth
Completeness in RDF data

Wikidata keeps lists of subject-relation pairs with missing values.

SELECT ?x WHERE { George of Trebizond placeOfBirth ?x }

George of Trebizond
Completeness in RDF data

Wikidata keeps lists of subject-relation pairs with missing values.

SELECT ?x WHERE { George of Trebizond placeOfBirth ?x }

[Incomplete query]
Completeness in RDF data

- Wikidata also provides *no value annotations*
Completeness in RDF data

- Wikidata also provides *no value annotations*

  ```sql
  SELECT ?x WHERE { USA officialLanguage ?x }
  ```
Completeness in RDF data

- Wikidata also provides *no value annotations*

```
SELECT ?x WHERE { USA officialLanguage ?x }
```

[Complete query]
Completeness in RDF data

- Wikidata also provides *no value annotations*

  ```
  SELECT ?x WHERE { USA officialLanguage ?x }
  ```

- Not applicable if we know some official language
Completeness in RDF data

- Wikidata also provides *no value annotations*

  ```sql
  SELECT ?x WHERE { USA officialLanguage ?x }
  ```

  ![Complete query]

- Not applicable if we know some official language

  ![Diagram with languages and flags]
Completeness in RDF data

- [S. Razniewski, W. Nutt, 2011]
  - Completeness formulation, table & query completeness, complexity analysis.
  - Reasoning over incomplete databases, TC-TC & TC-QC
- [X. Dong et al., 2014]
  - 71% of people in Freebase does not have a place of birth
- [F. Darari et al., 2013], [F. Darari et al., 2016]
  - Reasoning with RDF completeness statements and the available data.
Completeness in RDF data

- [E. Muñoz, M. Nickels, 2017]
  - Mine cardinalities for object values in order to assess completeness in KBs.

- [T. P. Tanon et al., 2017]
  - Obtain cardinality estimations to generate completeness statements to better assess the quality of rules learned from KBs.
Outline

- Completeness in RDF knowledge bases
- State of the art on completeness
- Completeness oracles [Our contribution]
- Vision on Completeness-aware Semantic Web
  - Representations for completeness oracles
  - Reasoning with completeness oracles
  - Enabling completeness in SPARQL
- Summary & conclusions
Completeness oracle

- Boolean function $\omega(q, K)$ that guesses the completeness of a query $q$ in a KB $K$. 
SR completeness oracle

- [Galárraga et. al., 2017] Function $\omega$ that guesses the completeness of queries of the form:

  SELECT ?x WHERE { subject relation ?x }
SR completeness oracle

• [Galárraga et. al., 2017] Function \( \omega \) that guesses the completeness of queries of the form:

\[
\text{SELECT } ?x \text{ WHERE } \{ \text{subject relation } ?x \} \]

• We use the notation \( \omega(subject, relation) \)
SR completeness oracle

- [Galárraga et. al., 2017] Function $\omega$ that guesses the completeness of queries of the form:

  \[
  \text{SELECT } ?x \text{ WHERE } \{ \text{subject relation } ?x \}\]

- We use the notation $\omega(subject, relation)$

- $\omega = pca(s, r) = \text{partial completeness assumption}$
SR completeness oracle

- [Galárraga et. al., 2017] Function $\omega$ that guesses the completeness of queries of the form:

  $$\text{SELECT } ?x \text{ WHERE } \{ \text{subject relation } ?x \}$$

- We use the notation $\omega(subject, relation)$

- $\omega = pca(s, r) = \text{partial completeness assumption}$
  - Query is \textbf{complete} in KB if at least one answer is known
Evaluating SR oracles

\( \omega = pca(s, r) = \) partial completeness assumption

**Gold standard:**
Complete instances in the domain of `officialLanguage`
Evaluating SR oracles

ω = pca(s, r) = partial completeness assumption

Gold standard: Complete instances in the domain of officialLanguage
 Evaluating SR oracles

\[ \omega = \text{american-country-oracle}(s, r) \]
Evaluating SR oracles

**PCA oracle**
- Precision = 3/5
- Recall = 3/4

**American country oracle**
- Precision = 1/2
- Recall = 1/4

*Gold standard:*
Complete instances in the domain of official Language
SR completeness oracles

- Closed World Assumption: \( \text{cwa}(s, r) = \text{true} \)
- PCA: \( \text{pca}(s, r) = \exists o : r(s, o) \)
- Cardinality: \( \text{card}(s, r) = \#(o : r(s, o)) \geq k \)
- Popular entities: \( \text{popularity}_{\text{pop}}(s, r) = \text{pop}(s) \)
- No-chg over time: \( \text{nochange}_{\text{chg}}(s, r) = \sim \text{chg}(s, r) \)
- Star: \( \text{star}_{r_1,\ldots,r_n}(s, r) = \forall i \in \{1,\ldots,n\} : \exists o : r_i(s, o) \)
- Class: \( \text{class}_c(s, r) = \text{type}(s, c) \)
- Rule mining oracle
Rule mining SR oracle

- Based on completeness rules

\[
\text{notype}(x, \text{Adult}), \text{type}(x, \text{Person}) \Rightarrow \text{complete}(x, \text{hasChild})
\]
\[
\text{dateOfDeath}(x, y), \text{lessThan}_1(x, \text{placeOfDeath}) \Rightarrow \text{incomplete}(x, \text{placeOfDeath})
\]
Rule mining SR oracle

• Based on completeness rules
  
  \[
  \text{notype}(x, \text{Adult}), \text{type}(x, \text{Person}) \Rightarrow \text{complete}(x, \text{hasChild}) \\
  \text{dateOfDeath}(x, y), \text{lessThan}_1(x, \text{placeOfDeath}) \Rightarrow \text{incomplete}(x, \text{placeOfDeath})
  \]

• Learned using the AMIE [Galárraga et. al, 2013] rule mining system
  
  – On gold standard built via crowdsourcing
Rule mining SR oracle

- Based on completeness rules

\[
\text{notype}(x, \text{Adult}), \text{type}(x, \text{Person}) \Rightarrow \text{complete}(x, \text{hasChild}) \\
\text{dateOfDeath}(x, y), \text{lessThan}_1(x, \text{placeOfDeath}) \Rightarrow \text{incomplete}(x, \text{placeOfDeath})
\]

- Learned using the AMIE [Galárraga et. al, 2013] rule mining system
  - On gold standard built via crowdsourcing
  - 100% F1-measure for functional relations, quite good for relations \textit{hasChild, graduatedFrom}
Performance of SR oracles

F1 measure of the oracles in YAGO3

<table>
<thead>
<tr>
<th>Relation</th>
<th>CWA</th>
<th>PCA</th>
<th>Class</th>
<th>AMIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>diedIn</td>
<td>60%</td>
<td>22%</td>
<td>99%</td>
<td>96%</td>
</tr>
<tr>
<td>directed</td>
<td>40%</td>
<td>96%</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>graduatedFrom</td>
<td>89%</td>
<td>4%</td>
<td>92%</td>
<td>87%</td>
</tr>
<tr>
<td>hasChild</td>
<td>71%</td>
<td>1%</td>
<td>78%</td>
<td>78%</td>
</tr>
<tr>
<td>hasGender</td>
<td>78%</td>
<td>100%</td>
<td>95%</td>
<td>100%</td>
</tr>
<tr>
<td>hasParent</td>
<td>1%</td>
<td>54%</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>isCitizenOf</td>
<td>4%</td>
<td>98%</td>
<td>5%</td>
<td>100%</td>
</tr>
<tr>
<td>isConnectedTo</td>
<td>87%</td>
<td>34%</td>
<td>88%</td>
<td>89%</td>
</tr>
<tr>
<td>isMarriedTo</td>
<td>55%</td>
<td>7%</td>
<td>57%</td>
<td>46%</td>
</tr>
<tr>
<td>wasBornIn</td>
<td>28%</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
</tr>
</tbody>
</table>
Outline

- Completeness in RDF knowledge bases
- State of the art on completeness
- Completeness oracles
- Vision on Completeness-aware Semantic Web
  - Representations for completeness oracles
  - Reasoning with completeness oracles
  - Enabling completeness in SPARQL
- Summary & conclusions
Outline

• Completeness in RDF knowledge bases
• State of the art on completeness
• Completeness oracles

• Vision on Completeness-aware Semantic Web
  – Representations for completeness oracles
  – Reasoning with completeness oracles
  – Enabling completeness in SPARQL

• Summary & conclusions
Representing completeness oracles

• Extensional approach [Darari, et al, 2013]
  – An oracle is a collection of completeness statements about queries
Representing completeness oracles

- Extensional approach [Darari, et al, 2013]
  - An oracle is a collection of completeness statements about queries

```
SELECT DISTINCT ?y WHERE { ?x hasOfficialLanguage ?y }
```
is complete in the KB
Representing completeness oracles

- Extensional approach [Darari, et al, 2013]
  - An oracle is a collection of completeness statements about queries

**SELECT DISTINCT ?y WHERE { ?x hasOfficialLanguage ?y }** is **complete** in the KB
Representing completeness oracles

- Extensional approach [Darari, et al, 2013]
  - A call to the oracle asks for the existence of the query in the graph

SELECT DISTINCT ?y WHERE { ?x hasOfficialLanguage ?y } is complete in the KB
Representing completeness oracles

• Intensional approach
  – The oracle logic is embedded as a lambda function or a link to a program or resource
Representing completeness oracles

- Intensional approach
  - The oracle logic is embedded as a lambda function or a link to a program or resource

$$\exists o : \text{isCitizenOf}(s, o)$$

<table>
<thead>
<tr>
<th>pca-citizenship</th>
<th>SR-Oracle</th>
<th>amie-oracle</th>
<th>RM-Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>precision</td>
<td>address</td>
<td><a href="http://example.org/rest/oracle">http://example.org/rest/oracle</a></td>
<td></td>
</tr>
<tr>
<td>96%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$\text{hasFormula}$$
Providing completeness guarantees

This list of results is complete with confidence X according to ω
Providing completeness guarantees

countries with official romance languages

Romance-speaking Europe - Wikipedia
https://en.wikipedia.org/wiki/Romance-speaking_Europe  ▼ Traducir esta página
Pasar a Countries - Countries / Territories, Romance languages used, Religion, Area ... it has a sole official language of English even though the vast ...
Providing completeness guarantees

SELECT ?country WHERE {
  ?lang family Romance .
}
Providing completeness guarantees

SELECT ?country WHERE {
  ?lang family Romance .
}
Outline

- Completeness in RDF knowledge bases
- State of the art on completeness
- Completeness oracles
- Vision on Completeness-aware Semantic Web
  - Representations for completeness oracles
  - Reasoning with completeness oracles
  - Enabling completeness in SPARQL
- Summary & conclusions
D completeness oracles

- Oracle $\omega_d$ for the completeness of queries:

  SELECT DISTINCT ?x WHERE { ?x relation ?y }
  SELECT DISTINCT ?y WHERE { ?x relation ?y }
D completeness oracles

- Oracle $\omega_d$ for the completeness of queries:
  
  SELECT DISTINCT ?x WHERE { ?x relation ?y }
  SELECT DISTINCT ?y WHERE { ?x relation ?y }

- We use the notation $\omega_d(relation)$ or $\omega_d(relation^{-1})$
  
  SELECT DISTINCT ?y WHERE { ?x officialLanguage ?y }
D completeness oracles

• Oracle $\omega_d$ for the completeness of queries:

  SELECT DISTINCT ?x WHERE { ?x relation ?y }
  SELECT DISTINCT ?y WHERE { ?x relation ?y }

• We use the notation $\omega_d(\text{relation})$ or $\omega_d(\text{relation}^{-1})$

  SELECT DISTINCT ?y WHERE { ?x officialLanguage ?y }

• If $\omega_d(\text{officialLanguage})$ returns true, $\omega_d$ states that the KB knows all languages that are official in some country
Completeness guarantees for arbitrary queries

- Write completeness annotations for every possible type of query
  - It requires a large amount of effort
Completeness guarantees for arbitrary queries

- Write completeness annotations for every possible type of query
  - It requires a large amount of effort
- Reuse existing SR and D oracles
Completeness guarantees for arbitrary queries

• Write completeness annotations for every possible type of query
  – It requires a large amount of effort

• Reuse existing SR and D oracles

```sql
SELECT ?country WHERE {
  ?lang family Romance .
}
```
Completeness guarantees for arbitrary queries

• Write completeness annotations for every possible type of query
  – It requires a large amount of effort

• Reuse existing SR and D oracles

```sql
SELECT ?country WHERE {
  ?lang family Romance .
}
```

$\omega' =$
Completeness guarantees for arbitrary queries

• Write completeness annotations for every possible type of query
  – It requires a large amount of effort

• Reuse existing SR and D oracles

```
SELECT ?country WHERE {
  ?lang family Romance .
}
```

\( \omega' = \omega(\text{Romance}, \text{family}^{-1}) \)
Completeness guarantees for arbitrary queries

- Write completeness annotations for every possible type of query
  - It requires a large amount of effort

- Reuse existing SR and D oracles

    \[
    \omega' = \omega(\text{Romance, family}^{-1}) \land (\bigwedge_{l: \text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1}))
    \]

    SQL:
    
    ```sql
    SELECT ?country WHERE {
        ?lang family Romance .
    }
    ```
Completeness guarantees for arbitrary queries

- Write completeness annotations for every possible type of query
  - It requires a large amount of effort

- Reuse existing SR and D oracles

\[
\omega' = \omega(\text{Romance, family}^{-1}) \land (\land_{l:\text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1}))
\]

\[
\text{SELECT } ?\text{country WHERE } \{ \\
\quad ?\text{country officialLanguage } ?\text{lang} . \\
\quad ?\text{lang family Romance} . \\
\}\]

It will generate false negatives
Completeness guarantees for arbitrary queries

• Write completeness annotations for every possible type of query
  – It requires a large amount of effort

• Reuse existing SR and D oracles

\[
\omega' = \omega(\text{Romance, family}^{-1}) \land (\bigwedge_{l:\text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1}))
\]

If the KB misses Ligurian, this term returns false

It will generate false negatives

SELECT ?country WHERE {
  ?lang family Romance .
}

It will generate false negatives
Completeness guarantees for arbitrary queries

- Write completeness annotations for every possible type of query
  - It requires a large amount of effort
- Reuse existing SR and D oracles

```
SELECT ?country WHERE {
  ?lang family Romance .
}
```

\( \omega' = \omega(\text{Romance, family}^{-1}) \land (\land_{l:\text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1})) \)
Completeness guarantees for arbitrary queries

- Multiple oracle expressions can offer completeness guarantees for a query.
Completeness guarantees for arbitrary queries

- Multiple oracle expressions can offer completeness guarantees for a query.

SELECT ?country WHERE {
  ?lang family Romance .
}

ω₁ = ω(Romance, family⁻¹) \land (\land_{l:family(l, Romance)} \omega(l, officialLanguage⁻¹))

ω² = ω(Romance, family⁻¹) \land (\land_{l:family(l, f)} \omega(l, officialLanguage⁻¹))

ω³ = ω(Romance, family⁻¹) \land \omega_d(officialLanguage) \land (\land_{c:officialLanguage(c, l)} \omega(c, officialLanguage))
Tightness for completeness guarantees

- $\omega' <_q \omega''$ for $q$ if $\forall K : \omega''(q, K) \land \omega'(q, K) :
  - \omega''(q, K') \Rightarrow \omega'(q, K') \forall K' \subseteq K.$

SELECT ?country WHERE {
  ?lang family Romance .
}

$\omega^1 = \omega(\text{Romance, family}^{-1}) \land (\bigwedge_{l: \text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1}))$

$\omega^2 = \omega(\text{Romance, family}^{-1}) \land (\bigwedge_{l: \text{family}(l, f)} \omega(l, \text{officialLanguage}^{-1}))$

$\omega^3 = \omega(\text{Romance, family}^{-1}) \land \omega_d(\text{officialLanguage}) \land (\bigwedge_{c: \text{officialLanguage}(c, l)} \omega(c, \text{officialLanguage}))$
Tightness for completeness guarantees

- $\omega' <_q \omega''$ for $q$ if $\forall K : \omega''(q, K) \land \omega'(q, K) :
  - \omega''(q, K') \Rightarrow \omega'(q, K') \forall K' \subseteq K.$

$$
\omega^1 = \omega(\text{Romance, family}^-1) \land (\land_l \text{family}(l, \text{Romance}) \land \omega(l, \text{officialLanguage}^-1))
$$

$$
\omega^2 = \omega(\text{Romance, family}^-1) \land (\land_l \text{family}(l, f) \land \omega(l, \text{officialLanguage}^-1))
$$

$$
\omega^3 = \omega(\text{Romance, family}^-1) \land \omega_d(\text{officialLanguage}) \land (\land_c \text{officialLanguage}(c, l) \land \omega(c, \text{officialLanguage}))
$$
Cost for completeness guarantees

- Number of oracle calls required for the answer

\[
\begin{align*}
\omega^1 &= \omega(\text{Romance, family}^{-1}) \land (\land_{l: \text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1})) \\
\omega^2 &= \omega(\text{Romance, family}^{-1}) \land (\land_{l: \text{family}(l, f)} \omega(l, \text{officialLanguage}^{-1})) \\
\omega^3 &= \omega(\text{Romance, family}^{-1}) \land \omega_d(\text{officialLanguage}) \land (\land_{c: \text{officialLanguage}(c, l)} \omega(c, \text{officialLanguage}))
\end{align*}
\]
Cost for completeness guarantees

• Number of oracle calls required for the answer

\[
\begin{align*}
\omega^1 &= \omega(\text{Romance}, \text{family}^{-1}) \land (\land_{l:\text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1})) \\
\omega^2 &= \omega(\text{Romance}, \text{family}^{-1}) \land (\land_{l:\text{family}(l, f)} \omega(l, \text{officialLanguage}^{-1})) \\
\omega^3 &= \omega(\text{Romance}, \text{family}^{-1}) \land \omega_d(\text{officialLanguage}) \land (\land_{c:\text{officialLanguage}(c, l)} \omega(c, \text{officialLanguage}))
\end{align*}
\]

\[
\text{SELECT } ?\text{country} \text{ WHERE } \{
\text{officialLanguage } ?\text{lang} .
\}
\]

\[
\text{cost}(\omega^1) = 1 + (\#l: \text{family}(l, \text{Romance})) \text{ Romance} .
\]

\[
\text{cost}(\omega^1) = 1 + (\#l: \text{family}(l, \text{Romance})) \text{ Romance} .
\]
SELECT ?country WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?lang family Romance .
}
Automatic oracle composition

Projection variable

```
SELECT ?country WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?lang family Romance .
}
```
Automatic oracle composition

SELECT ?country WHERE {
  ?country monarch ?monarch .
  ?country locatedIn Europe .
  ?lang family Romance .
}
Automatic oracle composition

SELECT ?country WHERE {
  ?country monarch ?monarch .
  ?country locatedIn Europe .
  ?lang family Romance .
}
Automatic oracle composition

SELECT ?country WHERE {
  ?country monarch ?monarch .
  ?country locatedIn Europe .
  ?lang family Romance .
}
Automatic oracle composition

SELECT ?country WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?lang family Romance .
}
Automatic oracle composition

SELECT ?country WHERE {
  ?country monarch ?monarch .
  ?country locatedIn Europe .
  ?lang family Romance .
}

1. $\omega_d(\text{monarch}) \land (\land_{\text{country}} \omega(\text{country, monarch}))$
2. $\omega_d(\text{monarch}^{-1}) \land (\land_{\text{country}} \omega(\text{country, monarch}^{-1}))$
Automatic oracle composition

SELECT ?country WHERE {
  ?country monarch ?monarch .
  ?country locatedIn Europe .
  ?lang family Romance .
}
Automatic oracle composition

\[ \omega_d(\text{monarch}) \land \\
(\land_{\text{country}} \omega(\text{country}, \text{monarch})) \\
\omega(\text{Europe}, \text{locatedIn}^{-1}) \]

```
SELECT ?country WHERE {
  ?country monarch ?monarch .
  ?country locatedIn Europe .
  ?lang family Romance .
}
```
Automatic oracle composition

\[ \ominus_d(\text{monarch}) \land \\
( \land_{\text{country}} \ominus(\text{country}, \text{monarch})) \\
\ominus(\text{Europe}, \text{locatedIn}^{-1}) \]

SELECT ?country WHERE {
  ?country --> monarch ?monarch .
  ?country --> locatedIn Europe .
  ?lang --> family Romance .
}
Automatic oracle composition

\[ \omega_d(\text{monarch}) \land (\land_{\text{country}} \omega(\text{country, monarch})) \land (\land_{\text{Europe}} \omega(\text{Europe, locatedIn}^{-1})) \land (\land_{\text{Romance}} \omega(\text{Romance, family}^{-1})) \]

SELECT ?country WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?lang family Romance .
}
Automatic oracle composition

\[
\omega_d(\text{monarch}) \land (\land_{\text{country}} \omega(\text{country, monarch}))
\]

\[
\omega(\text{Europe, locatedIn}^{-1})
\]

\[
\omega(\text{Romance, family}^{-1})
\]

SELECT ?country WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?lang family Romance .
}
Automatic oracle composition

\[ \lor_{d}(\text{monarch}) \land \\
(\land_{\text{country}} \lor(\text{country}, \text{monarch})) \\
\lor(\text{Europe}, \text{locatedIn}^{-1}) \]

\[ \land_{l : \text{family}(l, \text{Romance})} \lor(l, \text{officialLanguage}^{-1}) \]

\[ \lor(\text{Romance}, \text{family}^{-1}) \]

SELECT ?country WHERE {
?country monarch ?monarch .
?country locatedIn Europe .
?lang family Romance .
}

95
Automatic oracle composition

\[
\omega' = \omega_{\text{d}}(\text{monarch}) \land (\bigwedge_{\text{country}} \omega(\text{country, monarch})) \land \omega(\text{Europe, locatedIn}^{-1}) \land \\
\omega(\text{Romance, family}^{-1}) \land (\bigwedge_{l : \text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1}))
\]

\[
\omega = \bigwedge_{l : \text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1}) \\
\omega_{\text{d}}(\text{monarch}) \land \\
(\bigwedge_{\text{country}} \omega(\text{country, monarch})) \\
\omega(\text{Europe, locatedIn}^{-1}) \\
\omega(\text{Romance, family}^{-1}) \\
\omega(\text{officialLanguage}^{-1}) \\
\]

SELECT ?country WHERE {
?country monarch ?monarch .
?country locatedIn Europe .
?lang family Romance .
}
Automatic oracle composition

\[ \omega' = \omega_d(\text{monarch}) \land (\land_{\text{country}} \omega(\text{country}, \text{monarch})) \land \omega(\text{Europe}, \text{locatedIn}^{-1}) \land \omega(\text{Romance}, \text{family}^{-1}) \land (\land_{l : \text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1})) \]

Under bag semantics, the projection variable is irrelevant

\[
\begin{align*}
\Lambda_{l : \text{family}(l, \text{Romance})} (l, \text{officialLanguage}^{-1}) \\
\omega(\text{Romance}, \text{family}^{-1})
\end{align*}
\]

SELECT ?country WHERE {
?country monarch ?monarch .
?country locatedIn Europe .
?lang family Romance .
}
Automatic oracle composition

\[\omega'' = \omega_d(\text{monarch}) \land (\land_{\text{country}} \omega(\text{country}, \text{monarch})) \land \omega(\text{Europe}, \text{locatedIn}^{-1}) \land \omega(\text{Romance}, \text{family}^{-1}) \land (\land_{c : \text{locatedIn}(c, \text{Europe}), \text{monarch}(c, x)} \omega(l, \text{officialLanguage}^{-1}))\]
Automatic oracle composition

\[ \varnothing_d(\text{monarch}) \land (\land_c \varnothing(\text{country}, \text{monarch})) \land \varnothing(\text{Europe}, \text{locatedIn}^{-1}) \land (\land_c \varnothing(\text{ofcialLanguage}^{-1})) \land \varnothing(\text{Romance}, \text{family}^{-1}) \]

\[\varnothing'' = \varnothing_d(\text{monarch}) \land (\land_c \varnothing(\text{country}, \text{monarch})) \land \varnothing(\text{Europe}, \text{locatedIn}^{-1}) \land (\land_c \varnothing(\text{ofcialLanguage}^{-1})) \land \varnothing(\text{Romance}, \text{family}^{-1}) \land (\land_c \varnothing(\text{locatedIn}(c, \text{Europe}), \text{monarch}(c, x)) \varnothing(l, \text{ofcialLanguage}^{-1}))\]

SELECT ?country WHERE {
  ?country monarch ?monarch .
  ?country locatedIn Europe .
  ?lang family Romance .
}

Challenge:
Evaluate first the oracles with smaller cost!
Automatic oracle composition

\[ \omega_d(\text{monarch}) \land (\land_{\text{country}} \omega(\text{country, monarch})) \land \omega(\text{Europe, locatedIn}^{-1}) \land \omega(\text{Romance, family}^{-1}) \]

\[ \land_{c: \text{locatedIn(c, Europe), monarch(c, x)}} \omega(l, \text{officialLanguage}^{-1}) \]

SELECT DISTINCT ?lang WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?lang family Romance .
}

Projection variable matters under set semantics
Outline

- Completeness in RDF knowledge bases
- State of the art on completeness
- Completeness oracles
- Vision on Completeness-aware Semantic Web
  - Representations for completeness oracles
  - Reasoning with completeness oracles
  - Enabling completeness in SPARQL
- Summary & conclusions
Enabling completeness in SPARQL

- Calls to completeness oracles could be embedded in the query language
Enabling completeness in SPARQL

• Calls to completeness oracles could be embedded in the query language
  – Example: aggregated number of Spanish speakers in a county per state, only for those states with complete information
Enabling completeness in SPARQL

- Calls to completeness oracles could be embedded in the query language
  - Example: aggregated number of Spanish speakers in a county per state, only for those states with complete information

SELECT ?state sum(?nspeak) WHERE {
} GROUP BY ?state HAVING (complete(?nspeak))
Enabling completeness in SPARQL

- Calls to completeness oracles could be embedded in the query language
  - Example: aggregated number of Spanish speakers in a county per state, only for *those states with complete information*

```
SELECT ?state sum(?nspeak) WHERE {
} GROUP BY ?state HAVING (complete(?nspeak))
```
Enabling completeness in SPARQL

- For each value of ?state check if the bindings for ?nspeak are complete

<table>
<thead>
<tr>
<th>?state</th>
<th>?county</th>
<th>?nspeak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delaware</td>
<td>New Castle</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>Kent</td>
<td>4300</td>
</tr>
<tr>
<td></td>
<td>Sussex</td>
<td>1200</td>
</tr>
<tr>
<td>Hawaii</td>
<td>Hawaii</td>
<td>30000</td>
</tr>
<tr>
<td></td>
<td>Kalawao</td>
<td>1200</td>
</tr>
</tbody>
</table>

SELECT ?state sum(?nspeak) WHERE {
} GROUP BY ?state HAVING (complete(?nspeak))
Enabling completeness in SPARQL

- For each value of `?state` check if the bindings for `?nspeak` are complete

<table>
<thead>
<tr>
<th>?state</th>
<th>?county</th>
<th>?nspeak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delaware</td>
<td>New Castle</td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>Kent</td>
<td>4300</td>
</tr>
<tr>
<td></td>
<td>Sussex</td>
<td>1200</td>
</tr>
<tr>
<td>Hawaii</td>
<td>Hawaii</td>
<td>30000</td>
</tr>
<tr>
<td></td>
<td>Kalawao</td>
<td>1200</td>
</tr>
</tbody>
</table>

```
SELECT `complete`(?nspeak) WHERE {
    ?county inState `Delaware` .
}
```

```
SELECT ?state SUM(?nspeak) WHERE {
} GROUP BY ?state HAVING (`complete`(?nspeak))
```
Enabling completeness in SPARQL

- For each value of \(?state\) check if bindings for \(?nspeak\) are complete

```
SELECT \texttt{complete(?nspeak)} WHERE {
  ?county inState \texttt{Delaware} .
}
```

SELECT \(?state\) \texttt{sum(?nspeak)} WHERE {
  ?county inState \(?state\) .
} GROUP BY \(?state\) HAVING (\texttt{complete(?nspeak)})
Outline

- Completeness in RDF knowledge bases
- State of the art on completeness
- Completeness oracles
- Vision on Completeness-aware Semantic Web
  - Representations for completeness oracles
  - Reasoning with completeness oracles
  - Enabling completeness in SPARQL
- Summary & conclusions
Summary

• Completeness is a dimension of data quality
  – It determines the value and reliability of the data
  – The state of the art provides completeness statements and oracles for simple queries

• Semantic Web is not completeness-aware
  – Vision
    • Use completeness oracles for simpler queries to infer completeness for arbitrary queries
    • Embed completeness in the SPARQL query language
  – Goal: Increase the value of the results delivered by queries
Future work

- Augment existing RDF data with completeness statements and oracles
- Extend query engines with completeness reasoning
  - Efficient implementation for oracle composition
  - Extend SPARQL to support the complete agg function
  - Reasoning beyond SR and D oracles
    - Use oracles that guarantee the completeness of queries with arbitrary number of triple patterns.
  - Provide confidence value for completeness guarantees.