Semantic Web
SPARQL

Gerd Gröner, Matthias Thimm

{groener,thimm}@uni-koblenz.de

Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau

July 15, 2013

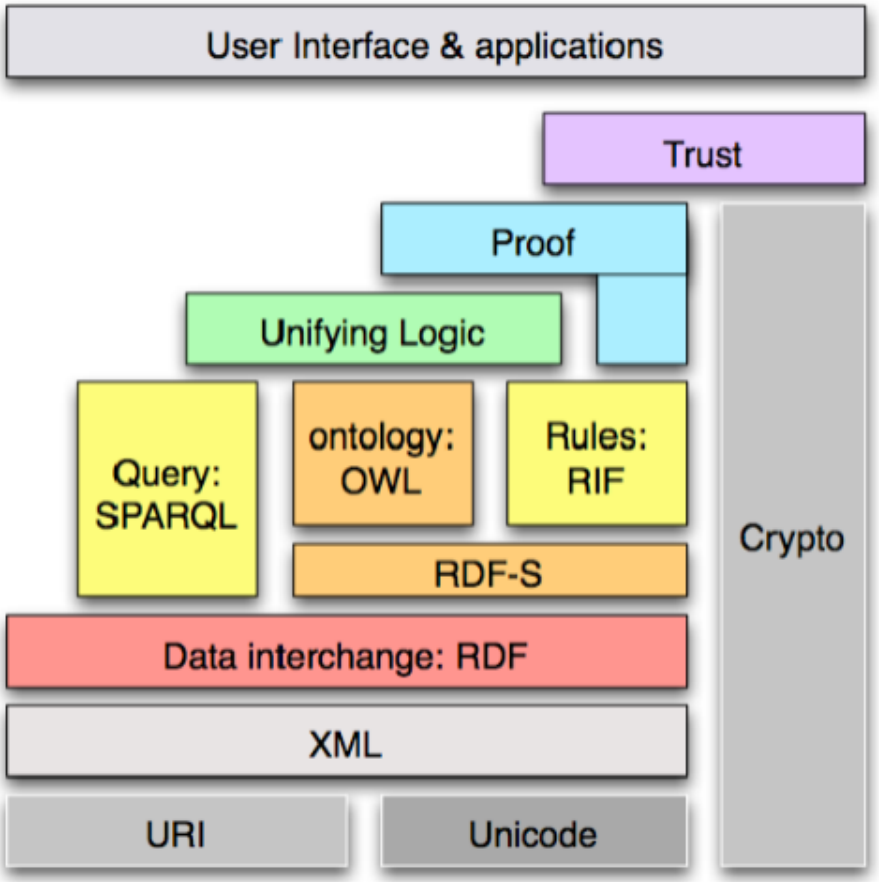
Agenda for this Week

- SPARQL (2 lectures)
- Semantic Search
- Ontology Engineering
- Pattern-based Ontology Design
- Semantic Web Applications, Closing

Outline

Outline

Semantic Web Layer Cake



Goals of this lecture

- Understanding basic and advanced queries using SPARQL
- Gain knowledge about how to query data sources

SPARQL

- W3C Recommendation 15 January 2008
  - `http://www.w3.org/TR/rdf-sparql-query/`
- Standard query language for RDF
  - Native RDF knowledge bases
  - knowledge bases viewed as RDF via middleware
- Language for querying for graph patterns
  - Includes unions, conjunctions and optional patterns
  - No support for inserts or updates
- Supports extensible testing for values and constraints

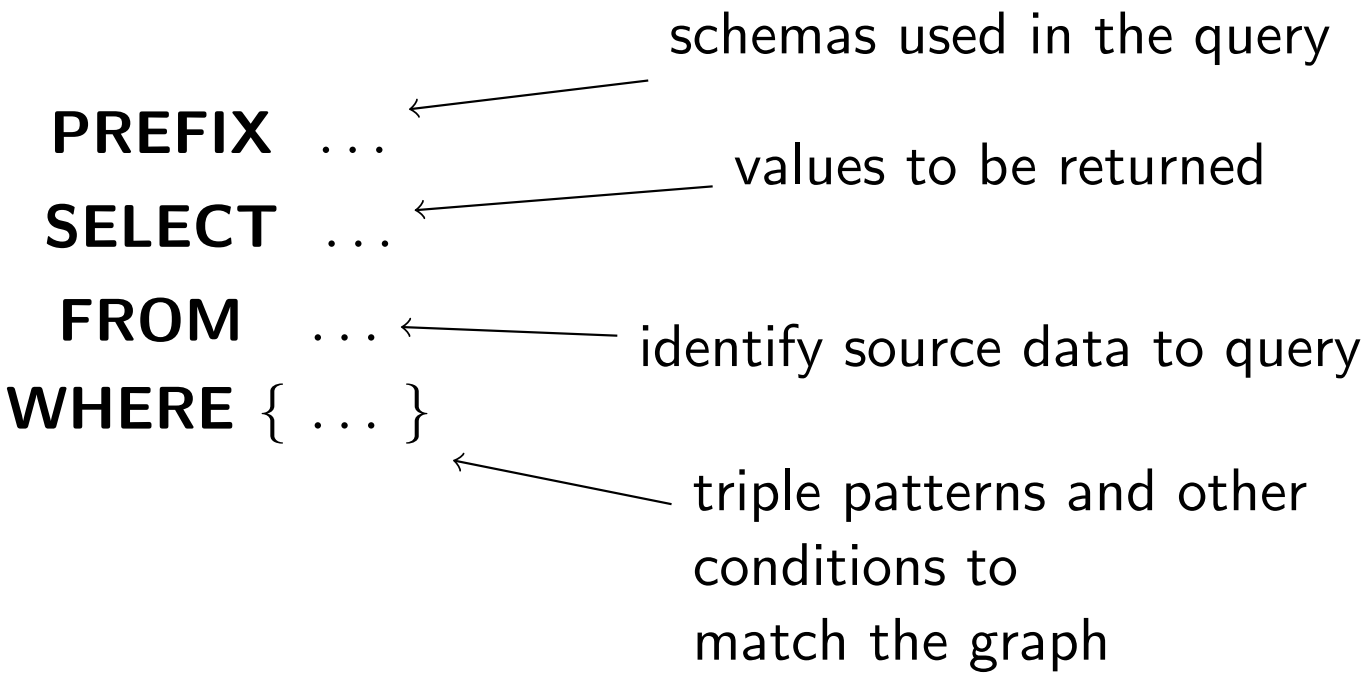
SPARQL 1.1

- SPARQL 1.1 (W3C Recommendation 21 March 2013)
  - `http://www.w3.org/TR/sparql11-query/`
- SPARQL 1.1 Query - Adds support for aggregates, subqueries, projected expressions, and negation to the SPARQL query language.
- SPARQL 1.1 Update - Defines an update language for RDF graphs.
- SPARQL 1.1 Protocol - Defines an abstract interface and HTTP bindings for a protocol to issue SPARQL Query and SPARQL Update statements against a SPARQL endpoint.
- SPARQL 1.1 Service Description - Defines a vocabulary and discovery mechanism for describing the capabilities of a SPARQL endpoint.
- SPARQL 1.1 Uniform HTTP Protocol for Managing RDF Graphs - Describes the use of the HTTP protocol for managing named RDF graphs on an HTTP server.
- SPARQL 1.1 Entailment Regimes - Defines conditions under which SPARQL queries can be used with entailment regimes such as RDF, RDF Schema, OWL, or RIF.
- SPARQL 1.1 Property Paths - Defines a more succinct way to write parts of basic graph patterns and also extend matching of triple pattern to arbitrary length paths.

SPARQL Query Types

- **SELECT**
  - returns the set of variables bound in a query pattern match
- **CONSTRUCT**
  - returns an RDF graph constructed by substituting variables in a set of triple templates
- **DESCRIBE**
  - returns an RDF graph that describes the resources found
- **ASK**
  - returns whether a query pattern matches any triples or not (true / false query)

SPARQL SELECT Query



SPARQL ASK Query – Example

Query: Is the river Amazon longer than the river Nile?

```
PREFIX  prop: <http://dbpedia.org/property/>
ASK
{
<http://dbpedia.org/resource/Amazon_River>  prop:length  ?amazon .
<http://dbpedia.org/resource/Nile> prop:length  ?nile .
  FILTER(?amazon > ?nile) .
}
```

Answer: true

SPARQL CONSTRUCT Query – Example

CONSTRUCT query is used to build an RDF graph.

```
PREFIX  ex:  <http://example.org/schema/>
PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT
{
  ?person rdf:type    ex:Adult.
}
WHERE
{
   ?person  ex:age  ?age
   FILTER (?age > 17)
}
```

Result is an RDF graph.

SPARQL DESCRIBE Query – Example

DESCRIBE query is used to provide further information about an RDF resource

```
DESCRIBE    <http://dbpedia.org/resource/Amazon_River>
```

Result is an extensive description of the resource “Amazon River”.

Here is just an excerpt:

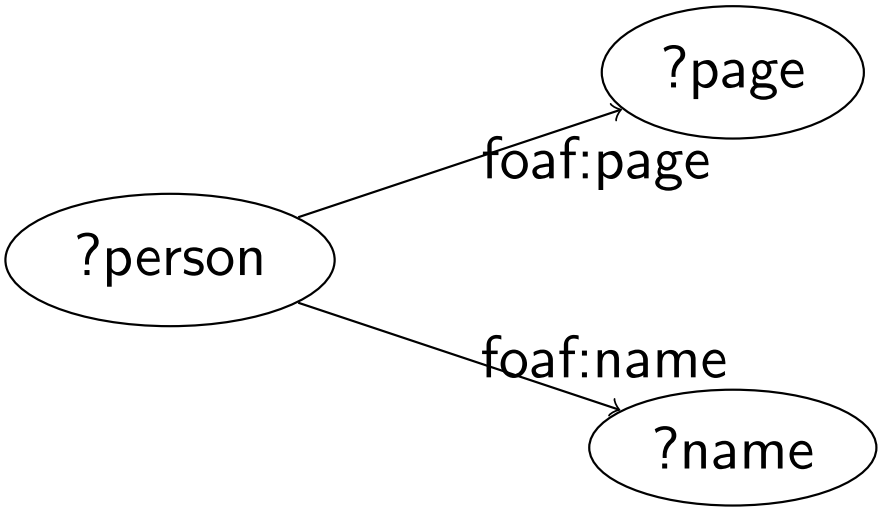
```
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbpedia:    <http://dbpedia.org/resource/> .
@prefix dbpedia-owl:   <http://dbpedia.org/ontology/> .

dbpedia:Amazon_River rdf:type dbpedia-owl:NaturalPlace ,
        dbpedia-owl:Place .
 ...
```

Basic SPARQL Patterns

- Triple Pattern
  - Similar to an RDF Triple
    - subject, predicate, object
  - Any component can be a query variable
  - Any combination of variables in the query is allowed
- Matching patterns in the **WHERE** clause
  - Matching conjunction of triple patterns
  - Matching a triple pattern to a graph
    - Finding bindings between variables and RDF Terms
  - Underneath use of reasoners
    - Inferring triples originally not present in the knowledge base

SPARQL Example

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
    ?person foaf:page ?page .
    ?person foaf:name ?name
}
```



Query Example

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
_:a foaf:name ''Steffen Staab'' .
_:a foaf:homepage <http://www.uni-koblenz.de/~staab> .
_:b foaf:name ''Gerd Groener'' .
_:b foaf:homepage <http://www.uni-koblenz.de/~groener> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:homepage ?page .
 ?person foaf:name ?name }
```

Query Result:

name	page
"Steffen Staab"	<http://www.uni-koblenz.de/~staab>
"Gerd Groener"	<http://www.uni-koblenz.de/~groener>

Querying for Blank Nodes

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
_:a  foaf:name ''Steffen Staab'' .
_:a  foaf:homepage  <http://www.uni-koblenz.de/~staab> .
_:b  foaf:name ''Gerd Groener'' .
_:b  foaf:homepage  <http://www.uni-koblenz.de/~groener> .
```

Query:

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?person ?name ?page
WHERE {
 ?person  foaf:homepage ?page .
 ?person  foaf:name ?name }
```

Query Result:

person	name	page
_:a	"Steffen Staab"	<http://www.uni-koblenz.de/~staab>
_:b	"Gerd Groener"	<http://www.uni-koblenz.de/~groener>

Filters

- Further constrain graph patterns
- Applied to the **whole group** of patterns
- FILTER clause:
  - Support for AND and OR logic operators
  - Extensive applications for testing literals
  - Support for numerical operations
  - Support for math equality operators for literals ($<, =, >$)
  - Use of regular expressions
  - support for data types defined in XSL (e.g., comparision of dates, time)
  - possible comparison of resources (equal, not equal)
  - even possible user extensions

FILTER – Value Constraints

Data:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
ex:book1  dc:title  ''SPARQL Tutorial'' .
ex:book1  ns:price  42 .
ex:book2  dc:title  ''The Semantic Web''  .
ex:book2  ns:price  23 .
```

Query:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x  ns:price  ?price .
        FILTER ?price < 30 .
        ?x  dc:title  ?title }
```

Query Result:

title	price
"The Semantic Web"	23

FILTER – Value Constraints

Data:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
ex:book1  dc:title  ''SPARQL Tutorial'' .
ex:book1  ns:price  42 .
ex:book2  dc:title  ''The Semantic Web''  .
ex:book2  ns:price  23 .
```

Query:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price
WHERE { ?x  ns:price  ?price .
        FILTER ?price < 30 .
        ?x  dc:title  ?title }
```

Query Result:

title	**price**
"The Semantic Web"	23

Scope of Filters

- Filters are applied to the whole group of patterns where they appear.

```
{ ?x foaf:name ?name .
  ?x foaf:homepage ?page .
  FILTER regex(?name, ``Steffen'') }
```

```
{ ?x foaf:name ?name .
  FILTER regex(?name, ``Steffen'') .
  ?x foaf:homepage ?page }
```

```
{ FILTER regex(?name, ``Steffen'') .
  ?x foaf:name ?name .
  ?x foaf:homepage ?page }
```

- These patters are equivalent — have the same solution.

Scope of Filters

- Filters are applied to the whole group of patterns where they appear.

```
{ ?x foaf:name ?name .
  ?x foaf:homepage ?page .
  FILTER regex(?name, ``Steffen'') }
```

```
{ ?x foaf:name ?name .
  FILTER regex(?name, ``Steffen'') .
  ?x foaf:homepage ?page }
```

```
{ FILTER regex(?name, ``Steffen'') .
  ?x foaf:name ?name .
  ?x foaf:homepage ?page }
```

- These patters are equivalent — have the same solution.

FILTER – Regular Expressions

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
 _:a foaf:name ''Steffen Staab'' .
 _:a foaf:homepage <http://www.uni-koblenz.de/~staab> .
 _:b foaf:name ''Gerd Groener'' .
 _:b foaf:homepage <http://www.uni-koblenz.de/~groener> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:homepage ?page .
 ?person foaf:name ?name .
 FILTER regex(?name, ''Steffen'')
}
```

Query Result:

name	**page**
"Steffen Staab"	<http://www.uni-koblenz.de/~staab>

FILTER – Regular Expressions

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
 _:a foaf:name ''Steffen Staab'' .
 _:a foaf:homepage <http://www.uni-koblenz.de/~staab> .
 _:b foaf:name ''Gerd Groener'' .
 _:b foaf:homepage <http://www.uni-koblenz.de/~groener> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:homepage ?page .
 ?person foaf:name ?name .
 FILTER regex(?name, ''i'', ''groener'')
}
```

Query Result: case insensitive

name	**page**
"Gerd Groener"	<http://www.uni-koblenz.de/~groener>

Optional Patterns

- include optional triple patterns to match
- optional is a pattern itself — can include further constraints

```
SELECT
WHERE {
    ...
   OPTIONAL { ... }
}
```

- OPTIONAL is left-associative (i.e., operators are grouped from left to right)
  pattern OPTIONAL {pattern } OPTIONAL { pattern }
  is the same as
  { pattern OPTIONAL { pattern } } OPTIONAL { pattern }

Query Example

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
 _:a foaf:name ''Steffen Staab'' .
 _:a foaf:homepage <http://www.uni-koblenz.de/~staab> .
 _:b foaf:name ''Gerd Groener'' .
 _:b foaf:mbox <groener@uni-koblenz.de> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:name ?name .
 ?person foaf:homepage ?page .
}
```

Query Result:

name	page
"Steffen Staab"	<http://www.uni-koblenz.de/~staab>

Query Example

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
 _:a foaf:name ''Steffen Staab'' .
 _:a foaf:homepage <http://www.uni-koblenz.de/~staab> .
 _:b foaf:name ''Gerd Groener'' .
 _:b foaf:mbox <groener@uni-koblenz.de> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:name ?name .
 ?person foaf:homepage ?page .
}
```

Query Result:

name	**page**
"Steffen Staab"	<http://www.uni-koblenz.de/~staab>

Query Example — OPTIONAL

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
 _:a foaf:name ''Steffen Staab'' .
 _:a foaf:homepage <http://www.uni-koblenz.de/~staab> .
 _:b foaf:name ''Gerd Groener'' .
 _:b foaf:mbox <groener@uni-koblenz.de> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:name ?name .
 OPTIONAL ?person foaf:homepage ?page .
}
```

Query Result:

name	page
"Steffen Staab"	<http://www.uni-koblenz.de/~staab>
"Gerd Groener"	

Query Example — OPTIONAL

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
 _:a foaf:name ''Steffen Staab'' .
 _:a foaf:homepage <http://www.uni-koblenz.de/~staab> .
 _:b foaf:name ''Gerd Groener'' .
 _:b foaf:mbox <groener@uni-koblenz.de> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:name ?name .
 OPTIONAL ?person foaf:homepage ?page .
}
```

Query Result:

name	**page**
"Steffen Staab"	<http://www.uni-koblenz.de/~staab>
"Gerd Groener"	

Union of Graph Pattern

UNION

- combining alternative graph patterns
- if more than one of the alternatives matches, all the possible pattern solutions are included in result

```
SELECT
WHERE {
      { pattern }
      UNION
      { pattern }
}
```

Query Example — UNION

Data:

```
@prefix dc10: <http://purl.org/dc/elements/1.0/> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
book1  dc10:title  ''SPARQL Tutorial'' .
book1  dc10:creator  ''Alice'' .
book2  dc11:title  ''The Semantic Web'' .
book2  dc11:creator  ''Robert'' .
```

Query:

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { { ?x dc10:title ?title }
        UNION
        { ?x dc11:title ?title } }
```

Query Result: title

"SPARQL Tutorial"
"The Semantic Web"

Query Example — UNION

Data:

```
@prefix dc10: <http://purl.org/dc/elements/1.0/> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
book1  dc10:title  ''SPARQL Tutorial'' .
book1  dc10:creator  ''Alice'' .
book2  dc11:title  ''The Semantic Web'' .
book2  dc11:creator  ''Robert'' .
```

Query:

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT ?title
WHERE { { ?x dc10:title ?title }
        UNION
        { ?x dc11:title ?title } }
```

Query Result: title

"SPARQL Tutorial"

"The Semantic Web"

Modification of the Result Set

Result of SPARQL query can be further modified

- ORDER BY
  - sort results alphabetically / numerically by specific variable
- LIMIT
  - Limit number of returned results (only top n results)
- OFFSET
  - Skip n top results and return the rest

These expressions can be combined in a query.

Sequencing and Limiting Results

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
   ?person  foaf:homepage  ?page .
   ?person  foaf:name  ?name
}
ORDER BY ?name
LIMIT 20
OFFSET 10
```

Bound Variables

- one of the FILTER expressions
- supports testing if a variable in a query can be bound to an instance in the knowledge base
- mostly used for negation as failure

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
  ?person foaf:name ?name .
  OPTIONAL { ?person foaf:knows ?x . }
  FILTER ( ! bound(?x))
}
```

Bound Variables – Example

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
 _:a foaf:name ''Steffen Staab'' .
 _:a foaf:homepage <http://www.uni-koblenz.de/~staab> .
 _:b foaf:name ''Gerd Groener'' .
 _:b foaf:mbox <groener@uni-koblenz.de> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:name ?name .
  OPTIONAL { ?person foaf:homepage ?page .}
  FILTER (bound(?page))
}
```

Query Result:

name	page
"Steffen Staab"	<http://www.uni-koblenz.de/~staab>

Bound Variables – Example

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
 _:a foaf:name ''Steffen Staab'' .
 _:a foaf:homepage <http://www.uni-koblenz.de/~staab> .
 _:b foaf:name ''Gerd Groener'' .
 _:b foaf:mbox <groener@uni-koblenz.de> .
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?page
WHERE {
 ?person foaf:name ?name .
  OPTIONAL { ?person foaf:homepage ?page .}
  FILTER (bound(?page))
}
```

Query Result:

name	**page**
"Steffen Staab"	<http://www.uni-koblenz.de/~staab>

Tricky Negation

Find people who do **not** know Steffen.
First attempt:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
   ?person  foaf:name  ?name .
   ?person  foaf:knows  ?x .
   FILTER ( ?x != ''Steffen'' ) }
```

Data:... we know that ...

```
''Paul''  foaf:knows  ''Steffen''
''Paul''  foaf:knows  ''Sergej''
```

... so "Paul" is still a valid answer (do you know why?)

... and we do not want this!

Negation with bound

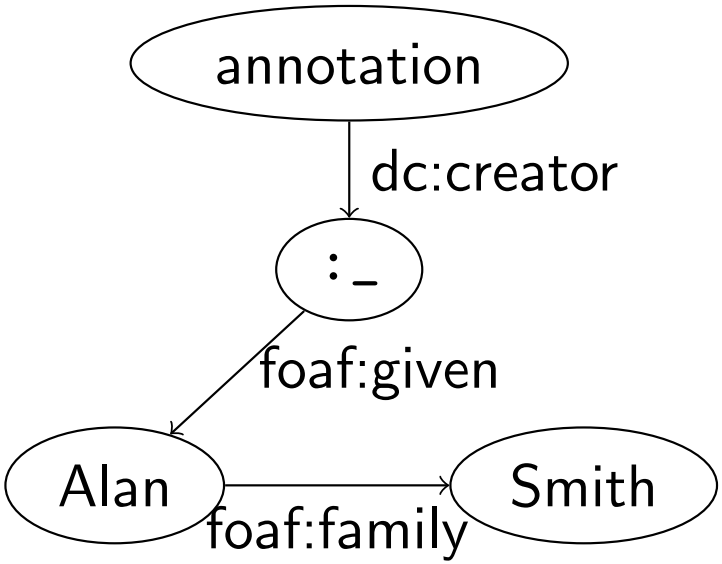
Find people who do **not** know Steffen.

- now the correct way using bound expression and optional graph pattern

```
PREFIX foaf: http://xmlns.com/foaf/0.1/
SELECT ?name
WHERE {
  ?person foaf:name ?name .
  OPTIONAL { ?person foaf:knows ?x .
             FILTER ( ?x = ``Steffen'' ) }
  FILTER ( ! bound(?x) )
}
```

Other SPARQL Filter Expressions

Testing if a bounded variable is a blank node.



Other SPARQL Filter Expressions (2)

- **isBlank:** Testing if a bounded variable is a blank node.

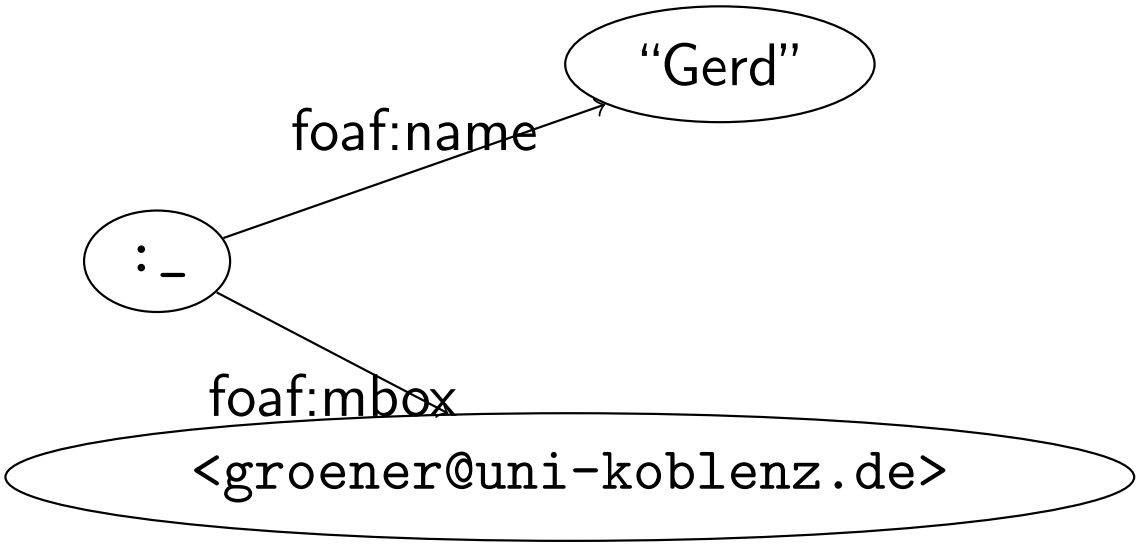
```
SELECT ?given ?family
WHERE { ?annotation  dc:creator ?c .
          OPTIONAL  {
            ?c foaf:given ?given .
            ?c foaf:family ?family }.
          FILTER isBlank(?c) }
```

- **lang:** Accessing the language of a literal

```
SELECT ?name ?mbox
WHERE { ?x foaf:name ?name .
          ?x foaf:mbox ?mbox .
          FILTER ( lang(?name) =  ''DE'' ) }
```

Other SPARQL Filter Expressions (3)

Testing if a bounded variable is a literal (not a resource).



Other SPARQL Filter Expressions (4)

- **isLiteral:** Testing if a bounded variable is a literal (not a resource)

```
SELECT ?name ?mbox
WHERE { ?x foaf:name ?name .
        ?x foaf:mbox ?mbox .
        FILTER isLiteral(?mbox) }
```

- **str:** Converting resource URI to string for regular expression matching

```
SELECT ?name ?mbox
WHERE { ?x foaf:name ?name .
        ?x foaf:mbox ?mbox .
        FILTER regex(str(?mbox), ''@uni-koblenz.de'') }
```

Equal Terms and Same Terms

- check if two terms are equal or if they describe the same entity

  ```
  term1 = term2 or
  sameTerm(term1, term2)
  ```

- returns true if
  - terms are of the same type (URI, literal, blank node)
  - two terms represent URIs are equivalent
  - two terms represent literals are equivalent
  - two terms are bound by the same blank node
- same entity can have even different URIs, but connected with owl:sameAs

Equal Terms

Find people who have the same email address, but use different names

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ''Alice''.
_:a foaf:mbox <mailto:alice@work.example> .
_:b foaf:name ''Ms A.'' .
_:b foaf:mbox <mailto:alice@work.example> .
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name1 ?name2
WHERE {
  ?x foaf:name ?name1 .
  ?x foaf:mbox ?mbox1 .
  ?y foaf:name ?name2 .
  ?y foaf:mbox ?mbox2 .
  FILTER ( sameTerm(?mbox1, ?mbox2) && ?name1 != ?name2) }
```

User-defined Functions

FILTER enables using user-defined expressions

```
PREFIX aGeo: <http://example.org/geo#>
SELECT ?neighbor WHERE {
   ?a   aGeo:placeName   ''Koblenz''  .
   ?a   aGeo:location    ?axLoc  .
   ?a   aGeo:location    ?ayLoc  .
   ?b   aGeo:placeName   ?neighbor  .
   ?b   aGeo:location    ?bxLoc  .
   ?b   aGeo:location    ?byLoc  .
   FILTER
    ( aGeo:distance(?axLoc, ?ayLoc, ?bxLoc, ?byLoc) < 5 )
}
```

Definition of user function geometric distance between two points described by (x, y) coordinates:

xsd:double aGeo:distance (numeric x1, numeric y1, numeric x2, numeric y2)

User-defined Functions

FILTER enables using user-defined expressions

```
PREFIX aGeo: <http://example.org/geo#>
SELECT ?neighbor WHERE {
   ?a   aGeo:placeName   ''Koblenz''  .
   ?a   aGeo:location    ?axLoc  .
   ?a   aGeo:location    ?ayLoc  .
   ?b   aGeo:placeName   ?neighbor  .
   ?b   aGeo:location    ?bxLoc  .
   ?b   aGeo:location    ?byLoc  .
   FILTER
    ( aGeo:distance(?axLoc, ?ayLoc, ?bxLoc, ?byLoc) < 5 )
}
```

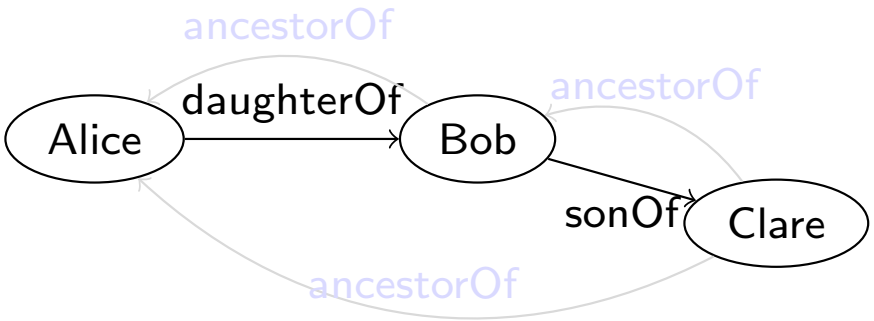
Definition of user function geometric distance between two points described by (x, y) coordinates:

xsd:double aGeo:distance (numeric x1, numeric y1, numeric x2, numeric y2)

Querying for Inferred Knowledge

- SPARQL does not have specific constructs for accessing inferred knowledge
  - Underlying knowledge base is responsible for supporting inference, e.g.,
    - Class hierarchy
    - Property hierarchy
    - Transitive or symmetric properties
    - OWL restrictions
    - Defining classes by unions and/or intersections
- Different knowledge bases can offer different level of support
  - Same knowledge in different knowledge bases may return different results for the same query, depending on supported entailment

Query Example



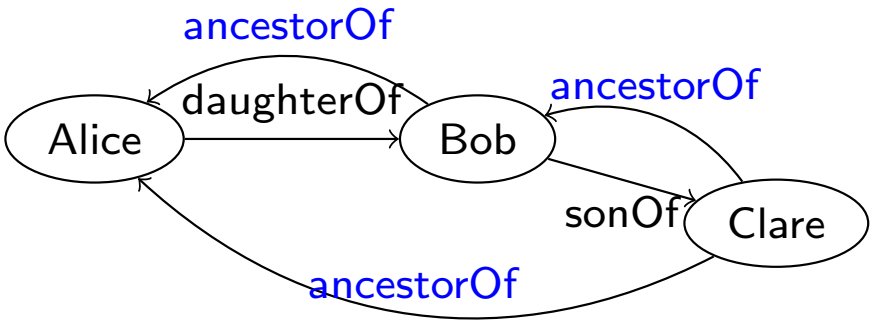
ancestorOf = owl:transitiveProperty $\wedge$ union(inverse(daughterOf), inverse(sonOf))

Find ancestors of Alice:

Query:

```
SELECT ?x
WHERE { ?x ancestorOf Alice }
```

Query Example



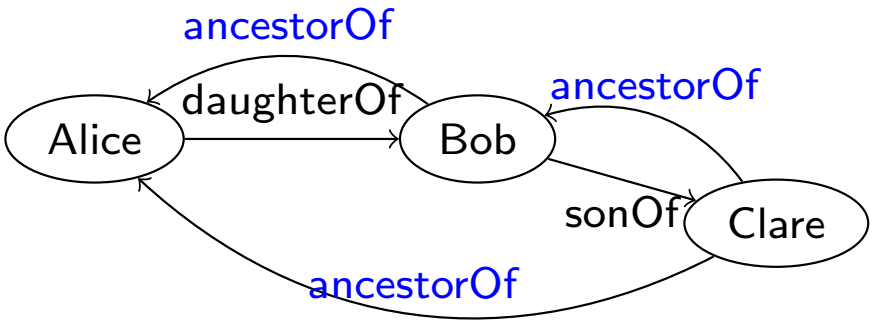
ancestorOf = owl:transitiveProperty $\wedge$ union(inverse(daughterOf), inverse(sonOf))

Find ancestors of Alice:

Query:

```
SELECT ?x
WHERE { ?x ancestorOf Alice }
```

Query Example



ancestorOf = owl:transitiveProperty $\wedge$ union(inverse(daughterOf), inverse(sonOf))

Find ancestors of Alice:

Query:

```
SELECT  ?x
WHERE  { ?x  ancestorOf  Alice }
```

CONSTRUCT Queries

- Special type of query to construct a new RDF graph from the existing knowledge base

```
PREFIX

CONSTRUCT
  {
     ... graph pattern ...
     ... definition of triples ...
  }
WHERE
   {
     constraint triple patterns,
     filters, etc.
   }
```

Constructing Graphs

Data:

```
@prefix foaf:
   <http://xmlns.com/foaf/0.1/> .
_:a foaf:givenname "Alice" .
_:a foaf:family_name "Hacker" .
_:b foaf:firstname "Bob" .
_:b foaf:surname "Hacker" .
```

Result:

```
@prefix vcard: <http://
  www.w3.org/2001/vcard-rdf/3.0#>
_:v1 vcard:N _:x .
 _:x vcard:givenName
"Alice" .
 _:x vcard:familyName
"Hacker" .
 _:v2 vcard:N _:z .
 _:z vcard:givenName "Bob" .
_:z vcard:familyName
"Hacker" .
```

Query:

```
PREFIX foaf:
  <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/
    2001/vcard-rdf/3.0#>
CONSTRUCT
{
?x vcard:N _:v .
_:v vcard:givenName ?gname .
 _:v vcard:familyName ?fname
}
WHERE
{
 { ?x foaf:firstname ?gname }
   UNION
 { ?x foaf:givenname ?gname } .
 { ?x foaf:surname ?fname }
   UNION
 { ?x foaf:family_name ?fname }
}
```

Constructing Graphs

Data:

```
@prefix foaf:
   <http://xmlns.com/foaf/0.1/> .
_:a foaf:givenname "Alice" .
_:a foaf:family_name "Hacker" .
_:b foaf:firstname "Bob" .
_:b foaf:surname "Hacker" .
```

Result:

```
@prefix vcard: <http://
  www.w3.org/2001/vcard-rdf/3.0#>
_:v1 vcard:N _:x .
 _:x vcard:givenName
"Alice" .
 _:x vcard:familyName
"Hacker" .
 _:v2 vcard:N _:z .
 _:z vcard:givenName "Bob" .
_:z vcard:familyName
"Hacker" .
```

Query:

```
PREFIX foaf:
  <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/
    2001/vcard-rdf/3.0#>
CONSTRUCT
{
?x vcard:N _:v .
_:v vcard:givenName ?gname .
 _:v vcard:familyName ?fname
}
WHERE
{
 { ?x foaf:firstname ?gname }
   UNION
 { ?x foaf:givenname ?gname } .
 { ?x foaf:surname ?fname }
   UNION
 { ?x foaf:family_name ?fname }
}
```

Constructing Graphs

Data:

```
@prefix foaf:
   <http://xmlns.com/foaf/0.1/> .
_:a foaf:givenname "Alice" .
_:a foaf:family_name "Hacker" .
_:b foaf:firstname "Bob" .
_:b foaf:surname "Hacker" .
```

Result:

```
@prefix vcard: <http://
  www.w3.org/2001/vcard-rdf/3.0#>
_:v1 vcard:N _:x .
 _:x vcard:givenName
"Alice" .
 _:x vcard:familyName
"Hacker" .
 _:v2 vcard:N _:z .
 _:z vcard:givenName "Bob" .
_:z vcard:familyName
"Hacker" .
```

Query:

```
PREFIX foaf:
  <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/
    2001/vcard-rdf/3.0#>
CONSTRUCT
{
?x vcard:N _:v .
_:v vcard:givenName ?gname .
 _:v vcard:familyName ?fname
}
WHERE
{
 { ?x foaf:firstname ?gname }
   UNION
 { ?x foaf:givenname ?gname } .
 { ?x foaf:surname ?fname }
   UNION
 { ?x foaf:family_name ?fname }
}
```

ASK Queries

- True / false queries — check if given set of triple patterns has at least one match in the knowledge base
- Does not include ORDER BY, LIMIT or OFFSET

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:homepage <http://work.example.org/alice/> .
_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@work.example>
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK { ?x foaf:name "Alice" .
      ?x foaf:mbox ?y }
```

Answer: NO

ASK Queries

- True / false queries — check if given set of triple patterns has at least one match in the knowledge base
- Does not include ORDER BY, LIMIT or OFFSET

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:homepage <http://work.example.org/alice/> .
_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@work.example>
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK { ?x foaf:name "Alice" .
      ?x foaf:mbox ?y }
```

Answer: NO

ASK Queries

- True / false queries — check if given set of triple patterns has at least one match in the knowledge base
- Does not include ORDER BY, LIMIT or OFFSET

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:homepage <http://work.example.org/alice/> .
_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@work.example>
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK { ?x foaf:name "Alice" .
      ?x foaf:mbox ?y }
```

Answer: NO

ASK Queries

- True / false queries — check if given set of triple patterns has at least one match in the knowledge base
- Does not include ORDER BY, LIMIT or OFFSET

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:homepage <http://work.example.org/alice/> .
_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@work.example>
```

Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK { ?x foaf:name "Alice" .
      ?x foaf:mbox ?y }
```

Answer: NO

DESCRIBE Queries

- Return a graph that includes description of specific resources
- Results of DESCRIBE queries reveal meta-information not returned by standard SELECT queries
  - Type of bounded resources
  - Types of relationships used in query pattern
- Exact description of resources is determined by the query service
  - No common standard of description
  - Can even include information about related resources

DESCRIBE Query Example

```
PREFIX ent: <http://org.example.com/employees#>
DESCRIBE ?x
WHERE { ?x ent:employeeId "1234" }
```

Result:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0> .
@prefix exOrg: <http://org.example.com/employees#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#>

_:a    exOrg:employeeId "1234" ;
       foaf:mbox_sha1sum "ABCD1234" ;
       vcard:N
          [ vcard:Family "Smith" ;
            vcard:Given "John" ] .

foaf:mbox_sha1sum rdf:type owl:InverseFunctionalProperty
```

Named Graphs

- RDF data stores may hold multiple RDF graphs:
  - record information about each graph
  - queries that involve information from more than one graph
  - default graph (does not have a name)
  - multiple named graphs (identified by URI reference)
  - direct implementation for reification
- Accessing named graphs
  - FROM
    - access knowledge in default graph
  - FROM NAMED
    - access information from specific named graph

Remember: Ad hoc reification (direct)

$\rightarrow$ remember: there are several alternatives for reification in RDF (modeling choices)

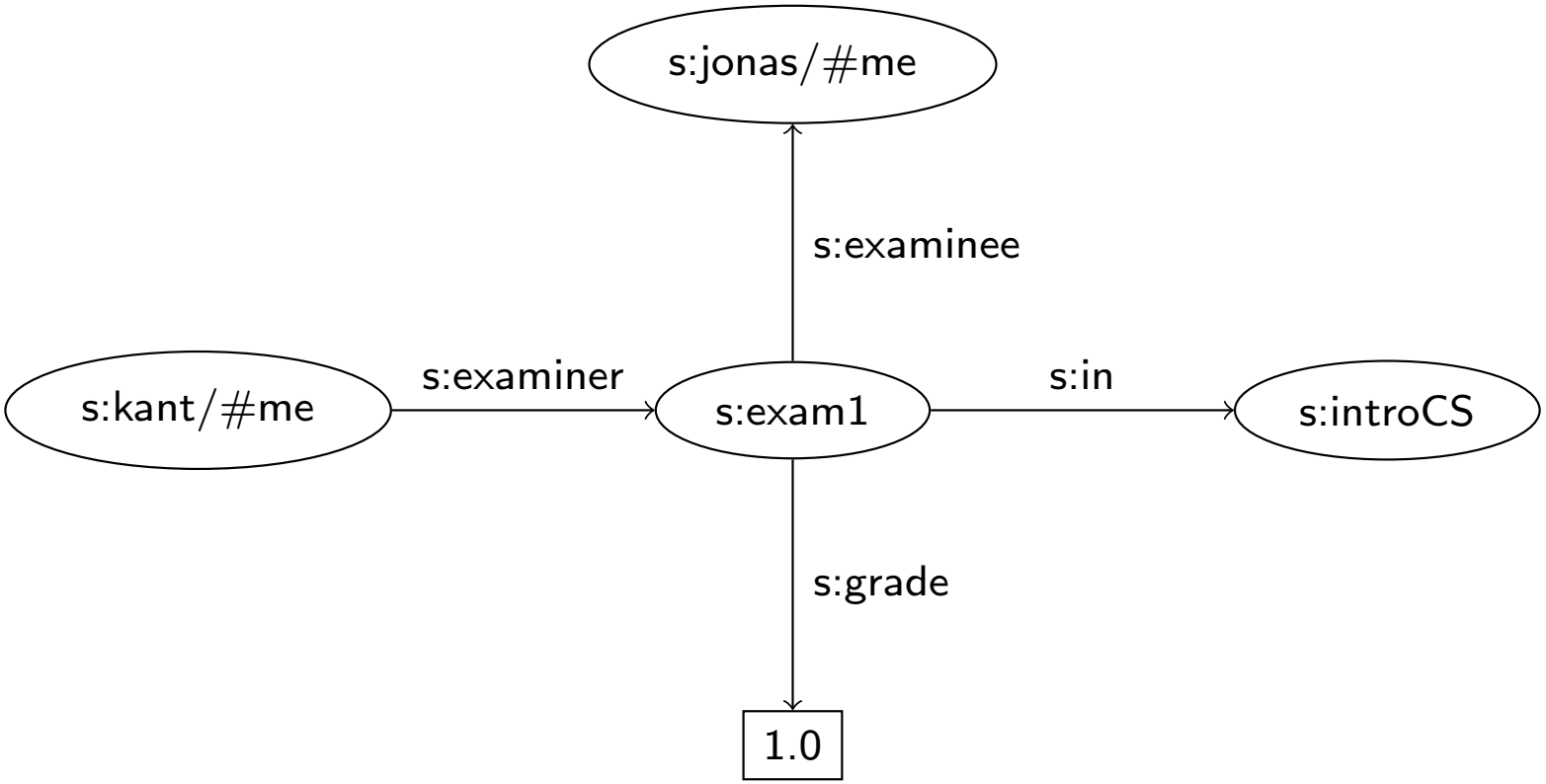
Repetition: Reification (statement on a statement):

Example:
"Kant" examined "Jonas" in "Introduction to CS" and gave him grade "1.0"

How can we model this in RDF?

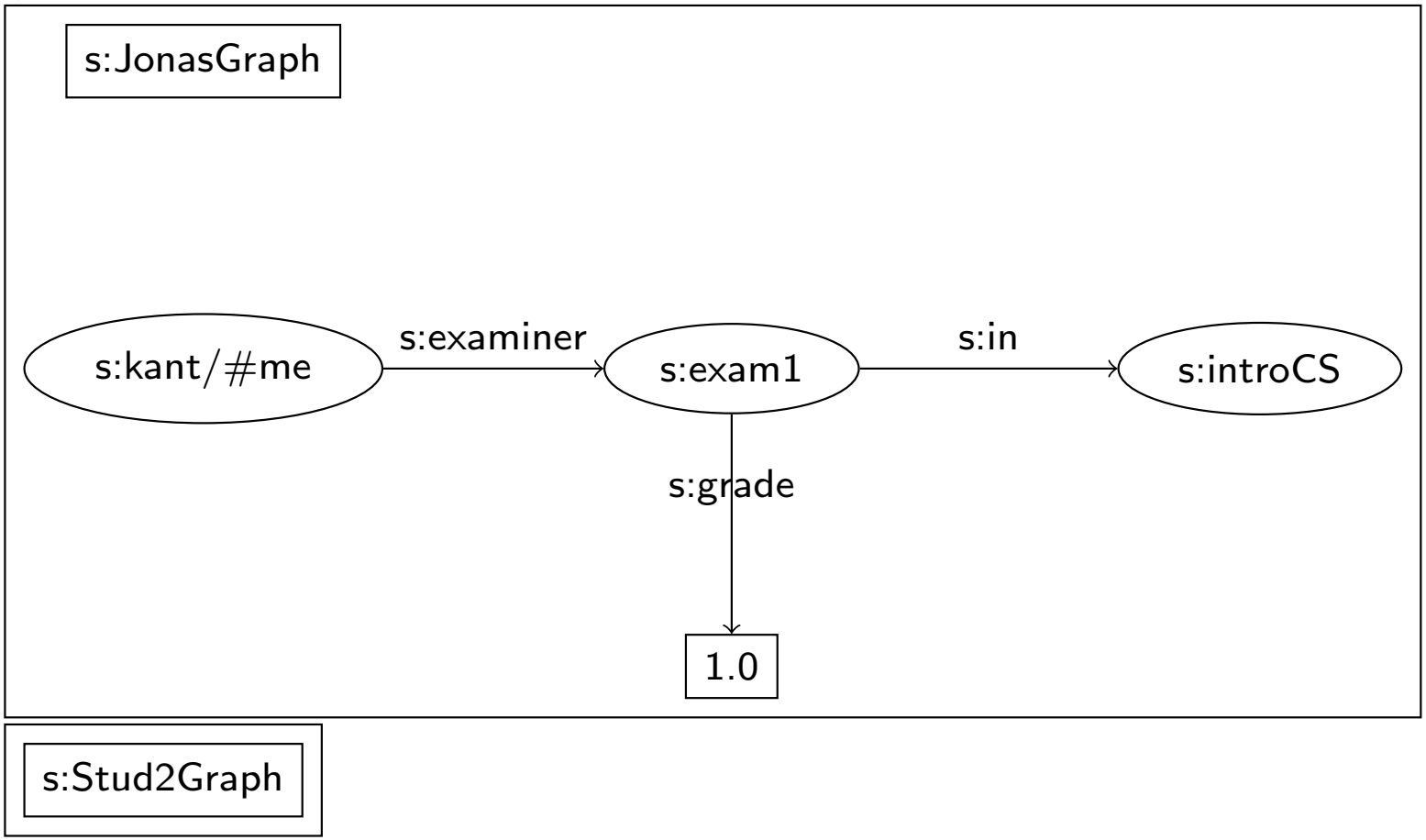
Ad hoc reification (direct)

"Kant" examined "Jonas" in "Introduction to CS" and gave him grade "1.0"



Named Graph for reification

"Kant" examined "Jonas" in "Introduction to CS" and gave him grade "1.0"



Named Graphs — Example

Data:

```
# Default graph (http://example.org/friends)
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<http://example.org/bob> dc:publisher ''Bob'' .
<http://example.org/alice> dc:publisher ''Alice'' .

# Bob's Graph: http://example.org/bob
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ''Bob'' .
_:a foaf:mbox <mailto:bob@oldcorp.example.org> .

# Alice's Graph: http://example.org/alice
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@work.example.org> .
```

Query:

```
SELECT ...
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
...
```

Named Graphs — Example

Data:

```
# Default graph (http://example.org/friends)
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<http://example.org/bob> dc:publisher ''Bob'' .
<http://example.org/alice> dc:publisher ''Alice'' .

# Bob's Graph: http://example.org/bob
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ''Bob'' .
_:a foaf:mbox <mailto:bob@oldcorp.example.org> .

# Alice's Graph: http://example.org/alice
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@work.example.org> .
```

Query:

```
SELECT ...
FROM NAMED <http://example.org/alice>
FROM NAMED <http://example.org/bob>
...
```

Relationships between Named Graphs

```
# Default graph
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:y foaf:name ``Alice''x .
_:y foaf:mbox <mailto:alice@work.example.org> .
_:y foaf:mbox <mailto:alice@oldcorp.org> .

# Graph: http://example.org/alice
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ``Alice'' .
_:a foaf:mbox <mailto:alice@work.example.org> .

# Graph: http://example.org/alice_prev
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name ``Alice'' .
_:a foaf:mbox <mailto:alice@oldcorp.org> .
```

Accessing the Name of Named Graphs

Data:

```
# Graph: http://example.org/alice
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@work.example.org> .
# Graph: http://example.org/alice_prev
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@oldcorp.org> .
```

Query:

```
SELECT ?src ?mbox
WHERE {
  GRAPH ?src
  { ?x foaf:name ''Alice'' .
    ?x foaf:mbox ?mbox
  } }
```

Result:

src	mbox
http://example.org/alice	mailto:alice@work.example.org
http://example.org/alice_prev	mailto:alice@oldcorp.org

Accessing the Name of Named Graphs

Data:

```
# Graph: http://example.org/alice
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@work.example.org> .
# Graph: http://example.org/alice_prev
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@oldcorp.org> .
```

Query:

```
SELECT ?src ?mbox
WHERE {
  GRAPH ?src
  { ?x foaf:name ''Alice'' .
    ?x foaf:mbox ?mbox
  } }
```

Result:

src	mbox
http://example.org/alice	mailto:alice@work.example.org
http://example.org/alice_prev	mailto:alice@oldcorp.org

Accessing the Name of Named Graphs

Data:

```
# Graph: http://example.org/alice
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@work.example.org> .
# Graph: http://example.org/alice_prev
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@oldcorp.org> .
```

Query:

```
SELECT ?src ?mbox
WHERE {
  GRAPH ?src
  { ?x foaf:name ''Alice'' .
    ?x foaf:mbox ?mbox
  } }
```

Result:

src	mbox
http://example.org/alice	mailto:alice@work.example.org
http://example.org/alice_prev	mailto:alice@oldcorp.org

Restricting Access by Graph Name

Data:

```
# Graph: http://example.org/alice
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@work.example.org> .
# Graph: http://example.org/alice_prev
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@oldcorp.org> .
```

Query:

```
PREFIX ex: <http://example.org/>
SELECT ?mbox
WHERE {
  GRAPH ex:alice
  { ?x foaf:mbox ?mbox }
}
```

Result: mbox

```
mailto:alice@work.example.org
```

Restricting Access by Graph Name

Data:

```
# Graph: http://example.org/alice
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@work.example.org> .
# Graph: http://example.org/alice_prev
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@oldcorp.org> .
```

Query:

```
PREFIX ex: <http://example.org/>
SELECT ?mbox
WHERE {
  GRAPH ex:alice
  { ?x foaf:mbox ?mbox }
}
```

Result: mbox

mailto:alice@work.example.org

Restricting Access by Graph Name

Data:

```
# Graph: http://example.org/alice
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@work.example.org> .
# Graph: http://example.org/alice_prev
_:a foaf:name ''Alice'' .
_:a foaf:mbox <mailto:alice@oldcorp.org> .
```

Query:

```
PREFIX ex: <http://example.org/>
SELECT ?mbox
WHERE {
  GRAPH ex:alice
  { ?x foaf:mbox ?mbox }
}
```

Result: mbox

```
mailto:alice@work.example.org
```

Outline