SPARQL (1): An RDF Query Language

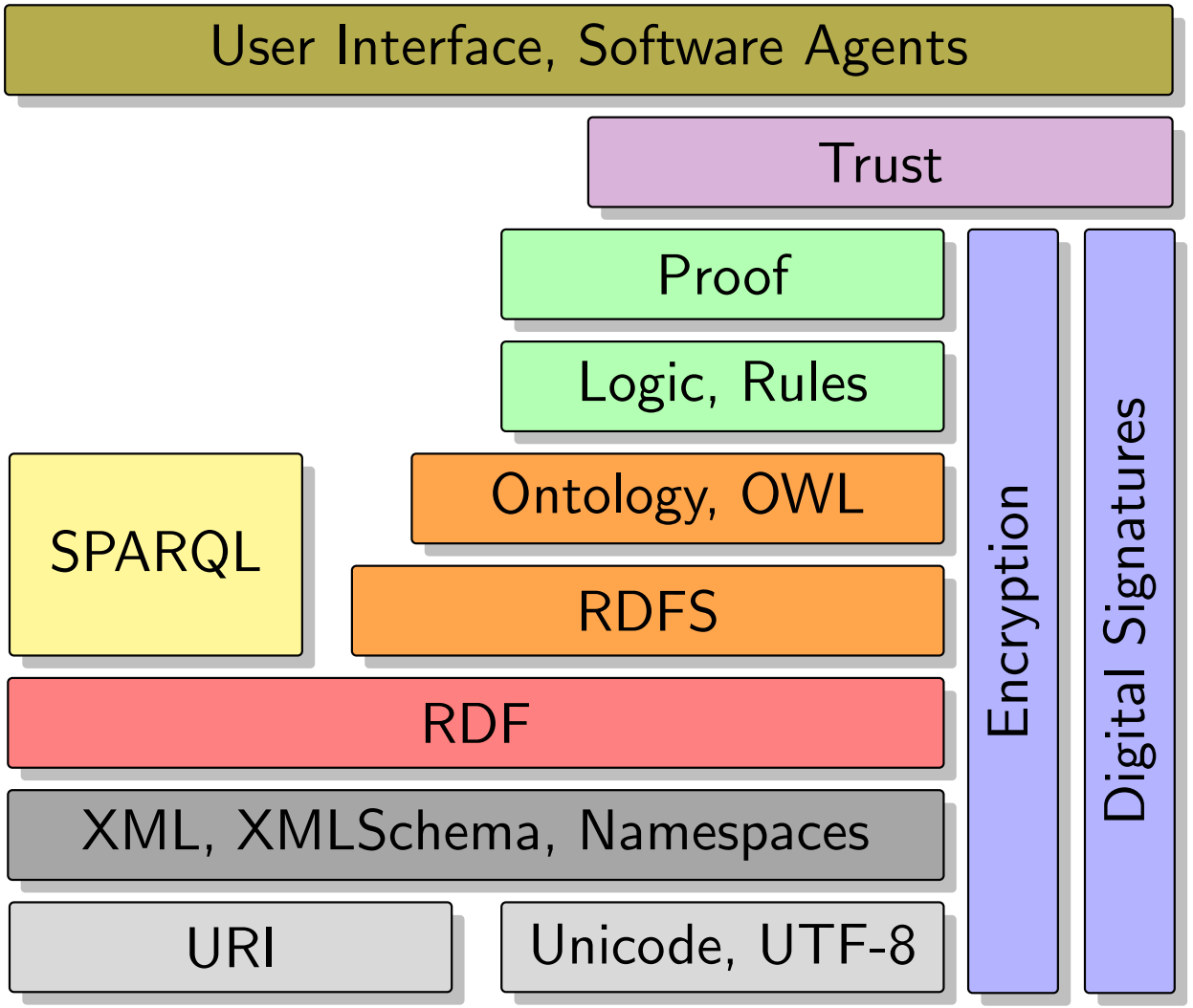
Camilo Thorne

Room 00.012
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
+49 (0) 711 685-81369
camilo.thorne@ims.uni-stuttgart.de

Semantic Web, SS 2017

Institut für Maschinelle Sprachverarbeitung

The Semantic Web Stack [W3C, Tim Berners-Lee]



Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

Accessing Information from Ontologies

- We have lots of information in RDF, RDFS
- We can use reasoning to make explicit information implicitly known by an ontology such as
  - What are the instances of class X?
  - Which classes are a subclass of Z?
  - To which classes does x belong?
  - ...

Institut für Maschinelle Sprachverarbeitung

Accessing Information from Ontologies

- We have lots of information in RDF, RDFS
- We can use reasoning to make explicit information implicitly known by an ontology such as
  - What are the instances of class X?
  - Which classes are a subclass of Z?
  - To which classes does x belong?
  - ...
- But sometimes we want to **retrieve** information stored in an ontology
  - Which German labels are contained in the ontology?
  - Which property connects the individuals x and y?
  - Which two pizzas have a common topping?
  - ...

Institut für Maschinelle Sprachverarbeitung

SPARQL: SPARQL Protocol and RDF Query Language

- SPARQL is a recursive acronym for "SPARQL Protocol and RDF Query Language"
- SPARQL is pronounced like "sparkle"
- SPARQL is a W3C recommendation since 2008
- SPARQL is used to query RDF(S) (remote) ontologies such as
  - DBpedia: http://dbpedia.org/sparql
  - YAGO: https://linkeddata1.calcul.u-psud.fr/sparql
  - ...
- Apart from the query language we are presenting here, SPARQL is also defines protocols for remote querying ontologies

▷ Inspired by relational DB query languages (= SQL)

Institut für Maschinelle Sprachverarbeitung

Triplestores and RDF/SPARQL APIs

- Systems supporting SPARQL querying over RDF(S) ontologies are known as **triplestores**:
  - Virtuoso: https://virtuoso.openlinksw.com/
  - Stardog: http://www.stardog.com/
  - Jena: https://jena.apache.org/
  - Ontop: http://ontop.inf.unibz.it/
  - ...
- Several APIs exist to communicate programmatically with triplestores over TCP-IP (and more in general, handle RDF(S) data plus more expressive standards)
  - Apache Jena: https://jena.apache.org/ (Java)
  - SPARQL wrapper: https://rdflib.github.io/sparqlwrapper/ (Python)
  - ...

Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

SPARQL Basic Syntax

Query composed of:

1. a head:

```
PREFIX p1: <uri1>
...
PREFIX p1: <uri1>
SELECT ?v1 ... ?vn
```

2. a body

```
WHERE{
 C(?v1,...,?vn,?vn+1,...,?vn+m)
}
```

`C[?v1,...,?vn,?vn+1,...,?vn+m]` denotes a sequence of RDF(S) triples; the `?v1,...,?vn` are the answer variables, whereas `?v1,...,?vn` serve to connect triples (join variables)

Institut für Maschinelle Sprachverarbeitung

SPARQL (Certain Answers) Semantics

Let $\mathcal{O}$ be an RDF(S) ontology. The semantics of a SPARQL query Q with body `C(?v1,...,?vn,?vn+1,...,?vn+m)` is the bag[a] of resources $\{\!|$ `r1`,..., `rn` $|\!\}$ that satisfy the RDF(S) entailment

$$\mathcal{O} \models \texttt{C[?v1,...,?vn,?vn+1,...,?vn+m]}[\texttt{r1,...,rn}/\texttt{?v1,...,?vn}]$$

where

- `?v1,...,?vn` are the head variables (variables we're asking for)
- `?v1,...,?vn` are the join variables (variables used to connect triples in the body)
- $[\texttt{r1,...,rn}/\texttt{?v1,...,?vn}]$ denotes the result of matching resource `ri` to variable `?vi`

[a] A bag $\{\!| \ldots |\!\}$ is a set with repetitions, i.e., $\{\!|a, a|\!\} \neq \{a\}$

Institut für Maschinelle Sprachverarbeitung

SPARQL Graph/Pattern Matching Semantics

- Certain answers semantics can be characterized (equivalently) as a graph matching task

- SPARQL matches graph patterns in the query with statements that form an RDF graph
- Every match of the query pattern is one returned line
- If the query contains variables, the variables get "bound" to the parts of the triple that correspond to them
- Several patterns are regarded as a conjunction
  (as if connected by `and`)

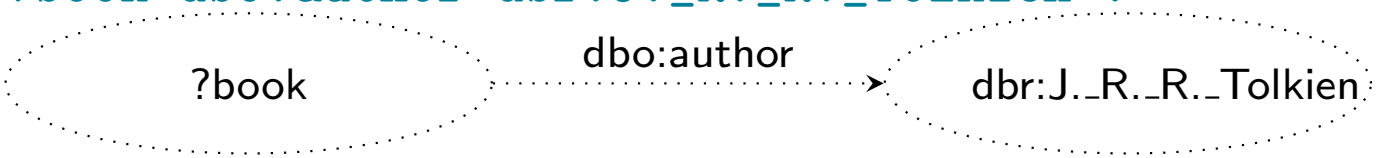
- We get as answers the same bags $\{\!|$ `r1`,. . . , `rn` $|\!\}$ of resources as with entailments

Institut für Maschinelle Sprachverarbeitung

SPARQL Basic Pattern Matching

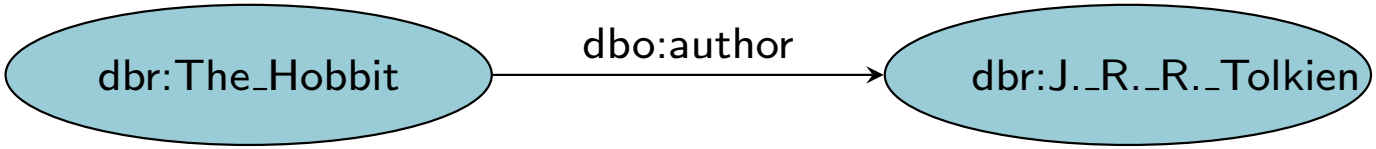
- Example query pattern:

```
?book dbo:author dbr:J._R._R._Tolkien .
```



- Example RDF triple:

```
dbr:The_Hobbit dbo:author dbr:J._R._R._Tolkien .
```

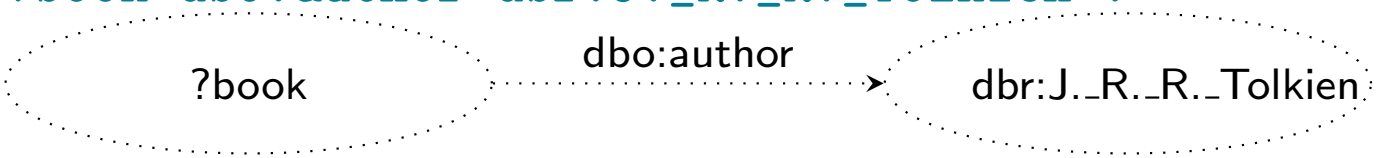


Institut für Maschinelle Sprachverarbeitung

SPARQL Basic Pattern Matching

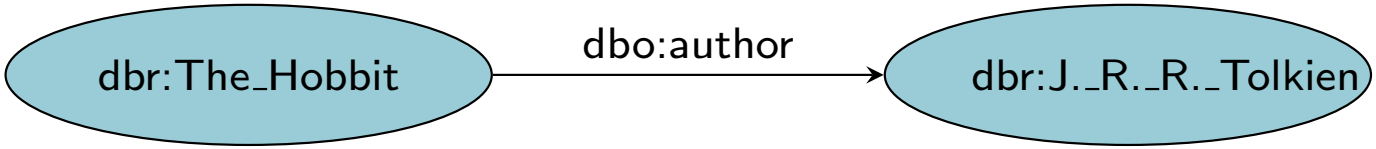
- Example query pattern:

```
?book dbo:author dbr:J._R._R._Tolkien .
```

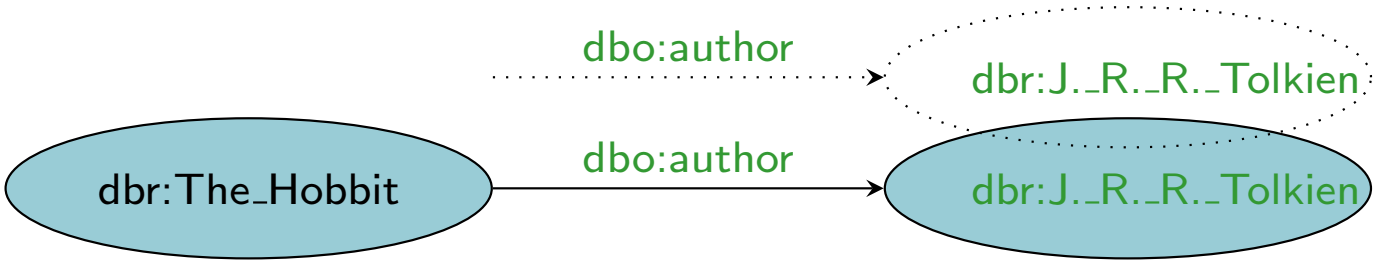


- Example RDF triple:

```
dbr:The_Hobbit dbo:author dbr:J._R._R._Tolkien .
```



- `dbo:author` and `dbr:J._R._R._Tolkien` match,

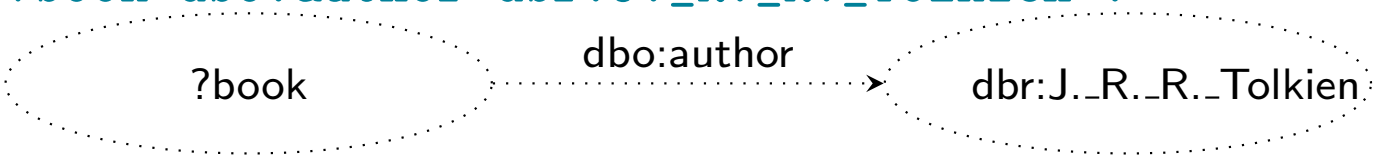


Institut für Maschinelle Sprachverarbeitung

SPARQL Basic Pattern Matching

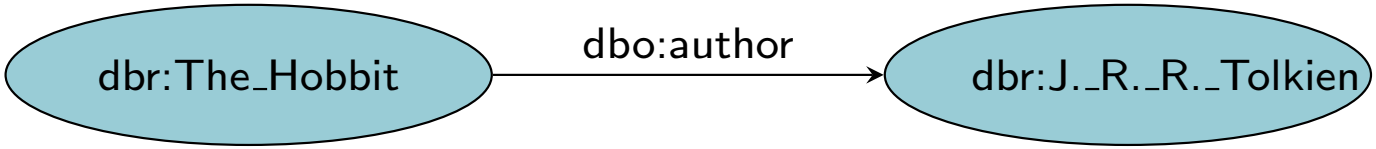
- Example query pattern:

```
?book dbo:author dbr:J._R._R._Tolkien .
```

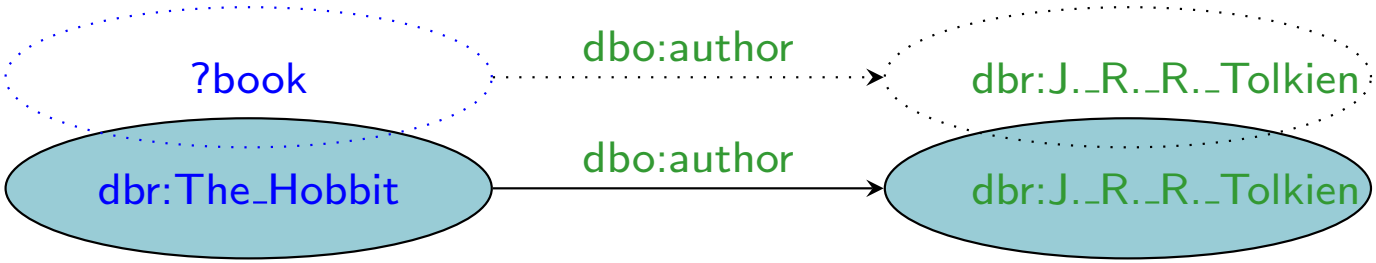


- Example RDF triple:

```
dbr:The_Hobbit dbo:author dbr:J._R._R._Tolkien .
```



- `dbo:author` and `dbr:J._R._R._Tolkien` match, `?book` gets bound to `dbr:The_Hobbit`.



Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

SPARQL Example (1)

- Query **body**:

```
?book dbo:author ?author .
?author rdf:type ?personclass .
?book dbo:literaryGenre ?genre .
```

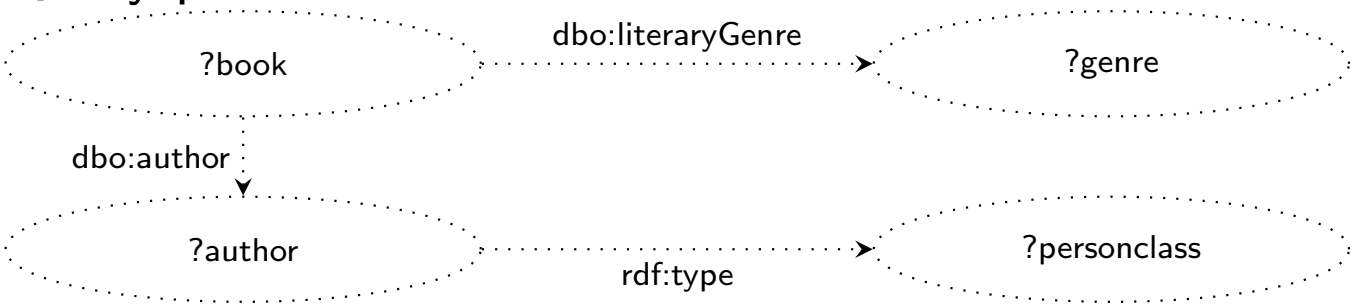
Institut für Maschinelle Sprachverarbeitung

SPARQL Example (1)

- Query **body**:

```
?book dbo:author ?author .
?author rdf:type ?personclass .
?book dbo:literaryGenre ?genre .
```

- Query pattern:



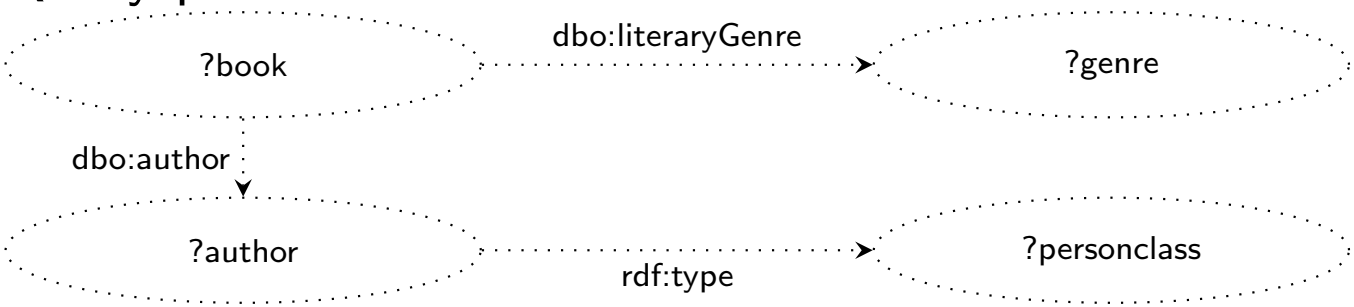
Institut für Maschinelle Sprachverarbeitung

SPARQL Example (1)

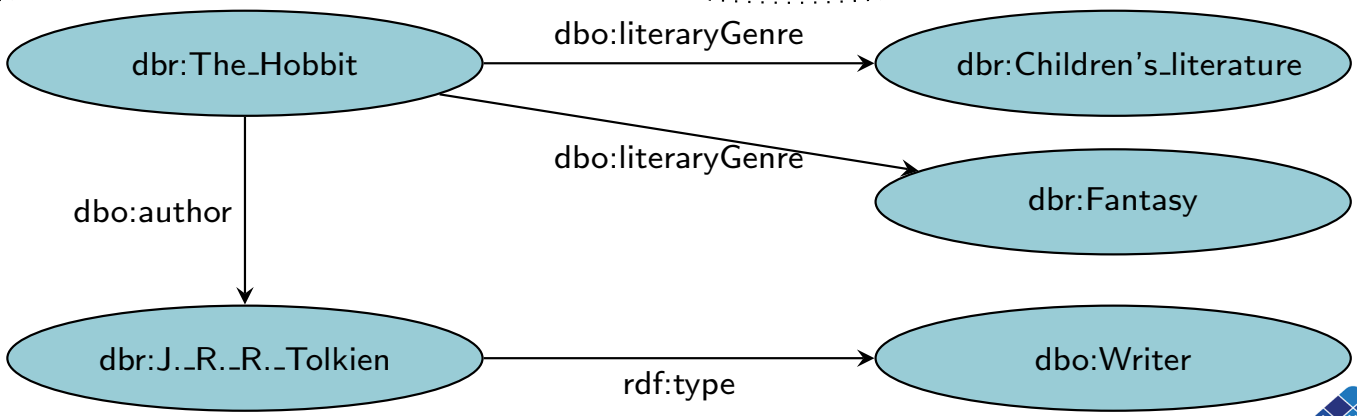
- Query **body**:

```
?book dbo:author ?author .
?author rdf:type ?personclass .
?book dbo:literaryGenre ?genre .
```

- Query pattern:



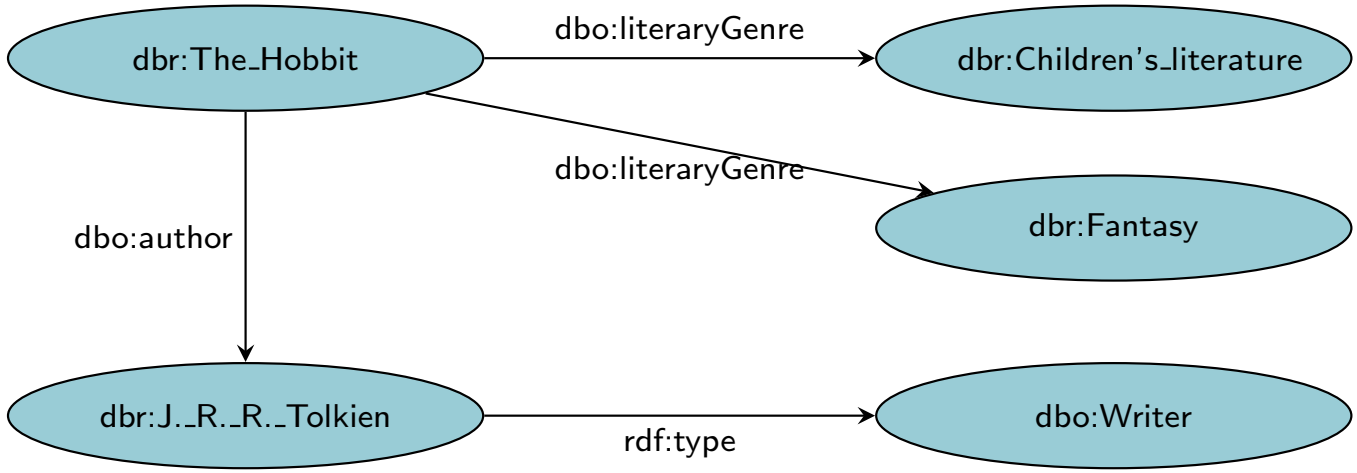
- RDF graph:



Institut für Maschinelle Sprachverarbeitung

SPARQL Example (2)

- Matching:



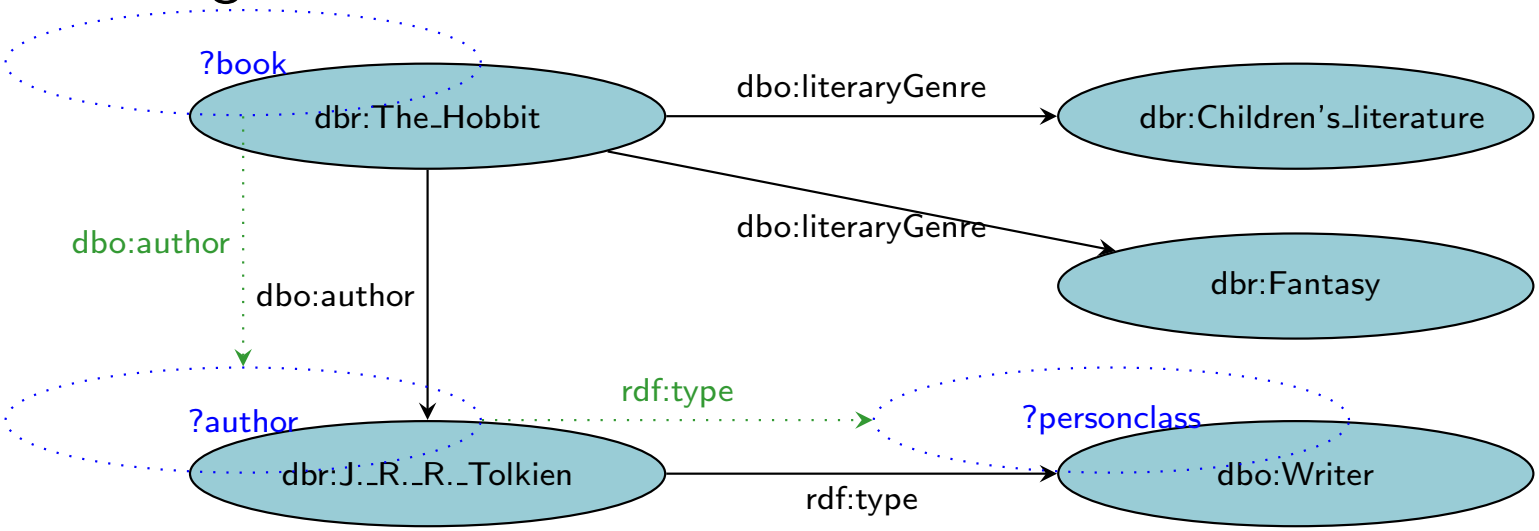
- Result:

?book	?author	?personclass	?genre

Institut für Maschinelle Sprachverarbeitung

SPARQL Example (2)

- Matching:



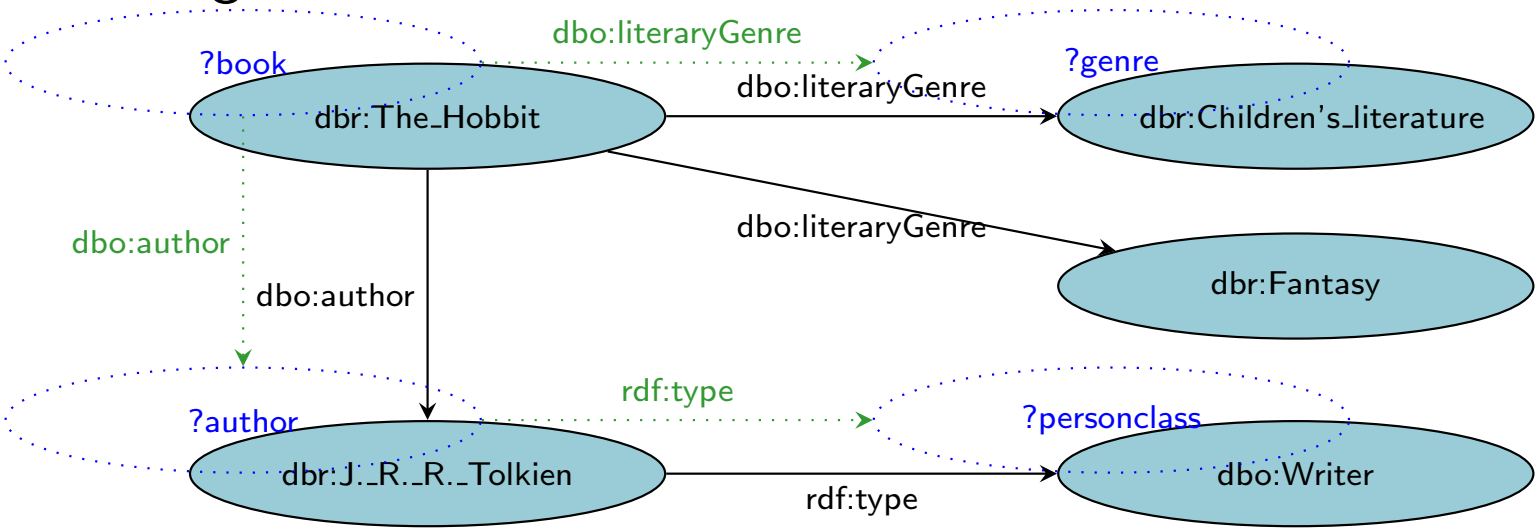
- Result:

?book	?author	?personclass	?genre
dbr:The_Hobbit	dbr:J._R._R._Tolkien	dbo:Writer	

Institut für Maschinelle Sprachverarbeitung

SPARQL Example (2)

- Matching:



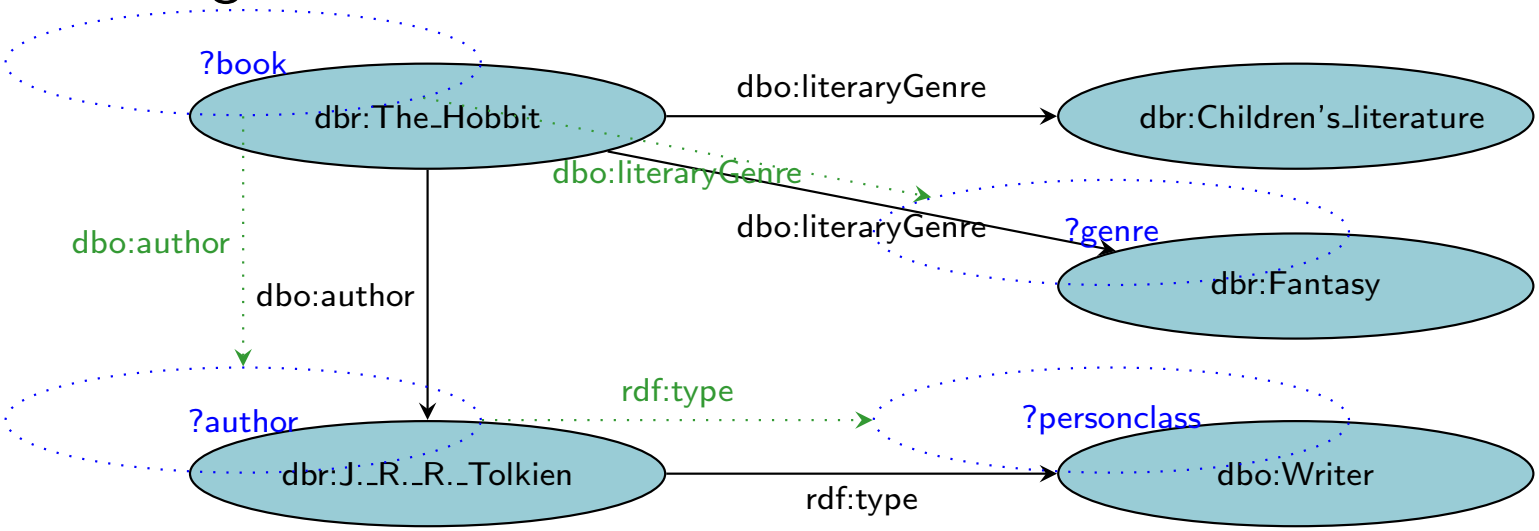
- Result:

?book	?author	?personclass	?genre
dbr:The_Hobbit	dbr:J._R._R._Tolkien	dbo:Writer	dbr:Children's_literature

Institut für Maschinelle Sprachverarbeitung

SPARQL Example (2)

- Matching:



- Result:

?book	?author	?personclass	?genre
dbr:The_Hobbit	dbr:J._R._R._Tolkien	dbo:Writer	dbr:Children's_literature
dbr:The_Hobbit	dbr:J._R._R._Tolkien	dbo:Writer	dbr:Fantasy

Institut für Maschinelle Sprachverarbeitung

SPARQL Example (3)

Complete query[1]:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?book ?author ?personclass ?genre
WHERE {
   ?book dbo:author ?author .
   ?author rdf:type ?personclass .
   ?book dbo:literaryGenre ?genre .
}
```

Result:

?book	?author	?personclass	?genre
dbr:The_Hobbit	dbr:J._R._R._Tolkien	dbo:Writer	dbr:Children's_literature
dbr:The_Hobbit	dbr:J._R._R._Tolkien	dbo:Writer	dbr:Fantasy
dbr:A_Memory_of_Light	dbr:Brandon_Sanderson	dbo:Artist	dbr:Fantasy_literature
dbr:A_Memory_of_Light	dbr:Brandon_Sanderson	dbo:Writer	dbr:Fantasy_literature
...			

[1] Depending on the query interface or **endpoint**, your results will be formatted with prefixes (as in the examples) or as full URIs

Institut für Maschinelle Sprachverarbeitung

SPARQL Syntax – Prefix

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
```

- `PREFIX` declares a namespace and the abbreviating prefix that can be used to access it.

Institut für Maschinelle Sprachverarbeitung

SPARQL Syntax – Result Clause

```
SELECT ?book ?author ?personclass ?genre
```

- The result clause identifies what information to return from the query
- Variables are denoted by a `?` in front of the name
- Only the information bound to the variables in the result clause is returned
- The number of lines returned is depending on the number of triples found, each line gives one possible way the variables can be bound to things in the graph

Institut für Maschinelle Sprachverarbeitung

SPARQL Syntax – Query Patterns

```
WHERE {
   ?book dbo:author ?author .
   ?author rdf:type ?personclass .
   ?book dbo:literaryGenre ?genre .
}
```

- The query pattern is the main part of the SPARQL query
- Query patterns are written in Turtle-Syntax for RDF
- Query patterns are **RDF graph patterns** for statements with subject, predicate and object
- Patterns may contain variables replacing any part of the statement
- Every pattern is terminated by a **dot**
- A sequence of patterns is interpreted as a **conjunction** ($\approx$ relational join)

Institut für Maschinelle Sprachverarbeitung

SPARQL and Data Types

```
ex:bsp1 ex:p "test" .
ex:bsp2 ex:p "test"^^xsd:string .
ex:bsp3 ex:p "test"@de .
ex:bsp4 ex:p "42"^^xsd:integer .
```

- Remember: Literals are typed in RDF
- SPARQL queries are sensitive to data types, exact matches are necessary
- The query `?subject ex:p "test".` will give `ex:bsp1` as the only result
- To get `ex:bsp3`, the query `?subject ex:p "test"@de .` must be used
- For numbers a short form `?subject ex:p 42 .` is possible, the number is interpreted as `xsd:integer`, `xsd:decimal`, or `xsd:double` according to the surface form

Institut für Maschinelle Sprachverarbeitung

Quiz: Basic SPARQL Queries

Given are RDF statements about mountains in this form:

```
dbr:Birkenkopf rdf:type dbo:Mountain .
dbr:Birkenkopf dbp:elevationM 511 .
dbr:Birkenkopf dbp:name "Birkenkopf" .
dbr:Birkenkopf dbo:locationArea dbr:Baden-Wurttemberg .
```

Complete the query to retrieve name and height of all mountains:

```
SELECT ?name ?elevation
WHERE {  ...  }
```

A)
```
?name rdfs:subClassOf ?restriction .
?restriction owl:onProperty dbp:elevationM .
?restriction owl:someValuesFrom ?elevation .
```

B)
```
?name rdf:type dbo:Mountain .
?name dbp:elevationM ?elevation .
```

C)
```
?mountain rdf:type dbo:Mountain .
?mountain dbp:elevationM ?elevation .
?mountain dbp:name ?name .
```

Institut für Maschinelle Sprachverarbeitung

SPARQL Algebra Semantics

- Certain answers semantics can be characterized also in terms of **relational algebra**
- Query answering is based on computing table joins (query body) and projecting away answer columns (query head), e.g., in

```
SELECT ?x WHERE {?x ?y z. ?m ?y z.}
```

  - ▷ triples are interpreted as tables over attributes and values: triple `?x ?y z.` is interpreted as table over columns X, Y and $\{z\}$ with named resource `z` as value z

$$T[X, Y, z]$$

  - ▷ shared (join) variables and resources are interpreted as table joins:

$$T[X, Y, z] \bowtie_{Y,z} T'[M, Y, z]$$

  - ▷ head variables correspond to projections:

$$\pi_X(T[X, Y, z] \bowtie_{Y,z} T'[M, Y, z])$$

- We get again the same bags $\{\!|$ `r1`,. . . , `rn` $|\!\}$ of answers

Institut für Maschinelle Sprachverarbeitung

Triplestores and Semantics

- Entailment, algebra and pattern matching semantics, while equivalent have different properties
- For us humans, the pattern matching semantics is the easiest to understand
- But in some cases algebra semantics is more computationally efficient, as it leverages relational indexing techniques
- Different triplestores implement different semantics
  - Virtuoso implements the algebra semantics
  - Stardog implements the graph matching semantics
  - Ontop implements the entailment semantics
  - ...
- Which one to choose will depend ultimately in your specific application use case

Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

Suggested Reading


Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph and York Sure. Semantic Web. Grundlagen. Springer textbook, 2008. (Chapter 7)


Pascal Hitzler, Markus Krötzsch and Sebastian Rudolph. Foundations of Semantic Web Technologies. Chapman & Hall/CRC, 2009. (Chapter 7)

Institut für Maschinelle Sprachverarbeitung