

Enabling Completeness-aware Querying in SPARQL

Luis Galárraga, Katja Hose, Simon Razniewski

May 14th, 2017

WebDB, Chicago

Outline

- Completeness in RDF knowledge bases
- Completeness oracles
- Our vision
 - Representations for completeness oracles
 - Reasoning with completeness oracles
 - Enabling completeness in SPARQL
- Summary & conclusions

Outline

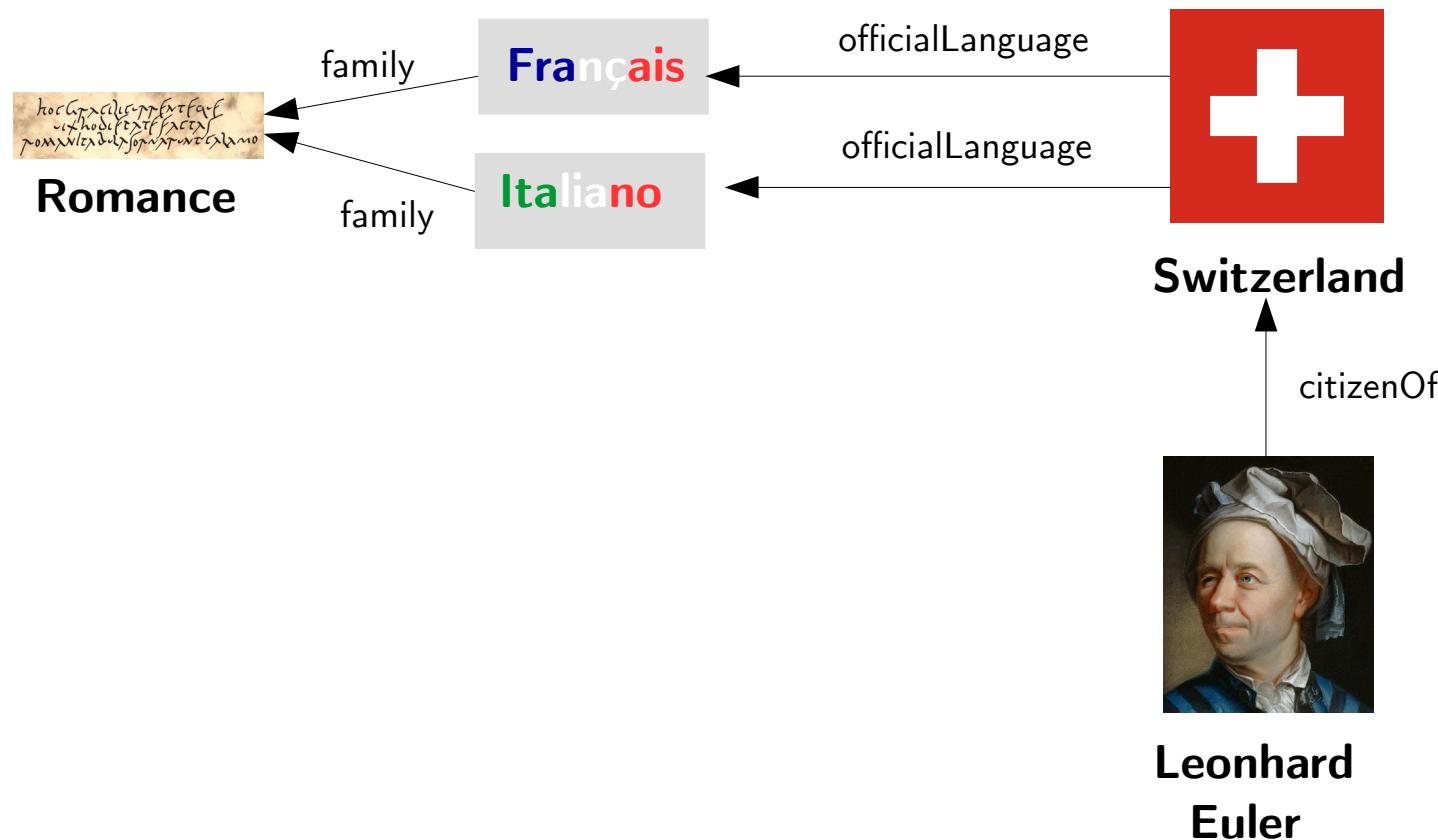
- Completeness in RDF knowledge bases
- Completeness oracles
- Our vision
 - Representations for completeness oracles
 - Reasoning with completeness oracles
 - Enabling completeness in SPARQL
- Summary & conclusions

Outline

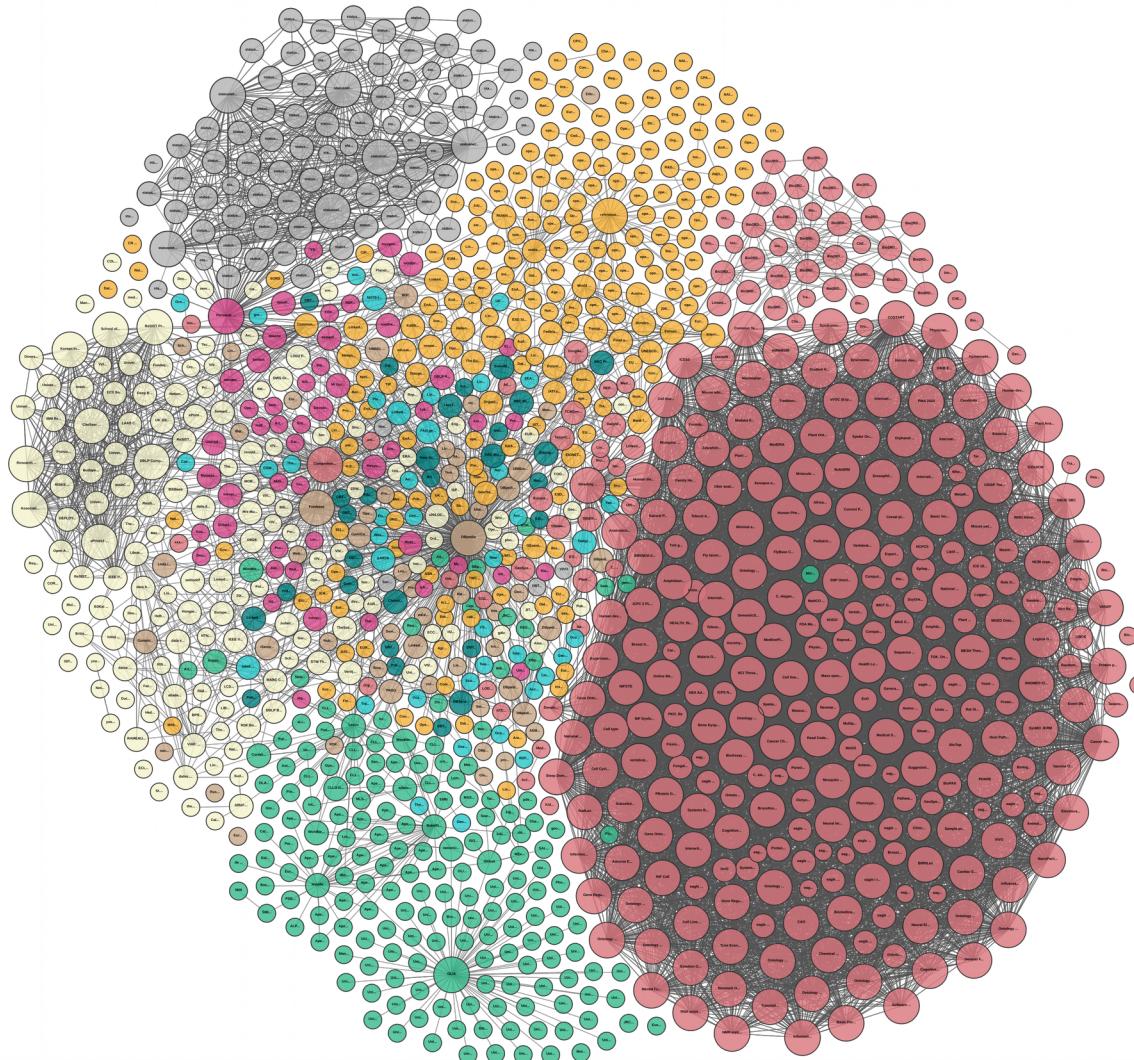
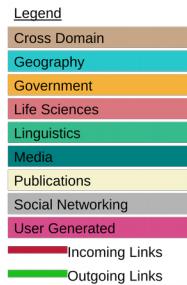
- Completeness in **RDF knowledge bases**
- Completeness oracles
- Our vision
 - Representations for completeness oracles
 - Reasoning with completeness oracles
 - Enabling completeness in SPARQL
- Summary & conclusions

RDF Knowledge Bases (KBs)

Collection of structured knowledge



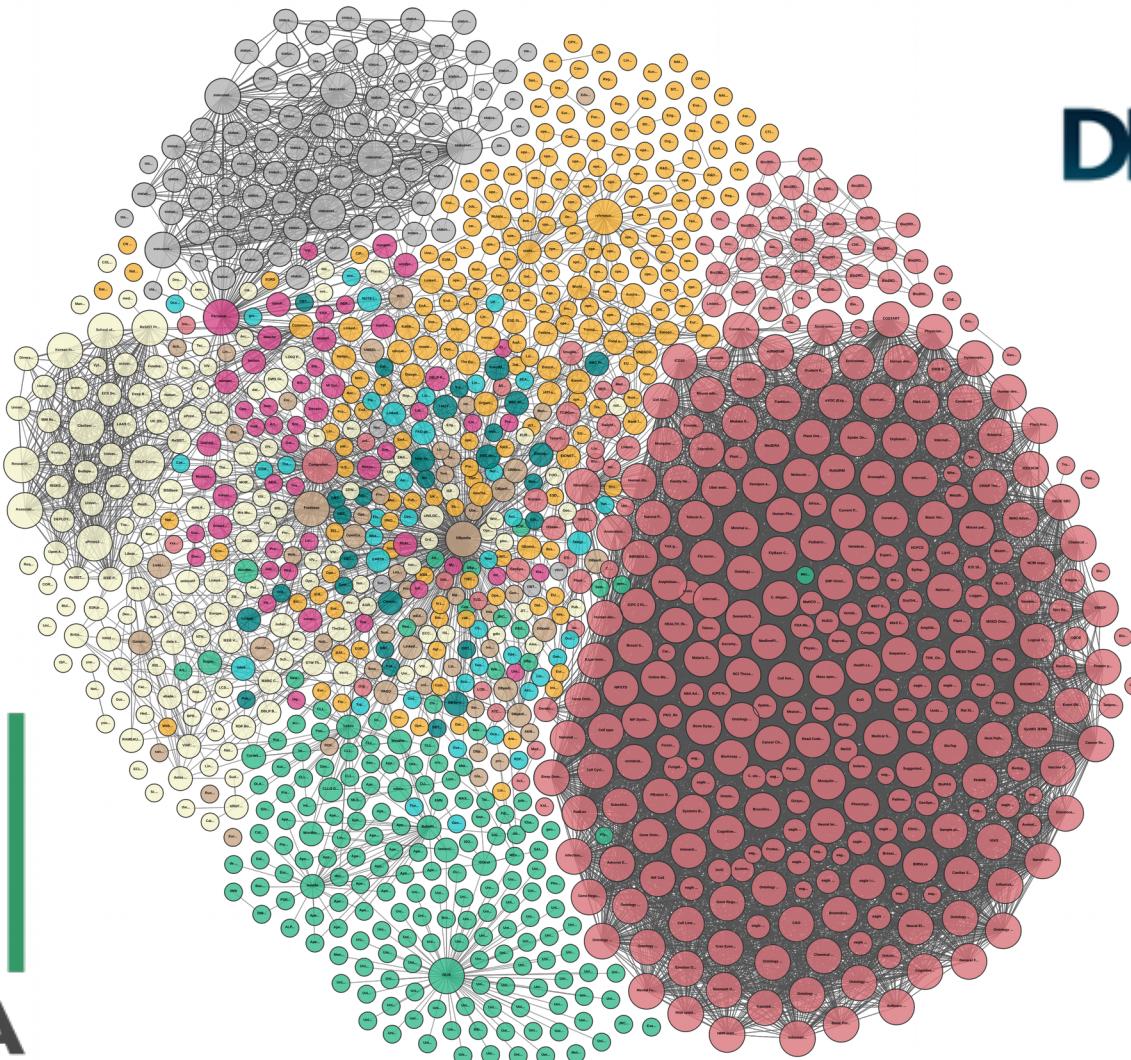
Plenty of KBs out there!



Plenty of KBs out there!

Legend

- Cross Domain
- Geography
- Government
- Life Sciences
- Linguistics
- Media
- Publications
- Social Networking
- User Generated
- Incoming Links
- Outgoing Links



KBs in action

official languages of switzerland 

All Shopping News Images Maps More Settings Tools

 Switzerland > Official languages

French	Romansh
German	Italian

Outline

- Completeness in RDF knowledge bases
- Completeness oracles
- Our vision
 - 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

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

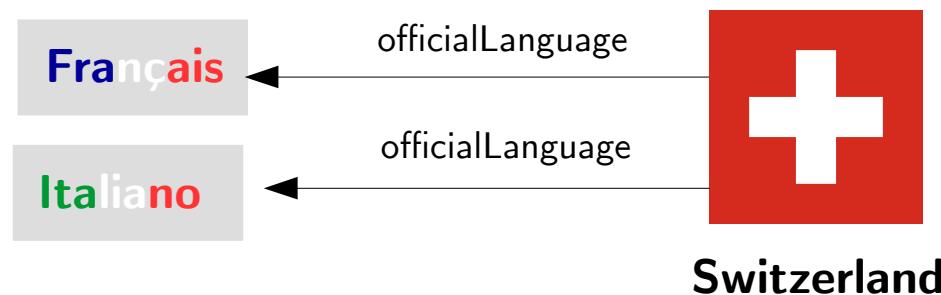
Completeness

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

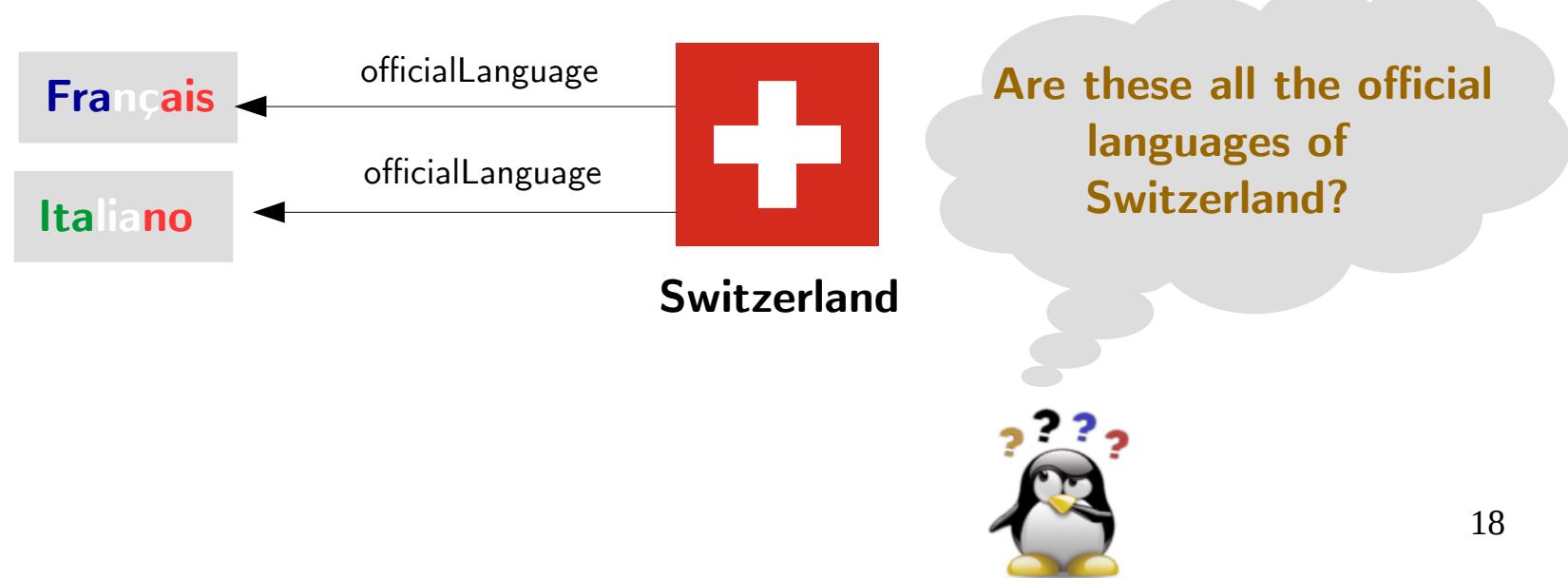
SELECT ?x WHERE { Switzerland officialLang ?x }



Completeness

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

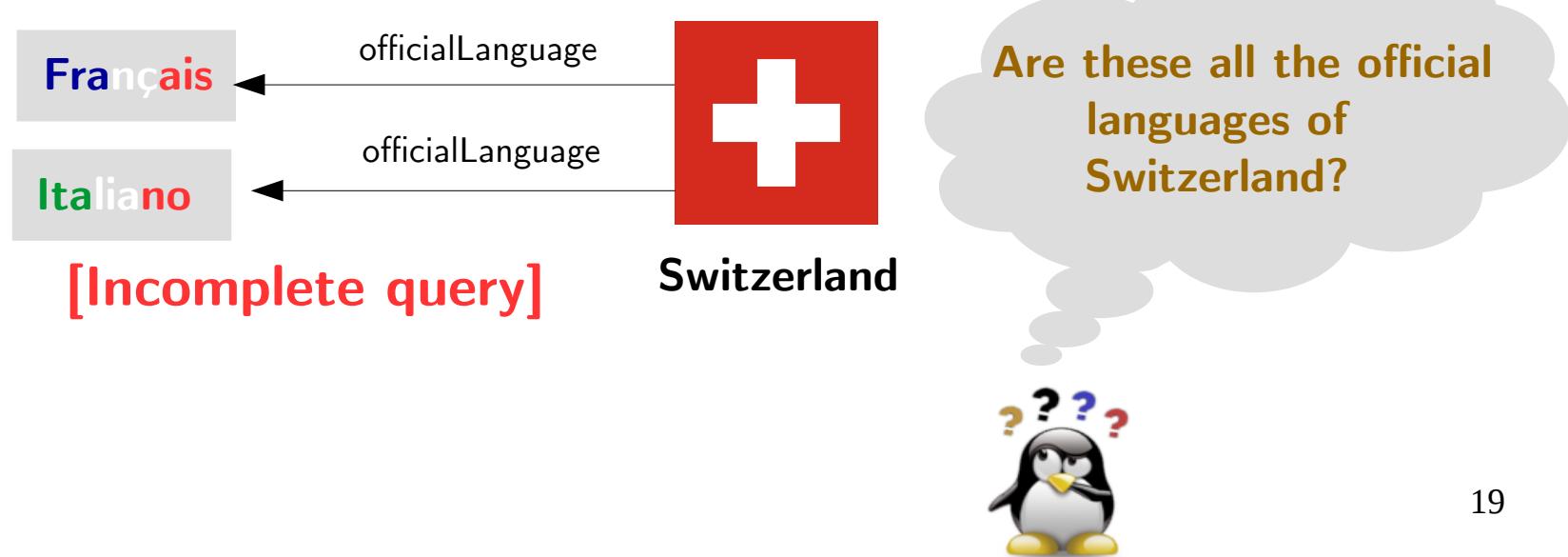
SELECT ?x WHERE { Switzerland officialLang ?x }



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 officialLang ?x }



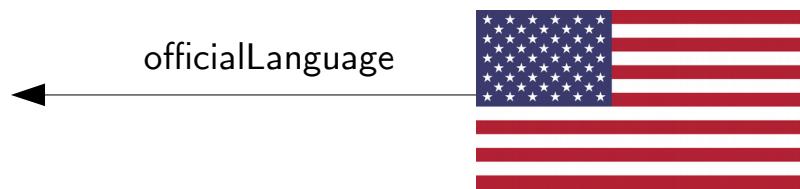
Completeness in RDF data

- Wikidata provides *no value annotations*

Completeness in RDF data

- Wikidata provides *no value annotations*

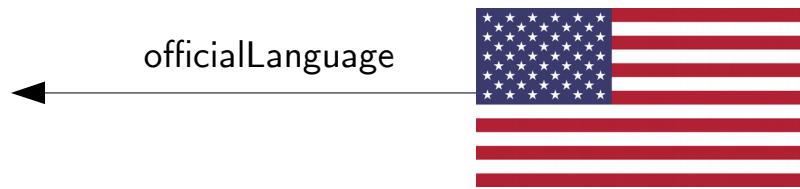
```
SELECT ?x WHERE { USA officialLang ?x }
```



Completeness in RDF data

- Wikidata provides *no value annotations*

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

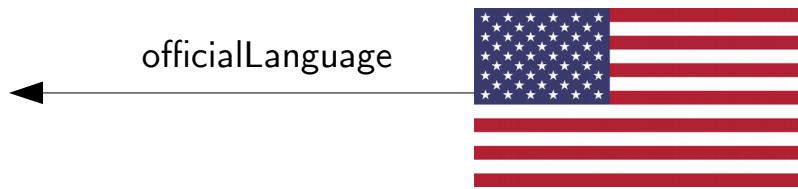


[Complete query]

Completeness in RDF data

- Wikidata provides *no value annotations*

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



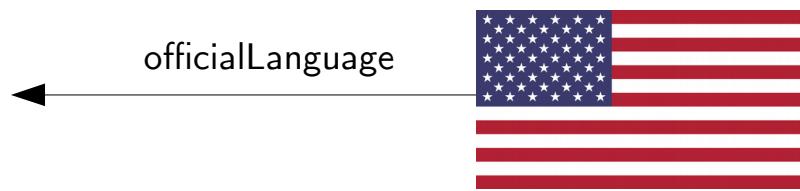
[Complete query]

- Not applicable if we know some official language

Completeness in RDF data

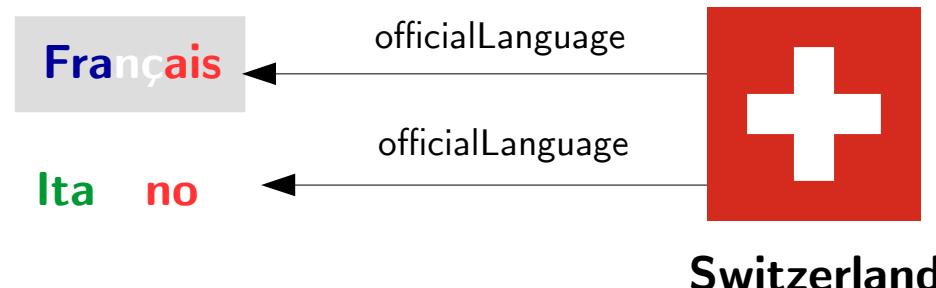
- Wikidata provides *no value annotations*

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



[Complete query]

- Not applicable if we know some official language



Outline

- Completeness in RDF knowledge bases
- **Completeness oracles**
- Our vision
 - 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

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

```
SELECT ?x WHERE { subject relation ?x }
```

SR completeness oracle

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

SELECT ?x WHERE { subject relation ?x }

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

SR completeness oracle

- Function ω that guesses the completeness of queries of the form [Galárraga et. al, 2017]:
$$\text{SELECT ?x WHERE \{} \text{subject relation ?x} \}$$
- We use the notation $\omega(\text{subject, relation})$
- $\omega = \text{pca}(s, r)$ = partial completeness assumption

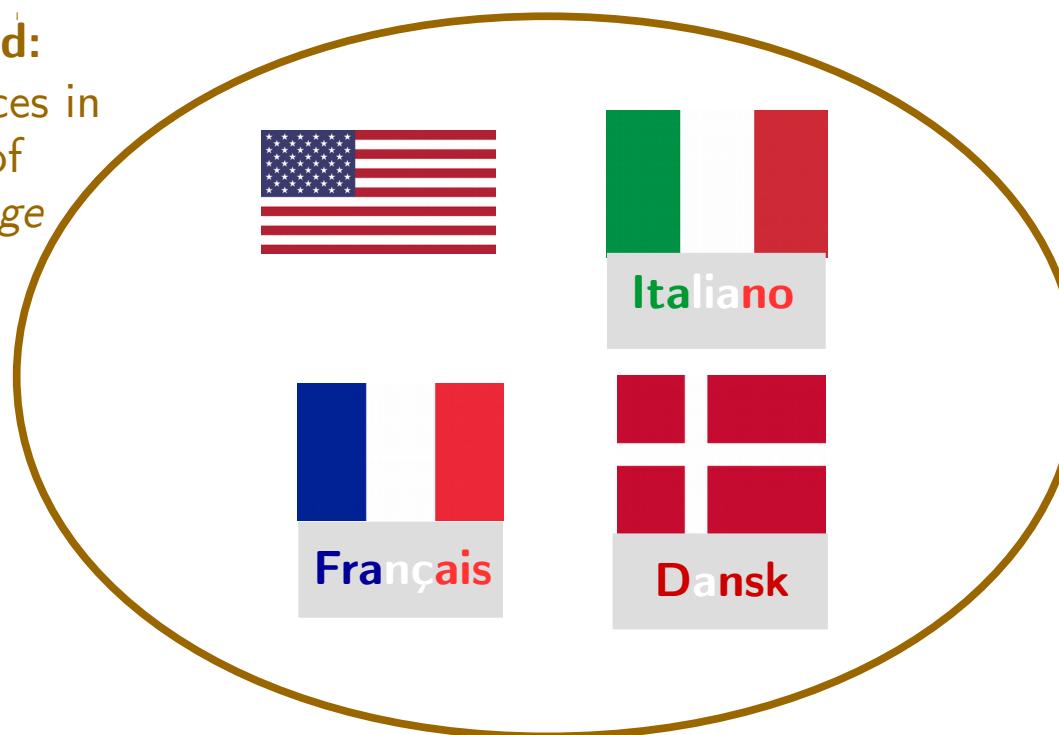
SR completeness oracle

- Function ω that guesses the completeness of queries of the form [Galárraga et. al, 2017]:
$$\text{SELECT ?x WHERE \{} \text{subject relation ?x} \}$$
- We use the notation $\omega(\text{subject, relation})$
- $\omega = \text{pca}(s, r)$ = partial completeness assumption
 - Query is **complete** in KB if at least one answer is known

Evaluating SR oracles

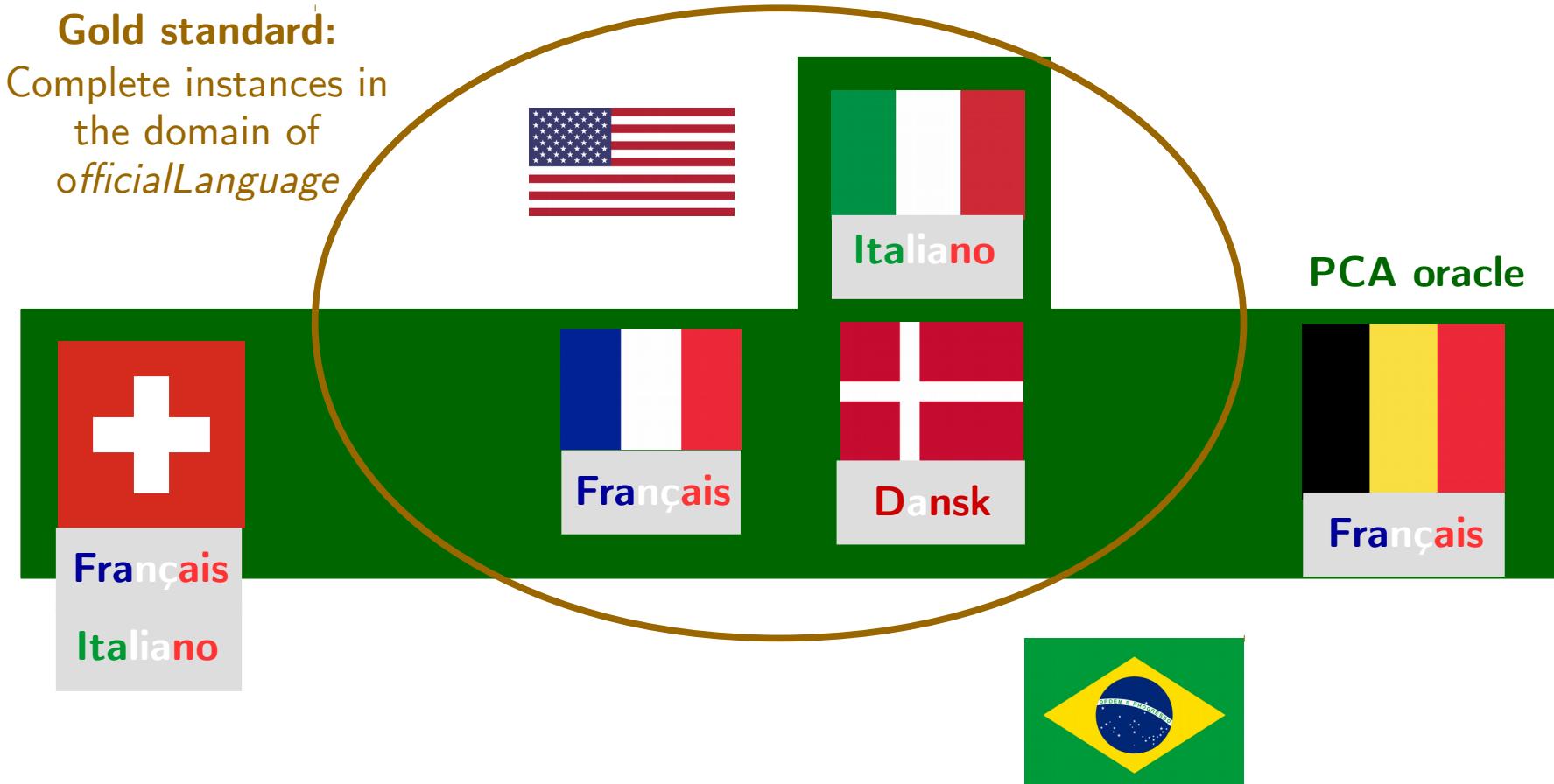
$\omega = pca(s, r) = \text{partial completeness assumption}$

Gold standard:
Complete instances in
the domain of
officialLanguage



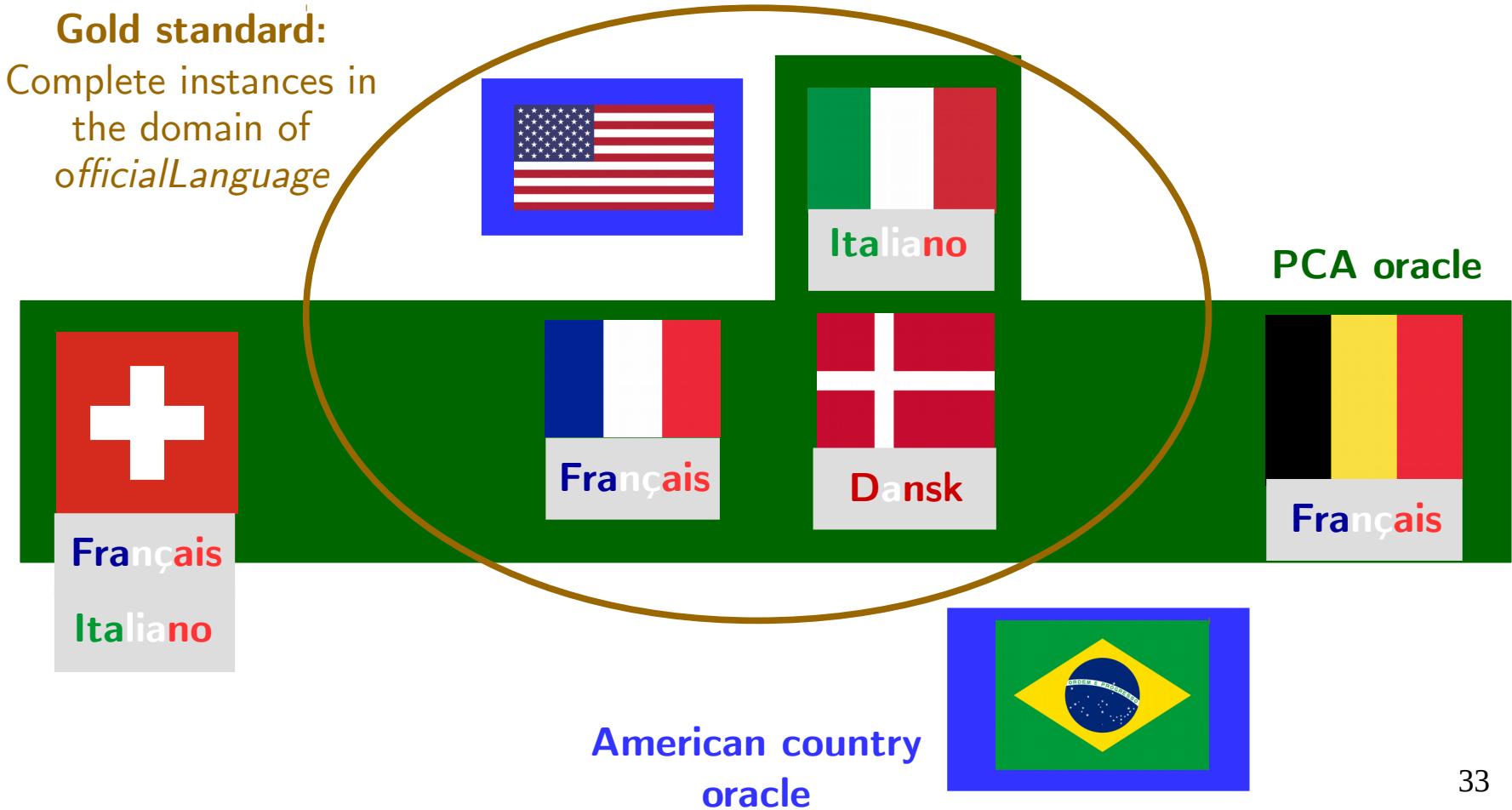
Evaluating SR oracles

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



Evaluating SR oracles

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



Evaluating SR oracles

PCA oracle

Precision = 3/5

Recall = 3/4

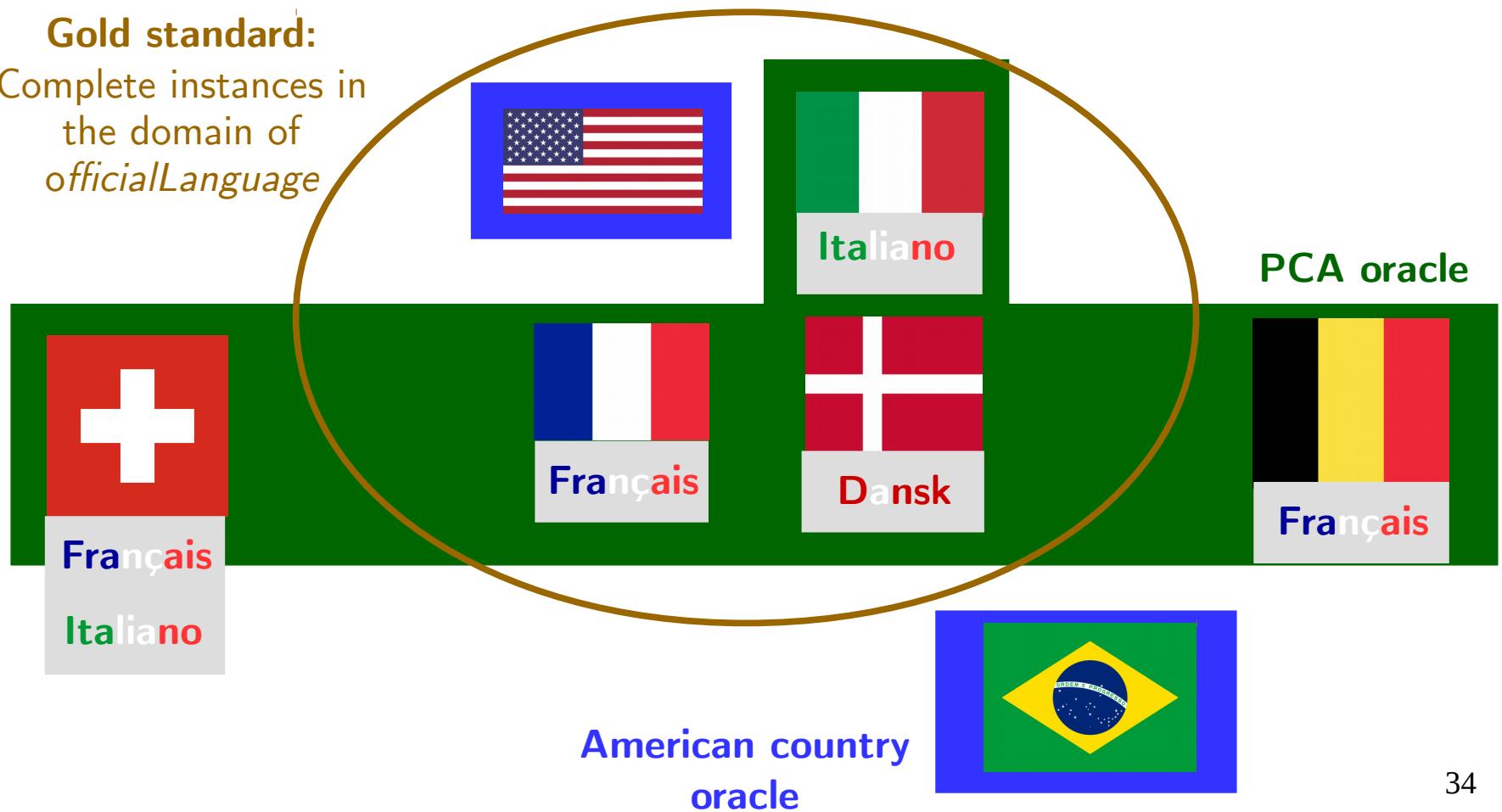
American country oracle

Precision = 1/2

Recall = 1/4

Gold standard:

Complete instances in
the domain of
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: $\text{popularity}_{\text{pop}}(s, r) = \text{pop}(s)$
- No-chg over time: $\text{nochange}_{\text{chg}}(s, r) = \sim \text{chg}(s, r)$
- Star : $\text{star}_{r_1, \dots, r_n}(s, r) = \forall i \in \{1, \dots, n\} : \exists o : r_i(s, o)$
- 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

`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*

Outline

- Completeness in RDF knowledge bases
- Completeness oracles
- Our vision
 - 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

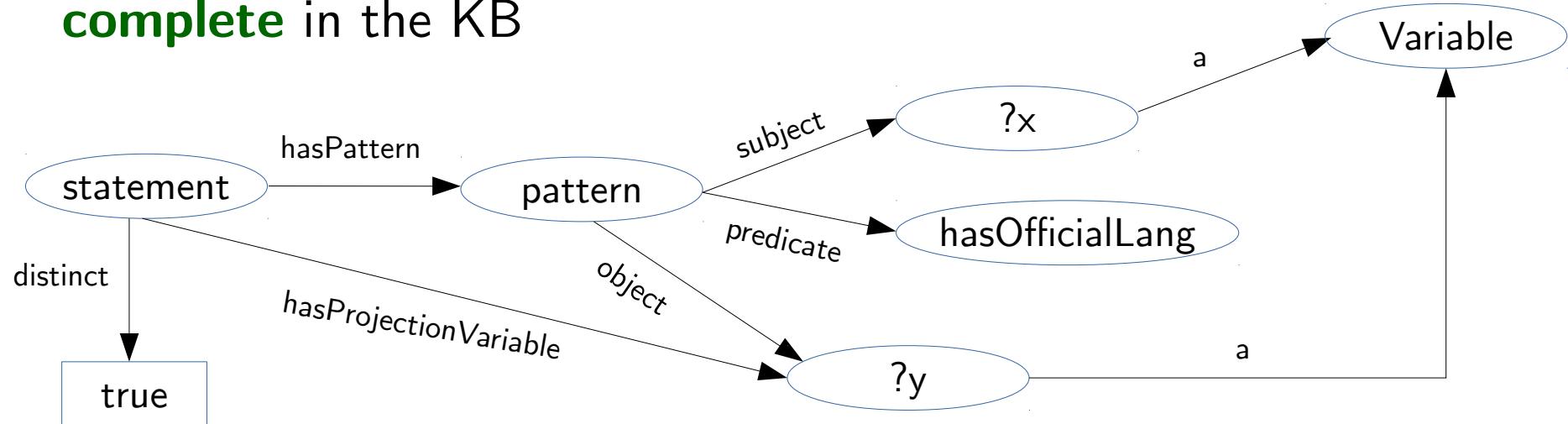
- Extensional approach [Darari, et. Al, 2013]
 - An oracle is a collection of completeness statements about queries

SELECT DISTINCT ?y WHERE { ?x hasOfficialLang ?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

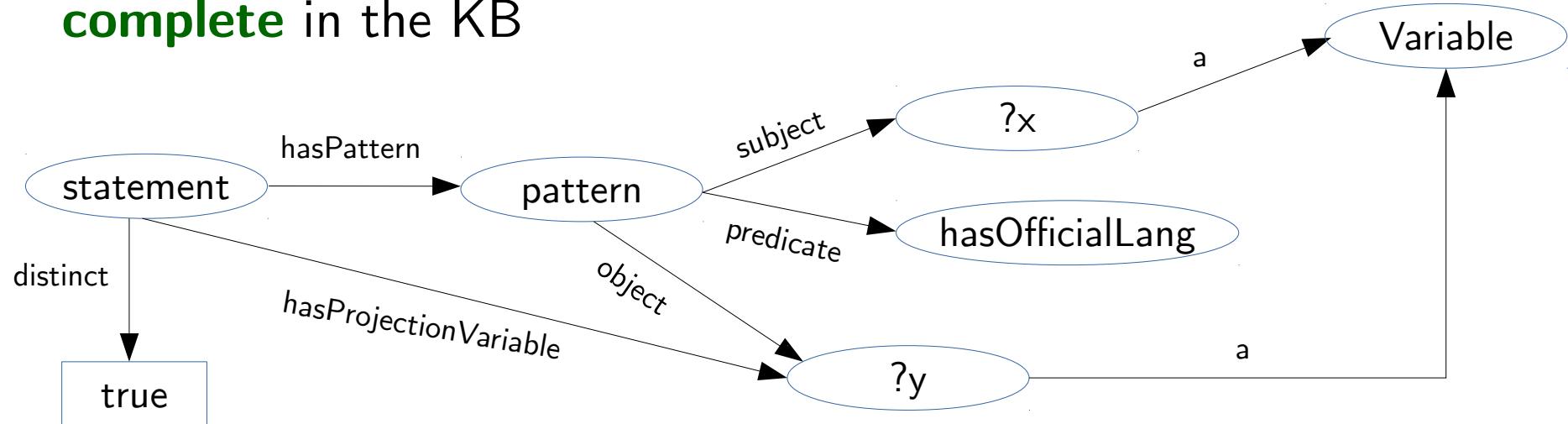
SELECT DISTINCT ?y WHERE { ?x hasOfficialLang ?y } is **complete** in the KB



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 hasOfficialLang ?y } is **complete** in the KB

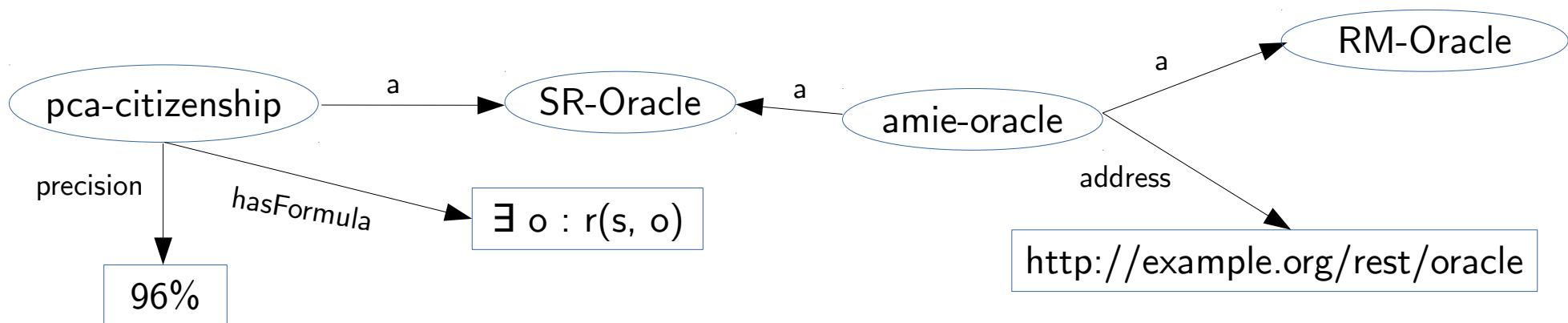


Representing completeness oracles

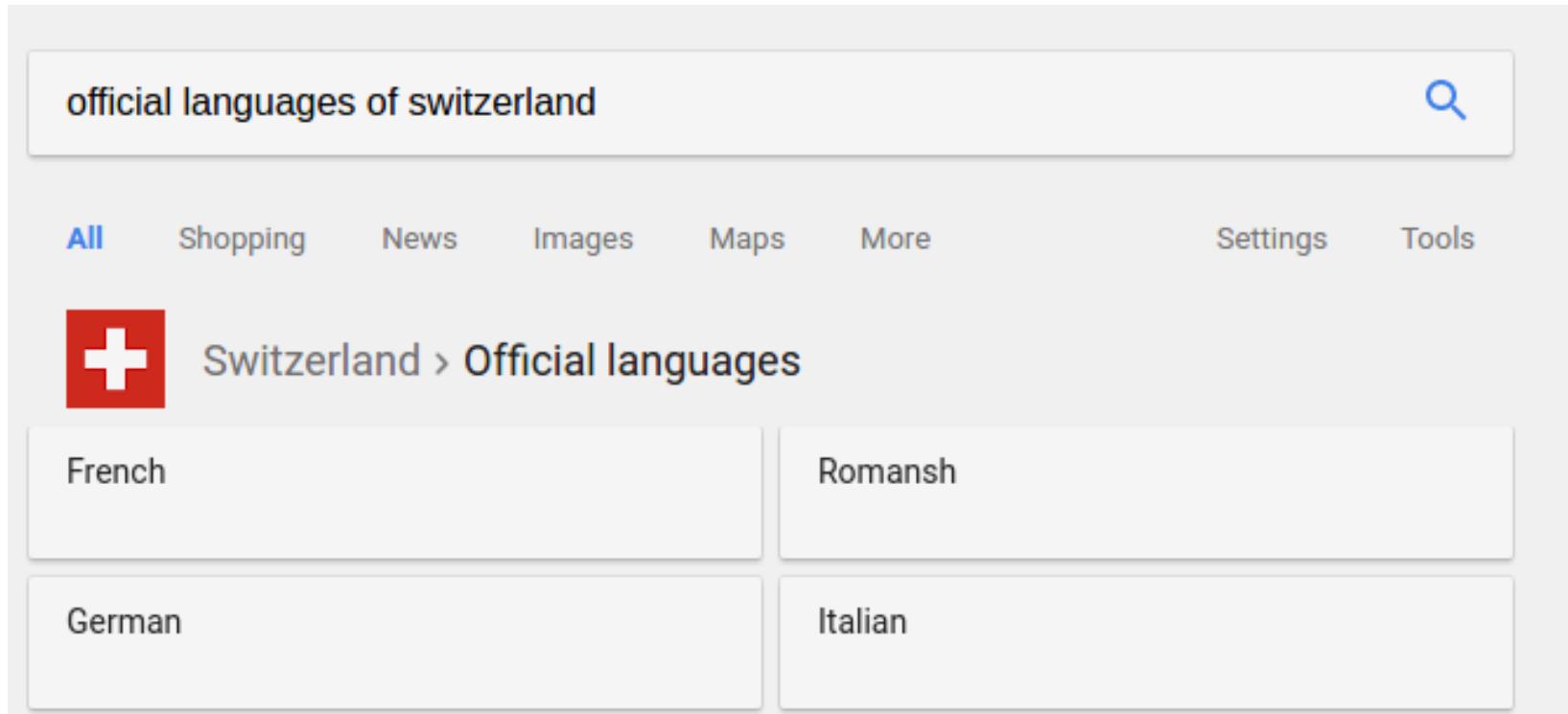
- Intensional approach
 - The oracle logic is embedded as a lambda function or a link to a program or resource

Representing completeness oracles

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



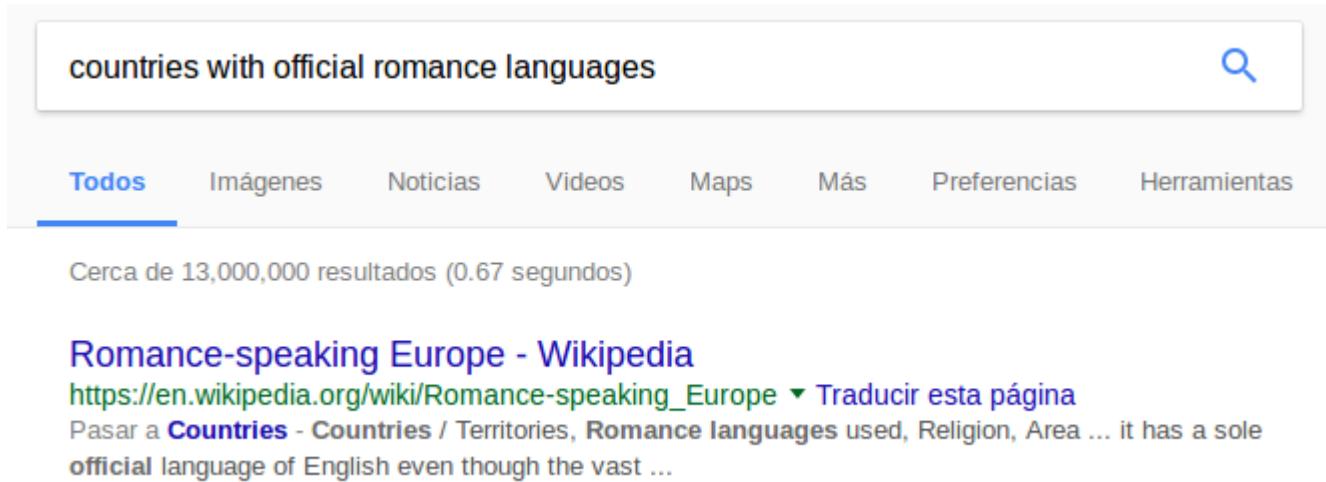
Providing completeness guarantees



A screenshot of a search results page. The search bar at the top contains the text "official languages of switzerland". Below the search bar is a navigation bar with categories: All, Shopping, News, Images, Maps, More, Settings, and Tools. The "All" category is highlighted in blue. The main content area shows a result for "Switzerland > Official languages". Below this, there are four cards arranged in a 2x2 grid. The top-left card contains the text "French". The top-right card contains the text "Romansh". The bottom-left card contains the text "German". The bottom-right card contains the text "Italian".

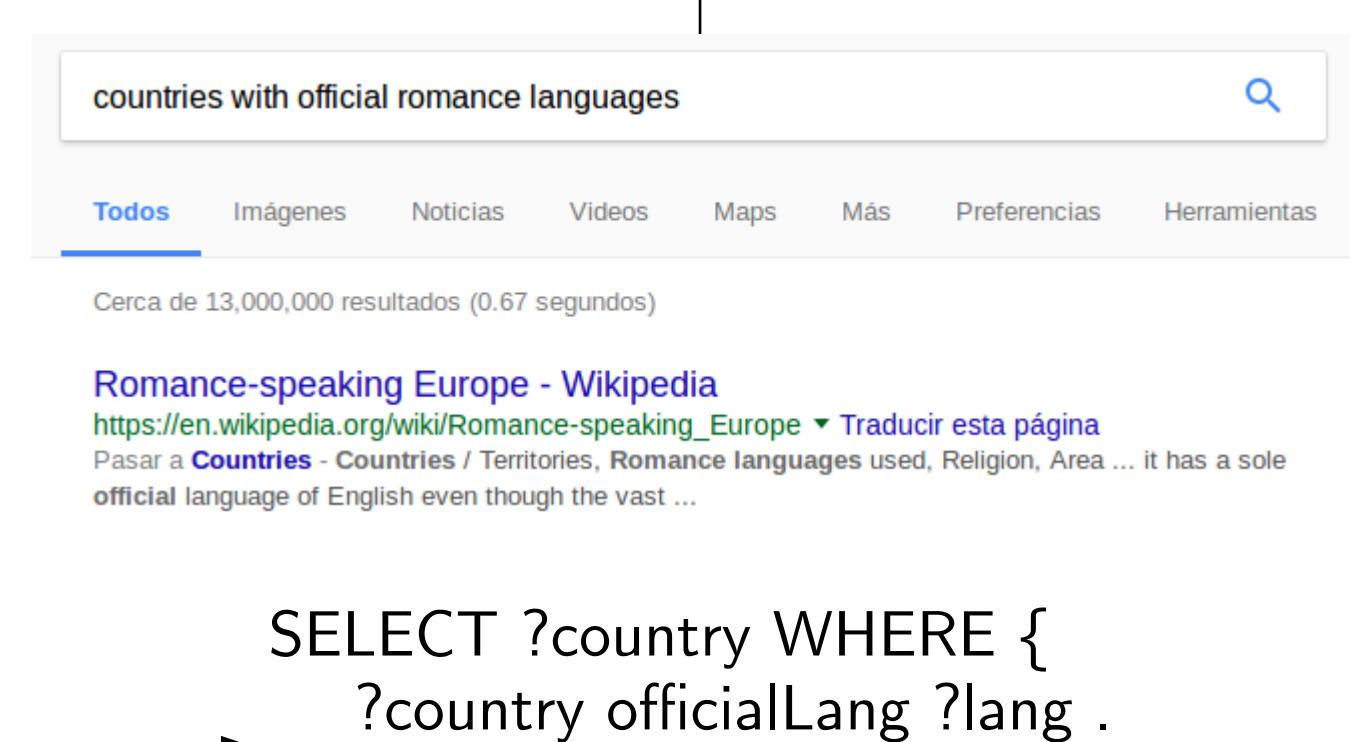
List of results is complete according to oracle ω with confidence X

Providing completeness guarantees



A screenshot of a search engine results page. The search bar at the top contains the query "countries with official romance languages". Below the search bar is a navigation menu with tabs: "Todos" (which is underlined in blue), "Imágenes", "Noticias", "Videos", "Maps", "Más", "Preferencias", and "Herramientas". The main content area shows a search result for "Romance-speaking Europe - Wikipedia". The result includes the title, a link to the page (https://en.wikipedia.org/wiki/Romance-speaking_Europe), a "Traducir esta página" button, and a snippet of text: "Pasar a **Countries** - Countries / Territories, Romance languages used, Religion, Area ... it has a sole official language of English even though the vast ...".

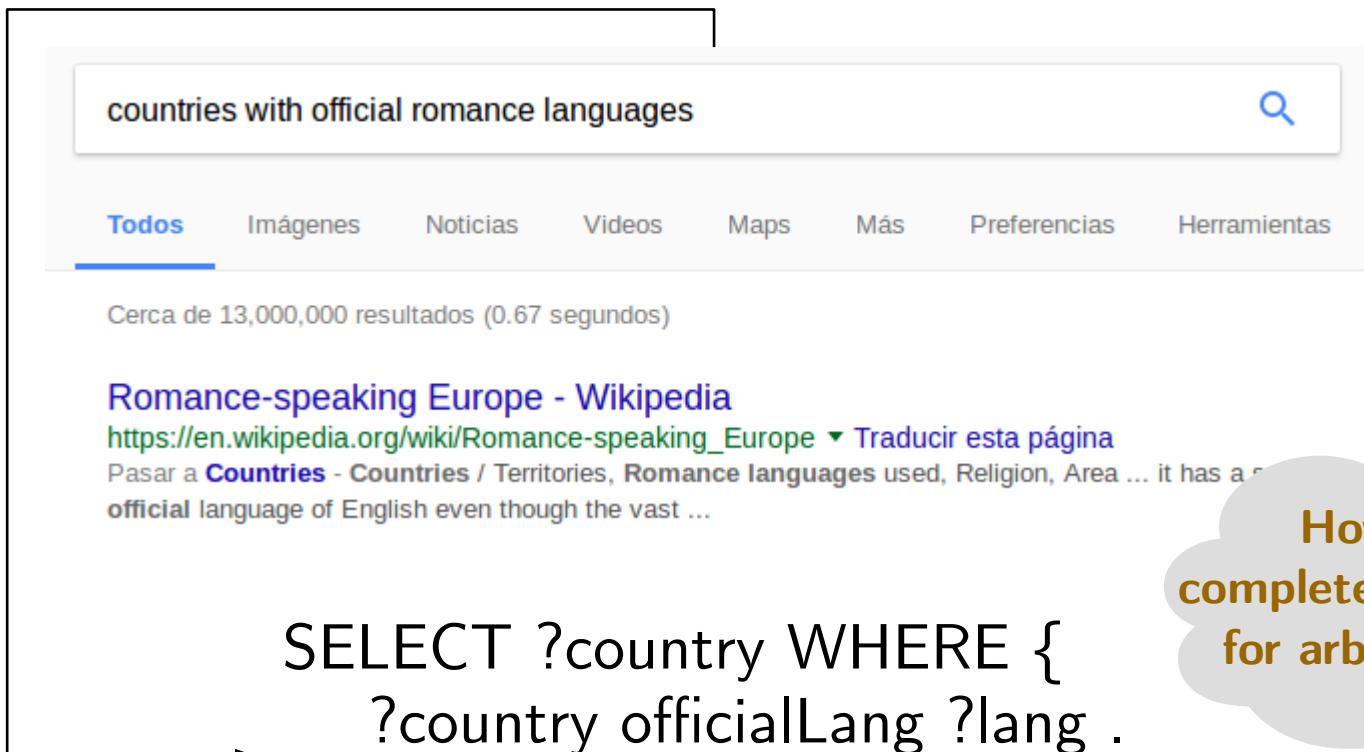
Providing completeness guarantees



A screenshot of a search engine results page. The search bar at the top contains the query "countries with official romance languages". Below the search bar is a navigation menu with tabs: "Todos" (which is underlined in blue), "Imágenes", "Noticias", "Videos", "Maps", "Más", "Preferencias", and "Herramientas". A status message below the menu says "Cerca de 13,000,000 resultados (0.67 segundos)". The first search result is a link to "Romance-speaking Europe - Wikipedia" with the URL https://en.wikipedia.org/wiki/Romance-speaking_Europe. A dropdown menu next to the link shows "Traducir esta página" and "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 officialLang ?lang .  
  ?lang family Romance .  
}
```

Providing completeness guarantees



A screenshot of a search engine results page. The search bar at the top contains the query "countries with official romance languages". Below the search bar is a navigation bar with tabs: "Todos" (which is underlined in blue), "Imágenes", "Noticias", "Videos", "Maps", "Más", "Preferencias", and "Herramientas". The main content area shows a search result for "Romance-speaking Europe - Wikipedia". The result includes the URL https://en.wikipedia.org/wiki/Romance-speaking_Europe, a "Traducir esta página" link, and a snippet of text: "Pasar a **Countries** - Countries / Territories, Romance languages used, Religion, Area ... it has a s official language of English even though the vast ...".

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

How to provide
completeness guarantees
for arbitrary queries?



Outline

- Completeness in RDF knowledge bases
- Completeness oracles
- Our vision
 - Representations for completeness oracles
 - Reasoning with completeness oracles
 - Enabling completeness in SPARQL

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 $\omega_d(\text{relation})$ or $\omega_d(\text{relation}^{-1})$

SELECT DISTINCT ?y WHERE { ?x officialLang ?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 $\omega_d(\text{relation})$ or $\omega_d(\text{relation}^{-1})$

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

- If ω_d 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 officialLang ?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 officialLang ?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 officialLang ?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 officialLang ?lang .  
  ?lang family Romance .  
}
```

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

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 officialLang ?lang  
  ?lang family Romance .  
}
```

It will generate
false negatives

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

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

If the KB misses Ligurian, this term returns false

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

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

Completeness guarantees for arbitrary queries

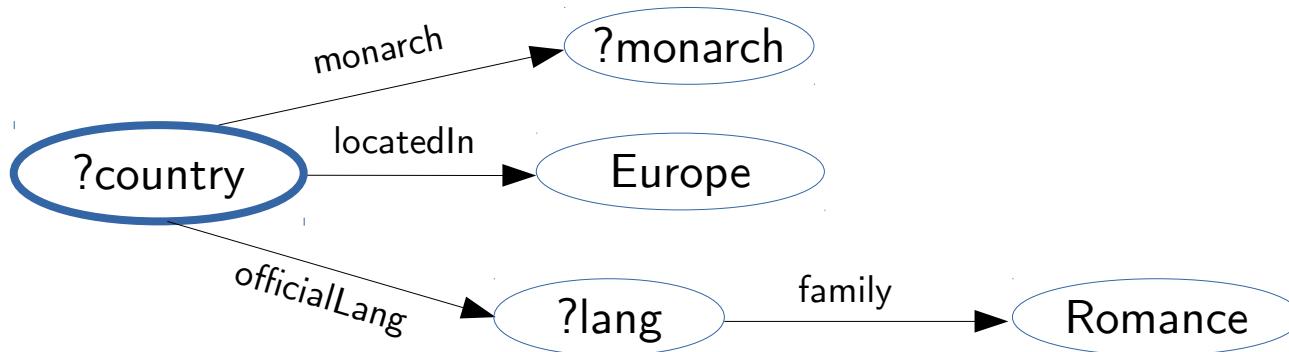
- Write completeness annotations for every possible type of query
 - It requires a large amount of effort
- Reuse existing SR and D oracles

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

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

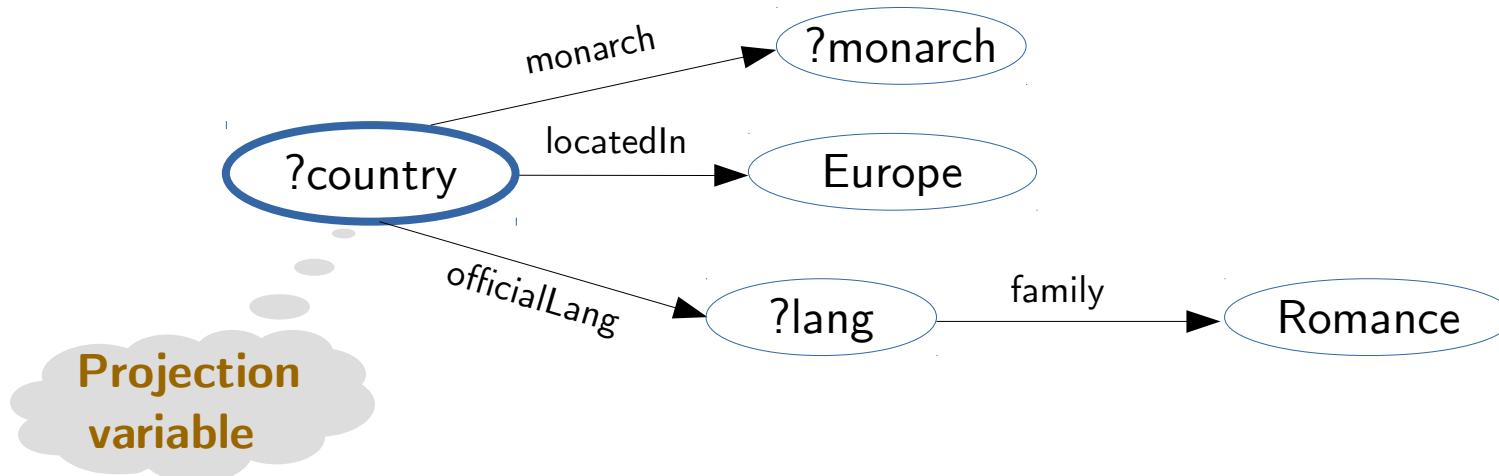
Even though this term does not care, because Ligurian is not official in any country

Automatic oracle composition



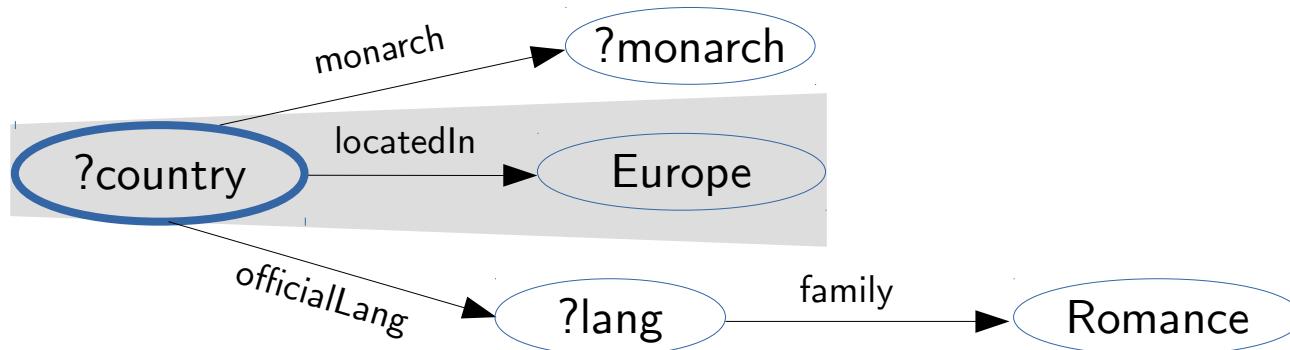
```
SELECT ?country WHERE {  
  ?country monarch ?monarch .  
  ?country locatedIn Europe .  
  ?country officialLang ?lang .  
  ?lang family Romance .  
}
```

Automatic oracle composition



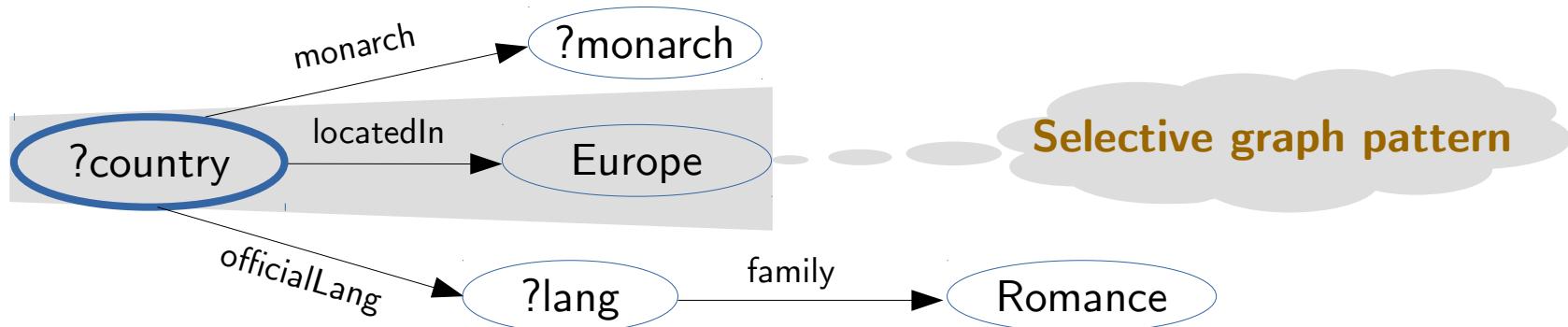
```
SELECT ?country WHERE {  
  ?country monarch ?monarch .  
  ?country locatedIn Europe .  
  ?country officialLang ?lang .  
  ?lang family Romance .  
}
```

Automatic oracle composition



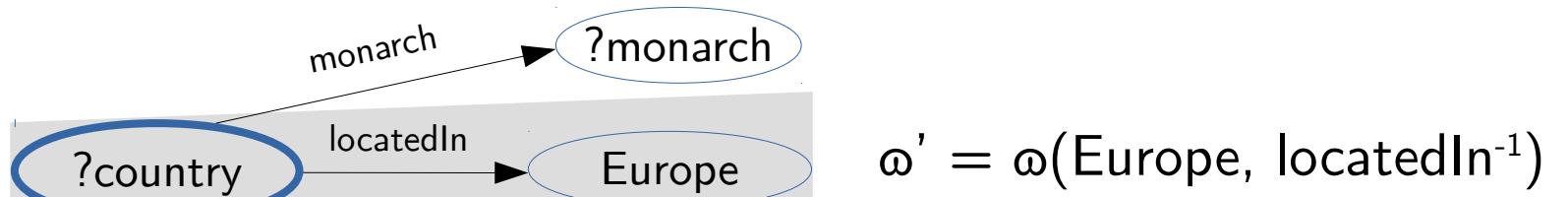
```
SELECT ?country WHERE {  
  ?country monarch ?monarch .  
  ?country locatedIn Europe .  
  ?country officialLang ?lang .  
  ?lang family Romance .  
}
```

Automatic oracle composition



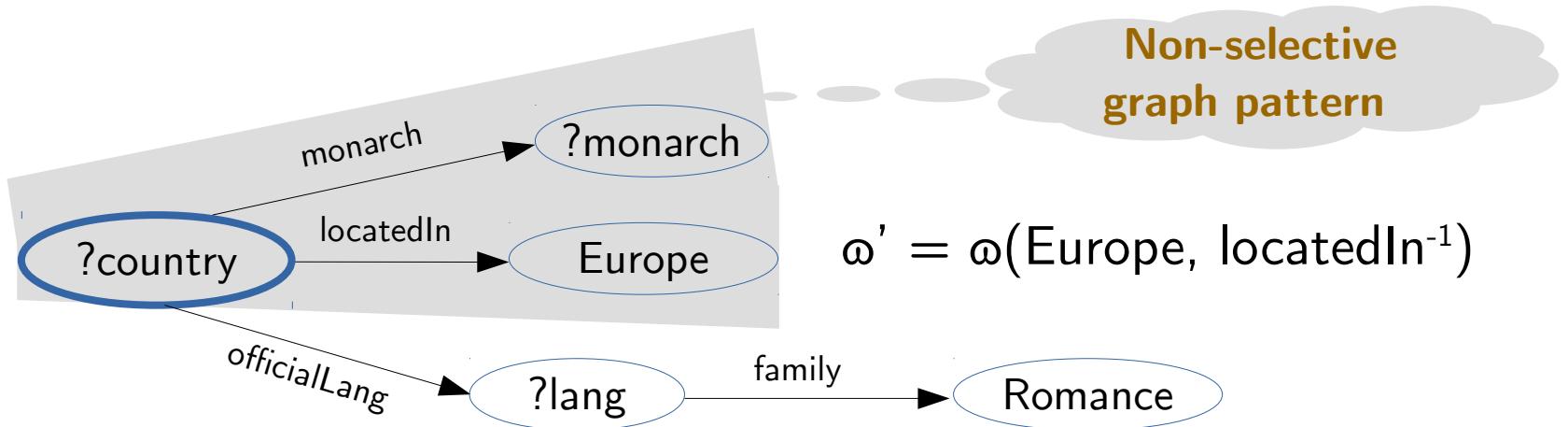
```
SELECT ?country WHERE {  
  ?country monarch ?monarch .  
  ?country locatedIn Europe .  
  ?country officialLang ?lang .  
  ?lang family Romance .  
}
```

Automatic oracle composition



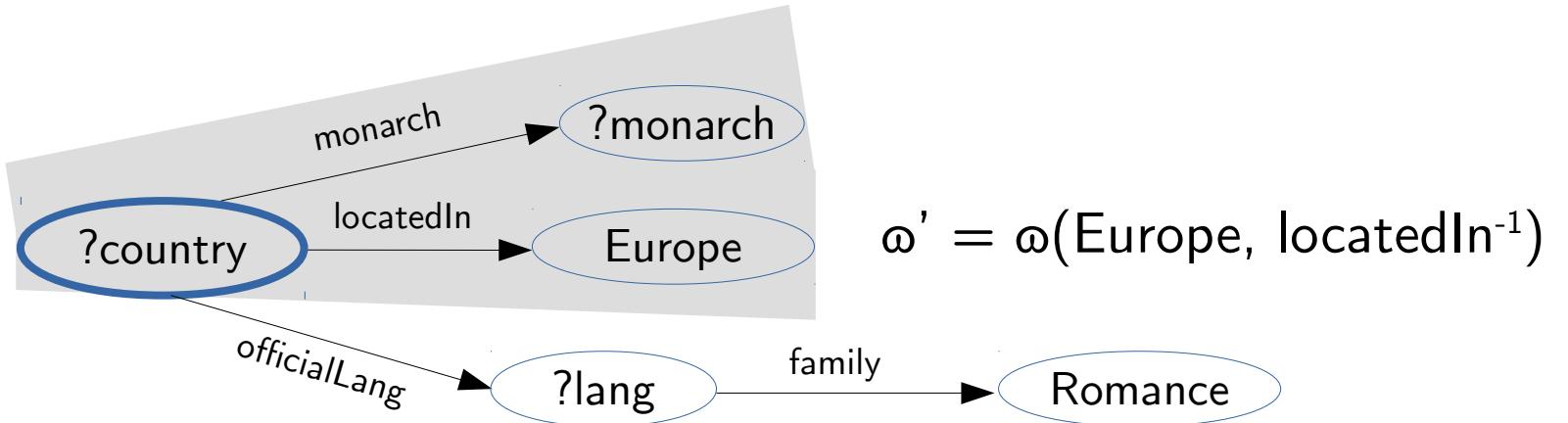
```
SELECT ?country WHERE {
  ?country monarch ?monarch .
  ?country locatedIn Europe .
  ?country officialLang ?lang .
  ?lang family Romance .
}
```

Automatic oracle composition



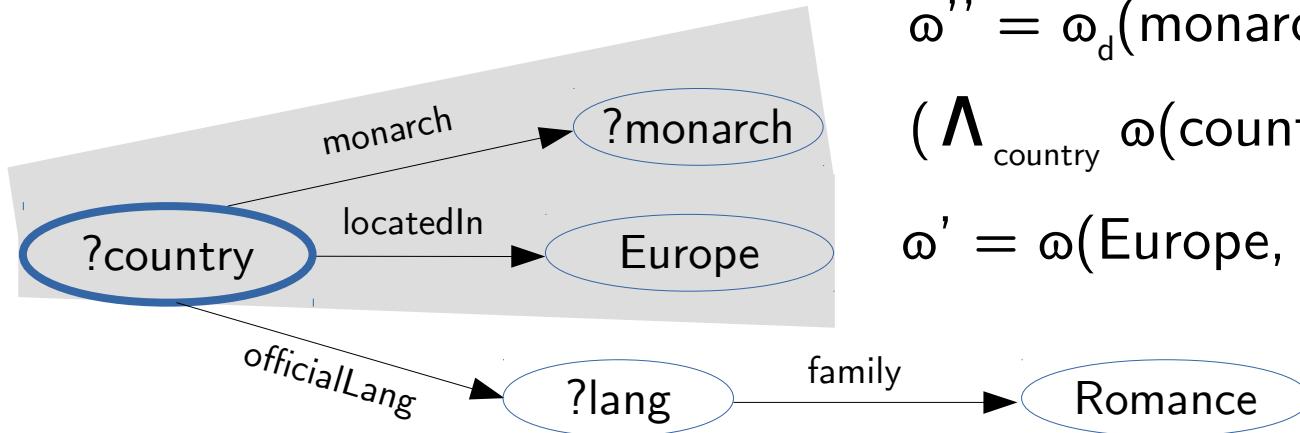
```
SELECT ?country WHERE {  
  ?country monarch ?monarch .  
  ?country locatedIn Europe .  
  ?country officialLang ?lang .  
  ?lang family Romance .  
}
```

Automatic oracle composition



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

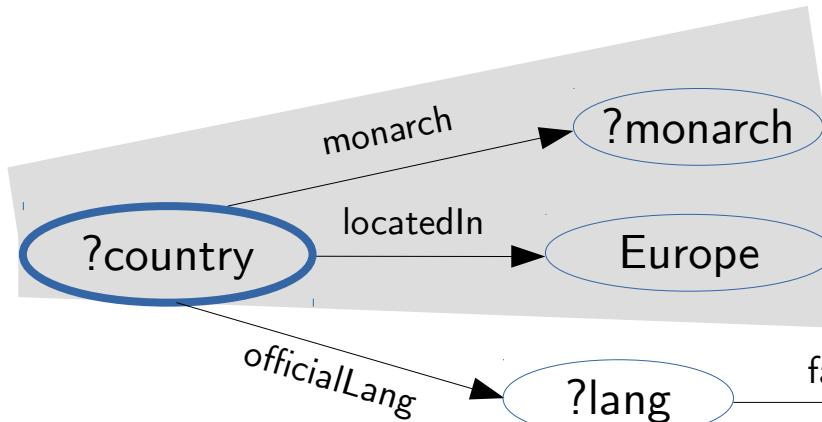
Automatic oracle composition



$$\begin{aligned}\omega'' &= \omega_d(\text{monarch}) \wedge \\ &(\bigwedge_{\text{country}} \omega(\text{country}, \text{monarch})) \\ \omega' &= \omega(\text{Europe}, \text{locatedIn}^{-1})\end{aligned}$$

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

Automatic oracle composition

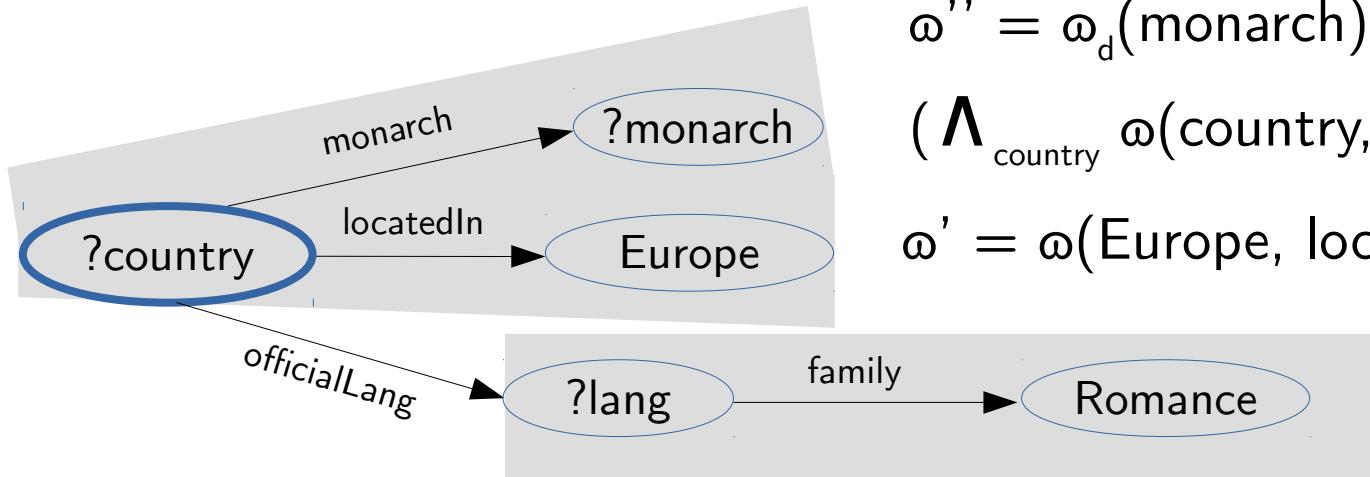


$$\omega'' = \omega_d(\text{monarch}) \wedge$$
$$(\Lambda_{\text{country}} \omega(\text{country}, \text{monarch}))$$
$$\omega' = \omega(\text{Europe}, \text{locatedIn}^{-1})$$

not needed in case of
Set semantics

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

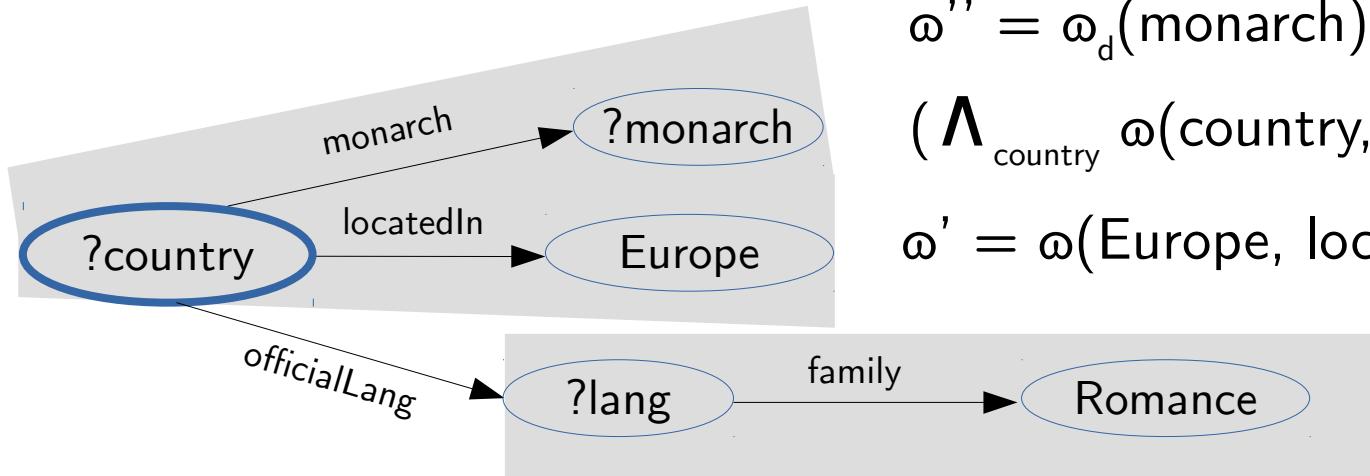
Automatic oracle composition



$$\begin{aligned}\omega'' &= \omega_d(\text{monarch}) \wedge \\ &(\bigwedge_{\text{country}} \omega(\text{country}, \text{monarch})) \\ \omega' &= \omega(\text{Europe}, \text{locatedIn}^{-1})\end{aligned}$$

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

Automatic oracle composition

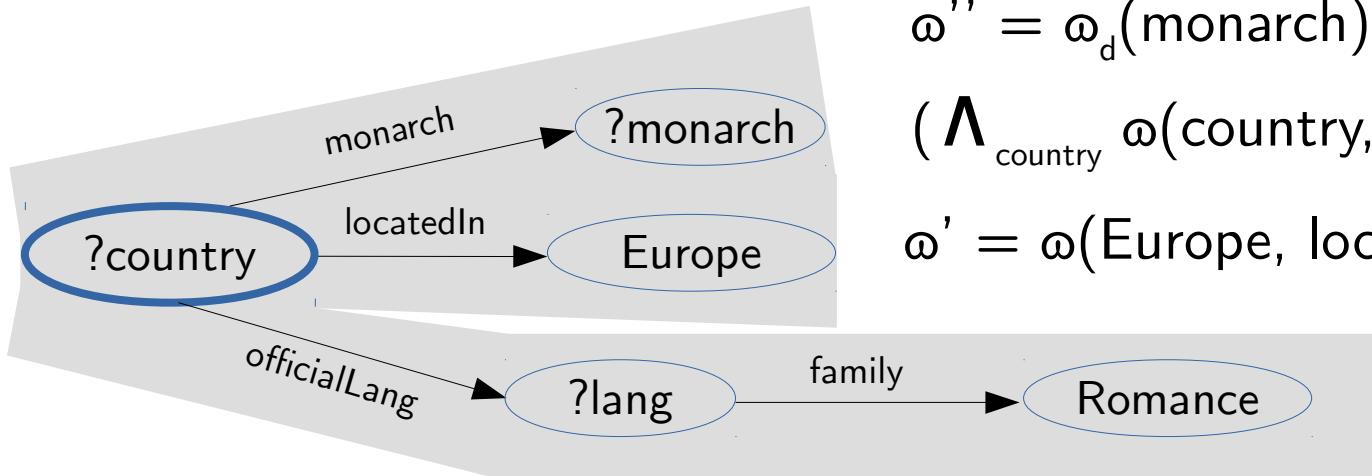


$$\omega'' = \omega_d(\text{monarch}) \wedge (\bigwedge_{\text{country}} \omega(\text{country}, \text{monarch}))$$
$$\omega' = \omega(\text{Europe}, \text{locatedIn}^{-1})$$

$$\omega^* = \omega(\text{Romance}, \text{family}^{-1})$$

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

Automatic oracle composition

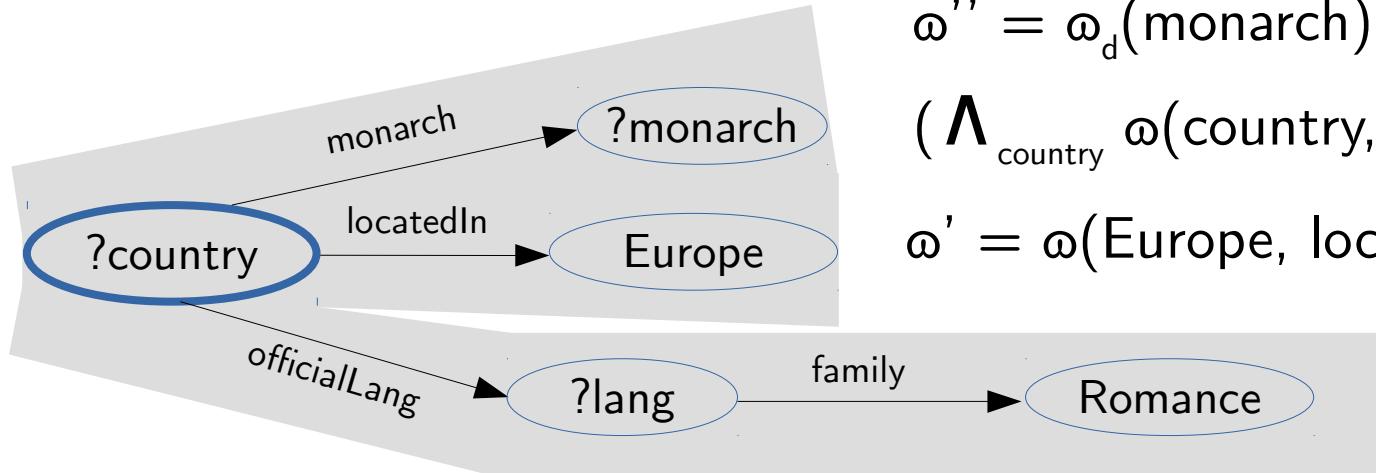


$$\begin{aligned}\omega'' &= \omega_d(\text{monarch}) \wedge \\ &(\bigwedge_{\text{country}} \omega(\text{country}, \text{monarch})) \\ \omega' &= \omega(\text{Europe}, \text{locatedIn}^{-1})\end{aligned}$$

$$\omega^* = \omega(\text{Romance}, \text{family}^{-1})$$

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

Automatic oracle composition



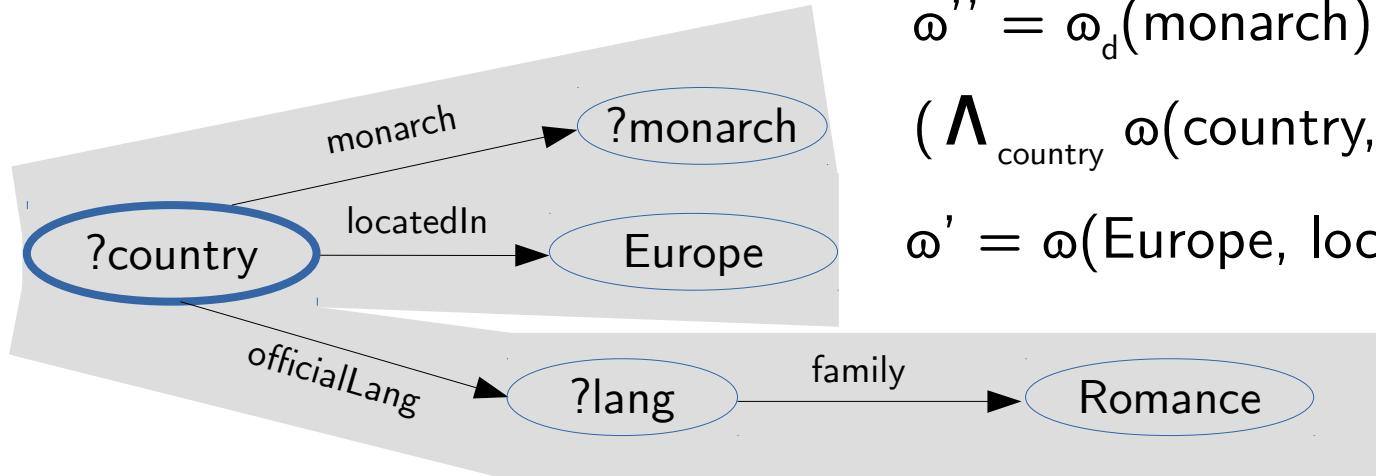
$$\omega^{**} = \Lambda_{l : \text{family}(l, \text{Romance})} \omega(l, \text{officialLang}^{-1}))$$

$$\begin{aligned}\omega'' &= \omega_d(\text{monarch}) \wedge \\ &(\Lambda_{\text{country}} \omega(\text{country}, \text{monarch})) \\ \omega' &= \omega(\text{Europe}, \text{locatedIn}^{-1})\end{aligned}$$

$$\omega^* = \omega(\text{Romance}, \text{family}^{-1})$$

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

Automatic oracle composition



$$\omega^{**} = \Lambda_{l : \text{family}(l, \text{Romance})} \omega(l, \text{officialLang}^{-1}))$$

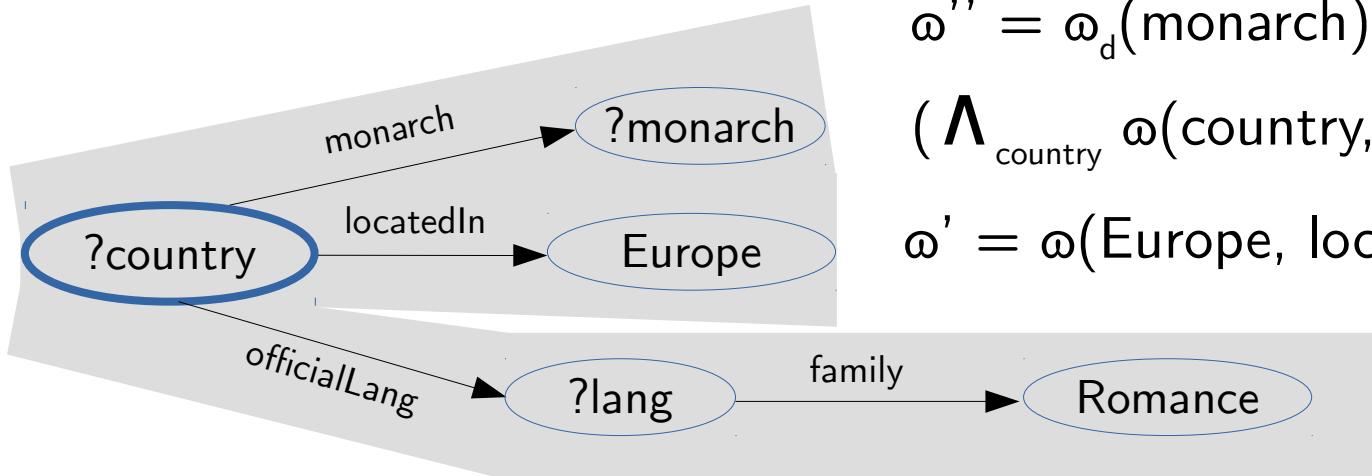
$$\begin{aligned}\omega'' &= \omega_d(\text{monarch}) \wedge \\ &(\Lambda_{\text{country}} \omega(\text{country}, \text{monarch})) \\ \omega' &= \omega(\text{Europe}, \text{locatedIn}^{-1})\end{aligned}$$

$$\omega^* = \omega(\text{Romance}, \text{family}^{-1})$$

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

Completeness verdict = $\omega^{**} \wedge \omega^* \wedge \omega' \wedge \omega''$

Automatic oracle composition



$$\omega^{**} = \Lambda_{l : \text{family}(l, \text{Romance})} \omega(l, \text{officialLang}^{-1}))$$

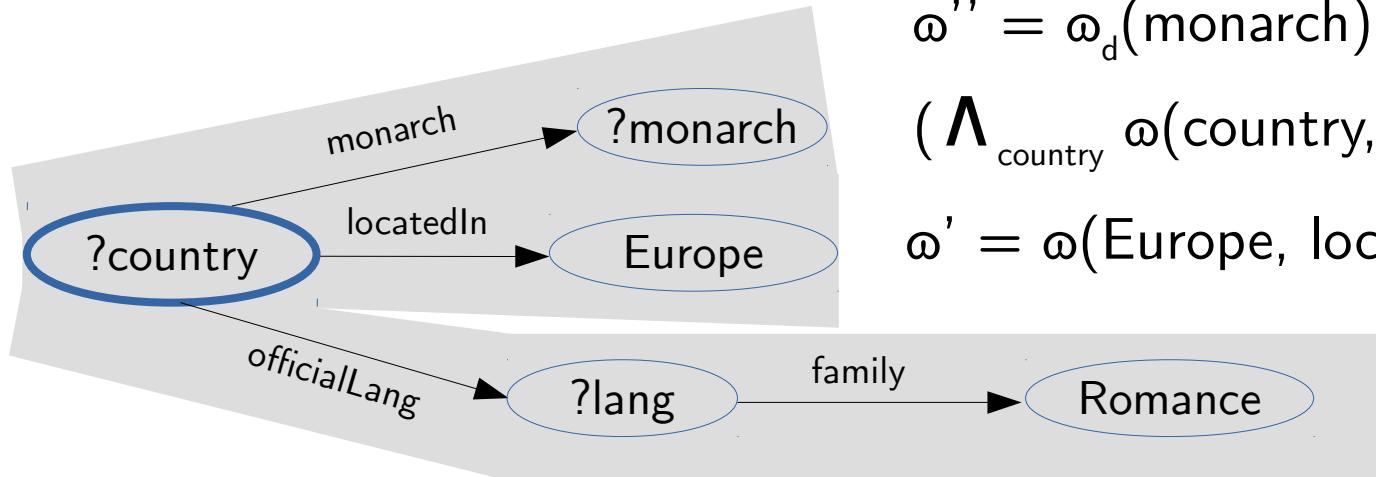
$$\omega^* = \omega(\text{Romance}, \text{family}^{-1})$$

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

Completeness verdict = $\omega^{**} \wedge \omega^* \wedge \omega' \wedge \omega''$

Confidence = $\text{prec}(\omega^{**}) \times \text{prec}(\omega^*) \times \text{prec}(\omega') \times \text{prec}(\omega'')$

Automatic oracle composition



$$\omega^{**} = \Lambda_{l : \text{family}(l, \text{Romance})} \omega(l, \text{officialLang}^{-1}))$$

$$\begin{aligned}\omega'' &= \omega_d(\text{monarch}) \wedge \\ &(\Lambda_{\text{country}} \omega(\text{country}, \text{monarch})) \\ \omega' &= \omega(\text{Europe}, \text{locatedIn}^{-1})\end{aligned}$$

$$\omega^* = \omega(\text{Romance}, \text{family}^{-1})$$

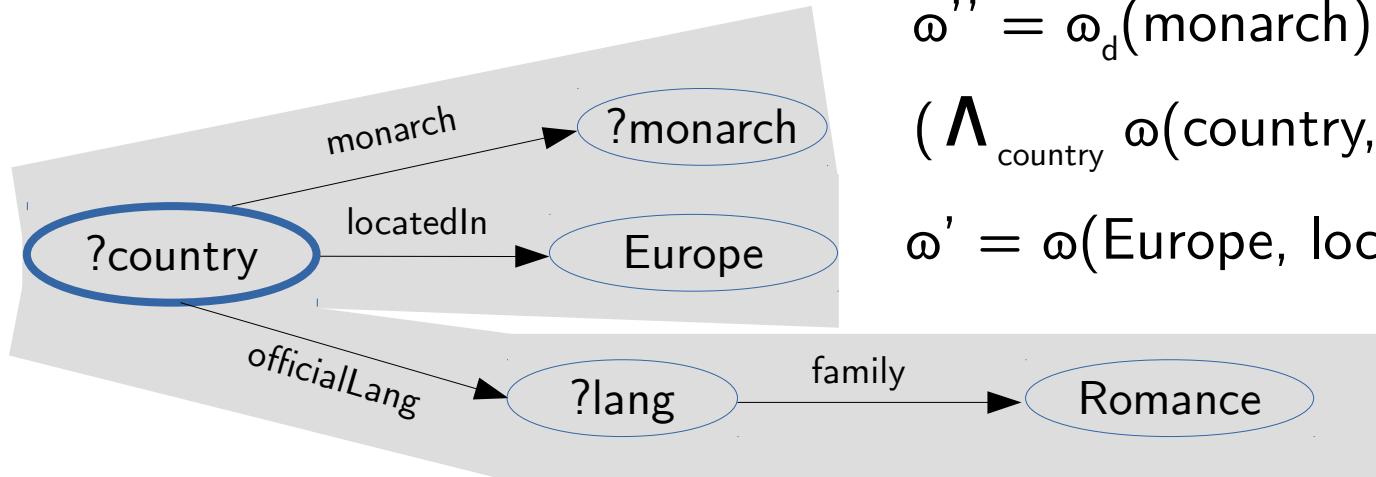
```
SELECT ?country WHERE {
  ?country monarch ?monarch .
  ?country locatedIn Europe .
  ?country officialLang ?lang
  ?lang family Romance .
}
```

*It could easily lead to
false negatives*

$$\text{Completeness verdict} = \omega^{**} \wedge \omega^* \wedge \omega' \wedge \omega''$$

$$\text{Confidence} = \text{prec}(\omega^{**}) \times \text{prec}(\omega^*) \times \text{prec}(\omega') \times \text{prec}(\omega'')$$

Automatic oracle composition



$$\omega^{**} = \Lambda_{l : \text{family}(l, \text{Romance})} \omega(l, \text{officialLang}^{-1}))$$

$$\begin{aligned}\omega'' &= \omega_d(\text{monarch}) \wedge \\ &(\Lambda_{\text{country}} \omega(\text{country}, \text{monarch})) \\ \omega' &= \omega(\text{Europe}, \text{locatedIn}^{-1})\end{aligned}$$

$$\omega^* = \omega(\text{Romance}, \text{family}^{-1})$$

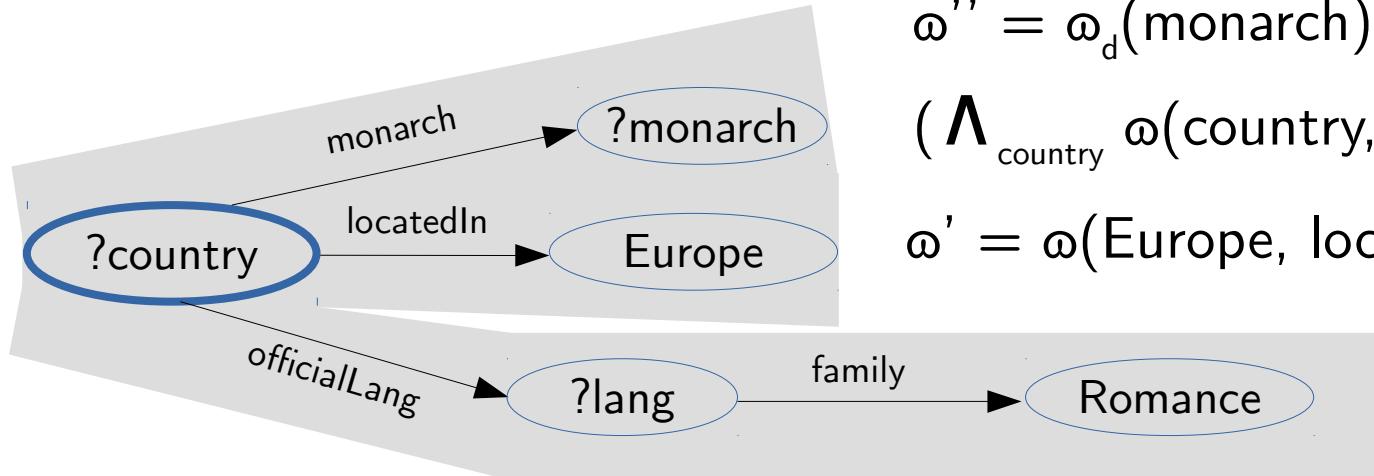
```
SELECT ?country WHERE {
  ?country monarch ?monarch
  ?country locatedIn Europe
  ?country officialLang ?lang
  ?lang family Romance .
}
```

We would like to
minimize the number
of used oracles

$$\text{Completeness verdict} = \omega^{**} \wedge \omega^* \wedge \omega' \wedge \omega''$$

$$\text{Confidence} = \text{prec}(\omega^{**}) \times \text{prec}(\omega^*) \times \text{prec}(\omega') \times \text{prec}(\omega'')$$

Automatic oracle composition



$$\omega^{**} = \Lambda_{l : \text{family}(l, \text{Romance})} \omega(l, \text{officialLang}^{-1}))$$

$$\begin{aligned}\omega'' &= \omega_d(\text{monarch}) \wedge \\ &(\Lambda_{\text{country}} \omega(\text{country}, \text{monarch})) \\ \omega' &= \omega(\text{Europe}, \text{locatedIn}^{-1})\end{aligned}$$

$$\omega^* = \omega(\text{Romance}, \text{family}^{-1})$$

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

Use more complex oracles
 that cover larger parts of
 the query graph at once

We would like to
 minimize the number
 of used oracles

Completeness verdict = $\omega^{**} \wedge \omega^* \wedge \omega' \wedge \omega''$

Confidence = $\text{prec}(\omega^{**}) \times \text{prec}(\omega^*) \times \text{prec}(\omega') \times \text{prec}(\omega'')$

Outline

- Completeness in RDF knowledge bases
- Completeness oracles
- Our vision
 - 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*

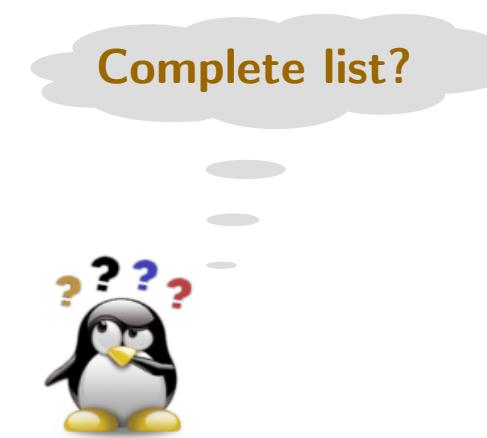
Boolean aggregation
function on sets of bindings

```
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

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



```
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

<code>?state</code>	<code>?county</code>	<code>?nspeak</code>
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 `?nspeak` for `?nspeak` are complete

Completeness oracles
to the rescue!

<code>?state</code>	<code>?county</code>	<code>?nspeak</code>
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
- Completeness oracles
- Our vision
 - 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
 - Existing work provides only 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
- Implement reasoning with completeness oracles in SPARQL query engines
 - Extend the SPARQL query language to support the *complete* aggregation function