

*AutoSPARQL –
Let Users Query Your Knowledge Base
<http://autosparql.dl-learner.org>*

Jens Lehmann, Lorenz Bühmann



UNIVERSITÄT LEIPZIG

2011-06-02

- 1 *How can we query RDF knowledge bases?*
- 2 *Screencast*
- 3 *Query Trees*
- 4 *Operations*
- 5 *QTL Algorithm and AutoSPARQL Interface*
- 6 *Evaluation*
- 7 *Conclusions and Future Work*

- 1 *How can we query RDF knowledge bases?*
- 2 *Screencast*
- 3 *Query Trees*
- 4 *Operations*
- 5 *QTL Algorithm and AutoSPARQL Interface*
- 6 *Evaluation*
- 7 *Conclusions and Future Work*

Query Interfaces:

- Knowledge Base Specific Interfaces

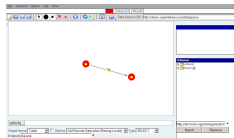
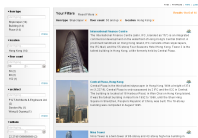
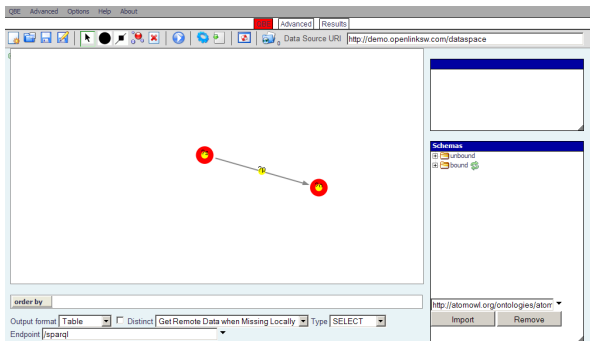
META	Sensor	<input type="text"/>
	Signature	<input type="text"/>
	Signature ID	<input type="text"/>
	Start Time	Year <input type="text"/> Month <input type="text"/> Day <input type="text"/> Hour <input type="text"/> Minute <input type="text"/>
	End Time	Year <input type="text"/> Month <input type="text"/> Day <input type="text"/> Hour <input type="text"/> Minute <input type="text"/>
IP	Source IP	<input type="text"/> OR Src Range <input type="text"/>
	Dest IP	<input type="text"/> OR Dest Range <input type="text"/>
	Length	<input type="text"/>
	Proto	<input type="text"/>
LAYER 4	Source Port(UDP and TCP)	<input type="text"/>
	Dest Port(UDP and TCP)	<input type="text"/>
	ICMP Type Code(ICMP only)	<input type="text"/>
DATA	Data(in ASCII)	<input type="text"/>
Other	Sort By:	<input type="text"/>
		<input type="button" value="Submit Query"/>

Query Interfaces:

- Knowledge Base
- Specific Interfaces
- Facet-Based Browsing

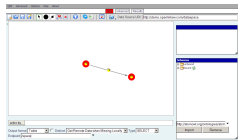
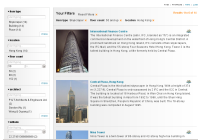
Query Interfaces:

- Knowledge Base Specific Interfaces
- Facet-Based Browsing
- Visual SPARQL Query Builders



Query Interfaces:

- Knowledge Base Specific Interfaces
- Facet-Based Browsing
- Visual SPARQL Query Builders
- Question Answering



AnswerBus

When was Ulysses S. Grant born?
Type in your question in English, French, Spanish, German, Italian or Portuguese.

Question:

When was Ulysses S. Grant born?

Possible answers: 286, 122

- S. Grant An American Hero Ulysses S. Grant was born July 3, 1822 he was originally named Hiram Ulysses Grant but it was changed to Ulysses S.
- Ulysses Simpson Grant - Ulysses Simpson Grant Born: 4/27/1822 Birthplace: Point Pleasant, Ohio Ulysses Simpson Grant was ...
- Grant Birthplace Picture 3 Clipped from historical marker at President Ulysses S. Grant birthplace, Hiram Ulysses Grant was born in this one-story, timber frame home on April 27, 1822 to Jesse and Hannah Simpson Grant.
- Grant (born Hiram Ulysses Grant) was born in Point Pleasant, Clermont County, Ohio, 25 miles (40 km) above Cincinnati on the Ohio River, to Jesse B. Grant and Hannah Simpson.
- Ulysses S. Grant Ulysses S. Grant was born in Point Pleasant Ohio, on April 27th, 1822 to a modest middle class family.
- Ulysses Simpson Grant was born Hiram Ulysses Grant on April 22, 1822, in Point Pleasant, Ohio, where his father worked as a tanner.
- Ulysses Sherman Grant was born in Moline, Illinois, on February 14, 1867, the son of Lewis Addison Grant and Mary Helen (Pierce) Grant.
- Hiram Ulysses Grant was born in Point Pleasant, Ohio, on April 27, 1822, the son of a tanner, Jesse Root Grant, and Hannah Simpson Grant.
- Ulysses Simpson Grant was born as Hiram Ulysses Grant) at Point Pleasant, Ohio, on April 27, 1822. He graduated from West Point in 1843 and served without particular distinction in the Mexican War.
- Grant Ulysses S. Grant was born in Point Pleasant Ohio, on April 27th, 1822 to a modest middle class family.

Try your question on other engines:

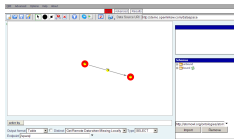
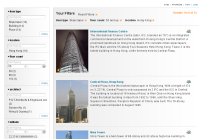
Alta Vista | CNN News Engine | Ask Jeeves | Excite | Google | Hotbot | Lycos | Start | Wondr | Yahoo

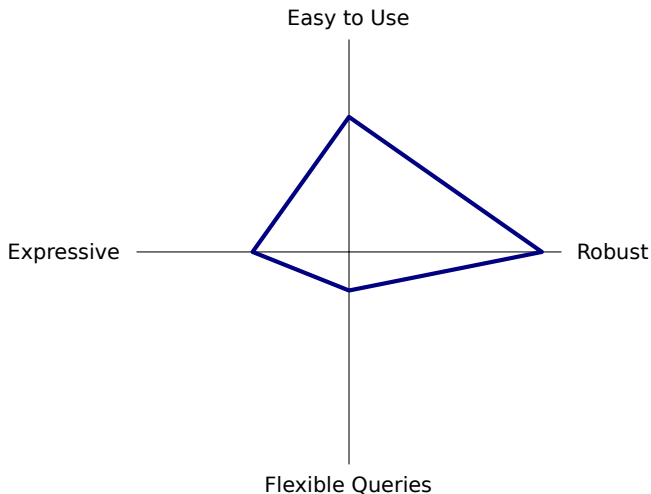
Query Interfaces:

- Knowledge Base Specific Interfaces
- Facet-Based Browsing
- Visual SPARQL Query Builders
- Question Answering

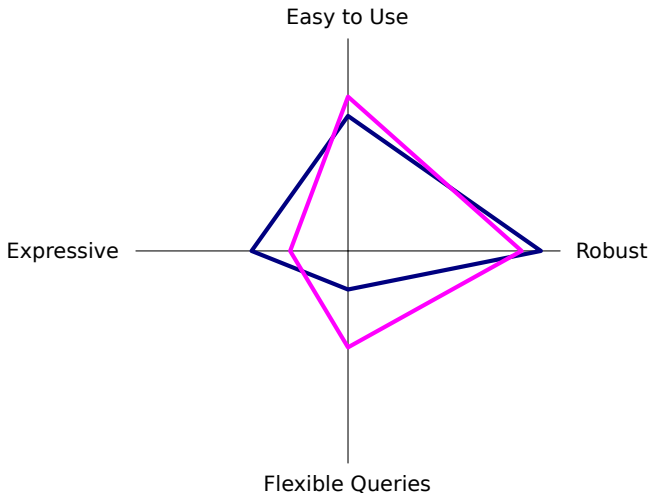


Are those the best tools to create SPARQL queries?



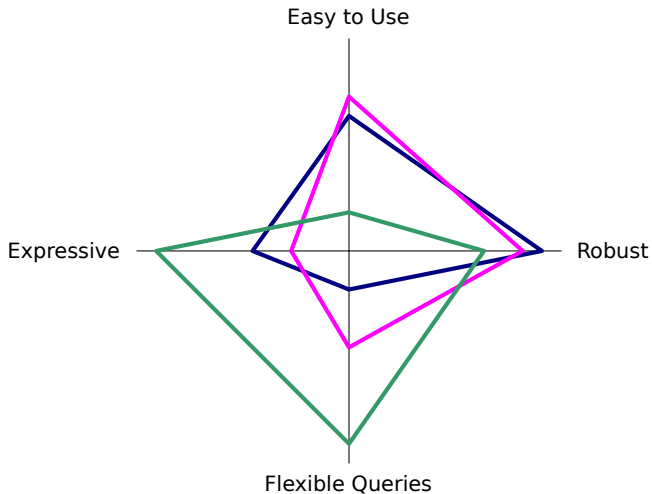


— Knowledge Base Specific Visual SPARQL Query Builders Facet-Based Browsing Question Answering



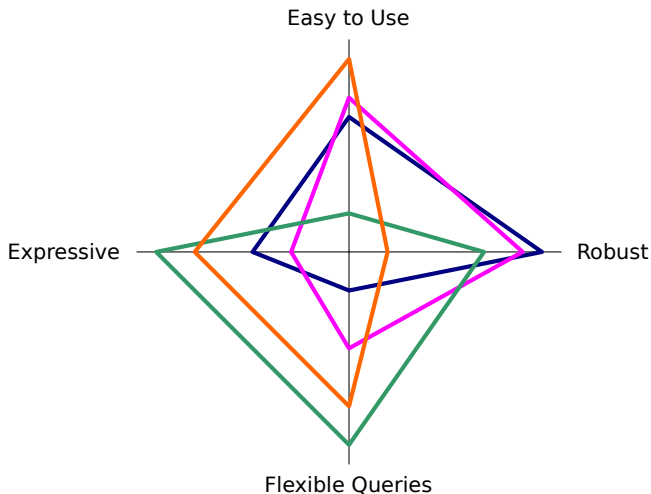
— Knowledge Base Specific
Visual SPARQL Query Builders

— Facet-Based Browsing
Question Answering



— Knowledge Base Specific
— Visual SPARQL Query Builders

— Facet-Based Browsing Question Answering



- 1 *How can we query RDF knowledge bases?*
- 2 *Screencast*
- 3 *Query Trees*
- 4 *Operations*
- 5 *QTL Algorithm and AutoSPARQL Interface*
- 6 *Evaluation*
- 7 *Conclusions and Future Work*

The screenshot shows the AutoSPARQL web interface. At the top left, there is a blue header with the text "AutoSPARQL". Below the header, there is a search area on the left and a results area on the right. The search area contains a "Search" box with a "Search" button and a message "No resources found." The results area contains a "Result" section with a table and a "Graph" button, and an "Examples" section with two columns: "Should belong to query result:" and "Should not belong to query result:". Both columns show "No examples selected." Four large, yellow, glowing numbers are overlaid on the interface: 1 is in the top left corner, 2 is in the search area, 3 is in the results area, and 4 is in the examples area.

1 Please search for a result, e.g. if you want to query "cities in France" you should search for "Paris". Once you have found a resource, you can mark it with "+", an interactive guide will be shown. Further questions, which lead you to your desired result, will be shown.

Search

Enter search term Search

No resources found.

Result

Table Query Graph

Label

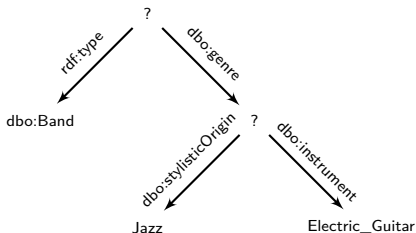
Examples

Should belong to query result: Should not belong to query result:

No examples selected No examples selected.

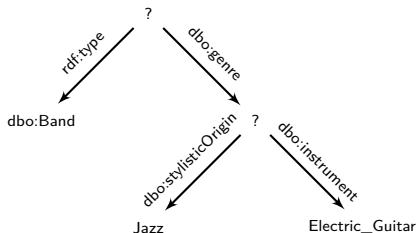
<http://localhost/screencast/autosparql.htm>

- 1 *How can we query RDF knowledge bases?*
- 2 *Screencast*
- 3 *Query Trees***
- 4 *Operations*
- 5 *QTL Algorithm and AutoSPARQL Interface*
- 6 *Evaluation*
- 7 *Conclusions and Future Work*



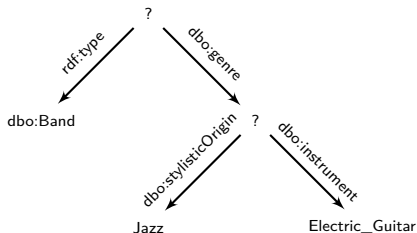
- query tree = rooted, directed, labelled tree
- nodes are labelled with IRIs, literals or ?
- edges are labelled with IRIs

QueryTree \longleftrightarrow SPARQL Query



```

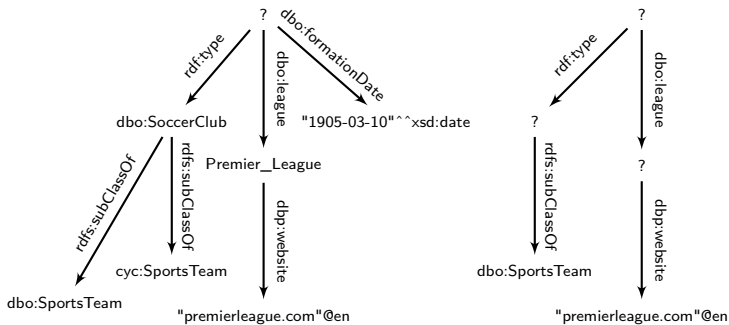
SELECT ?x0 WHERE {
  ?x0 rdf:type dbo:Band.
  ?x0 dbo:genre ?x1.
  ?x1 dbo:instrument dbp:Electric_guitar.
  ?x1 dbo:stylisticOrigin dbp:Jazz.
}
  
```



- each resource in an RDF graph can be mapped to a query tree
- follow links from resource
- avoid cycles
- fix recursion depth
- query trees act as bridge between RDF graph and SPARQL queries

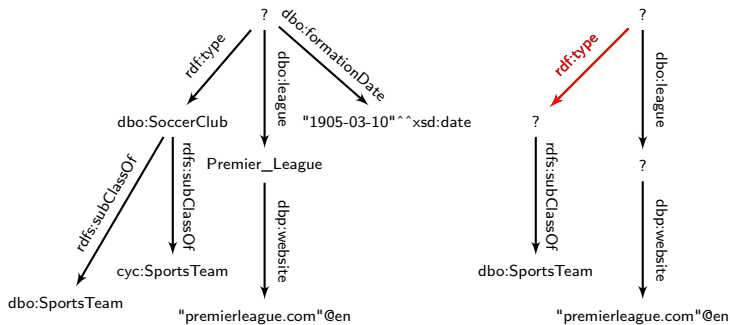
- 1 *How can we query RDF knowledge bases?*
- 2 *Screencast*
- 3 *Query Trees*
- 4 *Operations*
- 5 *QTL Algorithm and AutoSPARQL Interface*
- 6 *Evaluation*
- 7 *Conclusions and Future Work*

Query Tree Subsumption



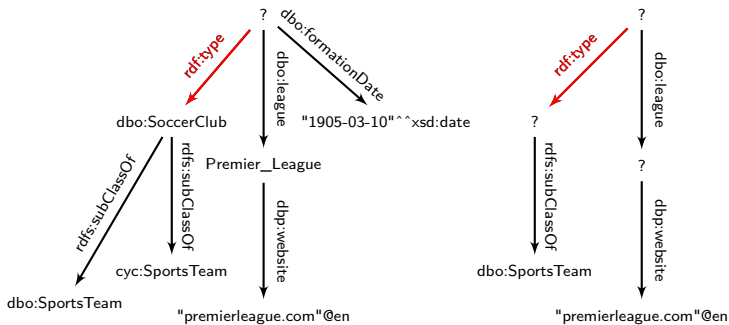
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



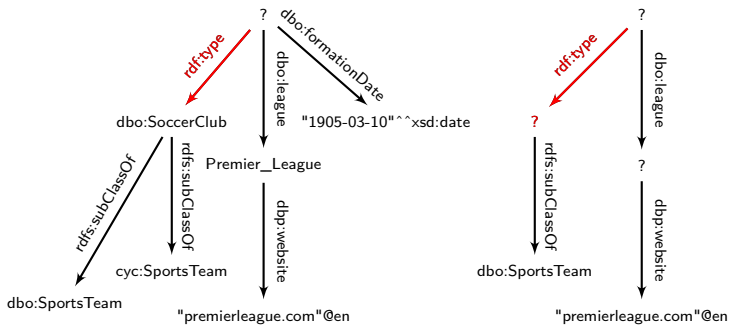
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



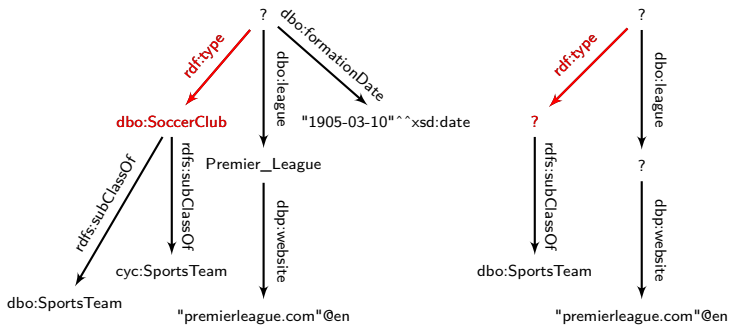
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



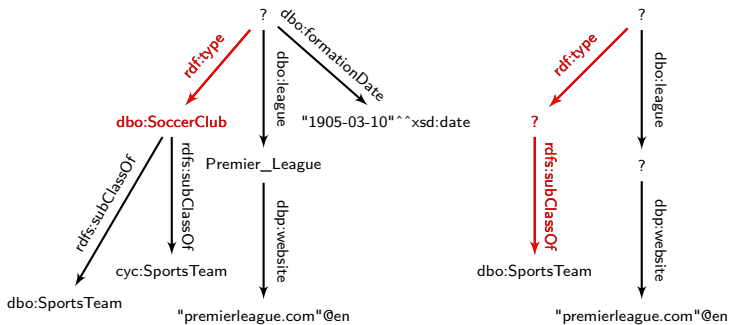
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



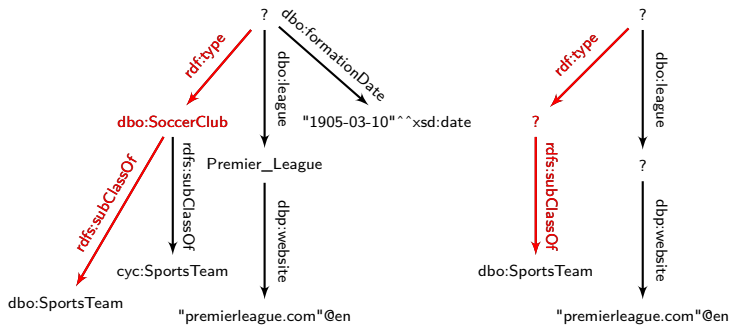
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



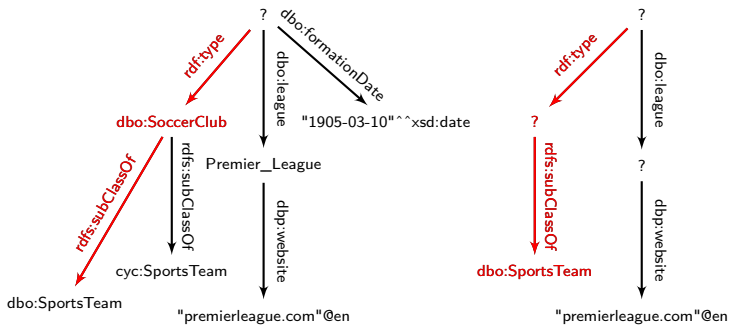
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



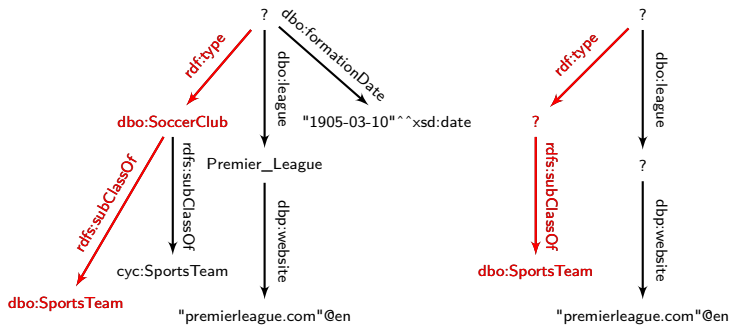
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



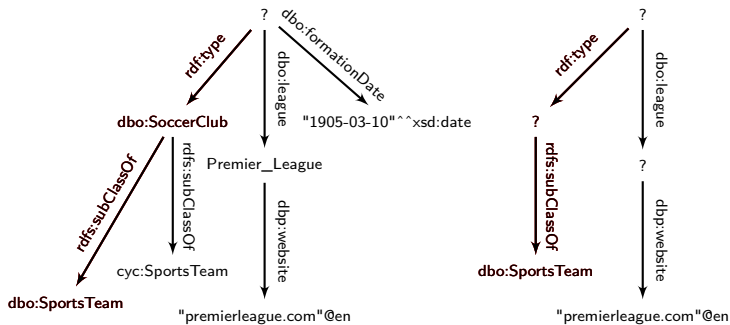
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



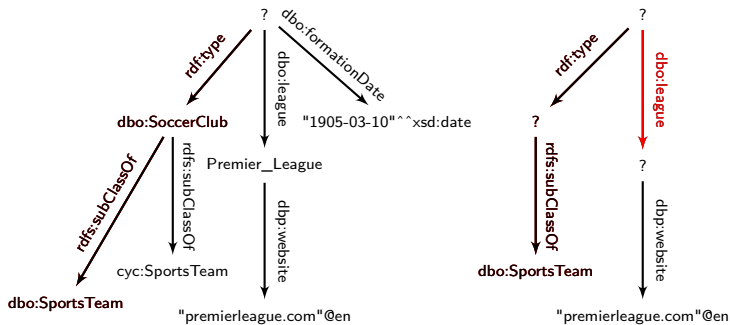
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



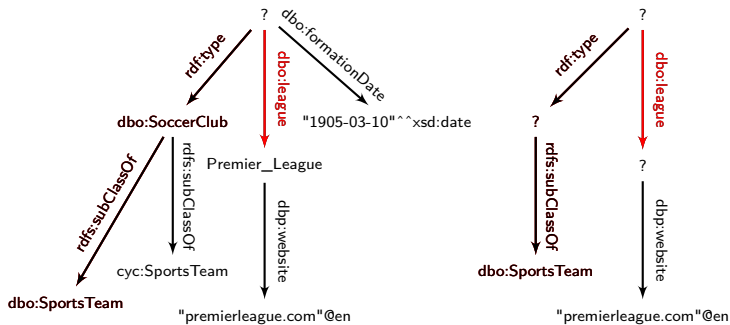
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



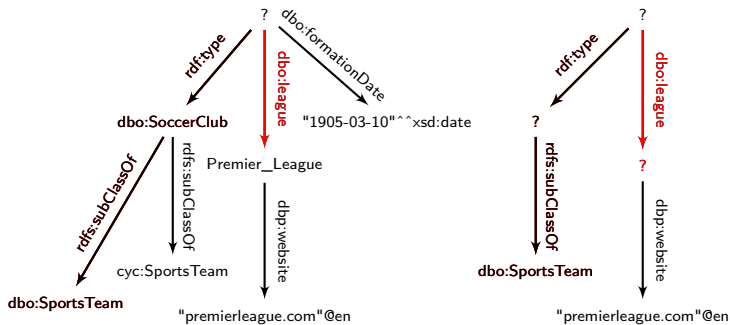
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



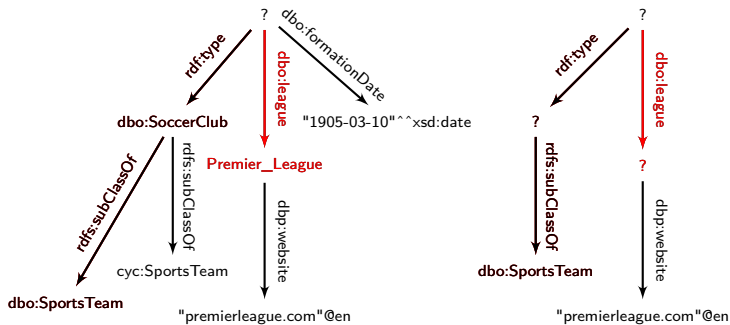
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



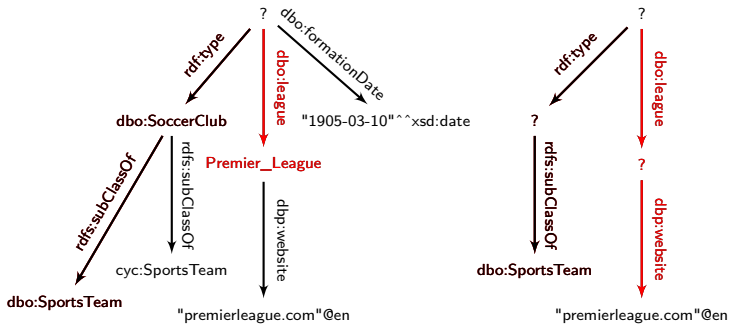
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



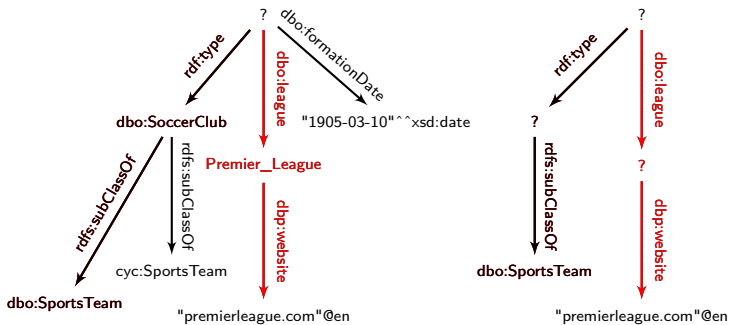
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



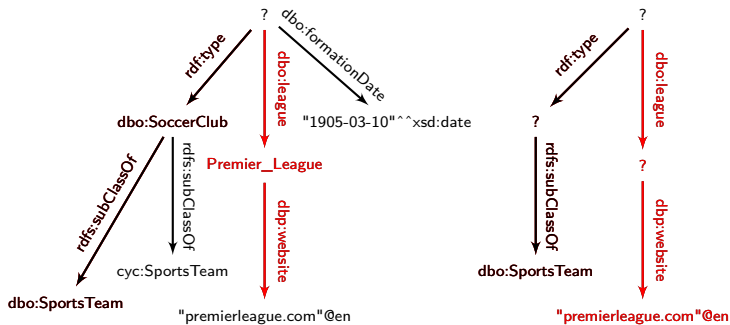
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



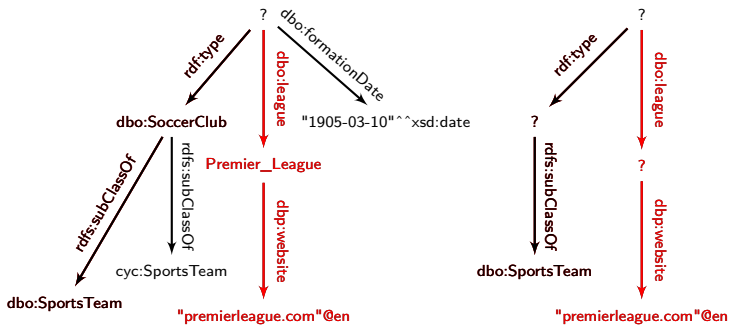
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



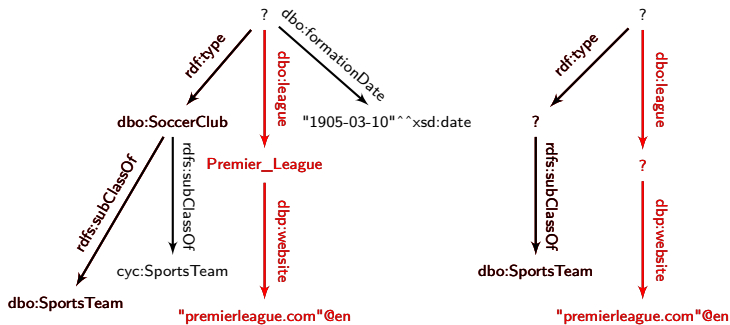
- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



- introduction of an order \leq on query trees
- left tree \leq right tree

Query Tree Subsumption



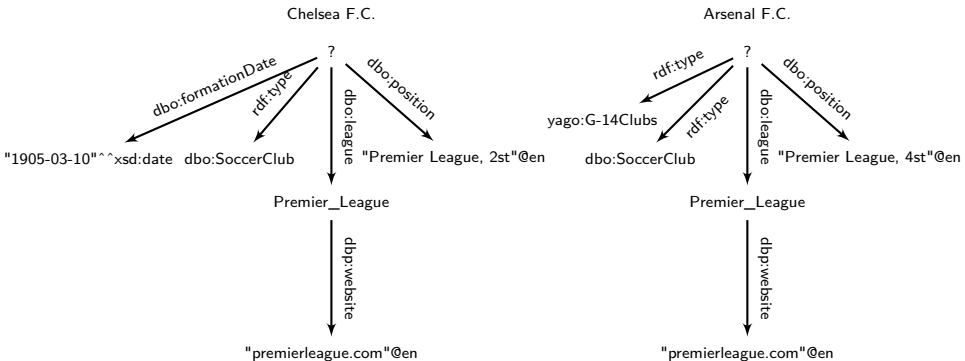
- introduction of an order \leq on query trees
- left tree \leq right tree

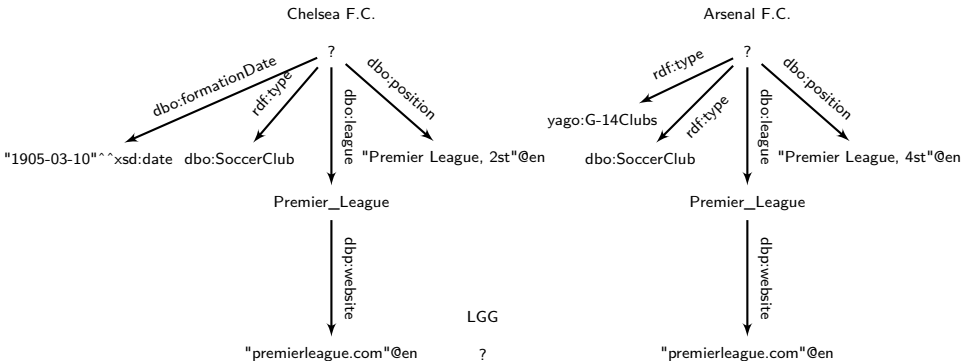
Compatible with SPARQL semantics:

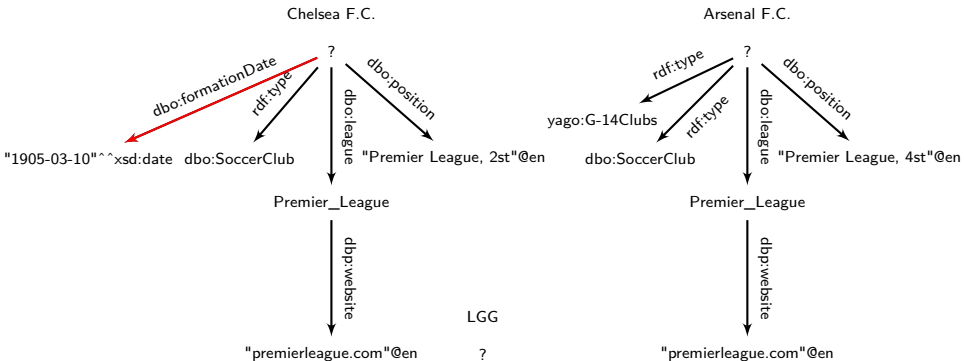
Proposition

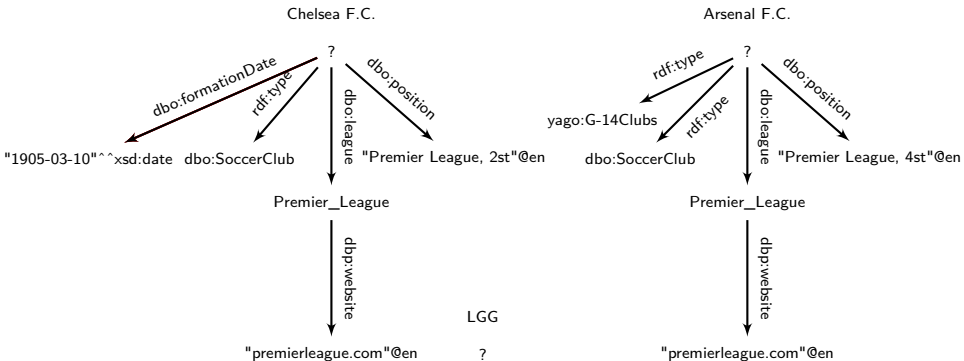
$T_1 \leq T_2$ implies $[[sparql(T_1)]]_G(?x_0) \subseteq [[sparql(T_2)]]_G(?x_0)$ for any RDF graph G .

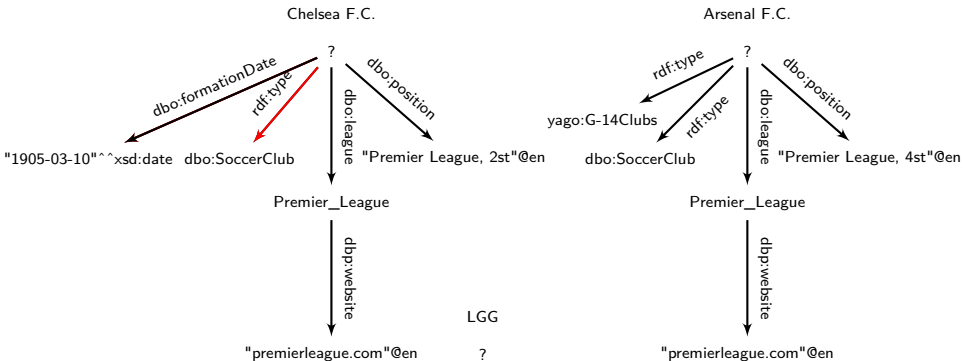
- LGG = least general generalisation
- operation takes two query trees as input and returns the most specific query tree, which subsumes both input trees
- well known in Inductive Logic Programming (ILP)

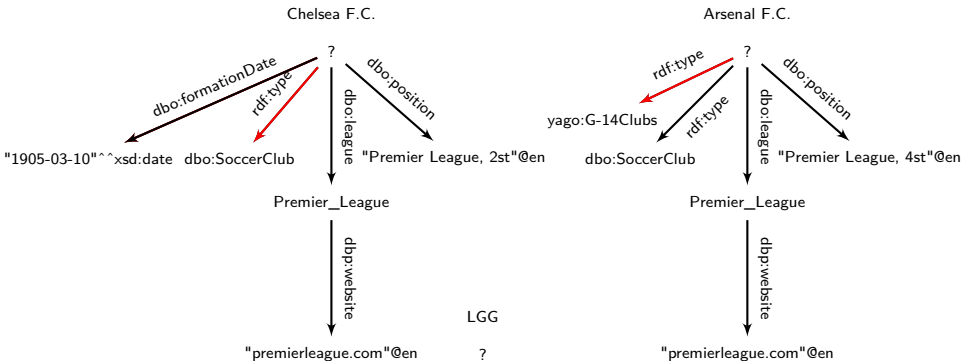


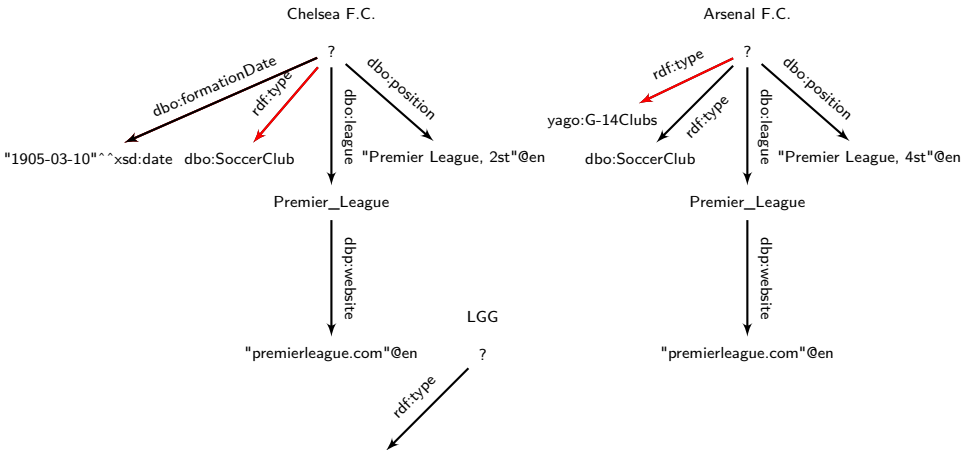


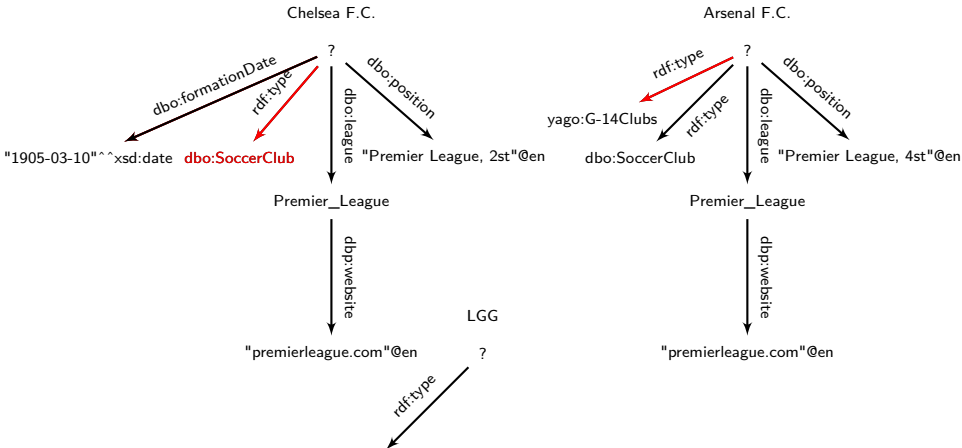


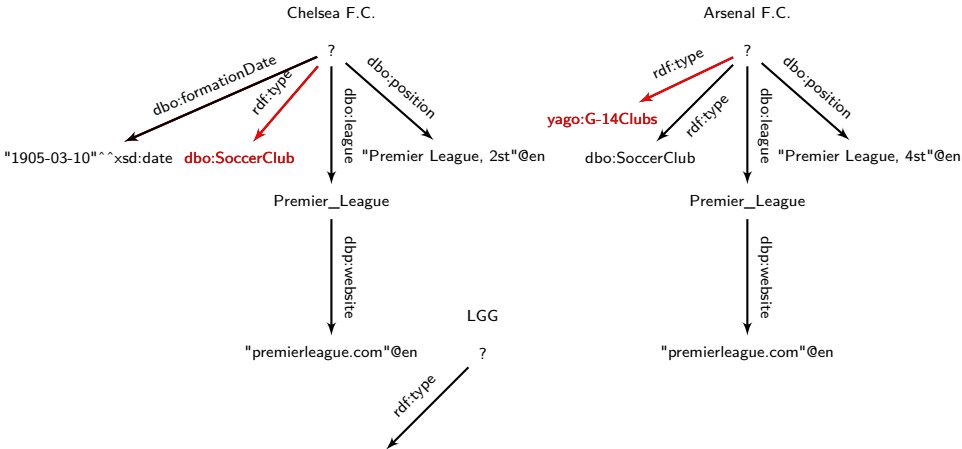


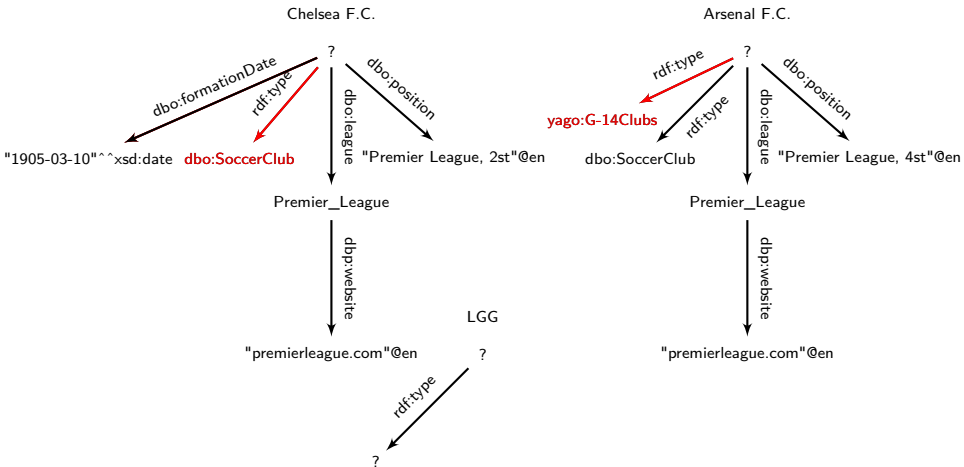


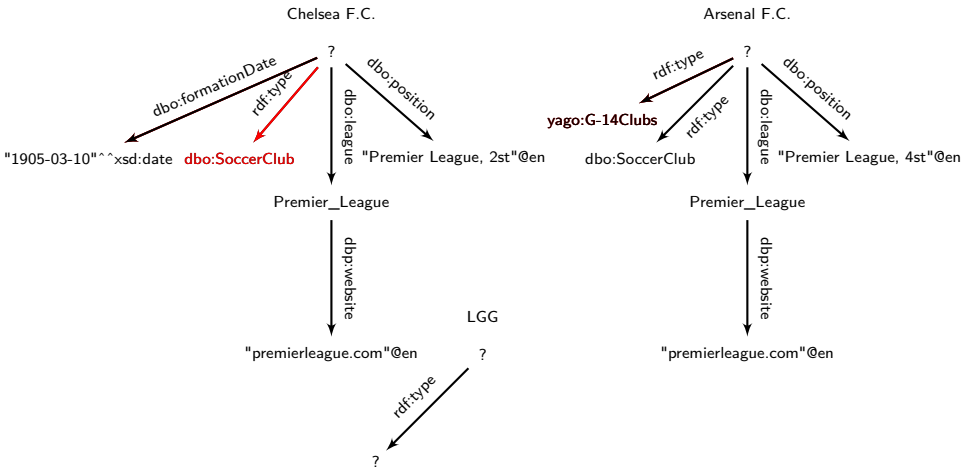


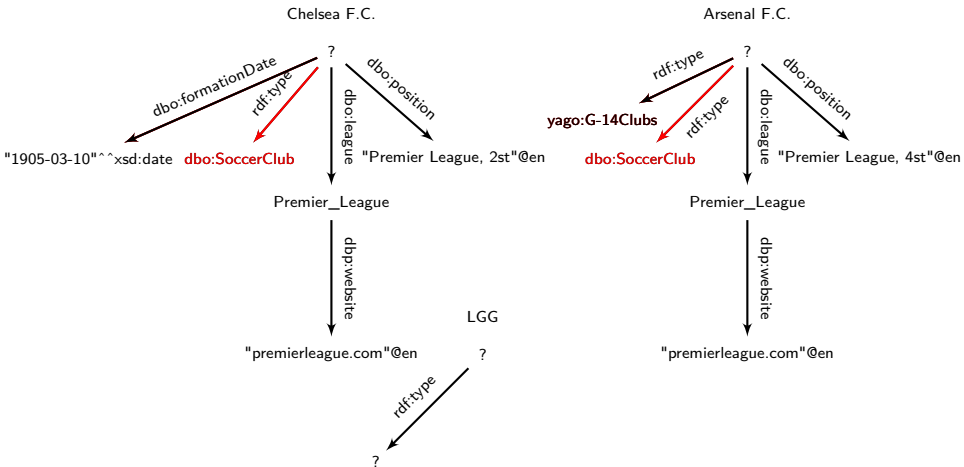


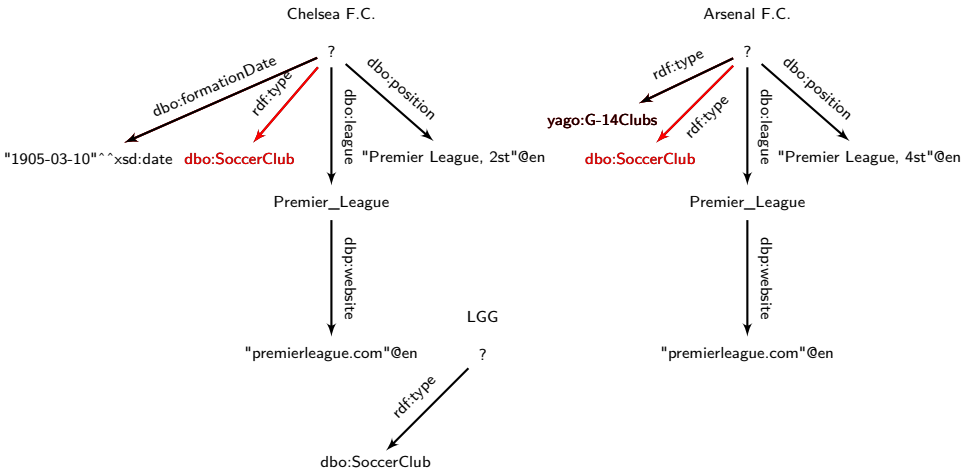


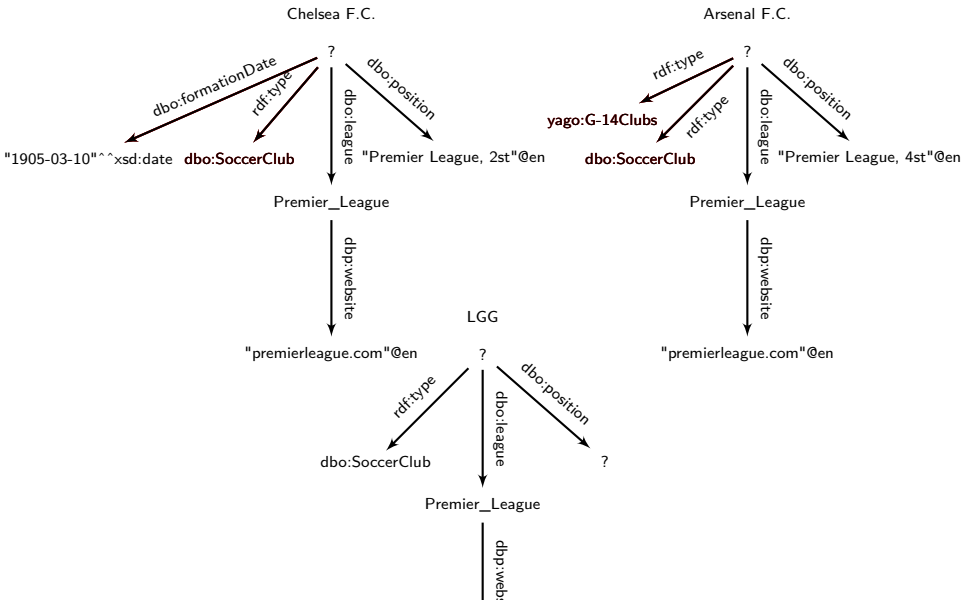




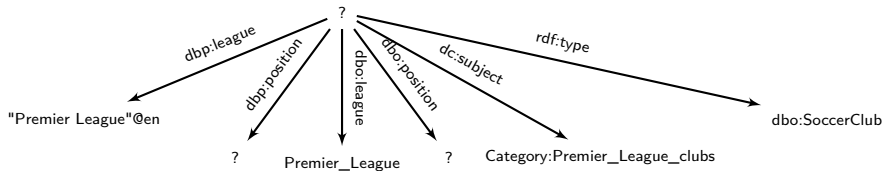


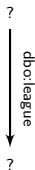






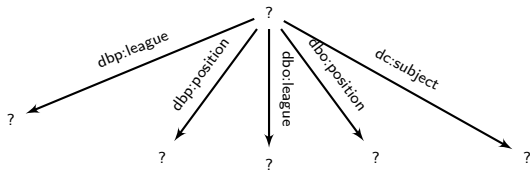
- NBR = negative based reduction
- input: positive query tree and set of negative query trees
- output: query tree which generalises positive query tree such that:
 - no negative query tree is subsumed
 - corresponding SPARQL query returns more results than the positive query tree





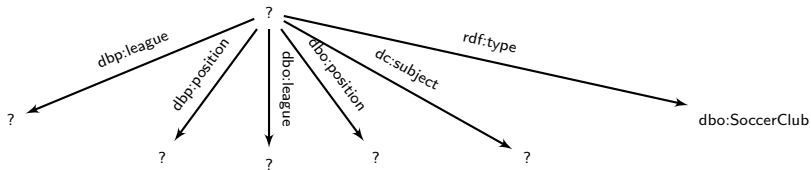
```

SELECT DISTINCT ?x0 WHERE{
  ?x0 dbo:league ?x1.
  OPTIONAL{?x0 dbo:position ?x6}
  OPTIONAL{?x0 dbp:league ?x7}
  OPTIONAL{?x0 dbp:position ?x8}
  OPTIONAL{?x0 dc:subject ?x9}
  FILTER(
    (?x1 != <Premier_League>) ||
    (!BOUND(?x6)) ||
    (!BOUND(?x7)) ||
    (!BOUND(?x8)) ||
    (!BOUND(?x9))
  )
}
    
```



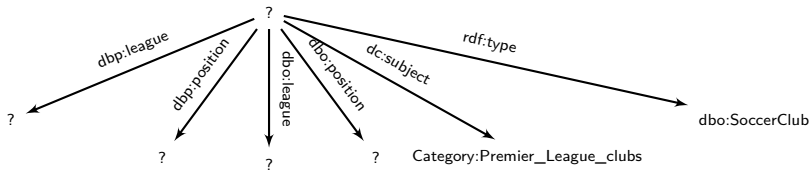
```

SELECT DISTINCT ?x0 WHERE{
  ?x0 dbo:league ?x1.
  ?x0 dbo:position ?x6.
  ?x0 dbp:league ?x7.
  ?x0 dbp:position ?x8.
  ?x0 dc:subject ?x9.
  FILTER(
    (?x1 != <Premier_League>) ||
    (?x7 != "Premier League"@en) ||
    (?x9 != <Category:Premier_League_clubs>))
}
  
```



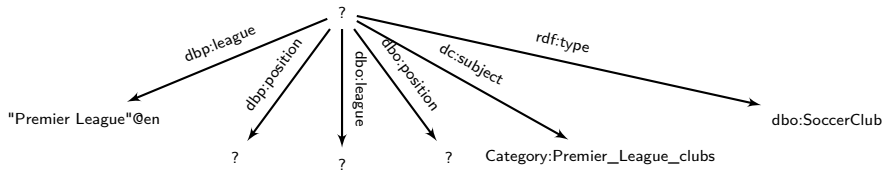
```

SELECT DISTINCT ?x0 WHERE{
  ?x0 dbo:league ?x1.
  ?x0 dbo:position ?x6.
  ?x0 dbp:league ?x7.
  ?x0 dbp:position ?x8.
  ?x0 dc:subject ?x9.
  ?x0 a dbo:SoccerClub.
  FILTER(
    (?x1 != <Premier_League>) ||
    (?x7 != "Premier League"@en) ||
    (?x9 != <Category:Premier_League_clubs>)
  )
}
    
```



```

SELECT DISTINCT ?x0 WHERE{
  ?x0 dbo:league ?x1.
  ?x0 dbo:position ?x6.
  ?x0 dbp:league ?x7.
  ?x0 dbp:position ?x8.
  ?x0 dc:subject <Category:Premier_League_clubs >.
  ?x0 a dbo:SoccerClub.
  FILTER(
    (?x1 != <Premier_League >) ||
    (?x7 != "Premier League"@en)
  )
}
    
```



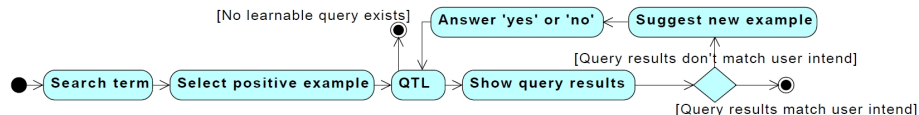
```

SELECT DISTINCT ?x0 WHERE{
  ?x0 dbo:league ?x1.
  ?x0 dbo:position ?x6.
  ?x0 dbp:league "Premier League"@en.
  ?x0 dbp:position ?x8.
  ?x0 dc:subject <Category:Premier_League_clubs>.
  ?x0 a dbo:SoccerClub.
  FILTER(
    (?x1 != <Premier_League>)
  )
}
  
```

- 1 *How can we query RDF knowledge bases?*
- 2 *Screencast*
- 3 *Query Trees*
- 4 *Operations*
- 5 *QTL Algorithm and AutoSPARQL Interface***
- 6 *Evaluation*
- 7 *Conclusions and Future Work*

QTL:

- QTL (query tree learner) combines previous operations
- examples (given by the user) are mapped to query trees
- computes LGG of positive examples
- NBR (or posonly generalisation in case of no negative examples) is used to generalise the LGG



AutoSPARQL:

- Active Learning method/interface on top of QTL
- resource returned by NBR but not by LGG is used as question for user (oracle)

Proposition (Completeness)

Every query tree will eventually be learned correctly by AutoSPARQL (see paper for technical restrictions).

- 1 *How can we query RDF knowledge bases?*
- 2 *Screencast*
- 3 *Query Trees*
- 4 *Operations*
- 5 *QTL Algorithm and AutoSPARQL Interface*
- 6 *Evaluation*
- 7 *Conclusions and Future Work*

- no active learning algorithm for SPARQL to compare against, but:
- benchmark data set of the 1st Workshop on Question Answering over Linked Data (QALD)¹

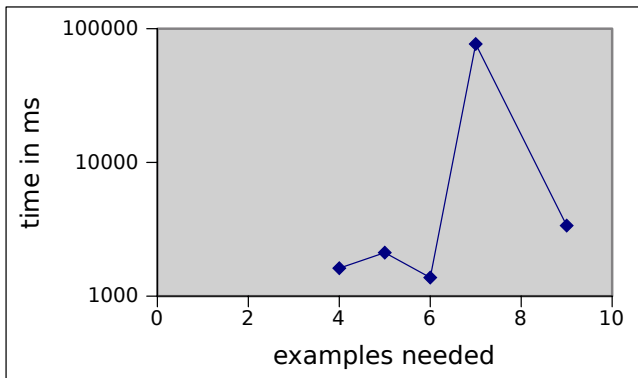
¹<http://www.sc.cit-ec.uni-bielefeld.de/qald-1>

- no active learning algorithm for SPARQL to compare against, but:
- benchmark data set of the 1st Workshop on Question Answering over Linked Data (QALD)¹
- contains 50 questions to DBpedia
- filtered questions with < 3 answers and unsupported constructs like UNION
- 15 queries remaining
- 3 positive and 1 negative examples used as starting point selected from initial search or randomly (if the search did not return 3 positives)

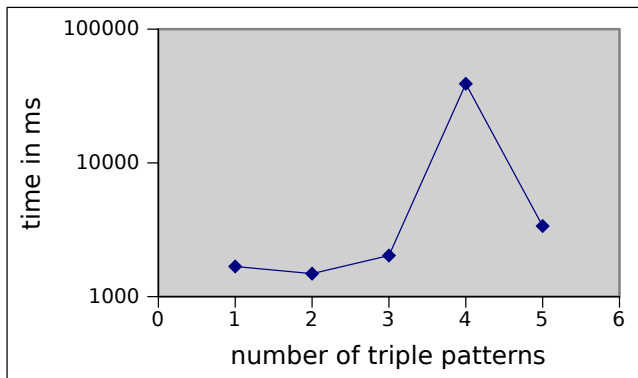
¹<http://www.sc.cit-ec.uni-bielefeld.de/qald-1>

- no active learning algorithm for SPARQL to compare against, but:
- benchmark data set of the 1st Workshop on Question Answering over Linked Data (QALD)¹
- contains 50 questions to DBpedia
- filtered questions with < 3 answers and unsupported constructs like UNION
- 15 queries remaining
- 3 positive and 1 negative examples used as starting point selected from initial search or randomly (if the search did not return 3 positives)
- parameters: recursion depth 2, NLP filter active
- hardware: 2 GB memory, 6 core

¹<http://www.sc.cit-ec.uni-bielefeld.de/qald-1>



- all 15 questions learned correctly (precision = recall = 1 ;-)
- 4 to 9 examples needed (5 on average)
- only low number of examples needed because of LGG and Active Learning



- performance: 7 seconds on average to learn query (max. 77 seconds)
- approx. $\frac{3}{4}$ for NBR, $\frac{1}{4}$ for SPARQL queries, $\leq 1\%$ for LGG
- more time need with increasing number of triple patterns

- 1 *How can we query RDF knowledge bases?*
- 2 *Screencast*
- 3 *Query Trees*
- 4 *Operations*
- 5 *QTL Algorithm and AutoSPARQL Interface*
- 6 *Evaluation*
- 7 *Conclusions and Future Work*

Benefits & Disadvantages

- + low number of questions: theoretical termination guarantee and low number of questions required (5 during test on average)
- + efficiency: by focusing on a fragment of SPARQL, QTL is efficient (about a second per question)
- ± expressiveness: common constructs supported in AutoSPARQL
- ± reasoning: no built-in reasoning → SPARQL 1.1 entailment regimes
- ± AutoSPARQL/QTL do not include noise handling
 - wrong answers lead to incorrect queries
 - users can, however, edit previous answers in UI
 - low number of examples makes noise handling less important (for the considered QTL use case)
- user often needs to give feedback in addition to asking a question → more work on initial QA phase needed

- first active learning algorithm and first induction algorithm for SPARQL
- AutoSPARQL lowers hurdle for users to create queries and can be used with any SPARQL endpoint
- flexible and suitable for non-experts
- several improvements planned over the next year ...

Thanks for your attention.

<http://autosparql.dl-learner.org>

<http://dl-learner.org>

<http://aksw.org>

<http://bis.informatik.uni-leipzig.de/LorenzBuehmann>

<http://jens-lehmann.org>

buehmann@informatik.uni-leipzig.de

lehmann@informatik.uni-leipzig.de

