

SPARQL, Named Graphs, Network Graphs

Steffen Staab

Maciej Janik

Semantic Web

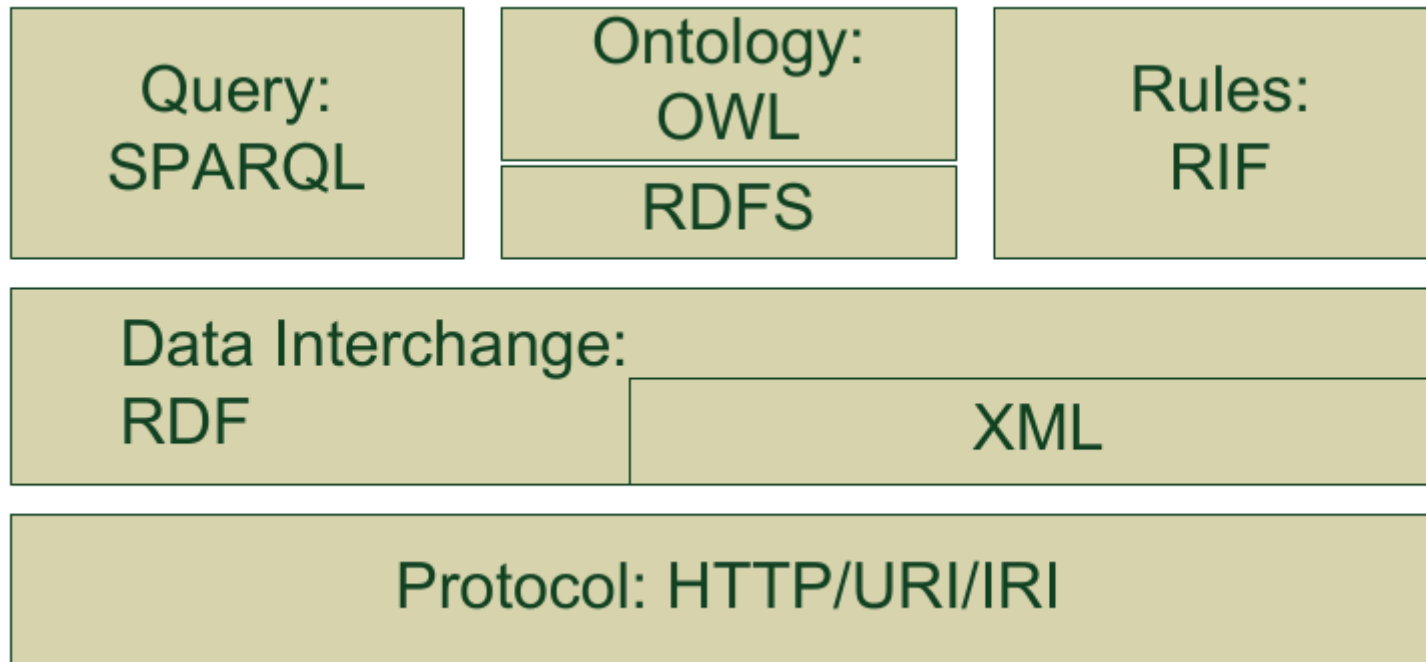
- **SPARQL Protocol and RDF Query Language**
- W3C Recommendation 15 January 2008
 - ◆ <http://www.w3.org/TR/rdf-sparql-query/>
- Standard query language for RDF
 - ◆ Native RDF knowledge bases
 - ◆ Knowledge bases viewed as RDF via middleware
- Language for querying for graph patterns
 - ◆ Includes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

Revision and extension

http://www.w3.org/2009/sparql/wiki/Main_Page

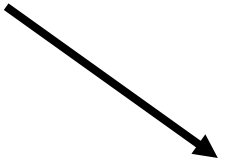
Working Draft documents, 2010-01-26:

- 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.



Existing standards

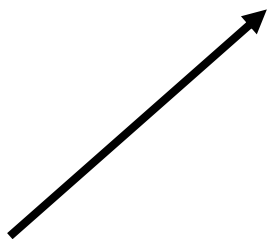
Schemas used in query



PREFIX ...

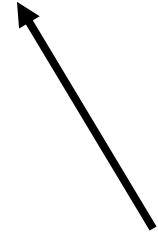
SELECT ... ← Values to be returned

FROM ...



WHERE { ... }

Identify source data to query



Triple patterns and other conditions to match the graph

- **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

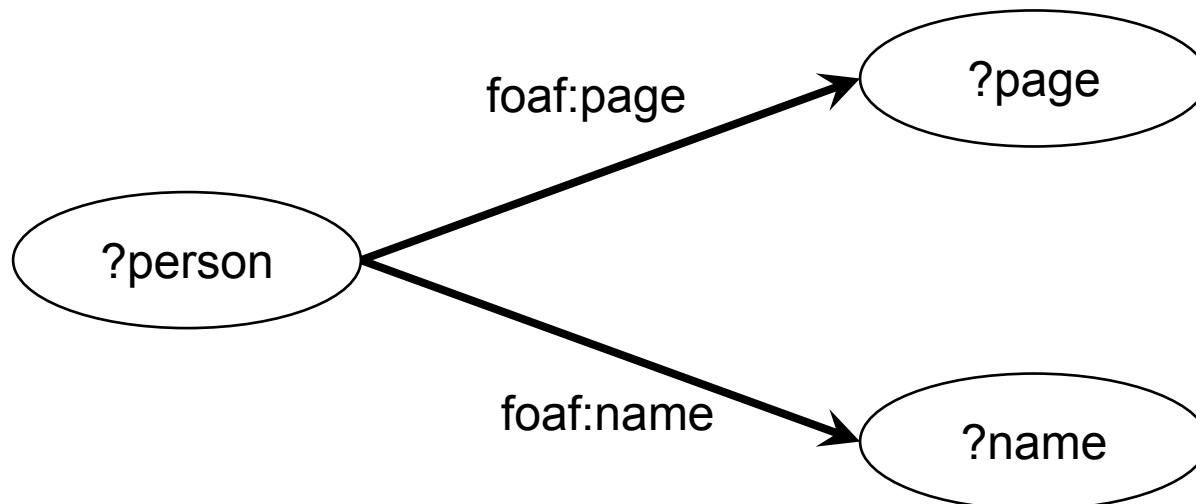
- **DESCRIBE**
 - ◆ returns an RDF graph that describes the resources found

- **ASK**
 - ◆ returns whether a query pattern matches any triples or not
True / False query

- 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

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



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 "Maciej Janik" .
_:b foaf:homepage <http://www.uni-koblenz.de/~janik> .
```

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

Query

Query Result

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

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 "Maciej Janik" .  
_:b foaf:homepage <http://www.uni-koblenz.de/~janik> .
```

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

Query

Query Result

person	name	homepage
_:c	"Steffen Staab"	<http://www.uni-koblenz.de/~staab>
_:d	"Maciej Janik"	<http://www.uni-koblenz.de/~janik>

- **FILTER**
 - ◆ Further constrain graph patterns
 - ◆ Applies to the **whole group** of triple 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
 - Less than ...equal ... greater than
 - ◆ Use of regular expressions
 - ◆ Support for datatypes defined in XSL
 - e.g. comparison of dates, time
 - ◆ Possible comparison of resources
 - Equal or not equal
 - ◆ Even possible user extensions

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

- Filters are applied to the whole group of patterns where it appears

```
{ ?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 patterns are equivalent – have the same solution.

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 "Maciej Janik" .
_:b foaf:homepage <http://www.uni-koblenz.de/~janik> .
```

```
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

Query Result

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

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 "Maciej Janik" .
_:b foaf:homepage <http://www.uni-koblenz.de/~janik> .
```

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", "janik")
}
```

Case insensitive

Query Result

name	page
"Maciej Janik"	<http://www.uni-koblenz.de/~janik>

- **OPTIONAL**

- ◆ Include optional triple patterns to the match
- ◆ Optional is a pattern itself – can include further constraints

SELECT

WHERE {

...

OPTIONAL { ... }

}

- OPTIONAL is left-associative

pattern OPTIONAL { pattern } OPTIONAL { pattern }

is the same as

{ pattern OPTIONAL { pattern } } OPTIONAL { pattern }

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 "Maciej Janik" .
_:b foaf:mbox <janik@uni-koblenz.de> .
```

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

Query

Query Result

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

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 "Maciej Janik" .
_:b foaf:mbox <janik@uni-koblenz.de> .
```

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

Query

Query Result

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

▪ 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 }  
}
```

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"

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 one query

Results 11 to 30 sorted by name

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

- 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 .
    ?person foaf:knows ?x .
    FILTER ( ! bound(?x) )
}
```

Find people who do not know Steffen

```
PREFIX foaf: < http://xmlns.com/foaf/0.1/>
```

```
SELECT ?name
```

```
WHERE {
```

```
  ?person foaf:name ?name .
```

```
  ?person foaf:knows ?x .
```

```
  FILTER ( ?x != "Steffen" )
```

```
}
```

... we know that ...

```
  "Maciej" foaf:knows "Steffen"
```

```
  "Maciej" foaf:knows "Sergej"
```

... so "Maciej" is still a valid answer, and we do not want it.

Find people who do not know Steffen

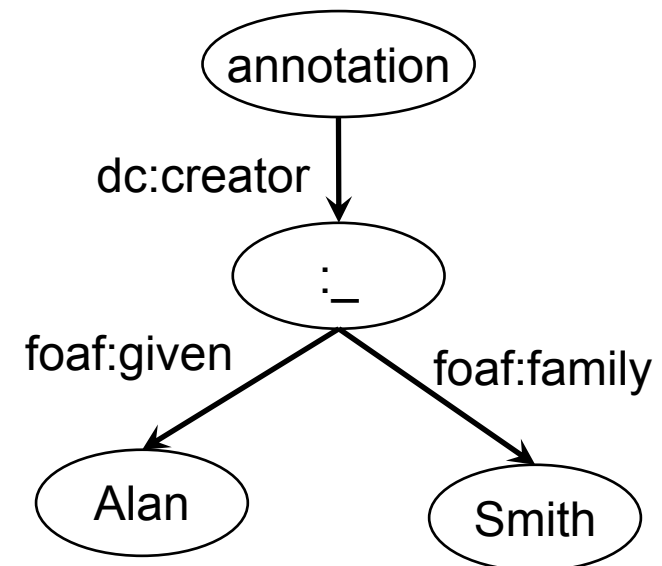
now the correct way using bound expression and optional graph pattern

```
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) ) }
}
```

▪ isBlank

- ◆ Testing if bounded variable is a blank node

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



▪ lang

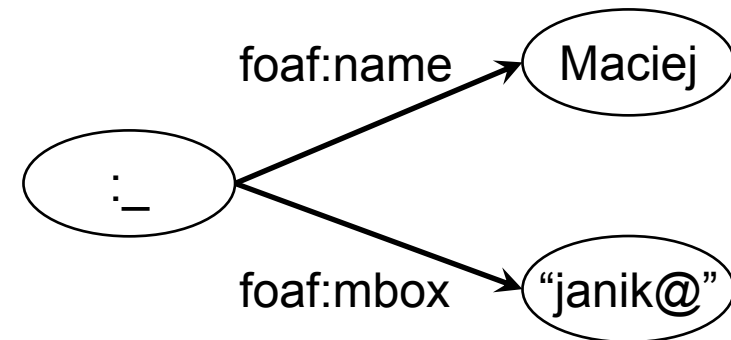
- ◆ Accessing the language of a literal

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

▪ isLiteral

- ◆ Testing if 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") }
```

- Check if two terms are equal or if they describe the same entity
 - ◆ Same entity can have even different URIs, but connected with owl:sameAs

`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

- 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) }
```

- FILTER enables using user-defined expressions

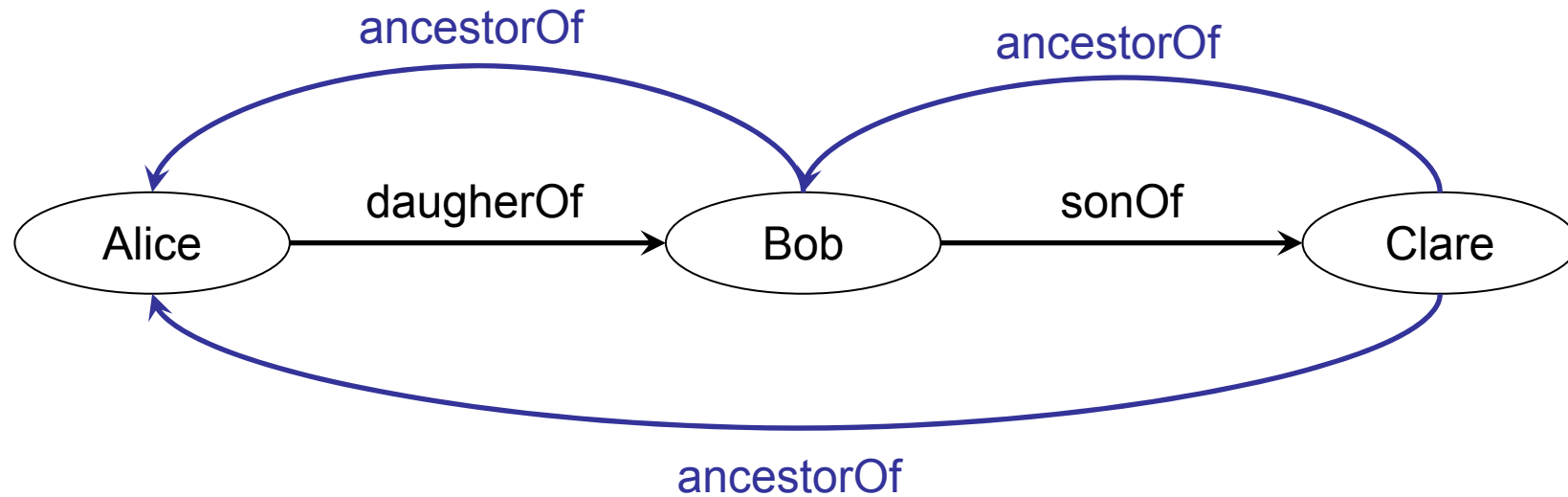
```
PREFIX aGeo: <http://example.org/geo#>
SELECT ?neighbor WHERE {
  ?a aGeo:placeName "Koblenz" .
  ?a aGeo:location ?axLoc .
  ?a aGeo:location ?ayLoc .
  ?b aGeo:placeName ?neighbor .
  ?b aGeo:location ?bxLoc .
  ?b aGeo:location ?byLoc .
  FILTER
    ( aGeo:distance(?axLoc, ?ayLoc, ?bxLoc, ?byLoc)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

- SPARQL do 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**



ancestorOf = owl:transitiveProperty + union (inverse(daughterOf), inverse(sonOf))

Find ancestors of Alice

Query

```
SELECT ?x  
WHERE ?x ancestorOf "Alice"
```

Result

"Clare"
"Bob"

- 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

}

▪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 }
}
```

- True / false queries – checks if given set of triple patterns have at least one match in 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>
```

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

Answer: NO

- Returns a graph that includes description of specific resources

- Results of DESCRIBE query reveal meta-information not returned by standard SELECT query
 - ◆ 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

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

```
@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#>

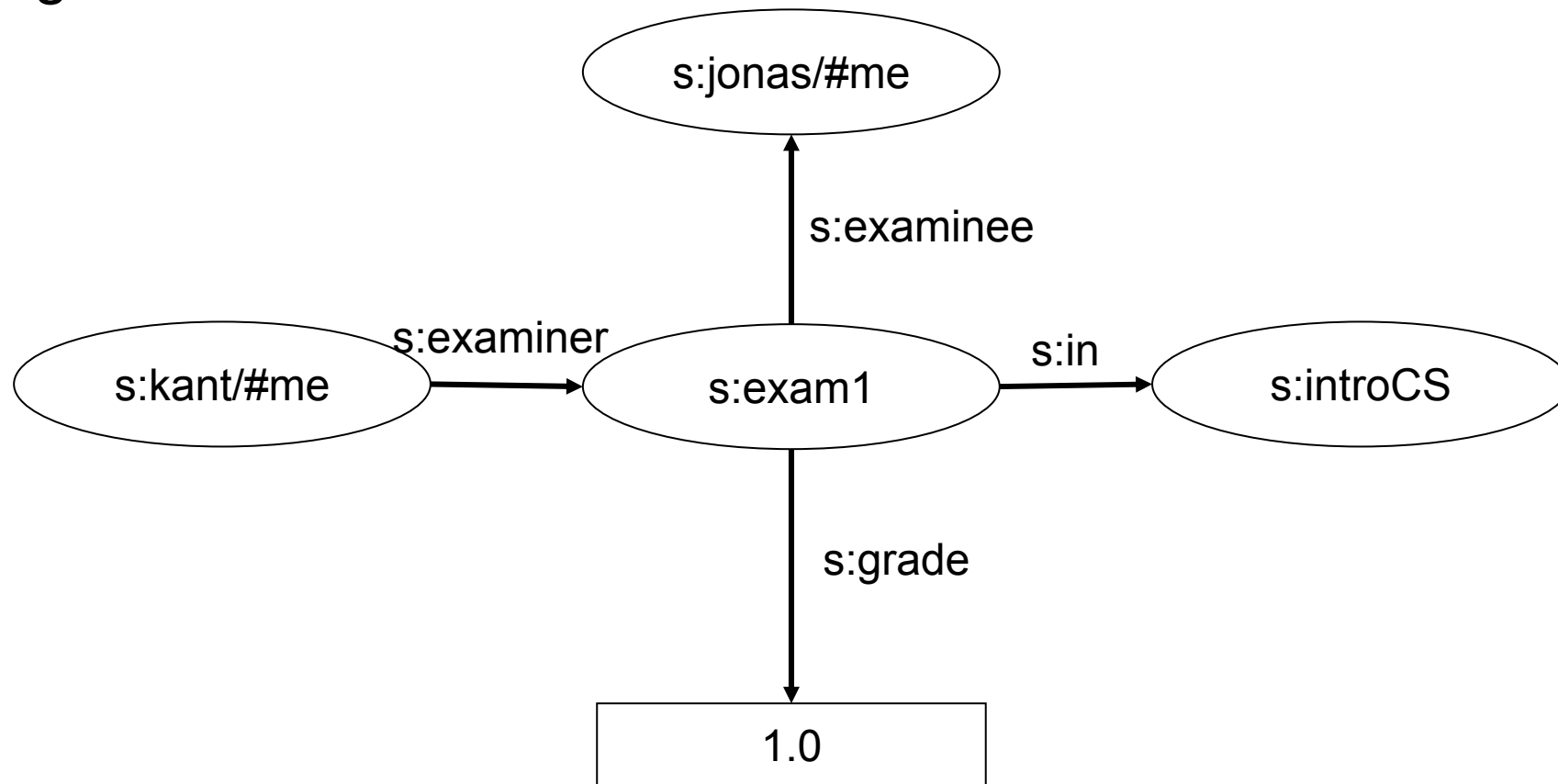
_:a      exOrg:employeeId "1234" ;
         foaf:mbox_sha1sum "ABCD1234" ;
         vcard:N
           [ vcard:Family "Smith" ;
             vcard:Given "John" ] .

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

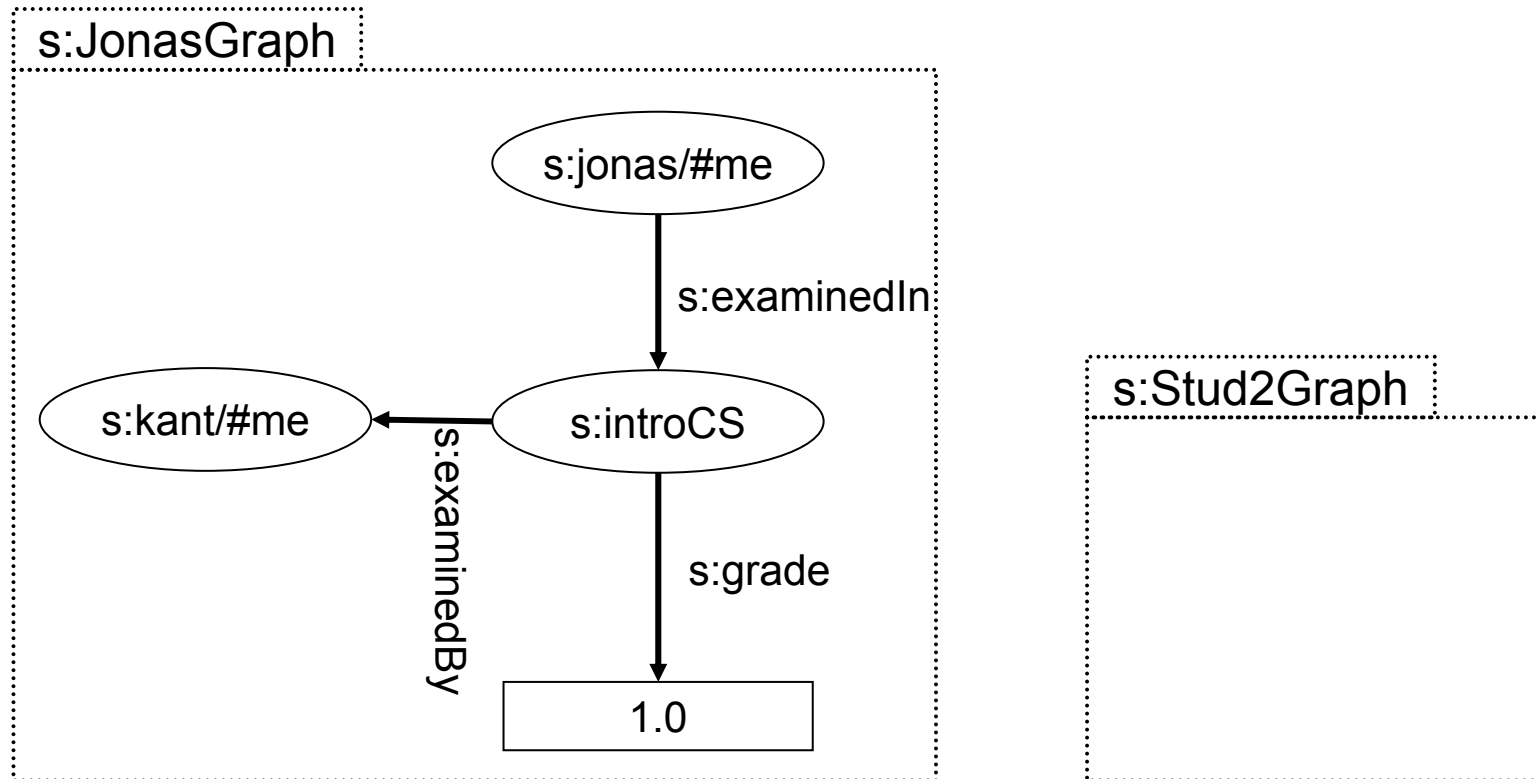
- RDF data stores may hold multiple RDF graphs:
 - ◆ record information about each graph
 - ◆ queries that involve information from more than one graph
 - ◆ default graph (does not have a name)
 - ◆ multiple named graphs (identified by URI reference)
 - ◆ direct implementation for reification

- Accessing named graphs
 - ◆ FROM
 - access knowledge in default graph
 - ◆ FROM NAMED
 - access information from specific named graph

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



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



```
# 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" .

# 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> .

# 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> .

SELECT ...

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

...
```

Default graph

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .  
_:y foaf:name "Alice" .  
_: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> .
```

```
# 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> .
```

```
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

```
# 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> .
```

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

Result:

mbox
mailto:alice@work.example.org

Networked Graphs

Simon Schenk and **Steffen Staab**.

“Networked Graphs: A Declarative Mechanism for SPARQL Rules, SPARQL Views and RDF Data Integration on the Web”, Proceedings of the 17th International World Wide Web Conference, **WWW2008**, Beijing, China. 2008.

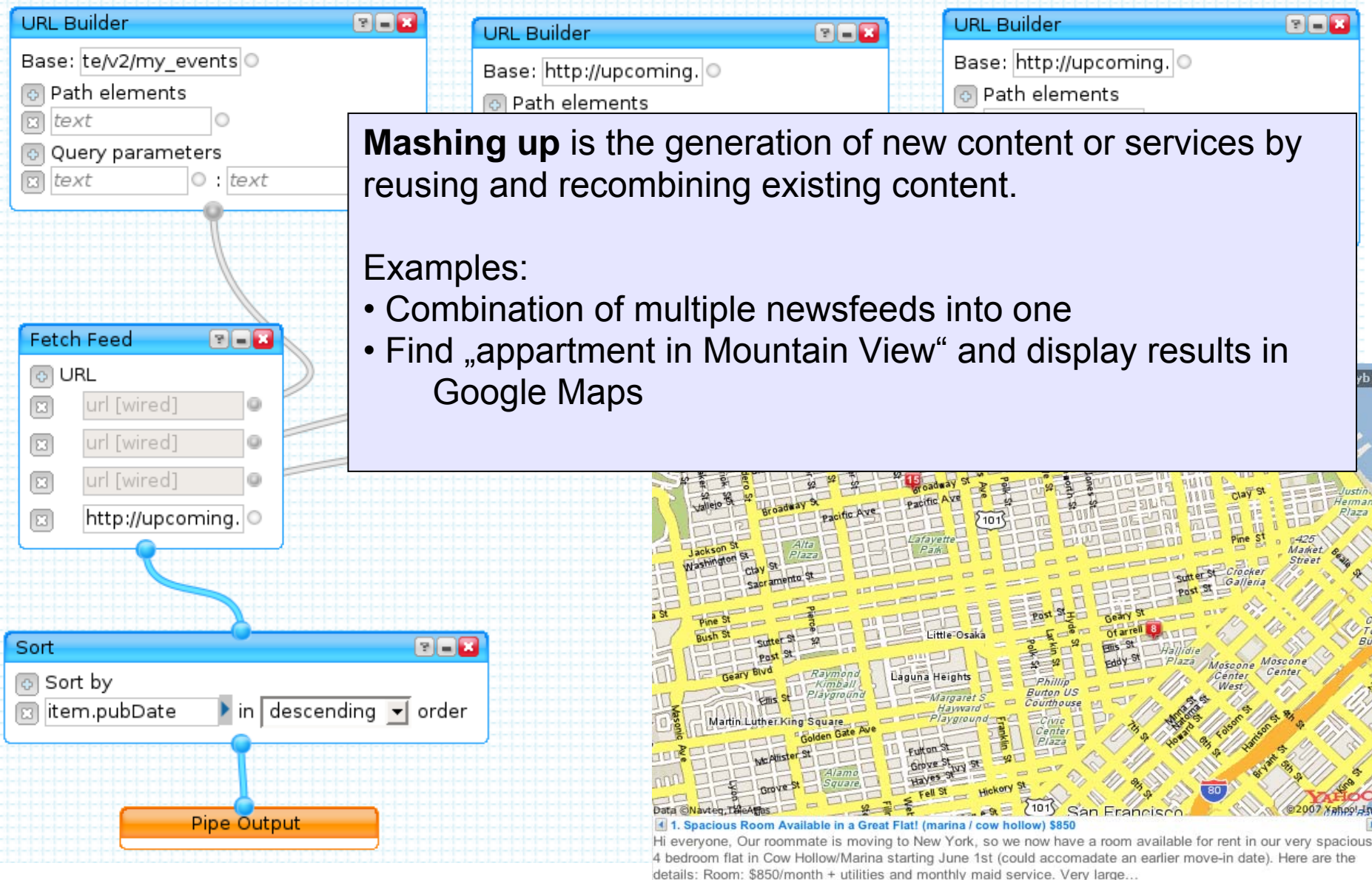
Basic Idea: Define RDF graphs

- ◆ *extensionally* by listing statements or
- ◆ *intensionally* using *views*
 - ◆ possible to have view within a view (recursion)

Integrate with existing SemWeb infrastructure

Easy exchange

Use existing data sources



Dominating Mashup-Modell:

- *Hack-and-Hope*

Disadvantages:

- **Screen-Scraping**
- No agreed-upon data model
- Sometimes one cannot help:
 - ♦ **Google Web Service**
 - ♦ **Amazon Web Service**

Dominating Semantic Web-Model:

- *Crawl-Integrate-and-Reason*

Disadvantages:

- Data are **stale**,
- Data integration is **not declarative**, but given by programmes with only implicit semantics
- Lack of **scalability** of one server
- **Access rights**: Not all data may be copied
- **Provenance** of data becomes unclear




FOAFer

Please enter a FOAF-Resource: [1]

[2]

Make your own
[FOAF-file](#) [3]!
[What](#) [4] [is](#) [5] [FOAF](#) [6]?

 -> <http://www.uni-koblenz.de/~sschenk/foaf.rdf>

Name:

Givenname:

Family Name:

MBOX_SHA1SUM:

[homepage](#) [7]

[workplace-homepage](#) [8]

[school-homepage](#) [9]

knows:

Steffen Staab

 -> [browse](#) [10]

Thomas franz

 -> [browse](#) [11]

Carsten Saathoff

[dblp.uni-trier.de](#)

DBLP

Simon Schenk

List of publications from the [DBLP Bibliography Server](#) - [FAQ](#)

[Coauthor Index](#) - Ask others: [ACMDL](#) - [ACM Guide](#) - [CiteSeer](#) - [CSB](#) - [Google](#)

2006	
3	EE Steffen Staab, Thomas Franz, Olaf Görnitz, Carsten Saathoff, Simon Schenk, Sergej Sizov: Lifecycle Knowledge Management: Getting the Semantics Across in X-Media. ISMIS 2006: 1-10
2005	
2	Simon Schenk: Introducing Social Aspects to Search in Peer-to-Peer Networks. Wissensmanagement 2005: 217-221
1	EE Simon Schenk: Introducing Social Aspects to Search in Peer-to-Peer Networks. Wissensmanagement (LNCS Volume) 2005: 234-242

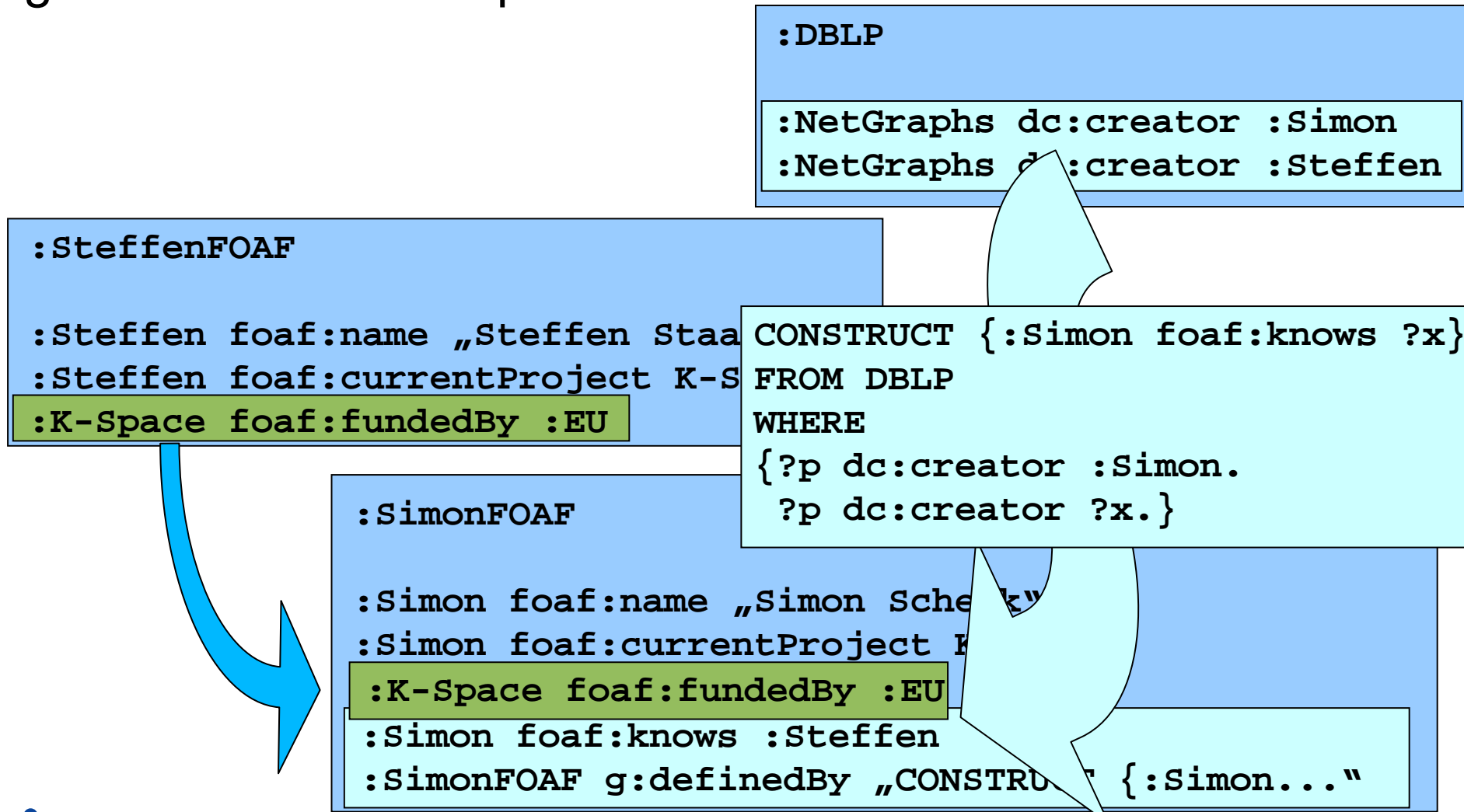
Coauthor Index

1	Thomas Franz [3]
2	Olaf Görnitz [3]
3	Carsten Saathoff [3]
4	Sergej Sizov [3]
5	Steffen Staab [3]

Fertig

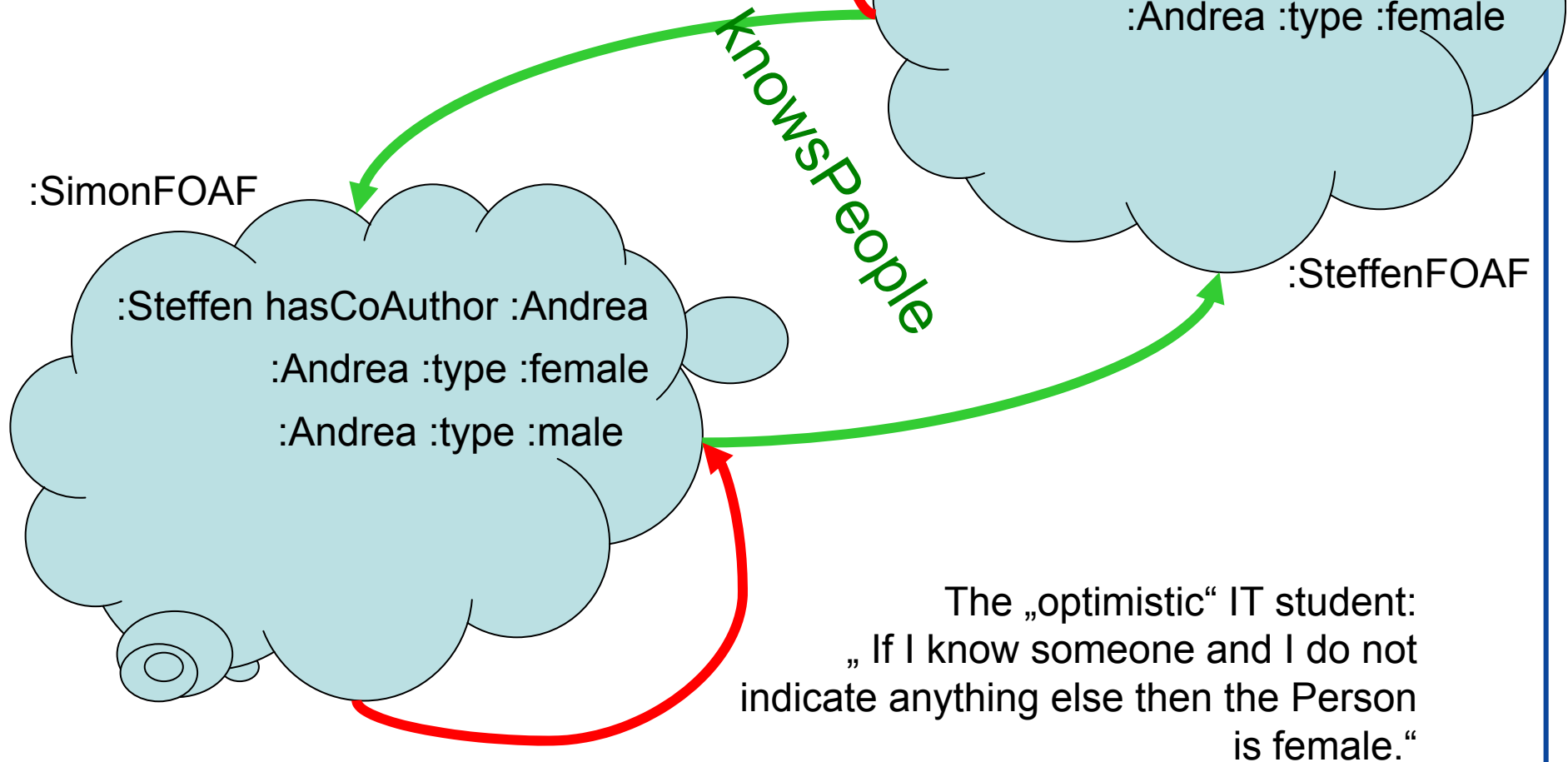


- Declarative, dynamic semantic mashups facilitate the generation of mashups



2 Networked Graphs

The „pessimistic“ IT student:
„If I know someone and I do not indicate
anything else then the Person is male.“



- In Semantic Web: Recursion and Negation unavoidable
- Solution:
Mapping of RDF Graphs and SPARQL Queries to logic programmes
 - ◆ Evaluation using three valued Well-founded Semantics or four valued stable model semantics
 - ◆ Non-monotonic logics with fixed point semantics
 - ◆ Conflicts with OWL Tarski-Semantik

Dominating Mashup-Modell:

- ***Hack-and-Hope***

Disadvantages:

- **Screen-Scraping**
- No agreed-upon data model
- Sometimes one cannot help:
 - ♦ **Google Web Service**
 - ♦ **Amazon Web Service**

Dominating Semantic Web-Model:

- ***Crawl-Integrate-and-Reason***

Disadvantages:

- Data are **stale**,
- Data integration is **not declarative**, but given by programmes with only implicit semantics
- Lack of **scalability** of one server
- **Access rights**: Not all data may be copied
- **Provenance** of data becomes unclear

Improvements:

- Reuse of data (telephone numbers!) instead of screen-scraping
- fresh views
- Definition of data integration may be exchanged as graph
- Evaluation possible on sources or on client sides
- Networked, dynamic Semantic Web

Formally: $G^N = (n, G, [G^N_1, \dots, G^N_n], v)$

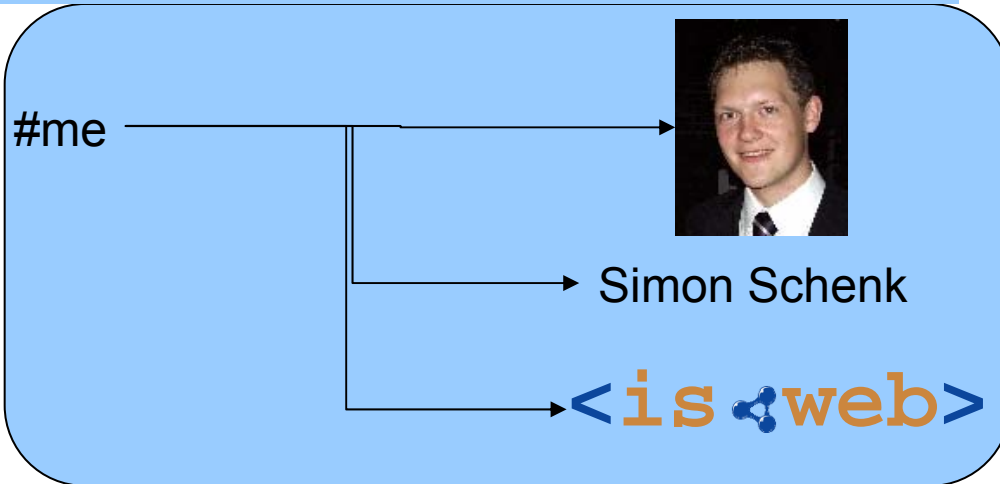
G : RDF Graph

G^N_i : Networked Graphs

v : view definition

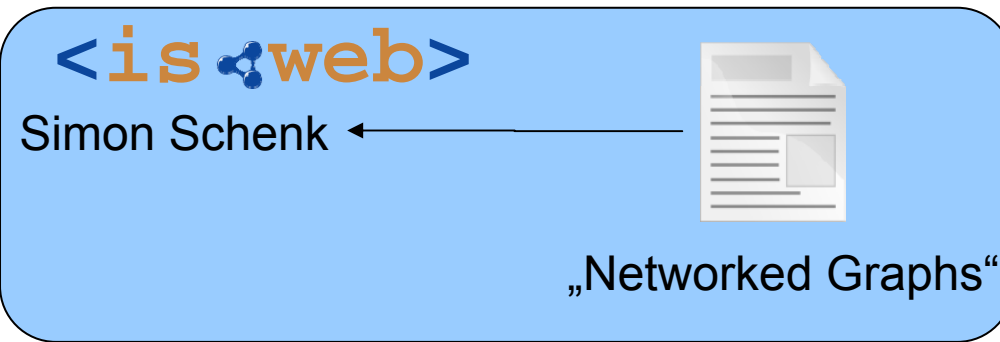
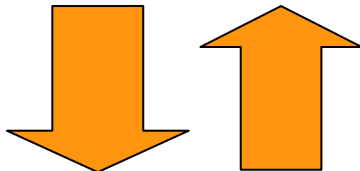
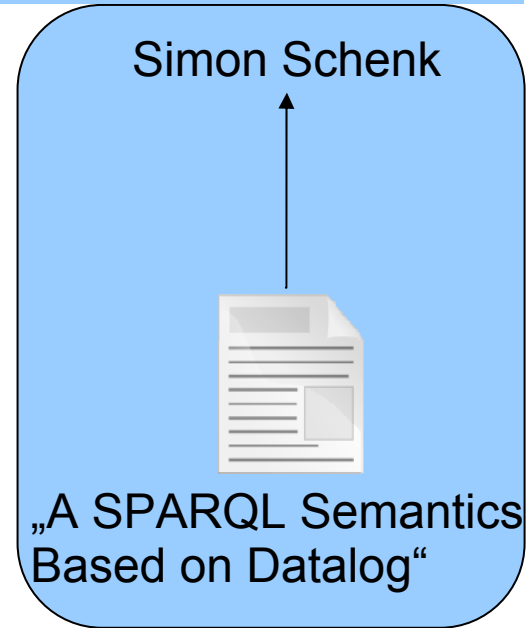
Motivation scenario

<http://www.uni-koblenz.de/~sschenk/foaf.rdf>



<http://www4.wiwiss.fu-berlin.de/dblp/>


`?x dc:creator [foaf:name „Simon Schenk“]`



`?x a isweb:ThesisProject.
NOT ?p past:Project ?x`



- Based on named graphs and SPARQL

```
foaf.rdf { n
```

```
#me foaf:name „Simon Schenk“.
```

```
#me foaf:depiction  .
```

```
foaf.rdf ng:definedBy
```

```
„CONSTRUCT {
```

```
?paper dc:creator #me; dc:title ?title; rdfs:seeAlso ?cr }
```

```
FROM DBLP
```

```
FROM ISWEB
```

```
WHERE {
```

```
?paper dc:creator dblp:person/352836; dc:title ?title
```

```
OPTIONAL {?paper dblp:crossref ?cr} .
```

```
?paper dc:date ?date}^^ng:Query.
```

```
foaf.rdf ng:definedBy „CONSTRUCT ...^^ng:Query.
```

```
...
```

```
}
```

G

V

$[G^N_1, G^N_2]$

V

Networked Graphs Semantics (sketch)

- Graph G_1^N depends on G_2^N , if G_1^N is defined by a view, which has G_2^N in its dataset.
- *Interdependence Set* of G^N : contains G^N and all graphs in the transitive closure of the depends on relation for G^N .
- Semantics of an interdependence set:
 - ♦ Iteratively evaluate all views in all graphs until a fixpoint is reached.

Problems:

- ♦ Need to prove termination
- ♦ Need to deal with negation

KI07: Map SPARQL to non-recursive Datalog with negation
Networked Graphs can be mapped to Datalog with negation
Evaluated under Well Founded Semantics (Gelder et al.)

SPARQL Protocol

- Protocol is used to send queries and results over the network
 - ◆ Query
 - HTTP binding
 - SOAP binding
 - ◆ XML result binding

HTTP binding

```
GET /sparql/?query=<encoded query> HTTP/1.1
Host: www.uni-koblenz.de
User-agent: neon-sparql-client/0.1
```

SOAP binding

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <query-request xmlns="http://www.w3.org/2005/09/sparql-
      protocol-types/#">
      <query>SELECT ?x ?y WHERE {?x isRelatedTo ?y}</query>
    </query-request>
  </soapenv:Body>
</soapenv:Envelope>
```

```
<?xml version="1.0"?>  
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
```

```
<head>  
  <variable name="name" />  
  <variable name="mbox" />  
  <link href="metadata.rdf" />  
</head>
```

```
<results>  
  <result>...  
</result>  
  <result>...  
</result>  
  ...  
</results>
```

```
</sparql>
```

```
<result>
  <binding name="x">
    <bnode>r2</bnode>
  </binding>
  <binding name="hpage">
    <uri>http://work.example.org/bob/</uri>
  </binding>
  <binding name="name">
    <literal xml:lang="en">Bob</literal>
  </binding>
  <binding name="age">
    <literal
      datatype="http://www.w3.org/2001/XMLSchema#integer">30
    </literal>
  </binding>
  <binding name="mbox">
    <uri>mailto:bob@work.example.org</uri>
  </binding>
</result>
```

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="name" />
    <variable name="mbox" />
  </head>

  <results>
    <result>
      <binding name="name" > ... </binding>
      <binding name="mbox" > ... </binding>
    </result>

    <result>
      <binding name="name" > ... </binding>
      <binding name="mbox" > ... </binding>
    </result>
    ...
  </results>
</sparql>
```

SPARQL extensions

(partially) discussed in the SPARQL-2 working group

<http://esw.w3.org/topic/SPARQL/Extensions/>

- Aggregate functions
 - ◆ COUNT, SUM, MIN, MAX, GROUP_BY, HAVING

- Paths expressions and property chains
 - ◆ Extend SPARQL beyond set of individual connected triples, allow variable length
 - ◆ PSPARQL SPARQLeR, SPARQ2L

- Imprecise matching
 - ◆ Use similarity measures to access triples
 - ◆ Similar idea as Soudex in relational database
 - ◆ iSPARQL

- SPARQL
 - ◆ Standard query language for RDF
 - **SELECT, CONSTRUCT, ASK, DESCRIBE**
 - Extensive filters, optional and alternative patterns
 - ◆ Protocol for queries and results
 - ◆ Based on triples model (subject-predicate-object)
 - No logic inference in language, only in underlying knowledge base
 - ◆ Named graphs
 - Separate or specialized knowledge
 - ◆ Networked graphs
 - Presenting (recursive) views of RDF data
 - Connecting external graphs over the network