

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

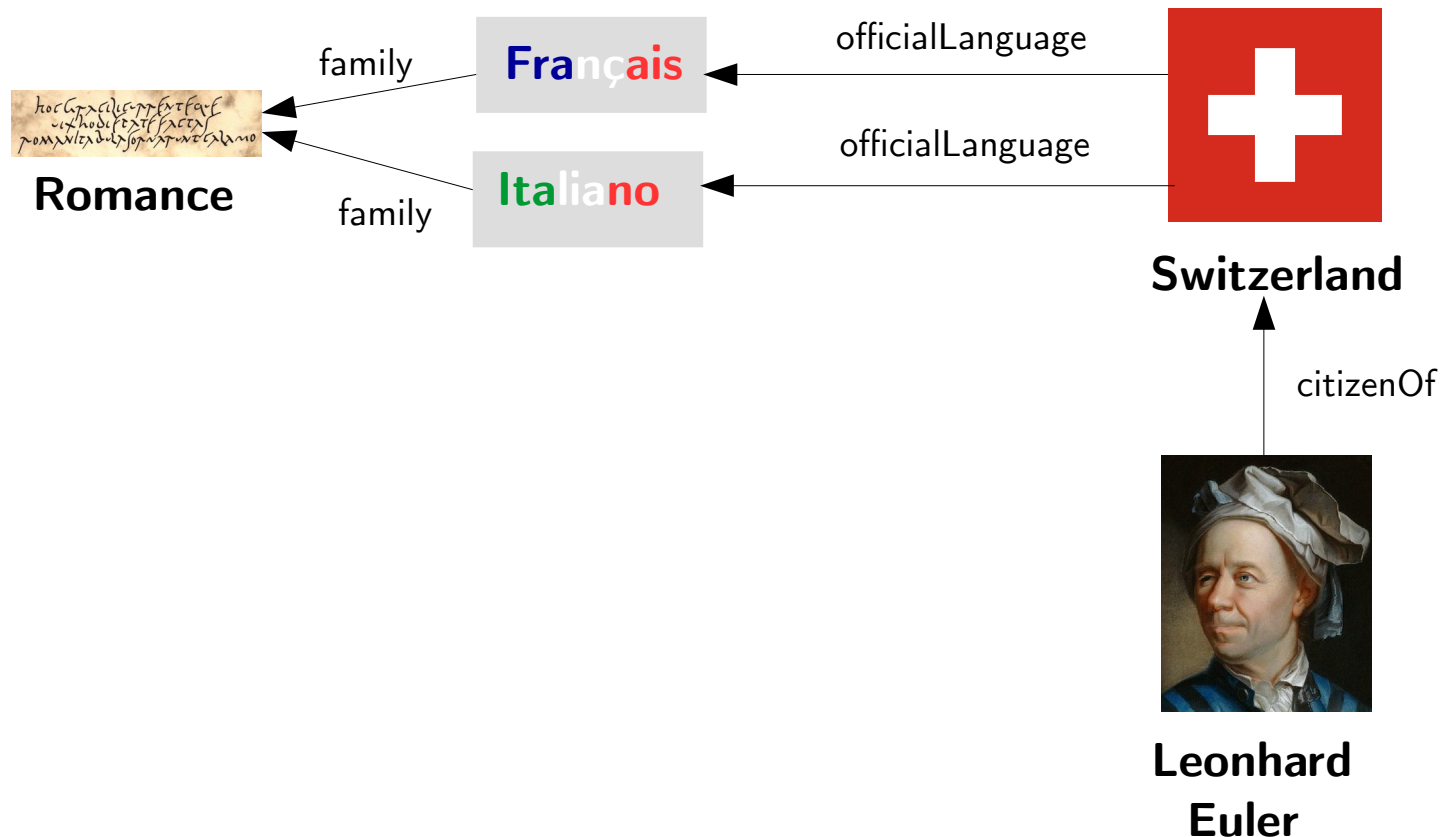
- Completeness in RDF knowledge bases
- State of the art on completeness
- Completeness oracles
- Vision on Completeness-aware Semantic Web
 - Representations for completeness oracles
 - Reasoning with completeness oracles
 - Enabling completeness in SPARQL
- Summary & conclusions

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

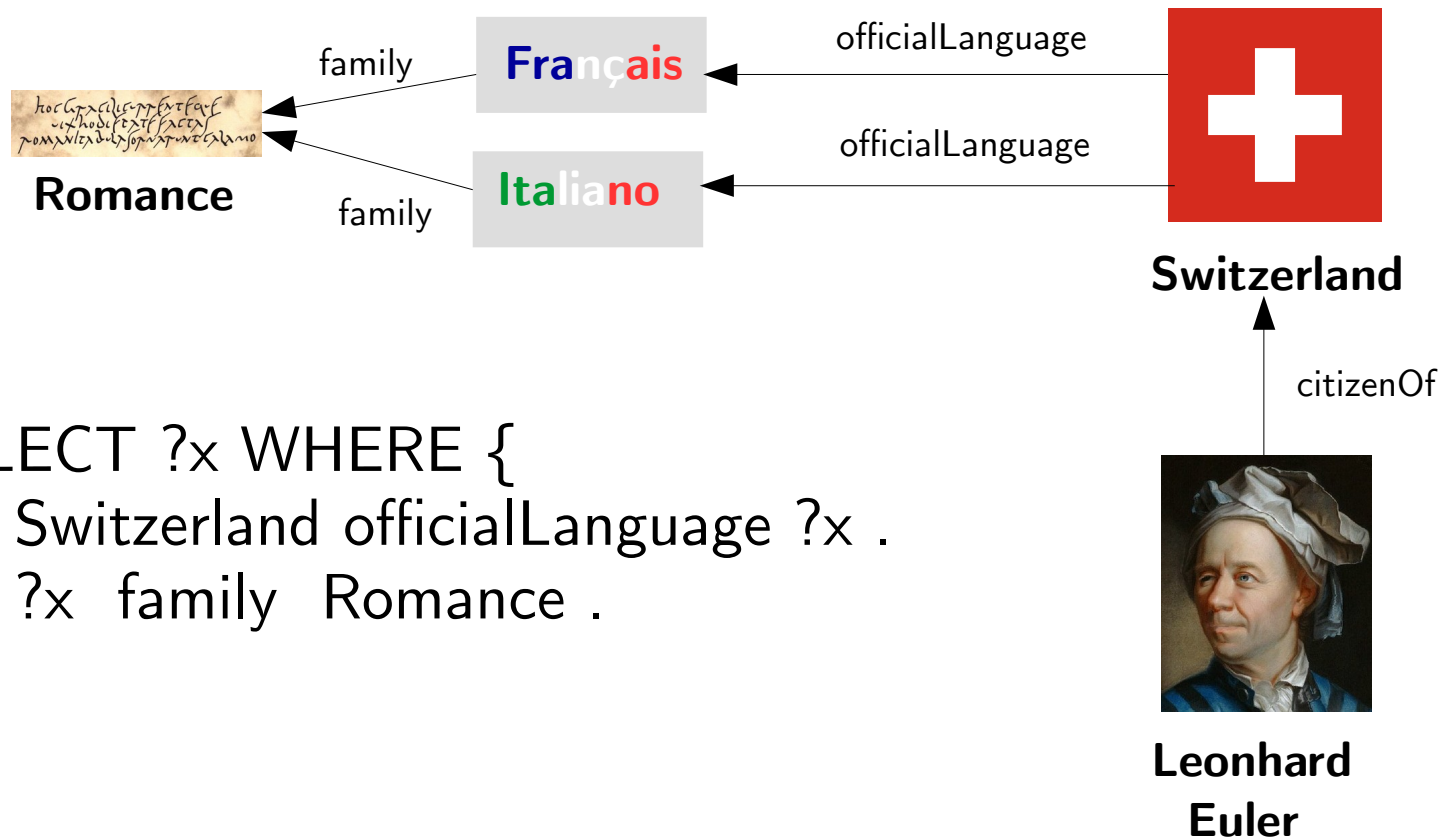
RDF Knowledge Bases (KBs)

Collection of structured knowledge



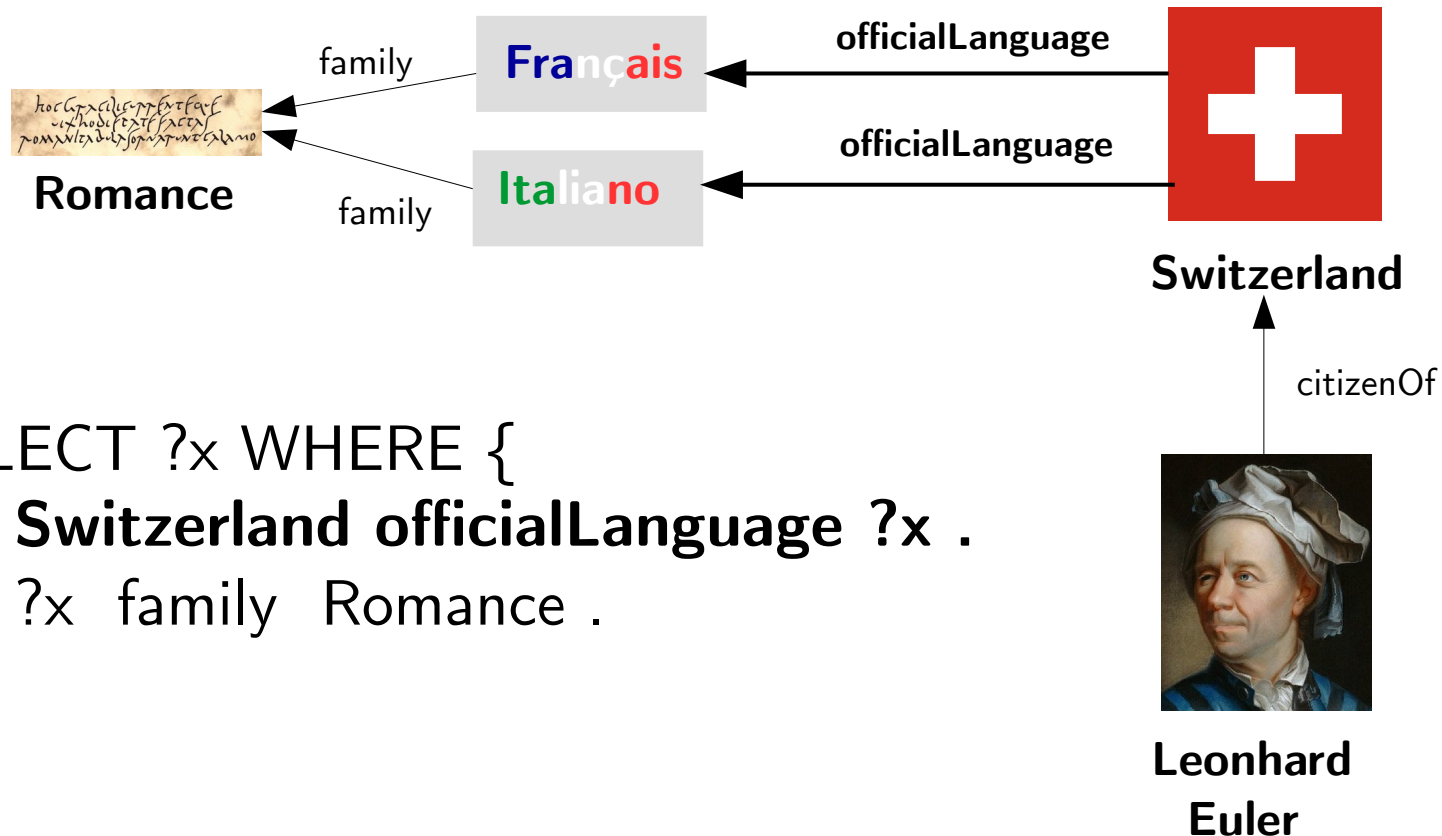
RDF Knowledge Bases (KBs)

RDF KBs can be queried using SPARQL



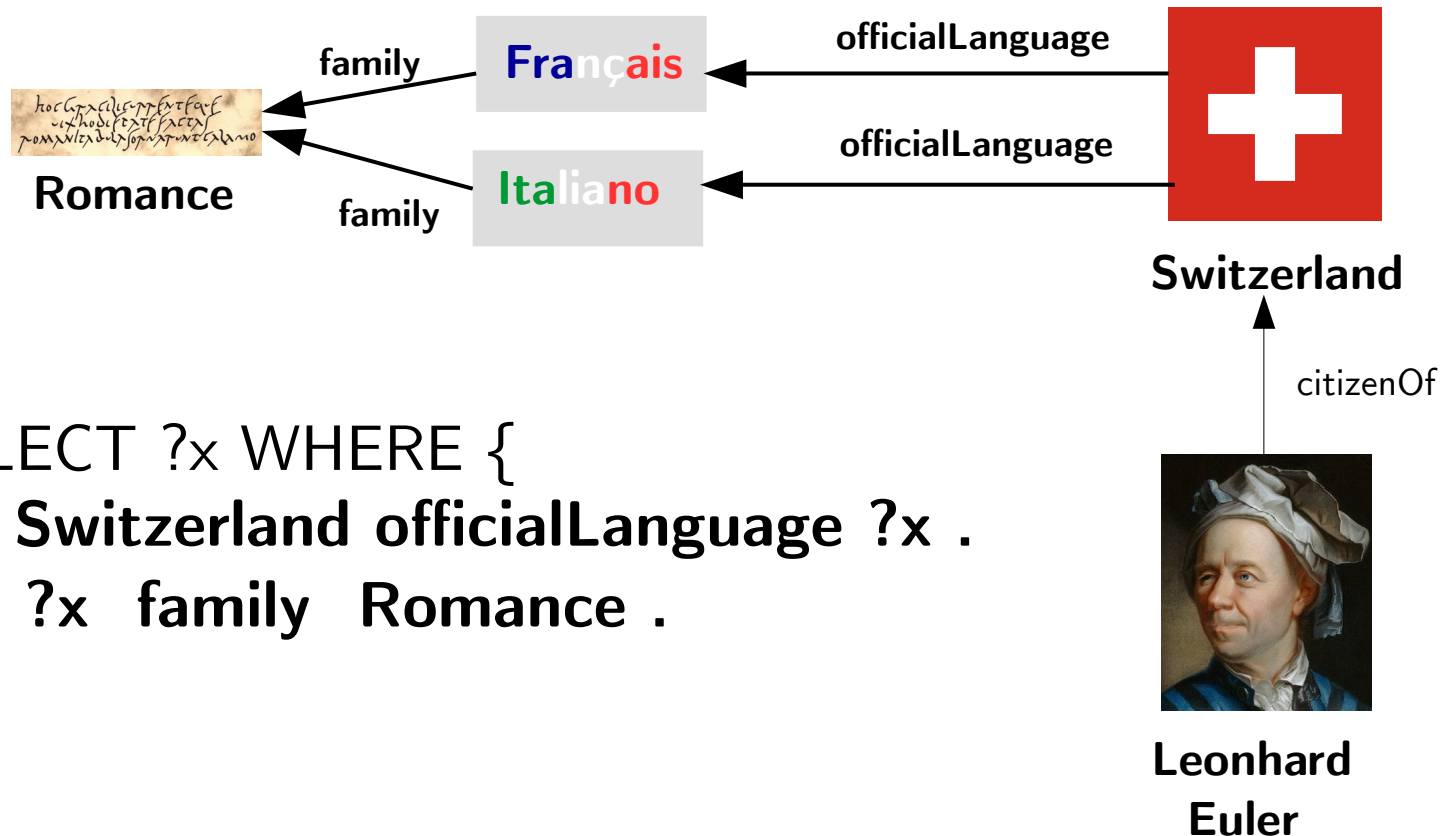
RDF Knowledge Bases (KBs)

RDF KBs can be queried using SPARQL



RDF Knowledge Bases (KBs)

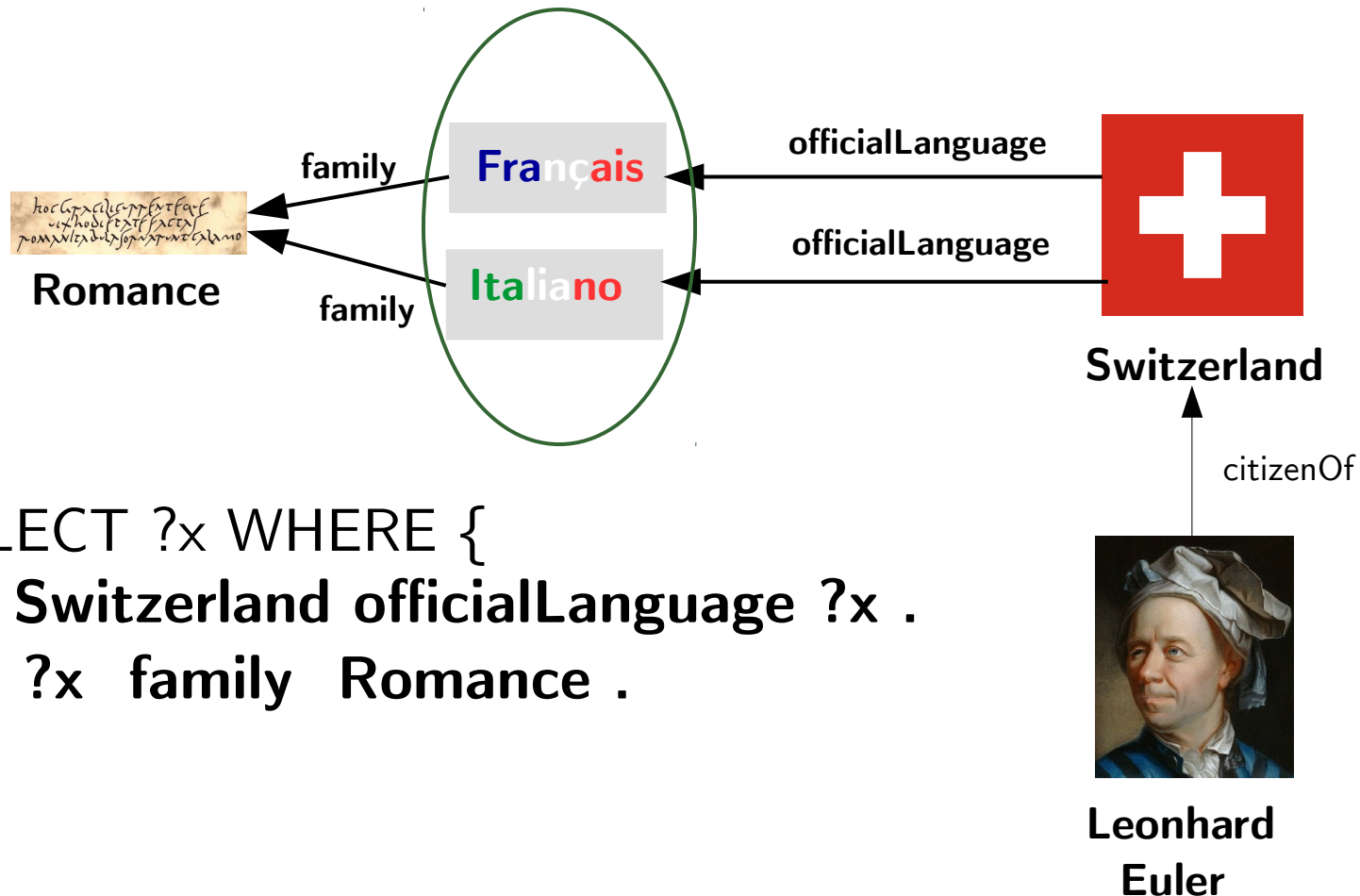
RDF KBs can be queried using SPARQL



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

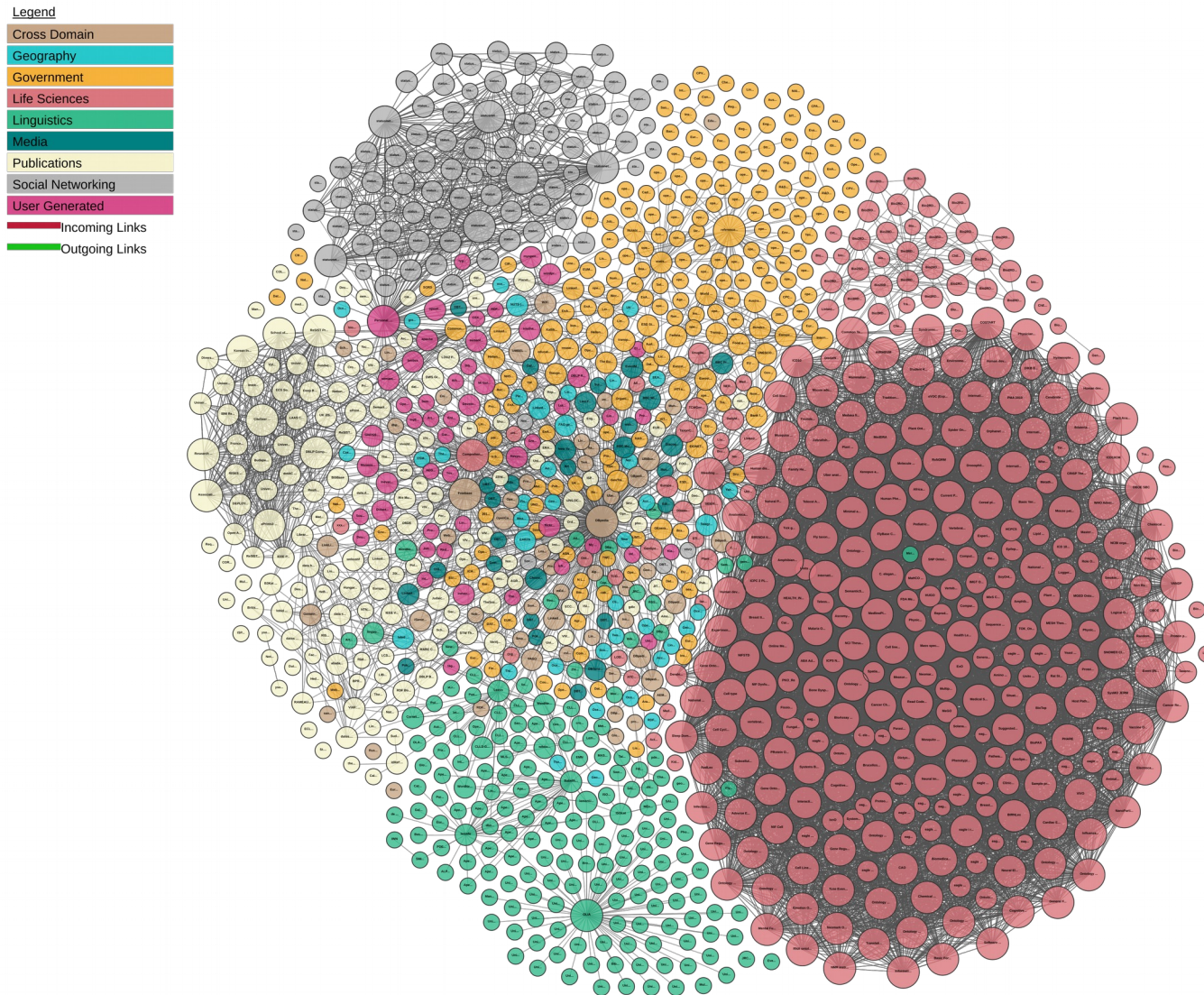
RDF Knowledge Bases (KBs)

RDF KBs can be queried using SPARQL

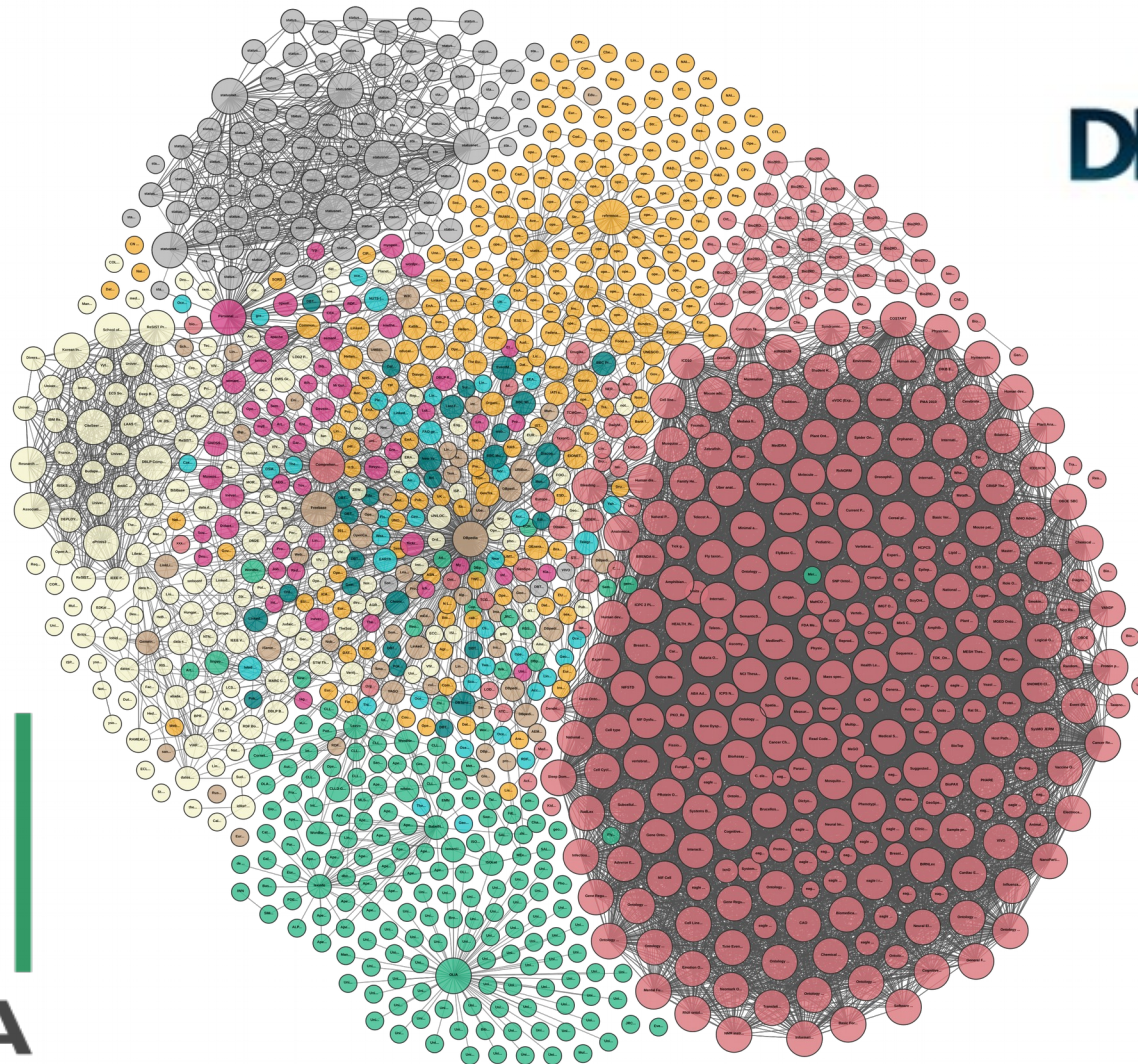


```
SELECT ?x WHERE {  
  Switzerland officialLanguage ?x .  
  ?x family Romance .  
}
```

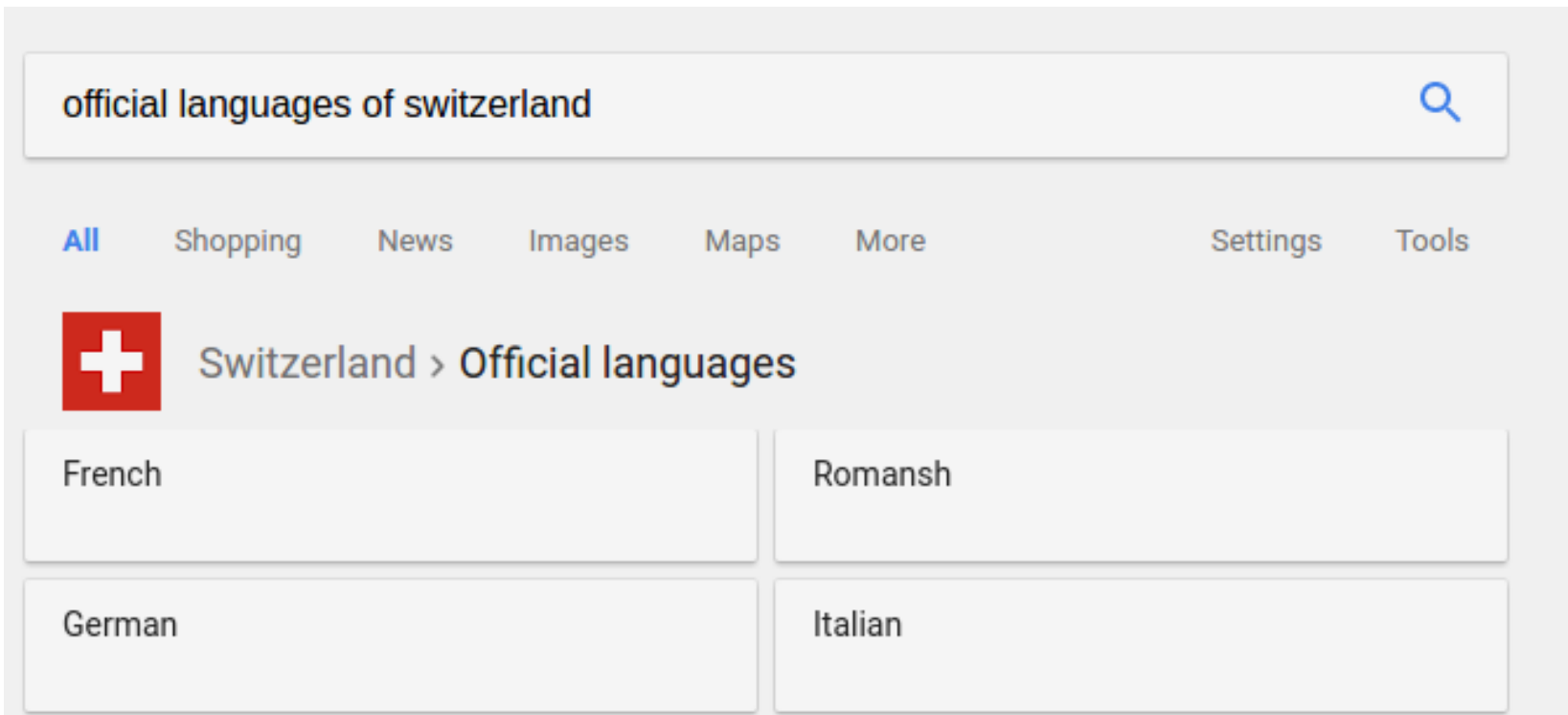
Plenty of KBs out there!



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

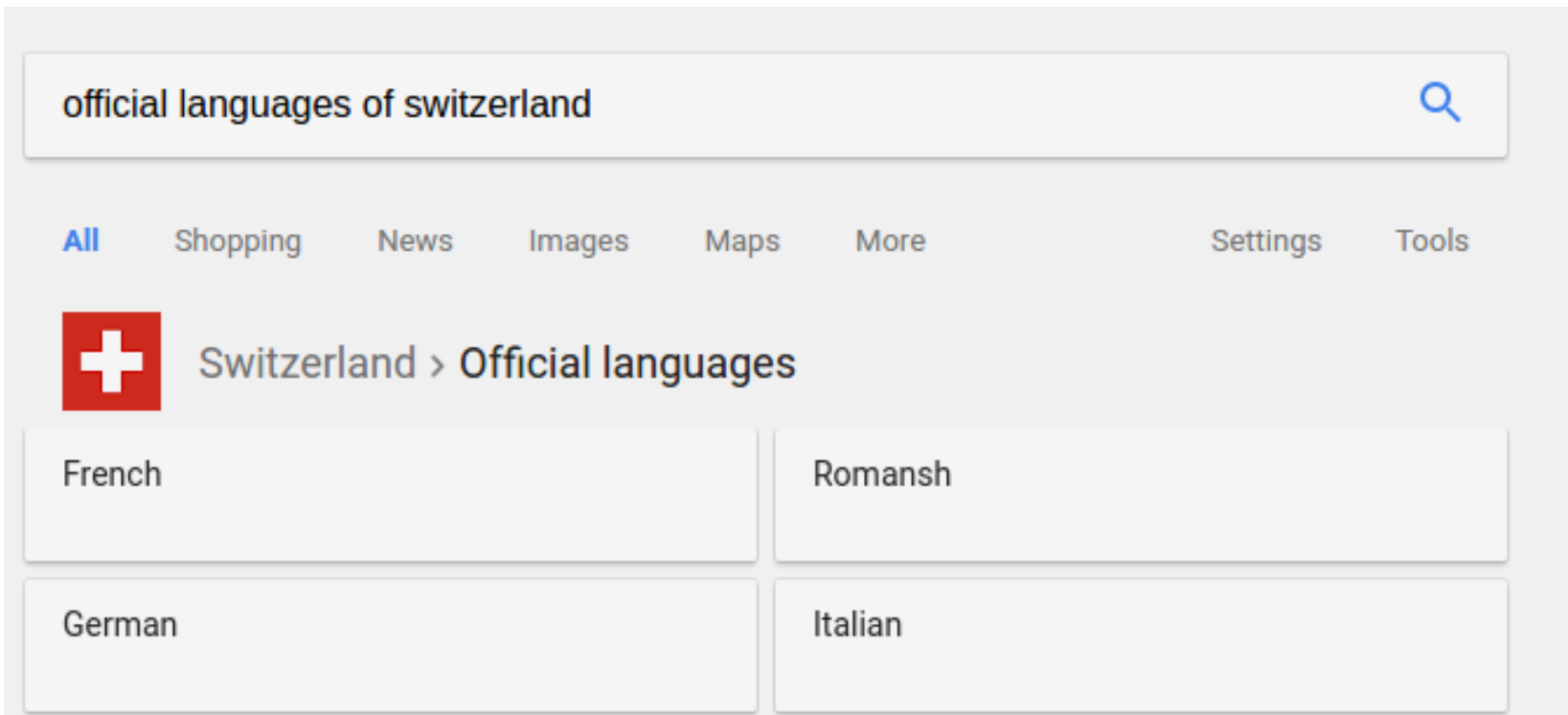
Completeness in RDF KBs

- 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

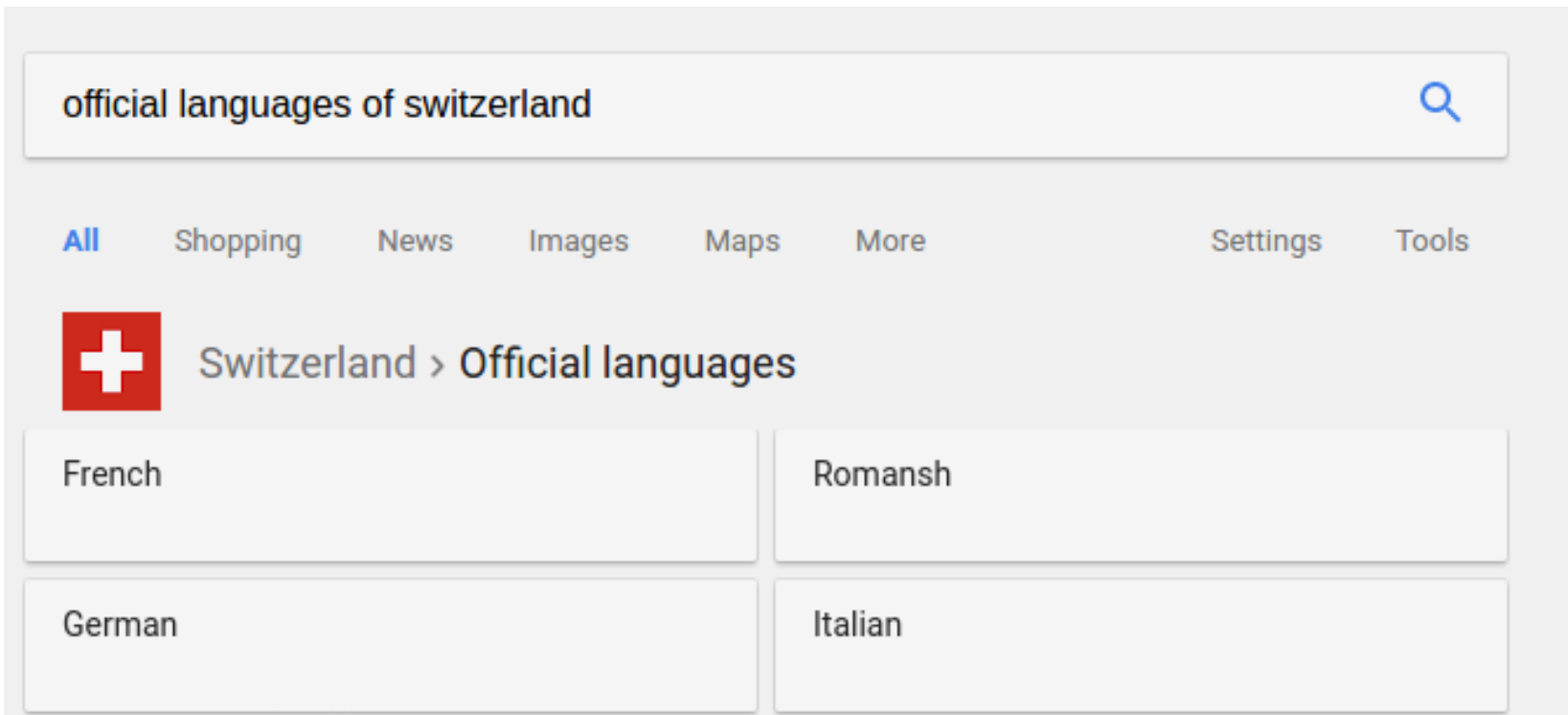
Completeness in RDF KBs

- 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^* .

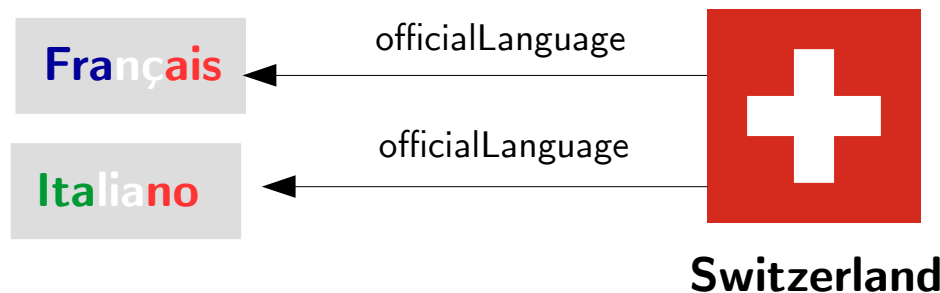
Completeness

- 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 $q(K^*) \subseteq q(K)$.

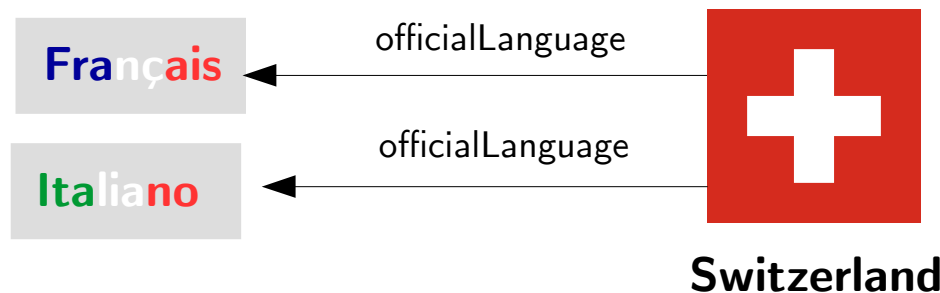
SELECT ?x WHERE { Switzerland officialLanguage ?x }



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 }



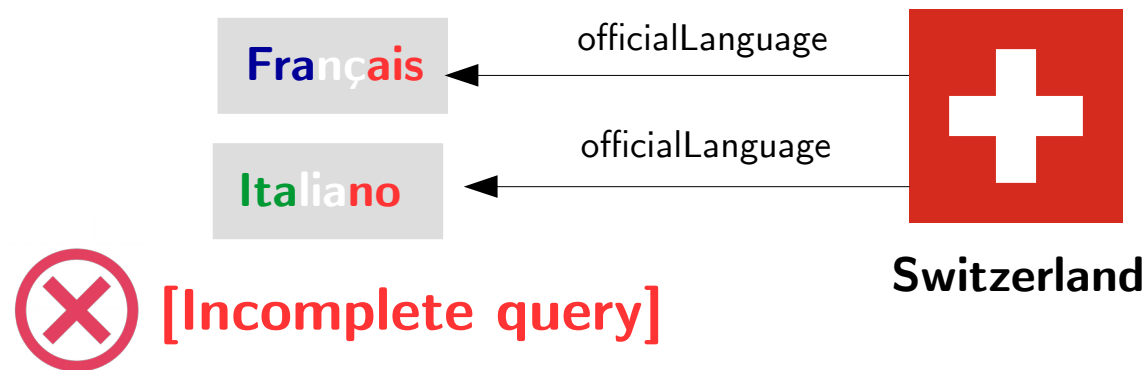
Are these all the official languages of Switzerland?



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?

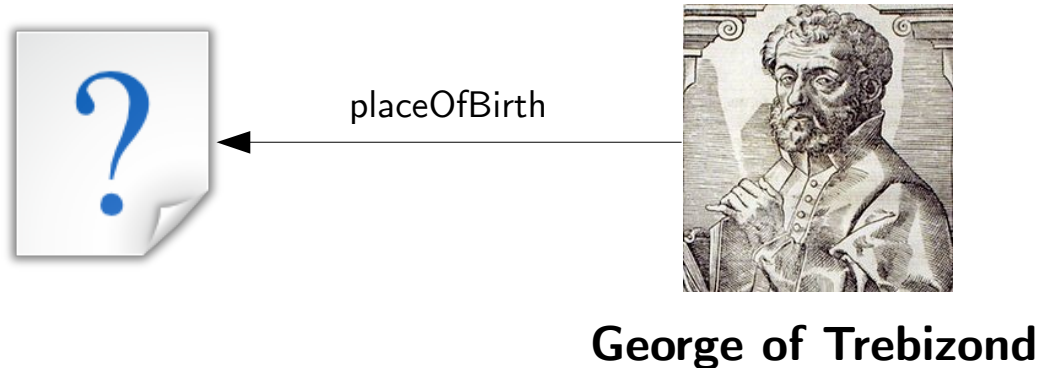


Outline

- 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

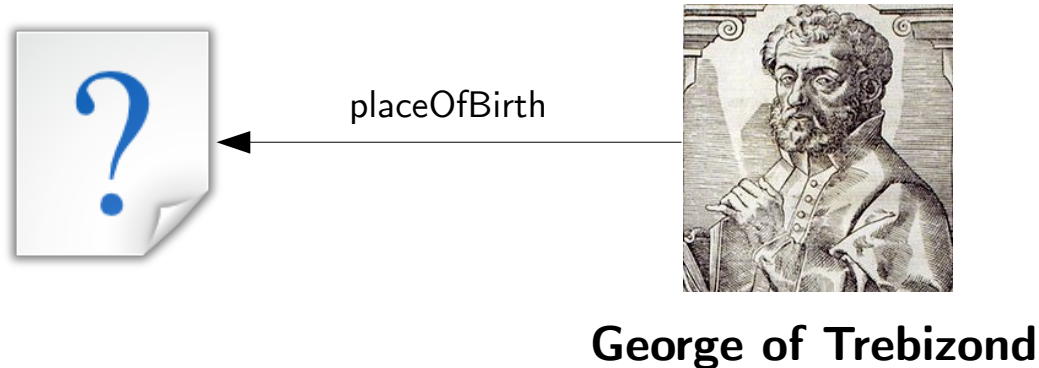
Completeness in RDF data

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



Completeness in RDF data

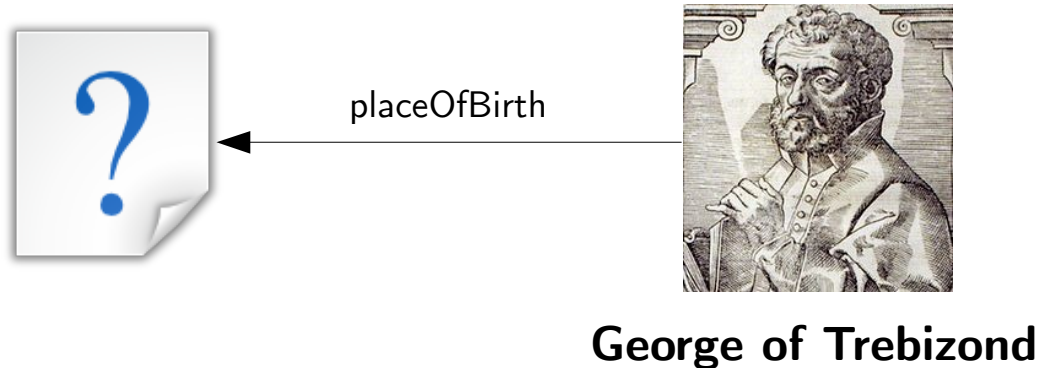
Wikidata keeps lists of subject-relation pairs with missing values.



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

Completeness in RDF data

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



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

⊗ [Incomplete query]

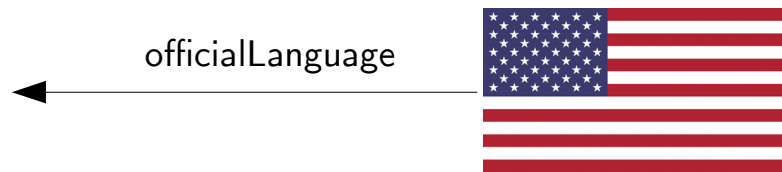
Completeness in RDF data

- Wikidata also provides *no value annotations*

Completeness in RDF data

- Wikidata also provides *no value annotations*

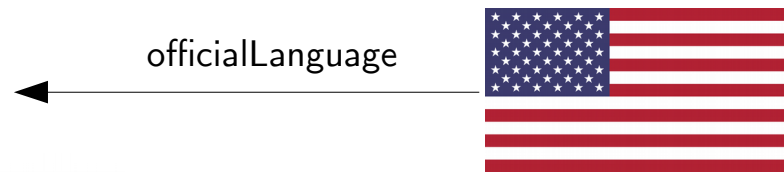
```
SELECT ?x WHERE { USA officialLanguage ?x }
```



Completeness in RDF data

- Wikidata also provides *no value annotations*

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

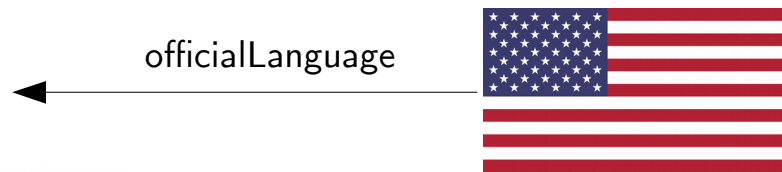


[Complete query]

Completeness in RDF data

- Wikidata also provides *no value annotations*

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



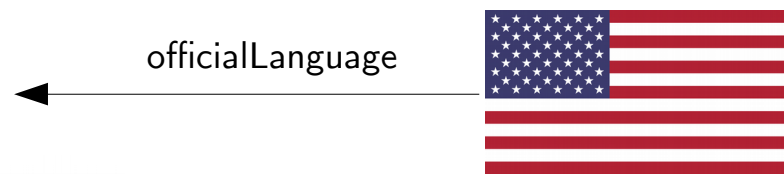
[Complete query]

- Not applicable if we know some official language

Completeness in RDF data

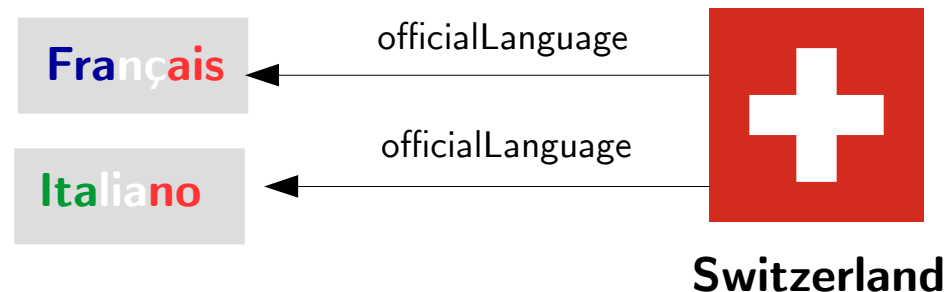
- Wikidata also provides *no value annotations*

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



[Complete query]

- Not applicable if we know some official language



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 ω that guesses the completeness of queries of the form:

SELECT ?x WHERE { subject relation ?x }

SR completeness oracle

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

SELECT ?x WHERE { subject relation ?x }

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

SR completeness oracle

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

SELECT ?x WHERE { subject relation ?x }

- We use the notation $\omega(\textit{subject}, \textit{relation})$
- $\omega = \textit{pca}(s, r) = \text{partial completeness assumption}$

SR completeness oracle

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

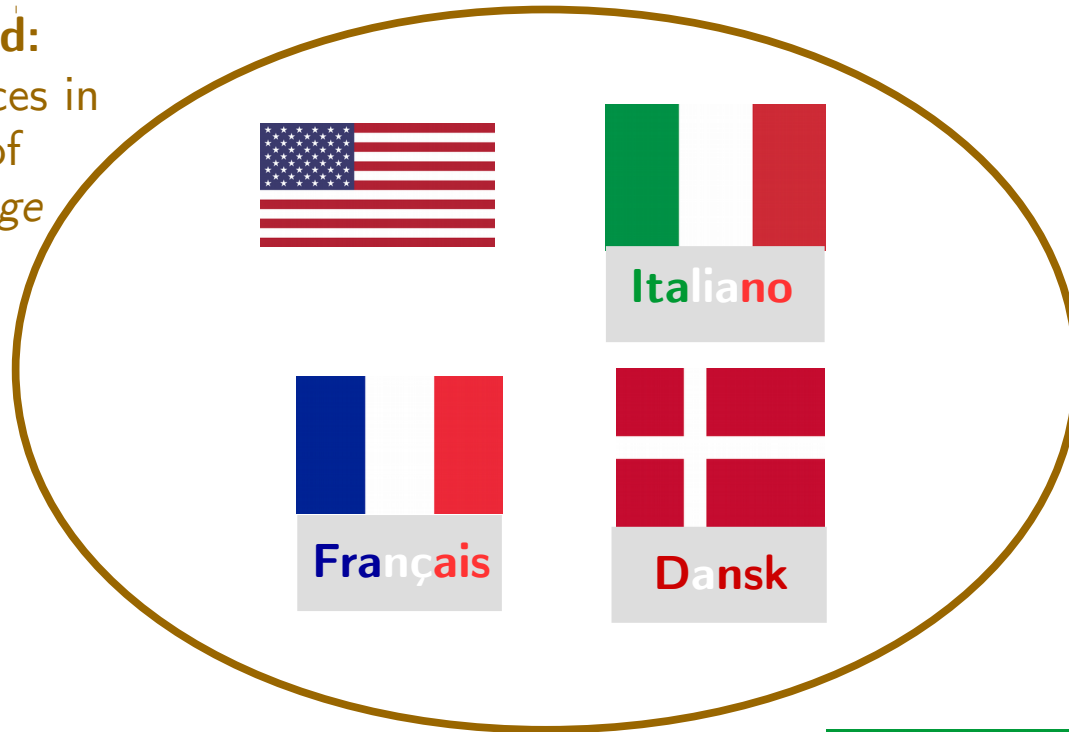
SELECT ?x WHERE { subject relation ?x }

- We use the notation $\omega(subject, relation)$
- $\omega = pca(s, r)$ = partial completeness assumption
 - Query is **complete** in KB if at least one answer is known

Evaluating SR oracles

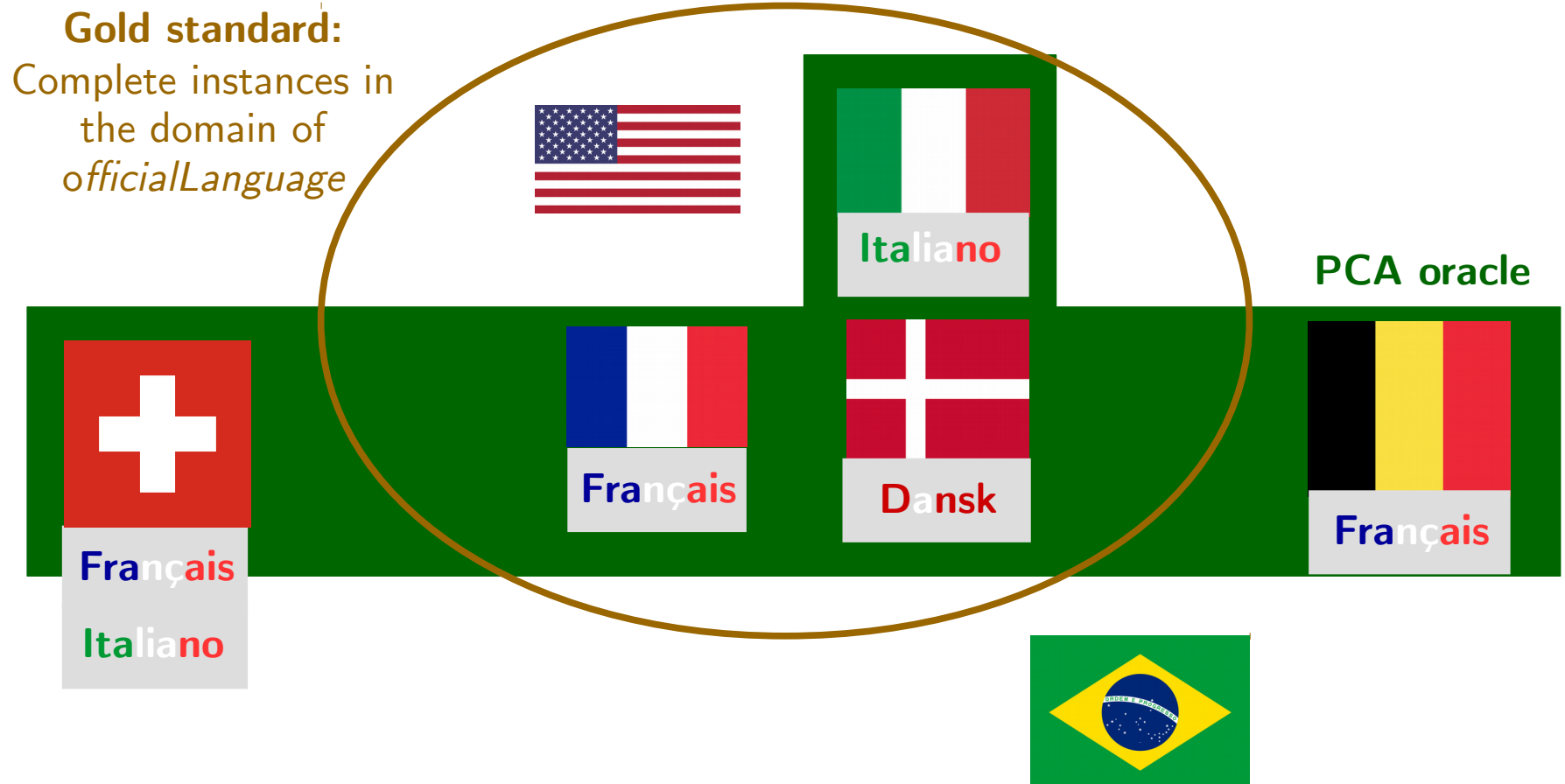
$\omega = pca(s, r) = \text{partial completeness assumption}$

Gold standard:
Complete instances in
the domain of
officialLanguage



Evaluating SR oracles

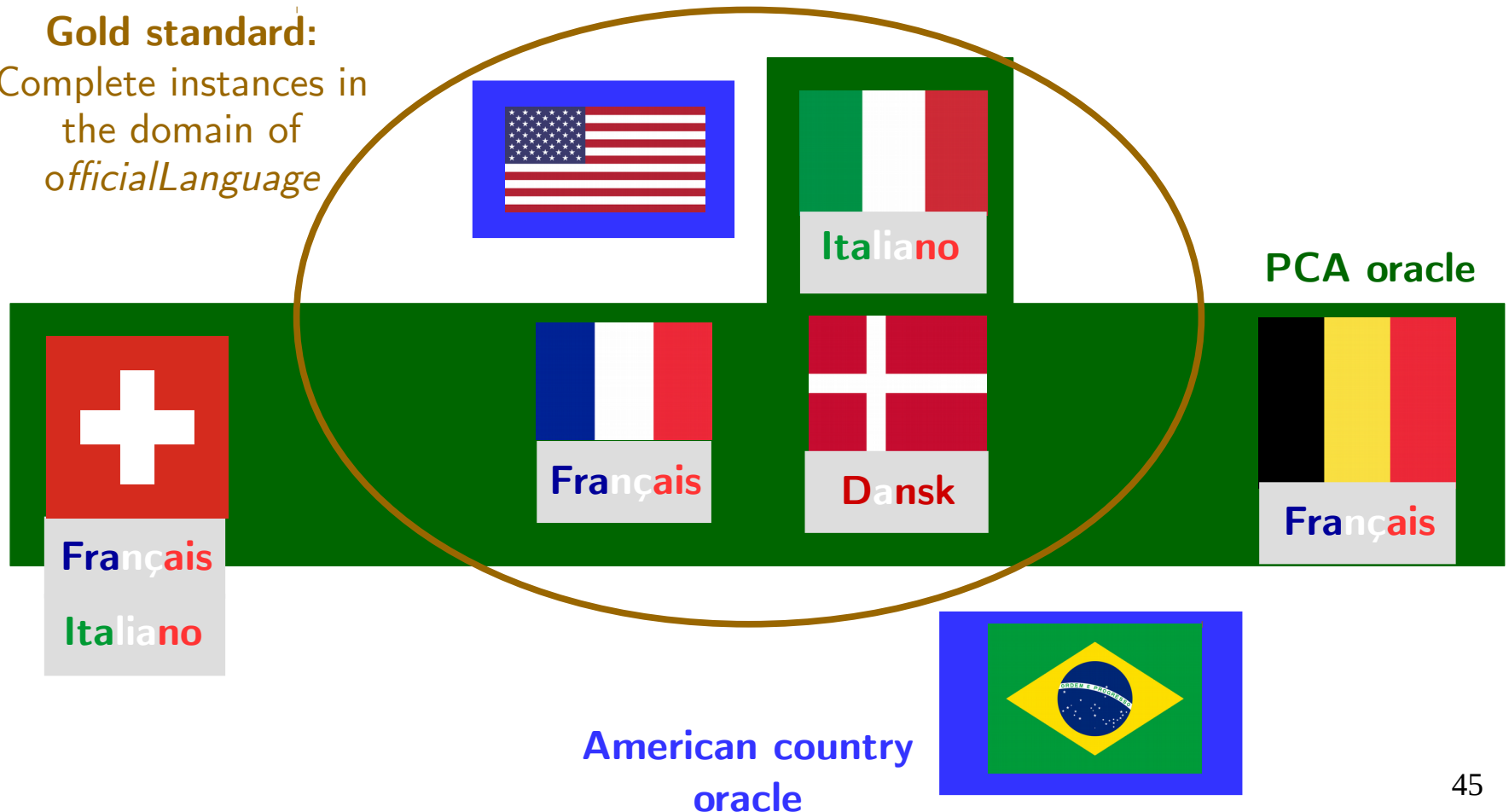
$\omega = pca(s, r) =$ partial completeness assumption



Evaluating SR oracles

$$\omega = \text{american-country-oracle}(s, r)$$

Gold standard:
Complete instances in
the domain of
officialLanguage



Evaluating SR oracles

PCA oracle

Precision = $3/5$

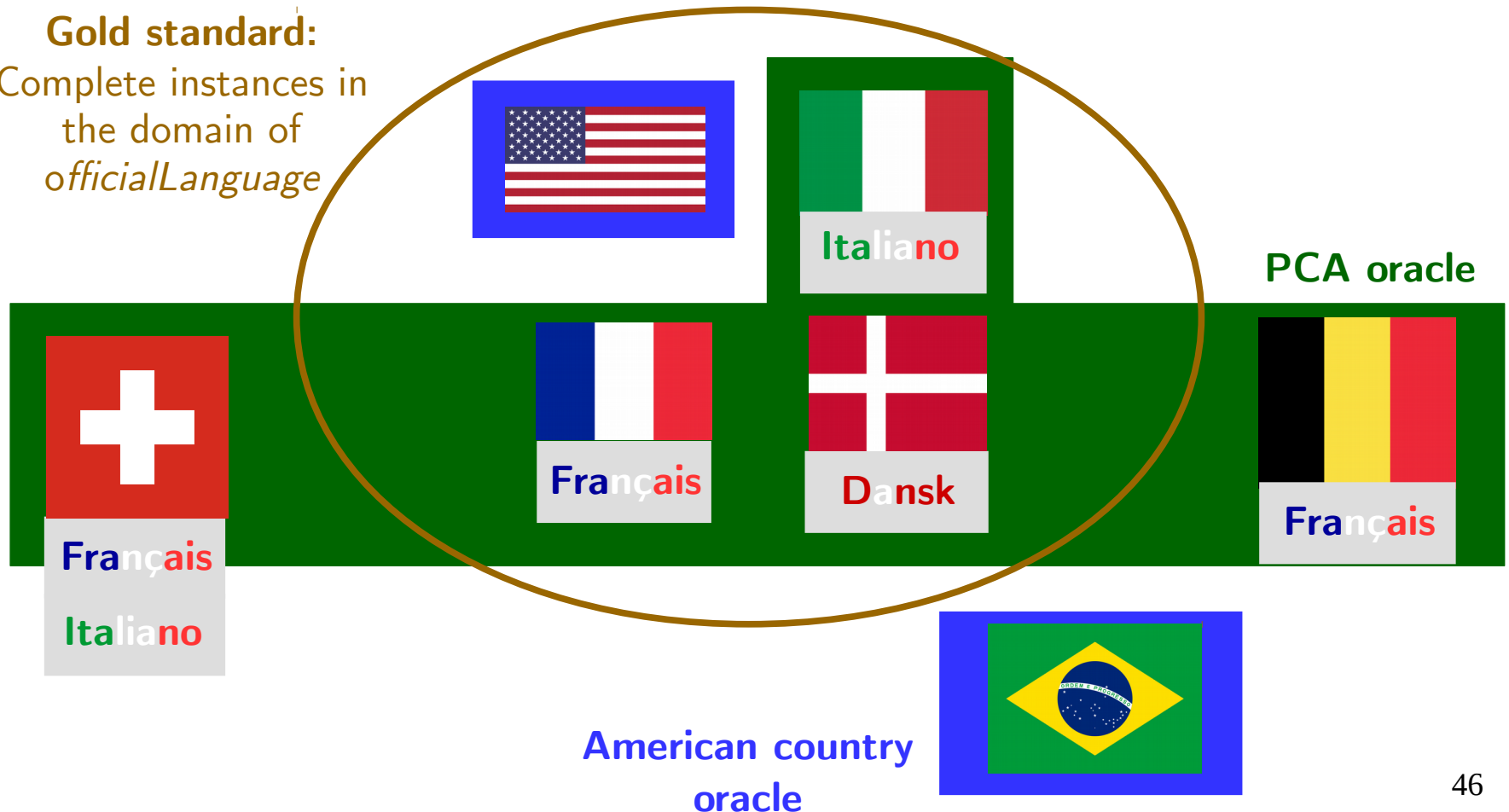
Recall = $3/4$

American country oracle

Precision = $1/2$

Recall = $1/4$

Gold standard:
Complete instances in
the domain of
officialLanguage



SR completeness oracles

- Closed World Assumption: $cwa(s, r) = \text{true}$
- PCA: $pca(s, r) = \exists o : r(s, o)$
- Cardinality: $card(s, r) = \#(o : r(s, o)) \geq k$
- Popular entities: $popularity_{pop}(s, r) = pop(s)$
- No-chg over time: $nochange_{chg}(s, r) = \sim chg(s, r)$
- Star : $star_{r_1, \dots, r_n}(s, r) = \forall i \in \{1, \dots, n\} : \exists o : r_i(s, o)$
- Class: $class_c(s, r) = type(s, c)$
- 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

`notype(x, Adult), type(x, Person) \Rightarrow complete(x, hasChild)`

`dateOfDeath(x, y), lessThan1(x, placeOfDeath) \Rightarrow incomplete(x, 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

`notype(x, Adult), type(x, Person) \Rightarrow complete(x, hasChild)`

`dateOfDeath(x, y), lessThan1(x, placeOfDeath) \Rightarrow incomplete(x, 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 *hasChild*, *graduatedFrom*

Performance of SR oracles

F1 measure of the oracles in YAGO3

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

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

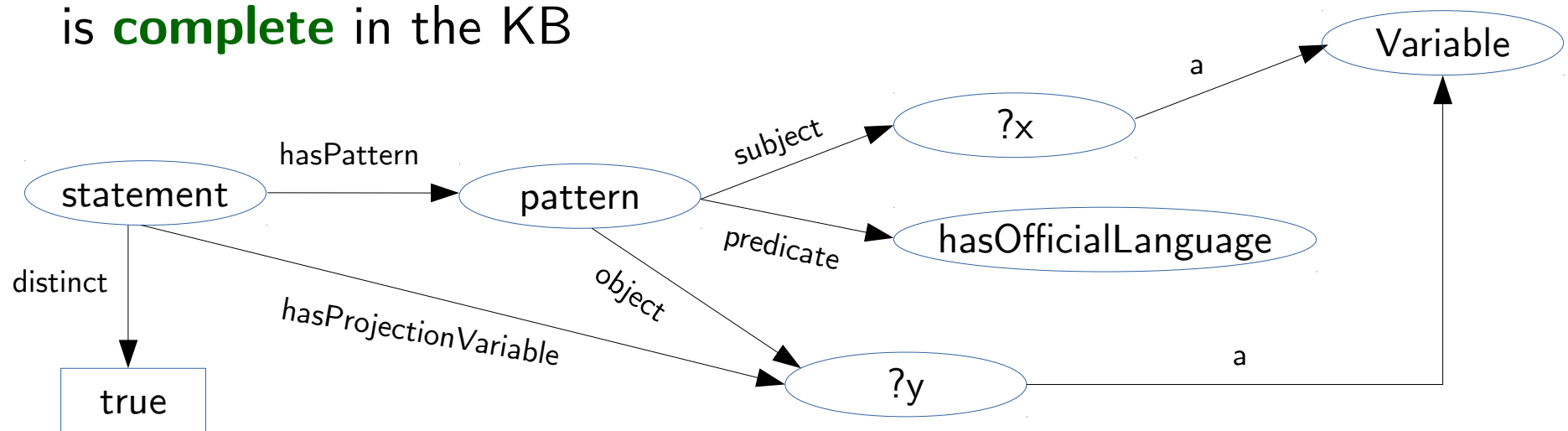
- Extensional approach [Darari, et al, 2013]
 - An oracle is a collection of completeness statements about queries

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

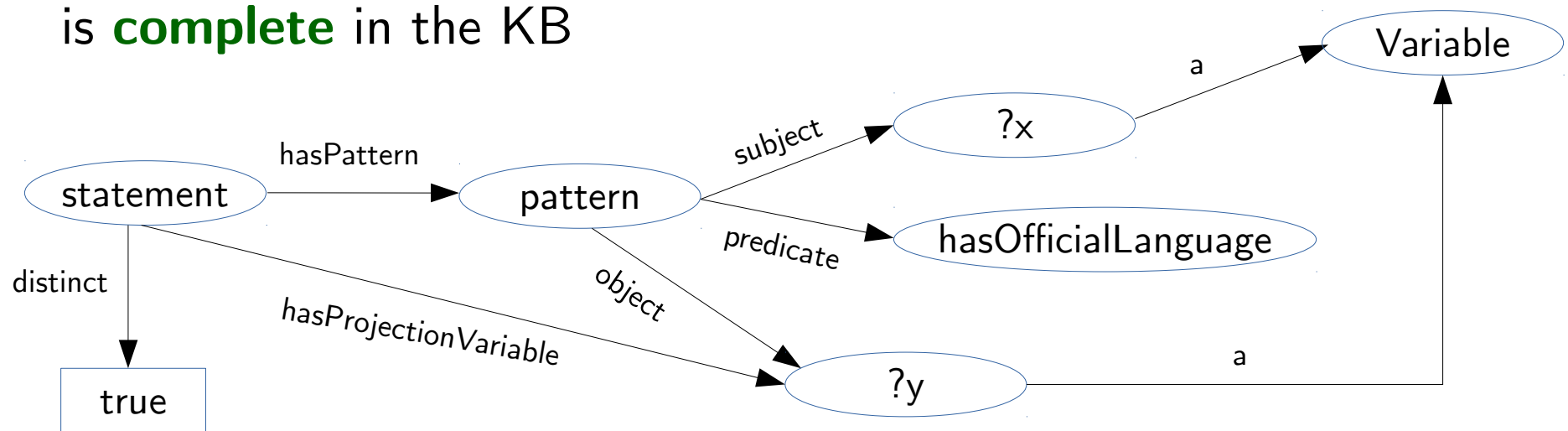
SELECT DISTINCT ?y WHERE { ?x hasOfficialLanguage ?y }
is **complete** in the KB



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

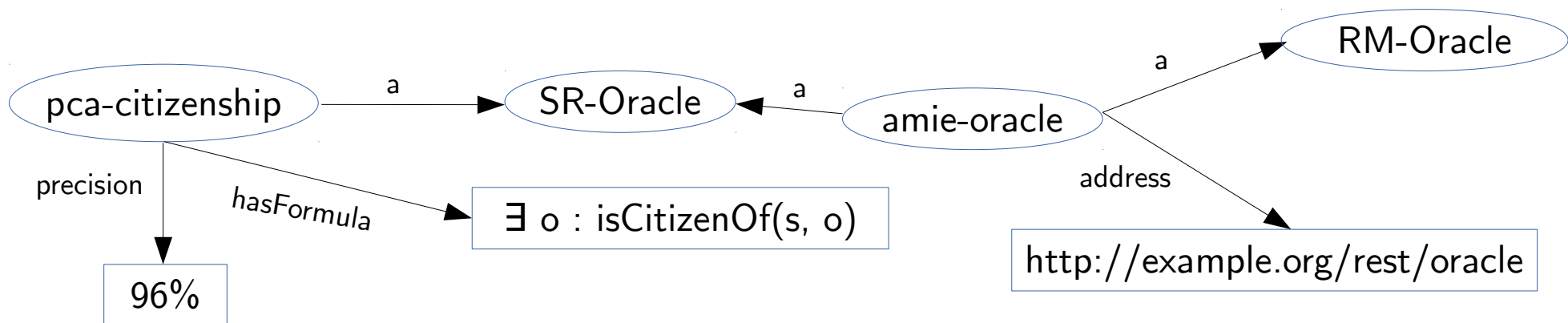


Representing completeness oracles

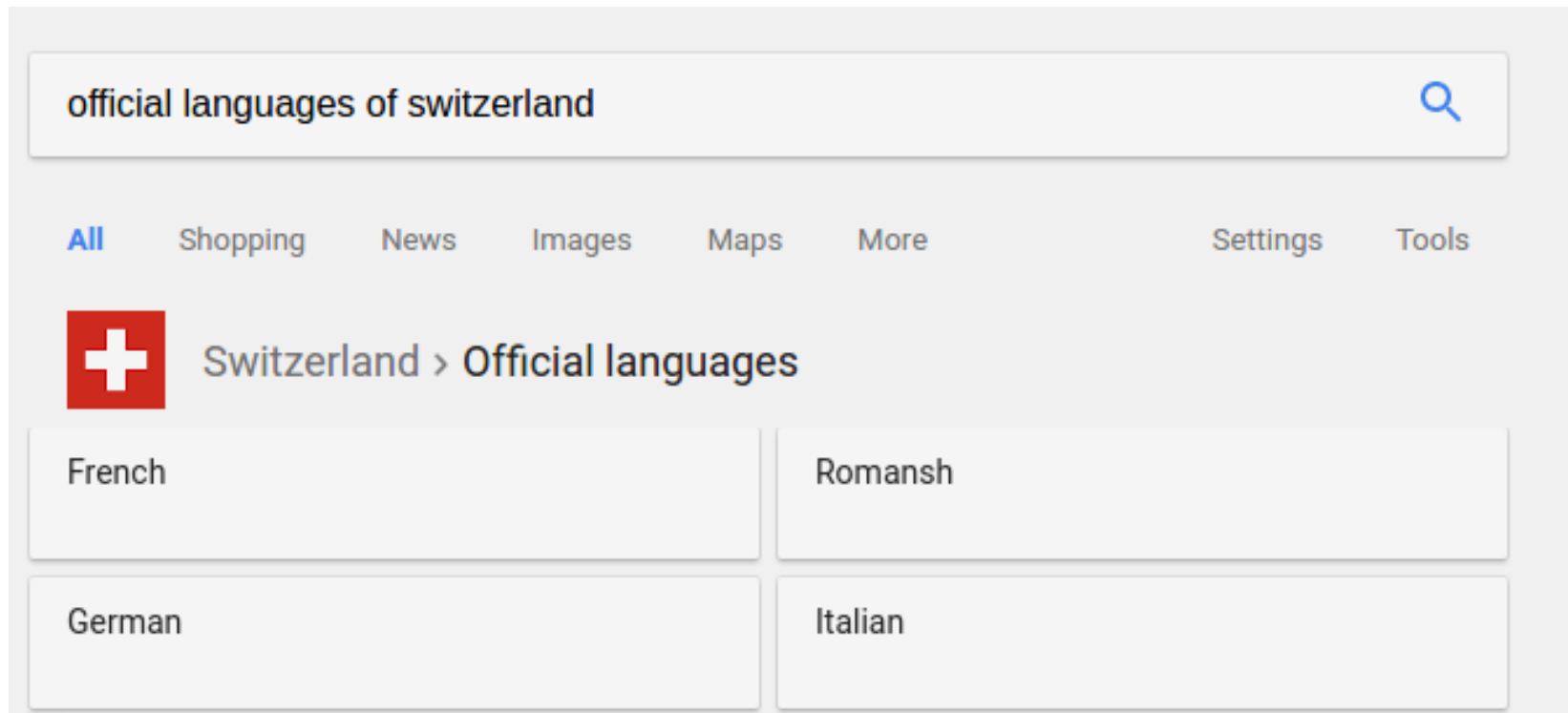
- Intensional approach
 - The oracle logic is embedded as a lambda function or a link to a program or resource

Representing completeness oracles

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

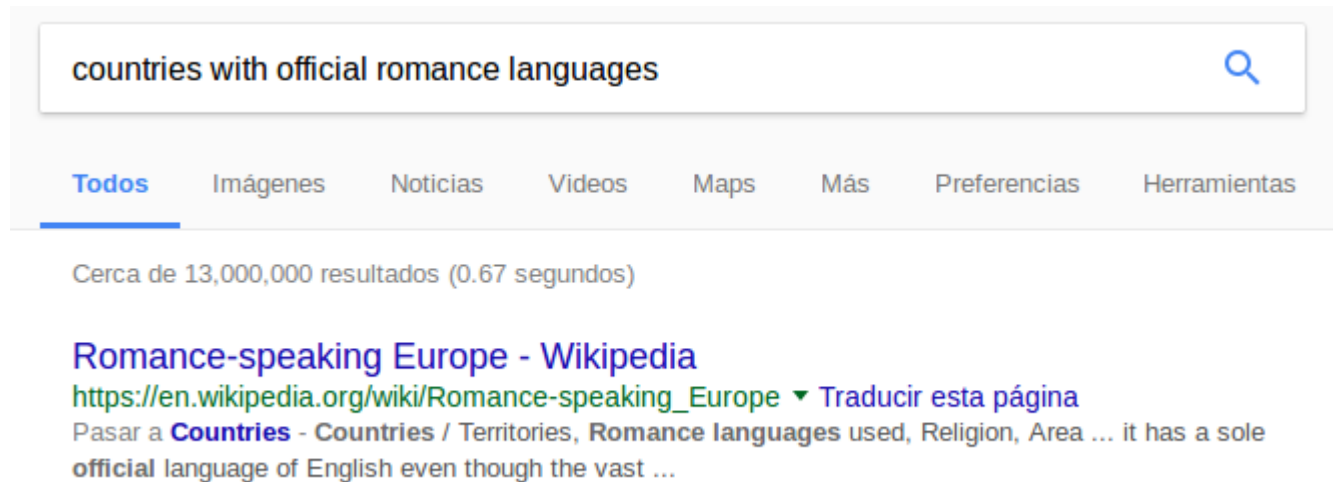


Providing completeness guarantees

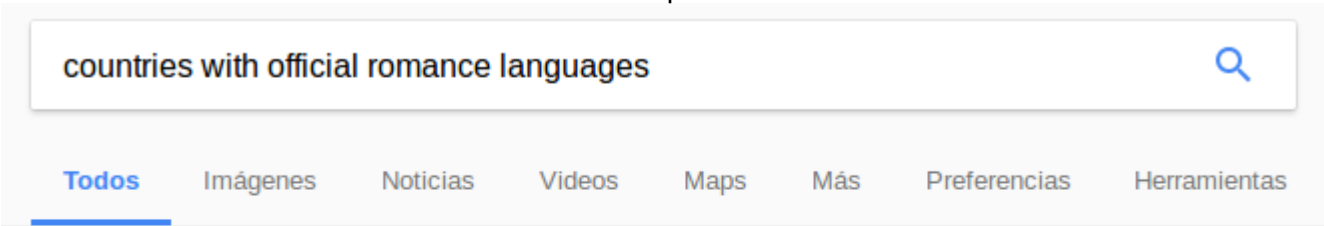


This list of results is complete with confidence X according to ω

Providing completeness guarantees



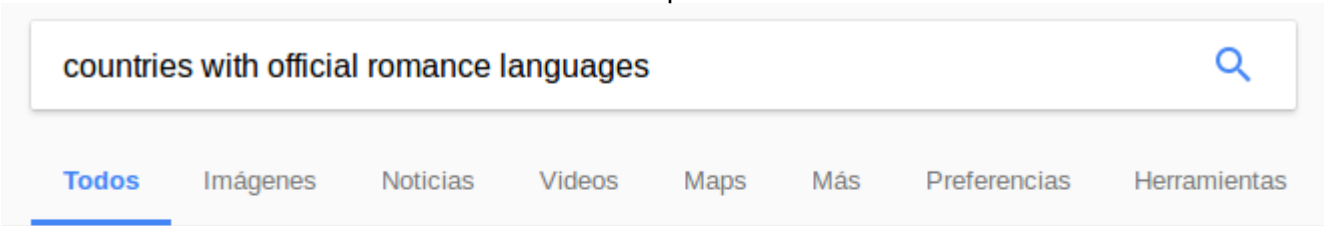
Providing completeness guarantees



The screenshot shows a Google search interface. The search bar contains the text "countries with official romance languages". Below the search bar, there are tabs for "Todos", "Imágenes", "Noticias", "Videos", "Maps", "Más", "Preferencias", and "Herramientas". The "Todos" tab is selected. Below the tabs, it says "Cerca de 13,000,000 resultados (0.67 segundos)". The first search result is "Romance-speaking Europe - Wikipedia" with the URL "https://en.wikipedia.org/wiki/Romance-speaking_Europe". Below the URL, there is a link "Traducir esta página". The snippet of the result reads: "Pasar a **Countries** - Countries / Territories, Romance languages used, Religion, Area ... it has a sole official language of English even though the vast ...".

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

Providing completeness guarantees



countries with official romance languages

[Todos](#) [Imágenes](#) [Noticias](#) [Videos](#) [Maps](#) [Más](#) [Preferencias](#) [Herramientas](#)

Cerca de 13,000,000 resultados (0.67 segundos)

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
official language of English even though the vast ...

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

How to provide
completeness guarantees
for arbitrary queries?



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 ω_d for the completeness of queries:

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

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

D completeness oracles

- Oracle ω_d for the completeness of queries:

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

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

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

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

D completeness oracles

- Oracle ω_d for the completeness of queries:

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

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

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

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

- If $\omega_d(\textbf{officialLanguage})$ returns true, ω_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

```
SELECT ?country WHERE {  
    ?country officialLanguage ?lang .  
    ?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

```
SELECT ?country WHERE {  
    ?country officialLanguage ?lang .  
    ?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 {  
    ?country officialLanguage ?lang .  
    ?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

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

$$\omega' = \omega(\text{Romance}, \text{family}^{-1}) \wedge \left(\bigwedge_{l: \text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1}) \right)$$

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 {  
    ?country officialLanguage ?lang .  
    ?lang family Romance .  
}
```

It will generate
false negatives

$$\omega' = \omega(\text{Romance}, \text{family}^{-1}) \wedge \left(\bigwedge_{l: \text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1}) \right)$$

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

If the KB misses
Ligurian, this term
returns false

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

It will generate
false negatives

$$\omega' = \omega(\text{Romance}, \text{family}^{-1}) \wedge \left(\bigwedge_{l: \text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1}) \right)$$

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 {  
    ?country officialLanguage ?lang  
    ?lang family Romance .  
}
```

Even though this term does not care, because Ligurian is not official in any country

$$\omega' = \omega(\text{Romance}, \text{family}^{-1}) \wedge \left(\bigwedge_{l: \text{family}(l, \text{Romance})} \omega(l, \text{officialLanguage}^{-1}) \right)$$

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 {  
    ?country officialLanguage ?lang .  
    ?lang family Romance .  
}
```

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

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

$$\omega^3 = \omega(\text{Romance}, \text{family}^{-1}) \wedge \omega_d(\text{officialLanguage}) \wedge \left(\bigwedge_{c:\text{officialLanguage}(c, l)} \omega(c, \text{officialLanguage}) \right)$$

Tightness for completeness guarantees

- $\omega' \prec_q \omega''$ for q if $\forall K : \omega''(q, K) \wedge \omega'(q, K) :$
 - $\omega''(q, K') \Rightarrow \omega'(q, K') \quad \forall K' \subseteq K.$

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

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

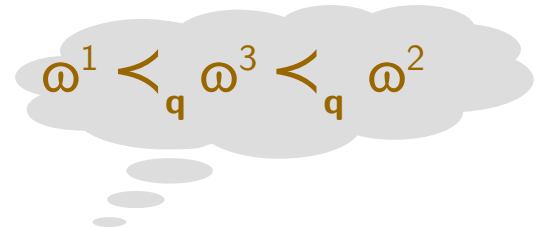
$$\omega^2 = \omega(\text{Romance}, \text{family}^{-1}) \wedge (\bigwedge_{l:\text{family}(l, f)} \omega(l, \text{officialLanguage}^{-1}))$$

$$\omega^3 = \omega(\text{Romance}, \text{family}^{-1}) \wedge \omega_d(\text{officialLanguage}) \wedge (\bigwedge_{c:\text{officialLanguage}(c, l)} \omega(c, \text{officialLanguage}))$$

Tightness for completeness guarantees

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

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



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

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

$$\omega^3 = \omega(\text{Romance}, \text{family}^{-1}) \wedge \omega_d(\text{officialLanguage}) \wedge \left(\bigwedge_{c:\text{officialLanguage}(c, l)} \omega(c, \text{officialLanguage}) \right)$$

Cost for completeness guarantees

- Number of oracle calls required for the answer

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

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

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

$$\omega^3 = \omega(\text{Romance}, \text{family}^{-1}) \wedge \omega_d(\text{officialLanguage}) \wedge \left(\bigwedge_{c:\text{officialLanguage}(c, l)} \omega(c, \text{officialLanguage}) \right)$$

Cost for completeness guarantees

- Number of oracle calls required for the answer

SELECT ?country WHERE {
 officialLanguage ?lang .

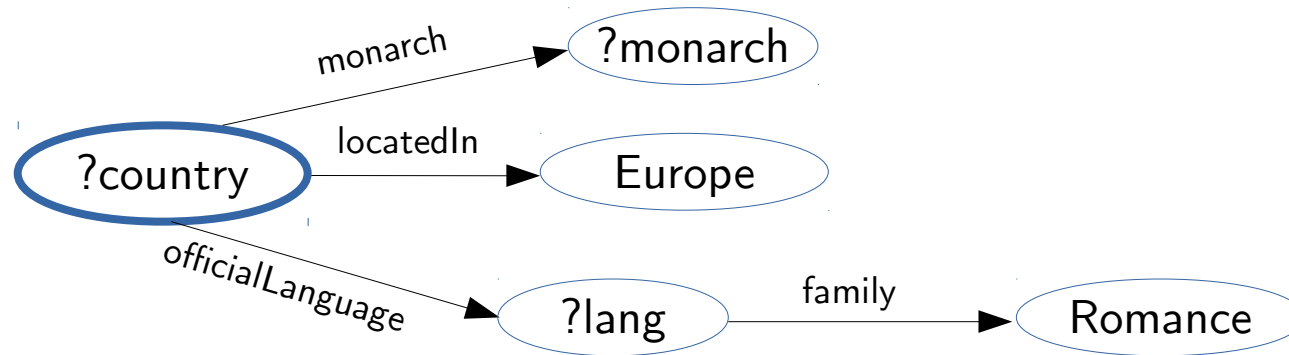
$\text{cost}(\omega^1) = 1 + (\#l: \text{family}(l, \text{Romance}))$

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

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

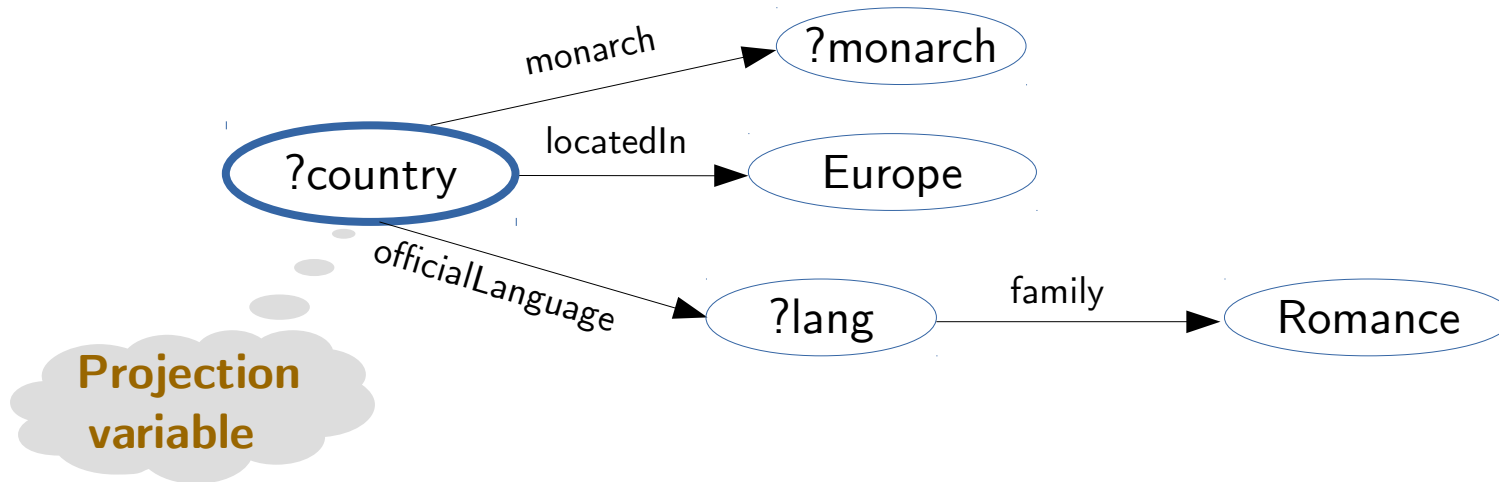
$$\omega^3 = \omega(\text{Romance}, \text{family}^{-1}) \wedge \omega_d(\text{officialLanguage}) \wedge \left(\bigwedge_{c: \text{officialLanguage}(c, l)} \omega(c, \text{officialLanguage}) \right)$$

Automatic oracle composition



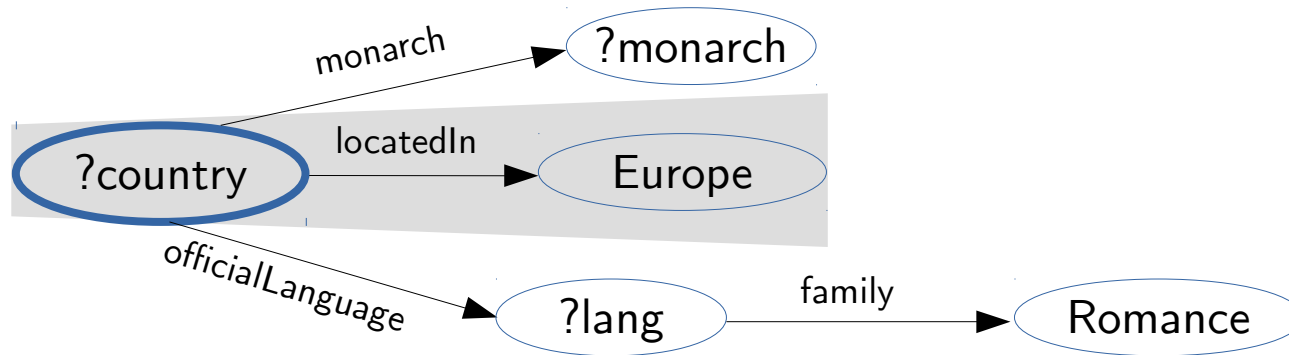
```
SELECT ?country WHERE {  
    ?country monarch ?monarch .  
    ?country locatedIn Europe .  
    ?country officialLanguage ?lang .  
    ?lang family Romance .  
}
```

Automatic oracle composition



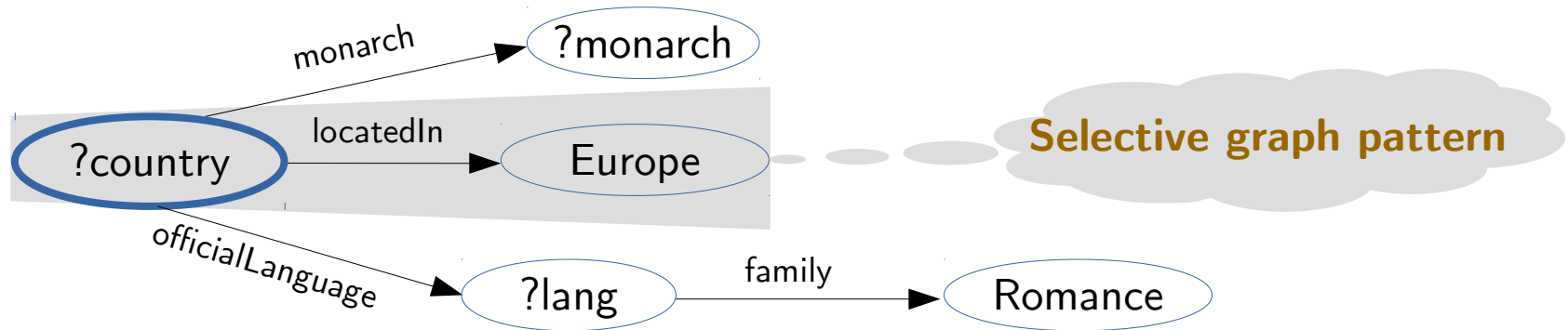
```
SELECT ?country WHERE {  
    ?country monarch ?monarch .  
    ?country locatedIn Europe .  
    ?country officialLanguage ?lang .  
    ?lang family Romance .  
}
```

Automatic oracle composition



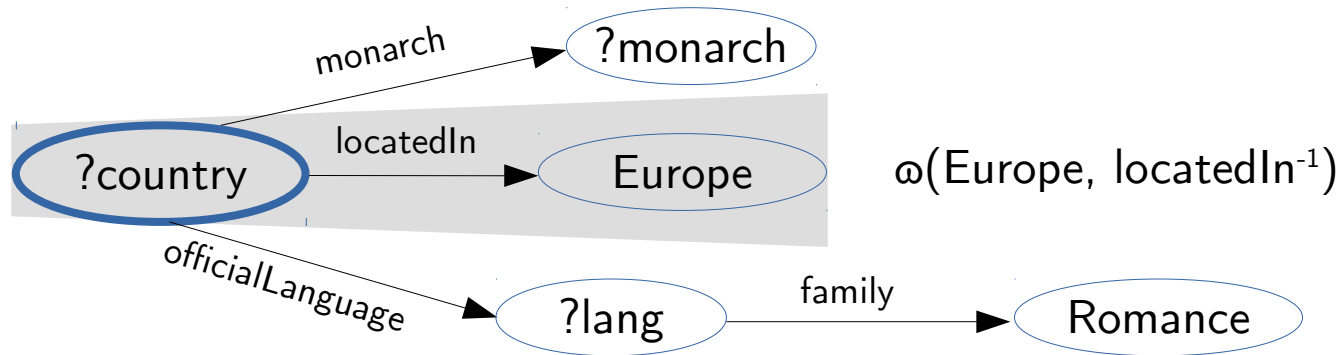
```
SELECT ?country WHERE {  
    ?country monarch ?monarch .  
    ?country locatedIn Europe .  
    ?country officialLanguage ?lang .  
    ?lang family Romance .  
}
```

Automatic oracle composition



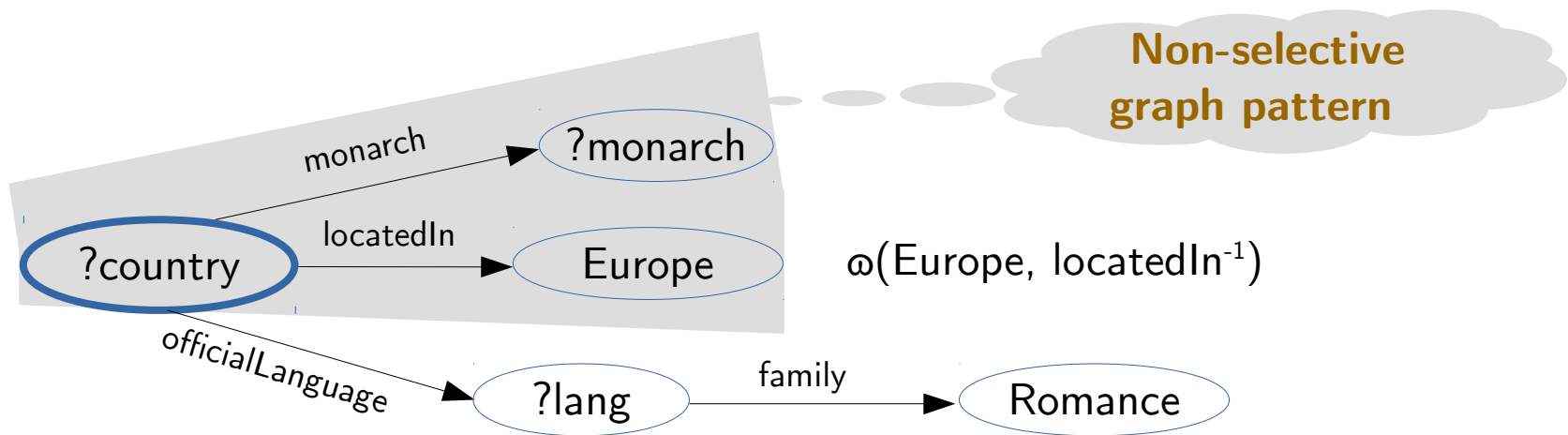
```
SELECT ?country WHERE {  
    ?country monarch ?monarch .  
    ?country locatedIn Europe .  
    ?country officialLanguage ?lang .  
    ?lang family Romance .  
}
```

Automatic oracle composition



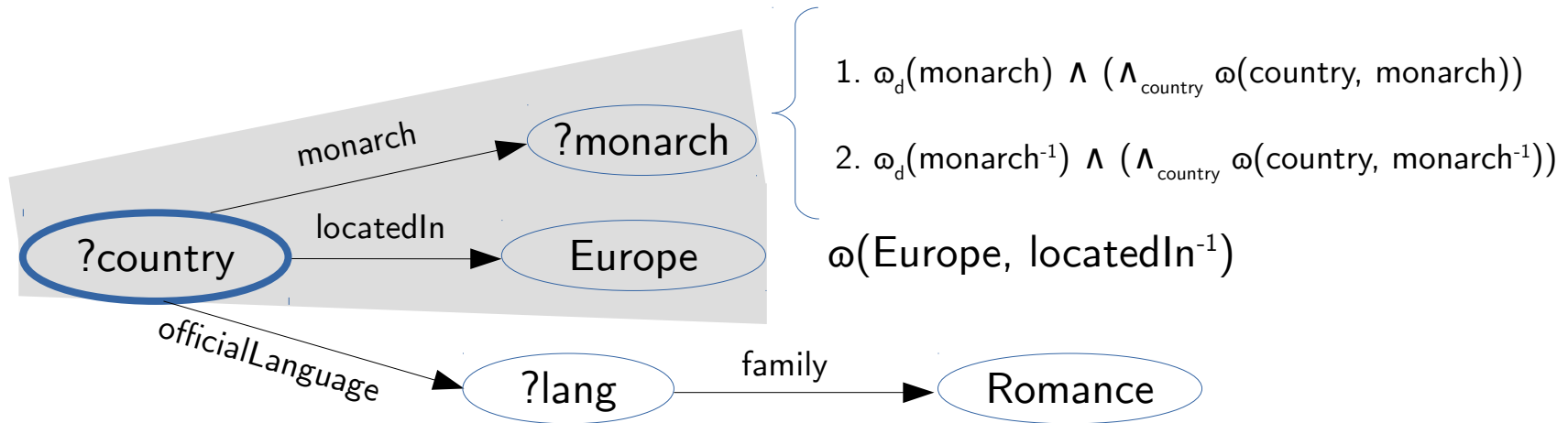
```
SELECT ?country WHERE {  
    ?country monarch ?monarch .  
    ?country locatedIn Europe .  
    ?country officialLanguage ?lang .  
    ?lang family Romance .  
}
```

Automatic oracle composition



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

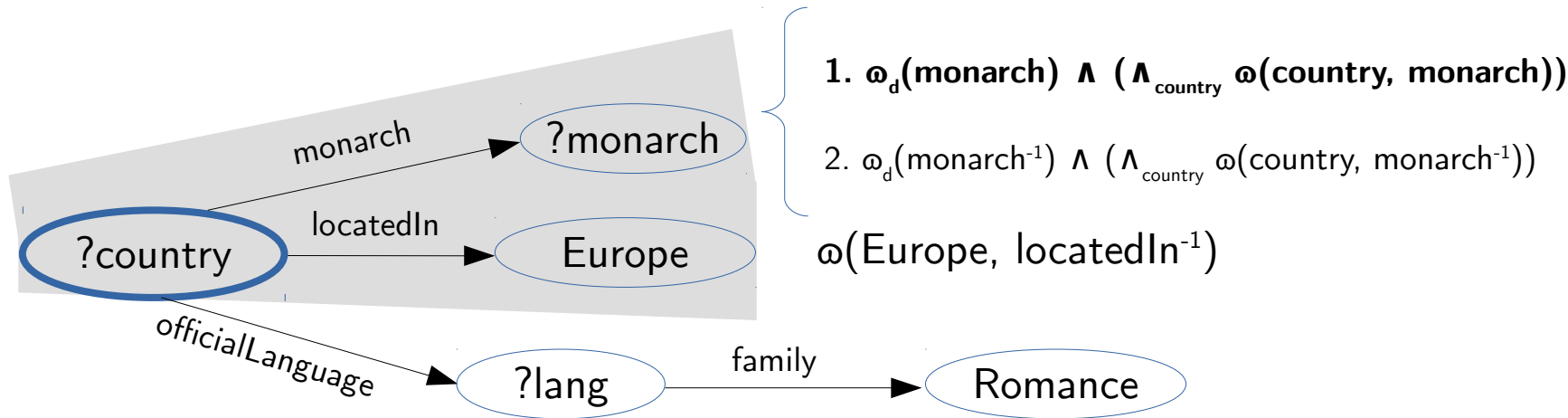
Automatic oracle composition



```

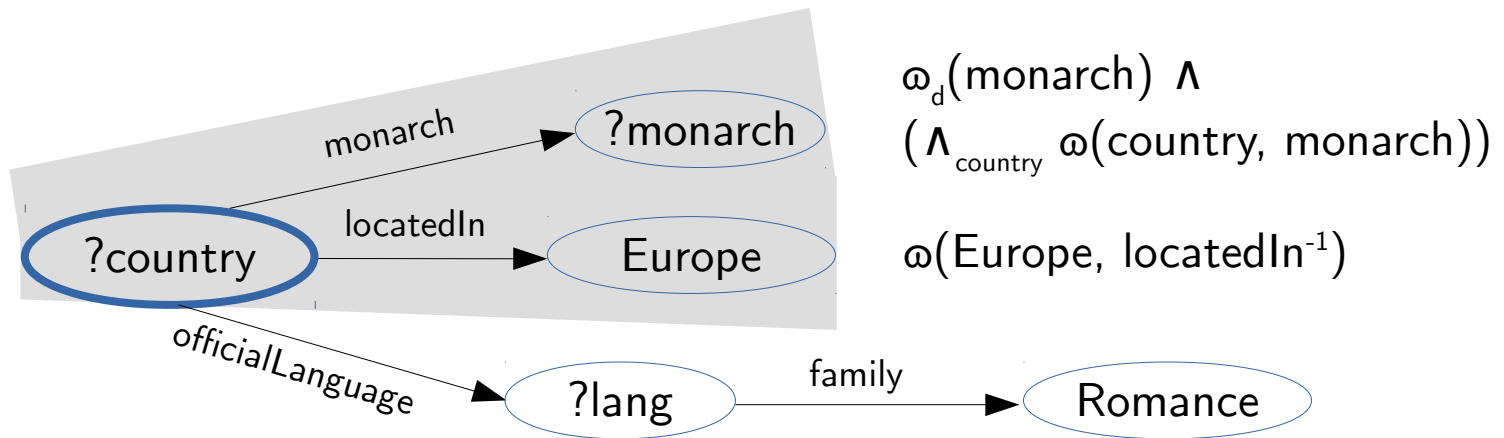
SELECT ?country WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?country officialLanguage ?lang .
    ?lang family Romance .
}
    
```

Automatic oracle composition



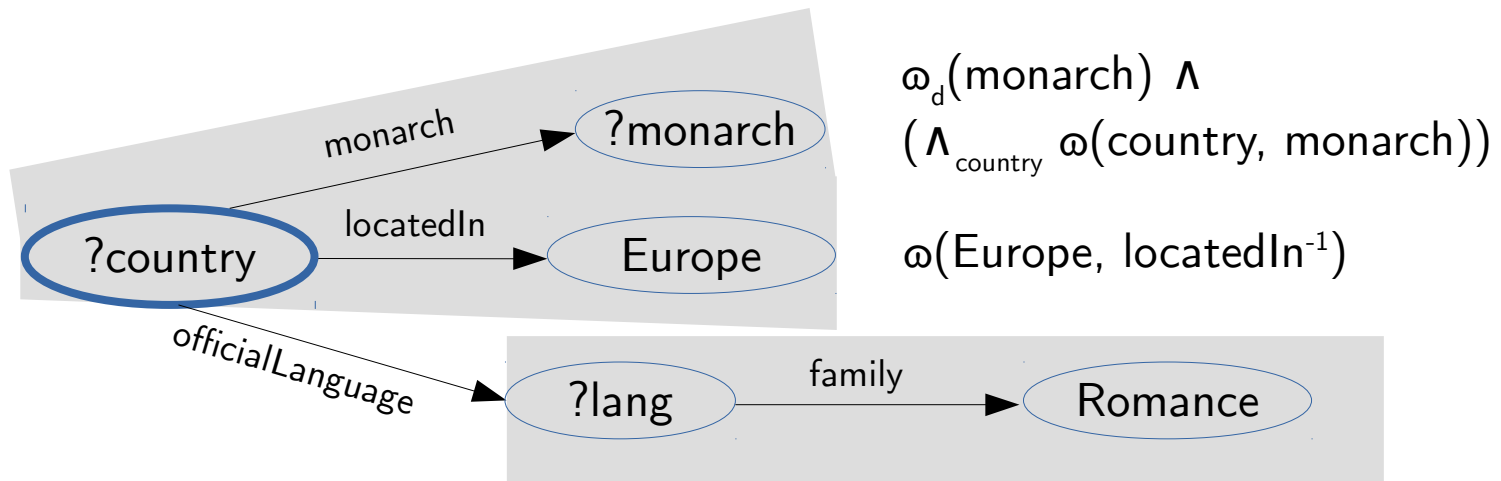
```
SELECT ?country WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?country officialLanguage ?lang .
    ?lang family Romance .
}
```

Automatic oracle composition



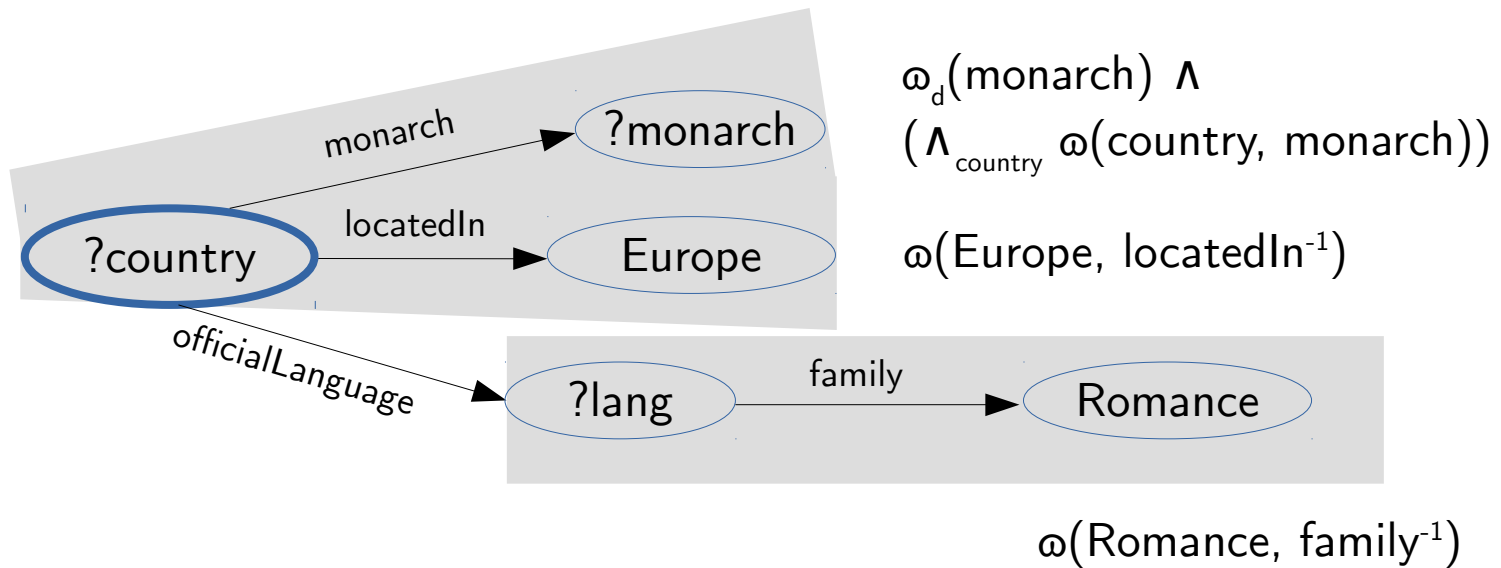
```
SELECT ?country WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?country officialLanguage ?lang .
    ?lang family Romance .
}
```

Automatic oracle composition



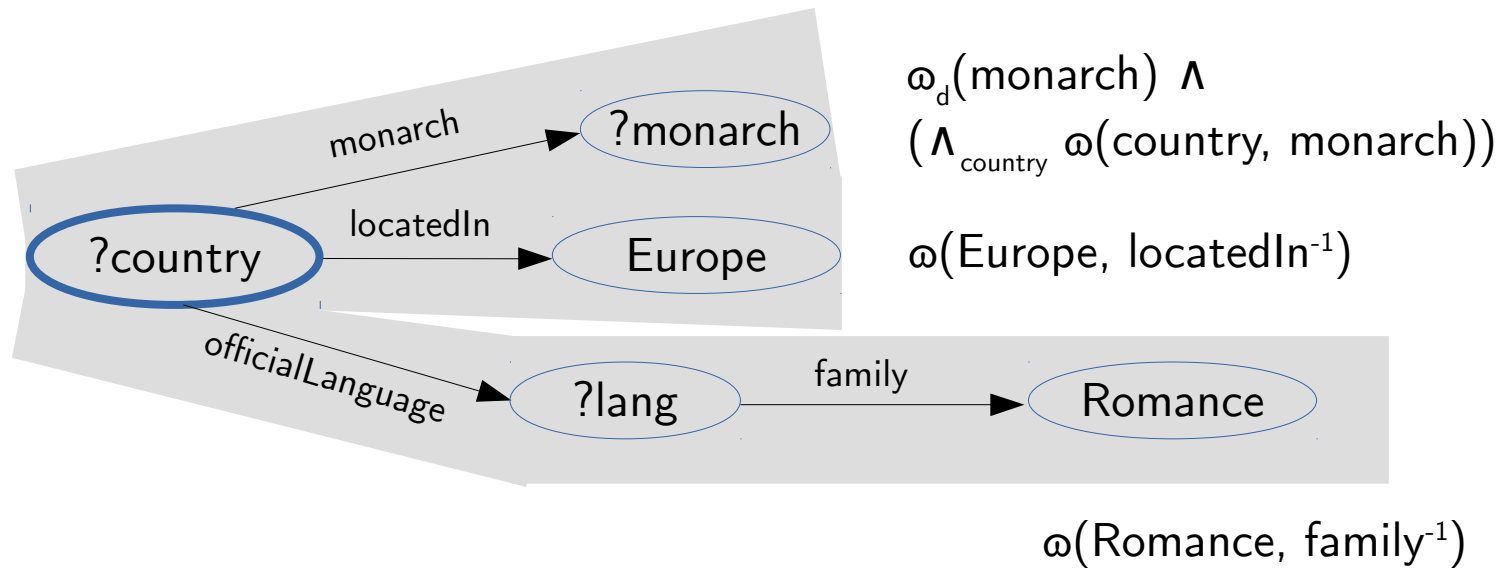
```
SELECT ?country WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?country officialLanguage ?lang .
    ?lang family Romance .
}
```

Automatic oracle composition



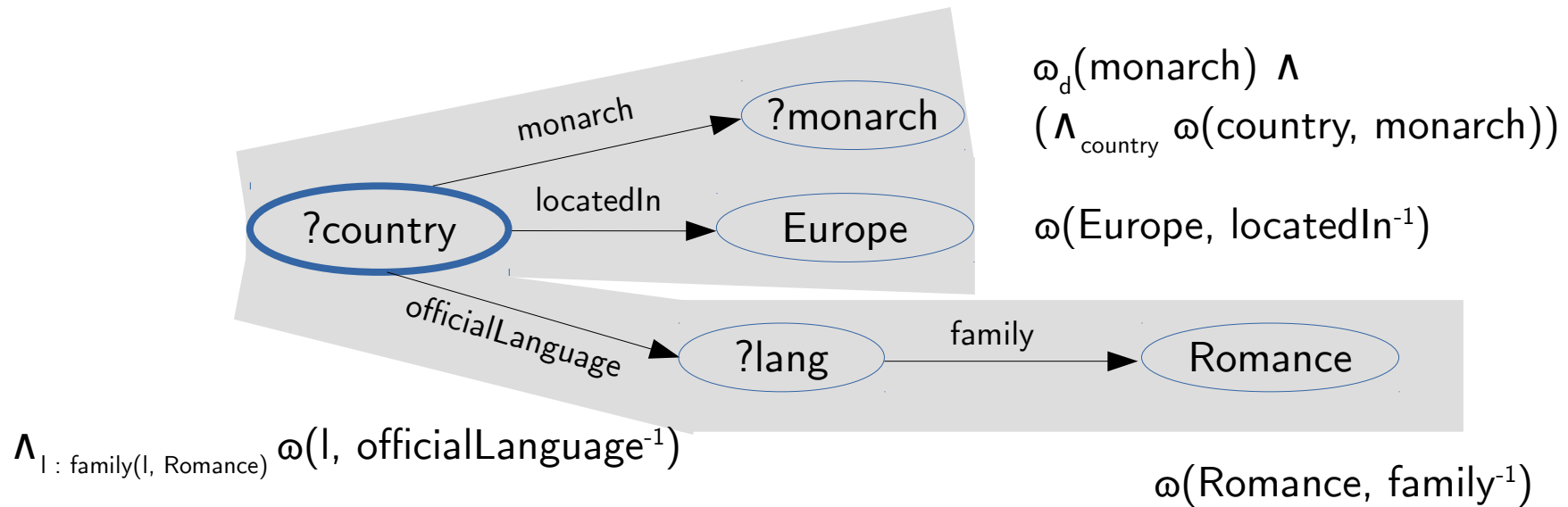
```
SELECT ?country WHERE {
    ?country monarch ?monarch .
    ?country locatedIn Europe .
    ?country officialLanguage ?lang .
    ?lang family Romance .
}
```

Automatic oracle composition



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

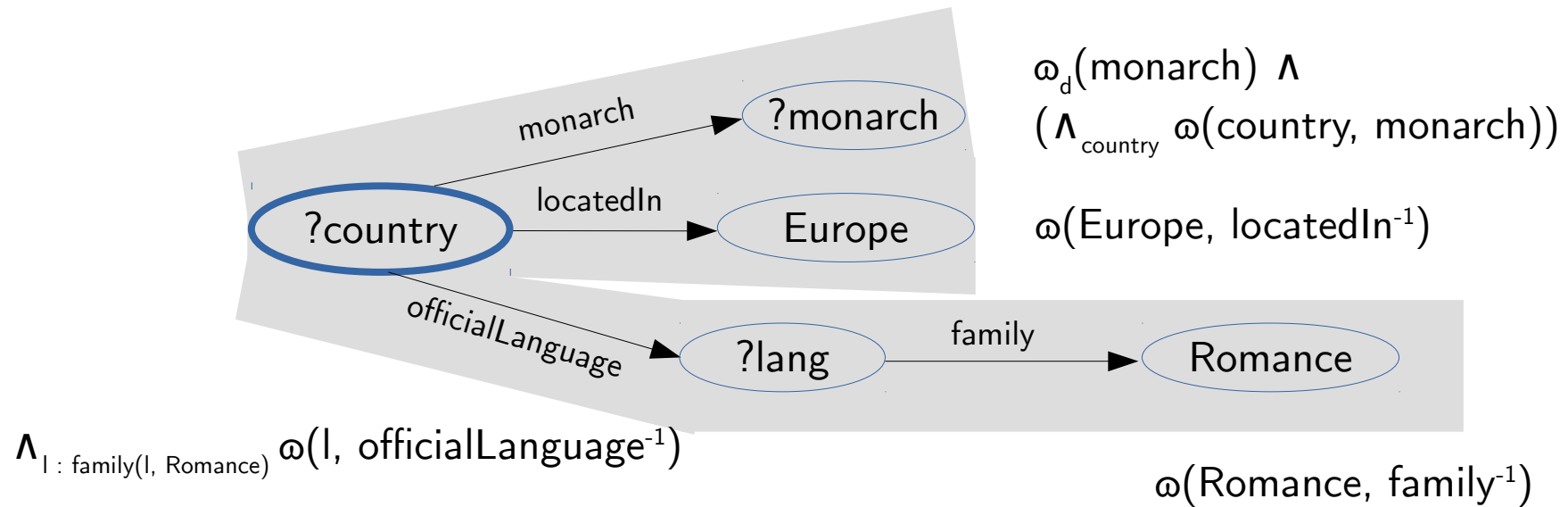
Automatic oracle composition



```

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

Automatic oracle composition

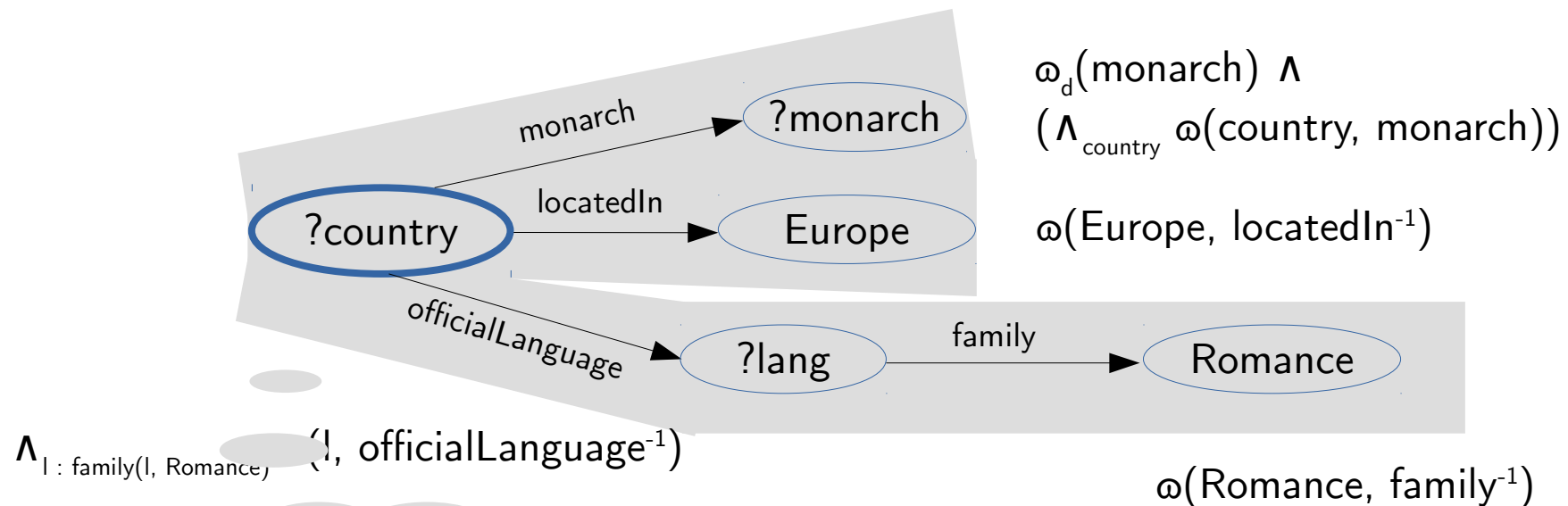


```

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

$$\omega' = \omega_d(\text{monarch}) \wedge (\Lambda_{\text{country}} \omega(\text{country}, \text{monarch})) \wedge \omega(\text{Europe}, \text{locatedIn}^{-1}) \wedge \omega(\text{Romance}, \text{family}^{-1}) \wedge (\Lambda_l : \text{family}(l, \text{Romance}) \omega(l, \text{officialLang}^{-1}))$$

Automatic oracle composition



Under bag semantics
the projection
variable is irrelevant

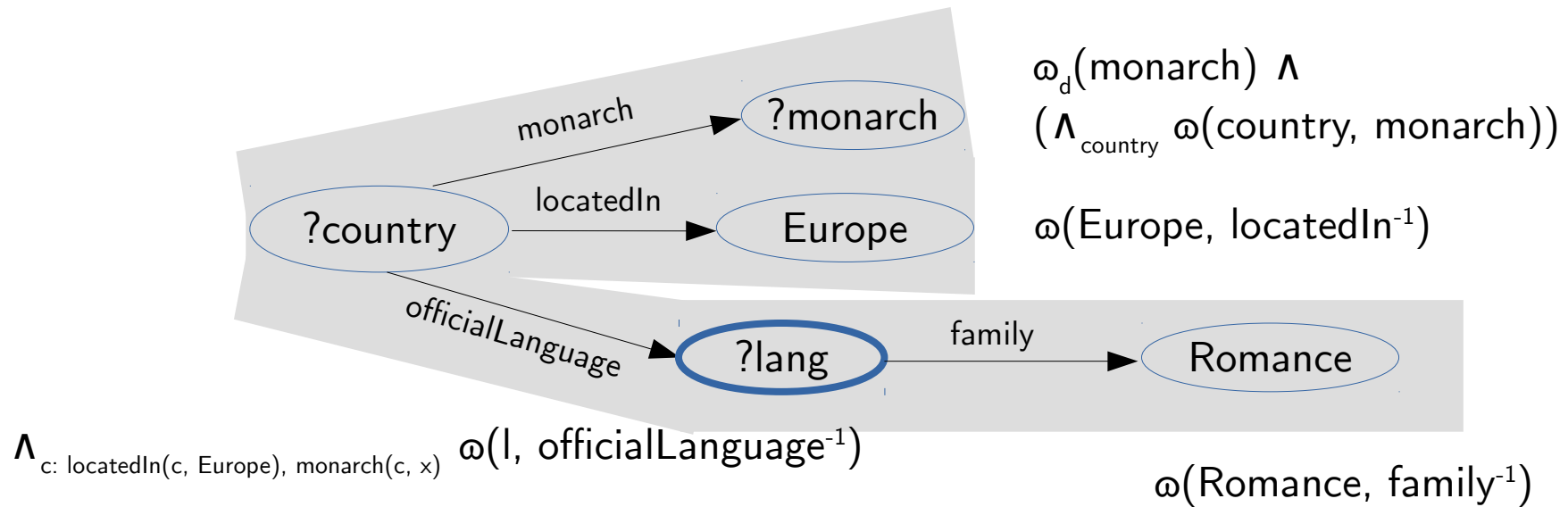
```

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

```

$$\omega' = \omega_d(\text{monarch}) \wedge (\Lambda_{\text{country}} \omega(\text{country}, \text{monarch})) \wedge \omega(\text{Europe}, \text{locatedIn}^{-1}) \wedge \omega(\text{Romance}, \text{family}^{-1}) \wedge (\Lambda_{l : \text{family}(l, \text{Romance})} \omega(l, \text{officialLang}^{-1}))$$

Automatic oracle composition

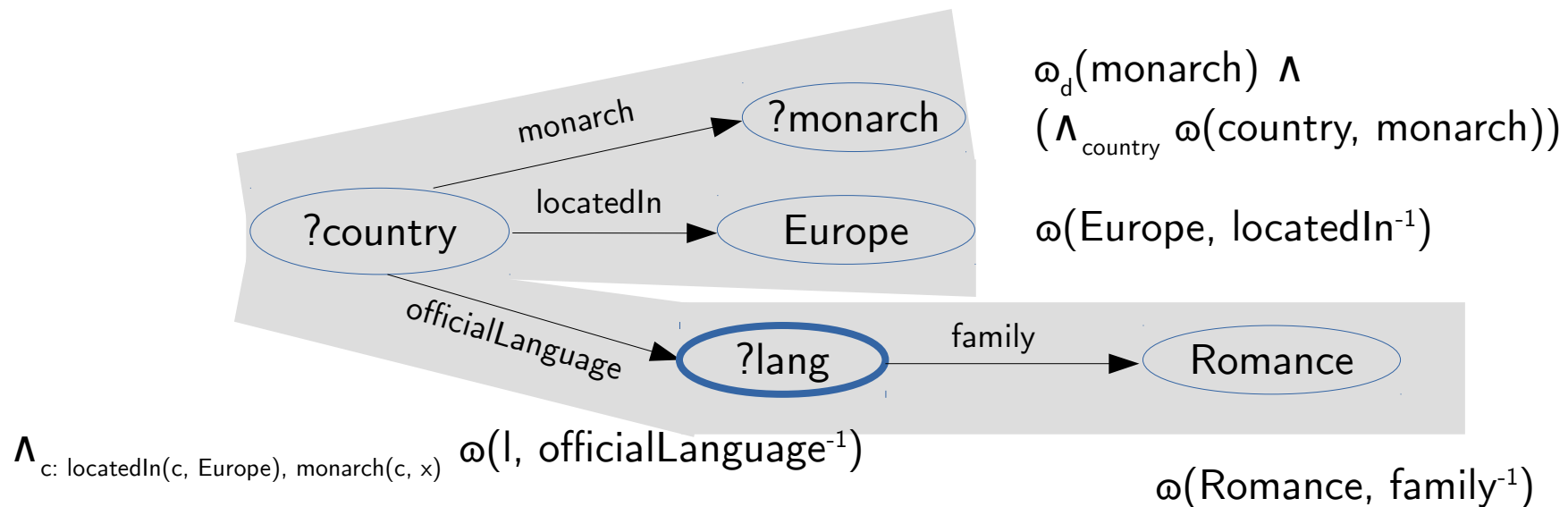


```

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

$$\omega'' = \omega_d(\text{monarch}) \wedge (\wedge_{\text{country}} \omega(\text{country}, \text{monarch})) \wedge \omega(\text{Europe}, \text{locatedIn}^{-1}) \wedge \omega(\text{Romance}, \text{family}^{-1}) \wedge (\wedge_{c: \text{locatedIn}(c, \text{Europe}), \text{monarch}(c, x)} \omega(l, \text{officialLanguage}^{-1}))$$

Automatic oracle composition

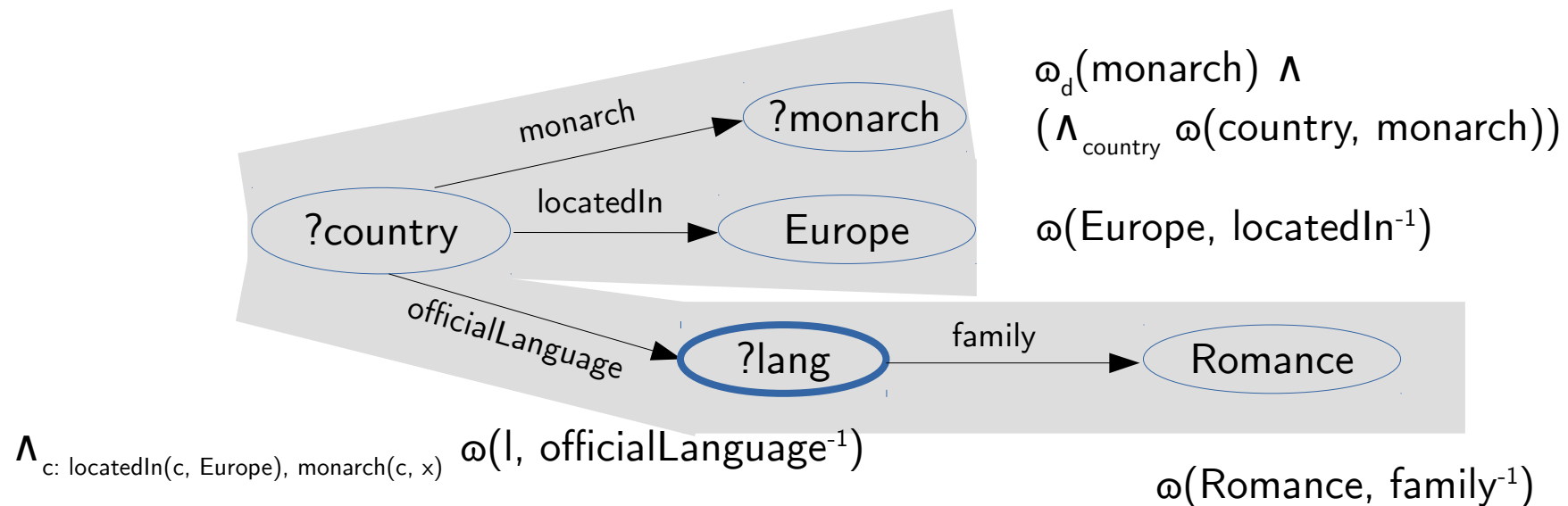


Challenge:
Evaluate first the oracles
with smaller cost!

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

$$\omega'' = \omega_d(\text{monarch}) \wedge (\Lambda_{\text{country}} \omega(\text{country}, \text{monarch})) \wedge \omega(\text{Europe}, \text{locatedIn}^{-1}) \wedge \omega(\text{Romance}, \text{family}^{-1}) \wedge (\Lambda_{c: \text{locatedIn}(c, \text{Europe}), \text{monarch}(c, x)} \omega(l, \text{officialLanguage}^{-1}))$$

Automatic oracle composition



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

Projection variable matters
under set semantics

$$\omega'' = \omega_d(\text{monarch}) \wedge (\Lambda_{\text{country}} \omega(\text{country}, \text{monarch})) \wedge \omega(\text{Europe}, \text{locatedIn}^{-1}) \wedge \omega(\text{Romance}, \text{family}^{-1}) \wedge (\Lambda_{c: \text{locatedIn}(c, \text{Europe}), \text{monarch}(c, x)} \omega(l, \text{officialLanguage}^{-1}))$$

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 {  
    ?county inState ?state .  
    ?county spanishSpeakers ?nspeak .  
} 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*



Boolean aggregation
function on sets of bindings

```
SELECT ?state sum(?nspeak) WHERE {  
    ?county inState ?state .  
    ?county spanishSpeakers ?nspeak .  
}  
GROUP BY ?state HAVING (complete(?nspeak))
```

Enabling completeness in SPARQL

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

<i>?state</i>	<i>?county</i>	<i>?nspeak</i>
Delaware	New Castle	2000
	Kent	4300
	Sussex	1200
Hawaii	Hawaii	30000
	Kalawao	1200

Complete list?



```
SELECT ?state sum(?nspeak) WHERE {  
  ?county inState ?state .  
  ?county spanishSpeakers ?nspeak .  
} GROUP BY ?state HAVING (complete(?nspeak))
```

Enabling completeness in SPARQL

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

<i>?state</i>	<i>?county</i>	<i>?nspeak</i>
Delaware	New Castle	2000
	Kent	4300
	Sussex	1200
Hawaii	Hawaii	30000
	Kalawao	1200

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

```
SELECT ?state sum(?nspeak) WHERE {  
  ?county inState ?state .  
  ?county spanishSpeakers ?nspeak .  
} GROUP BY ?state HAVING (complete(?nspeak))
```

Enabling completeness in SPARQL

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

Completeness oracles
to the rescue!

?state	?county	?nspeak
Delaware	New Castle	2000
	Kent	4300
	Sussex	1200
Hawaii	Hawaii	30000
	Kalawao	1200

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

```
SELECT ?state sum(?nspeak) WHERE {  
  ?county inState ?state .  
  ?county spanishSpeakers ?nspeak .  
} GROUP BY ?state HAVING (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.