

XML and Semantic Web Technologies

II. Semantic Web / 1. Ressource Description Framework (RDF)

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute of Economics and Information Systems
& Institute of Computer Science
University of Hildesheim
<http://www.ismll.uni-hildesheim.de>

II. Semantic Web / 1. Ressource Description Framework (RDF)

1. RDF Basics

2. RDF Built-in Resources

3. RDF/XML Syntax

4. RDF(S) Semantics

The RDF / RDF-S specification consists of the following parts:

- an abstract language for statements (RDF, RDF-S):
 - Resource Description Framework (RDF): Concepts and Abstract Syntax (REC-2004/02/10),
 - RDF Vocabulary Description Language 1.0: RDF Schema (REC-2004/02/10),
 - RDF Semantics (REC-2004/02/10),
- concrete representations thereof:
 - RDF/XML Syntax Specification (Revised; REC-2004/02/10),
 - RDF/N3 Syntax (not a W3C recommendation).
- introductory materials:
 - RDF Primer (REC-2004/02/10),

as well as further documents on test cases.

RDF Triples

RDF allows the formalization of statements in form of two-adic relations (called **predicates** or **properties**).

The first argument of the relation is called its **subject**, the second its **object**.

Subject, predicate, and object form so-called **triples**:

subject predicate object

Subjects, predicates, and objects can be stated in different manners:

- identified by an URI (optionally with a fragment identifier),
- identified by a name of local scope (anonymous),
- given explicitly as literal (optionally having a **type**, that in turn is identified by an URI).

	URI	literal	anonymous
predicate	+		
subject	+		+
object	+	+	+

RDF Triples

Thus, a valid RDF triple is, e.g.,

`http://www.cgnm.de/ http://www.cgnm.de/ http://www.cgnm.de/`

Figure 1: RDF triple, informal notation.

But what does it mean?

RDF Triples

The semantics of an URI in an RDF-triple must be defined by some means, either informally or formally.

For example, if we agree on the following semantics,

URI	meaning
http://www.cgnm.de/rdf/relatives#Anne	person Anne
http://www.cgnm.de/rdf/relatives#Clara	person Clara
http://www.cgnm.de/rdf/relatives#motherOf	property to be mother of

then

1 <http://www.cgnm.de/rdf/relatives#Anne>
2 <http://www.cgnm.de/rdf/relatives#motherOf>
3 <http://www.cgnm.de/rdf/relatives#Clara>

Figure 2: RDF triple, informal notation.

means, that *Anne is the mother of Clara*.

N-triples Representation

N-triples is a very simple, but verbose format to express RDF triples (see <http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/#ntriples>).

Each triple is written as a whitespace-separated sequence of subject, predicate, and object, terminated with a ".".

syntax	meaning
<code>< <i>absoluteURI</i> ></code>	URI reference
<code>" <i>string</i> "</code>	untyped literal
<code>" <i>string</i> "@@< <i>absoluteURI</i> ></code>	typed literal
<code>_: <i>NCName</i></code>	anonymous node name

```

1 <http://www.cgnm.de/rdf/relatives#Anne> <http://www.cgnm.de/rdf/relatives#mother
2 <http://www.cgnm.de/rdf/relatives#Clara> .

```

Figure 3: RDF triple in N-triples representation (should be on one line).

Notation 3 Representation (N3)

Notation 3 extends N-triples with many additional syntactical possibilities (see <http://www.w3.org/DesignIssues/Notation3>).

The most important extension is prefix declaration:

- With the declaration

`@prefix` $\langle QName \rangle$ $\langle URI \rangle$.

a prefix (the prefix of the $\langle QName \rangle$, its local name usually is empty) is bound to an URI.

- $\langle QName \rangle$ s (with prefixes bound by a `@prefix` declaration) can be used as subjects, predicates, and objects.

syntax	meaning
$\langle QName \rangle$	URI reference (prefix URI concatenated with local name)

Notation 3 Representation / example

¹ @prefix r: <http://www.cgnm.de/rdf/relatives#> .
² r:Anne r:motherOf r:Clara .

Figure 4: RDF triple in N3 representation.

¹ @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
² @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
³ @prefix xs: <http://www.w3.org/2001/XMLSchema#> .
⁴ @prefix : <#> .

Figure 5: Default prefixes in N3 examples used in the following.

Anonymous Subjects and Objects

Anonymous subjects or objects model an existential quantifier:

$$\exists x : r:\text{motherOf}(r:\text{Anne}, x) \wedge r:\text{name}(x, \text{"Dennis"})$$

¹ @prefix xs: <http://www.w3.org/2001/XMLSchema#> .

² @prefix r: <http://www.cgnm.de/rdf/relatives#> .

³ r:Anne r:motherOf r:Clara .

⁴ r:Anne r:name "Anne" .

⁵ r:Anne r:age "32"^^xs:integer .

⁶ r:Clara r:name "Clara" .

⁷ r:Anne r:motherOf _:xyz1 .

⁸ _:xyz1 r:name "Dennis" .

Figure 6: RDF/N3 example with anonymous resource.

N3 / Abbreviations

- `;` denotes repetition of the subject (**predicate list**).
- `,` denotes repetition of the subject and predicate (**object list**).
- `[...]` denotes **creation of an anonymous resource**, to which the enclosed predicate+object is attributed.

```
1 @prefix xs: <http://www.w3.org/2001/XMLSchema#> .  
2 @prefix r: <http://www.cgnm.de/rdf/relatives#> .  
3 r:Anne r:name "Anne" ;  
4     r:age "32"^^xs:integer ;  
5     r:motherOf r:Clara , [ r:name "Dennis" ] .  
6 r:Clara r:name "Clara" .
```

Figure 7: RDF/N3 example with abbreviations.

Graph Representation

RDF can be represented as directed graph with labeled nodes and edges.

- subjects and objects are nodes of the graph, their URI or literal node (typed) labels,
- anonymous subjects or objects are nodes w./o. label (**blank nodes**),
- predicates are edges, pointing from subject to object, their URI the edge label.

Graph Representation / example

```

1 @prefix xs: <http://www.w3.org/2001/XMLSchema#> .
2 @prefix r: <http://www.cgnm.de/rdf/relatives#> .
3 r:Anne r:motherOf r:Clara .
4 r:Anne r:name "Anne" .
5 r:Anne r:age "32"^^xs:integer .
6 r:Clara r:name "Clara" .
7 r:Anne r:motherOf _:xyz1 .
8 _:xyz1 r:name "Dennis" .
  
```

Figure 8: RDF triple in N3 representation.

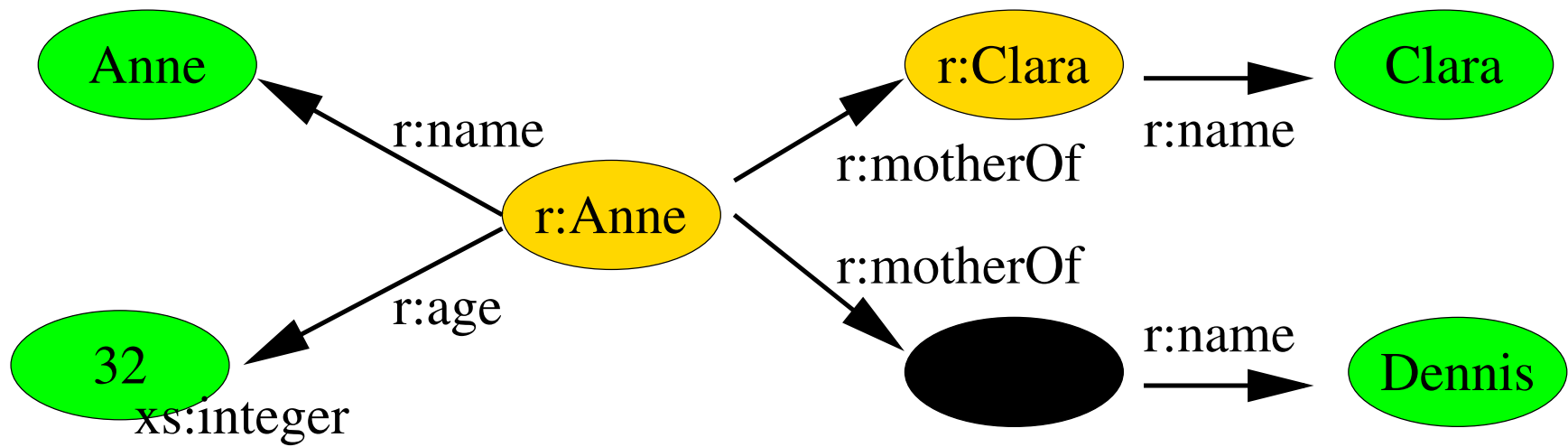


Figure 9: Graph representation of example above.

II. Semantic Web / 1. Resource Description Framework (RDF)

1. RDF Basics

2. RDF Built-in Resources

3. RDF/XML Syntax

4. RDF(S) Semantics

RDF Built-in Predicates and Resources

RDF has built-in predicates and resources for several tasks:

1. assigning types and classes
(`rdf:type`),
2. top-level classes
(`rdfs:Resource`, `rdf:Property`, etc.),
3. predicates for defining subclasses and subproperties
(`rdfs:subClassOf`, `rdfs:subPropertyOf`),
4. classes for RDF statements
(reification; `rdf:Statement`, `rdf:subject`, etc.),
5. container classes
(`rdfs:Container`, `rdf:Alt`, etc.).

RDF Classes and Types

- The predicate `rdf:type` (`rdfs:Resource`, `rdfs:Class`) states that the subject is of a given class or type ("instance of").
- The predicate `rdfs:subClassOf` (`rdfs:Class`, `rdfs:Class`) states that the subject is a subclass of a given class or type, i.e.,

$$\text{rdfs:subClassOf}(A, B) :\Leftrightarrow \forall x : \text{rdf:type}(x, A) \Rightarrow \text{rdf:type}(x, B)$$

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

3 @prefix : <#> .

4 :Mortal rdf:type rdfs:Class .

5 :Human rdf:type rdfs:Class .

6 :Human rdfs:subClassOf :Mortal .

7 :Sokrates rdf:type :Human .

Figure 10: Definition of two classes and an instance.

RDF Top-level Classes

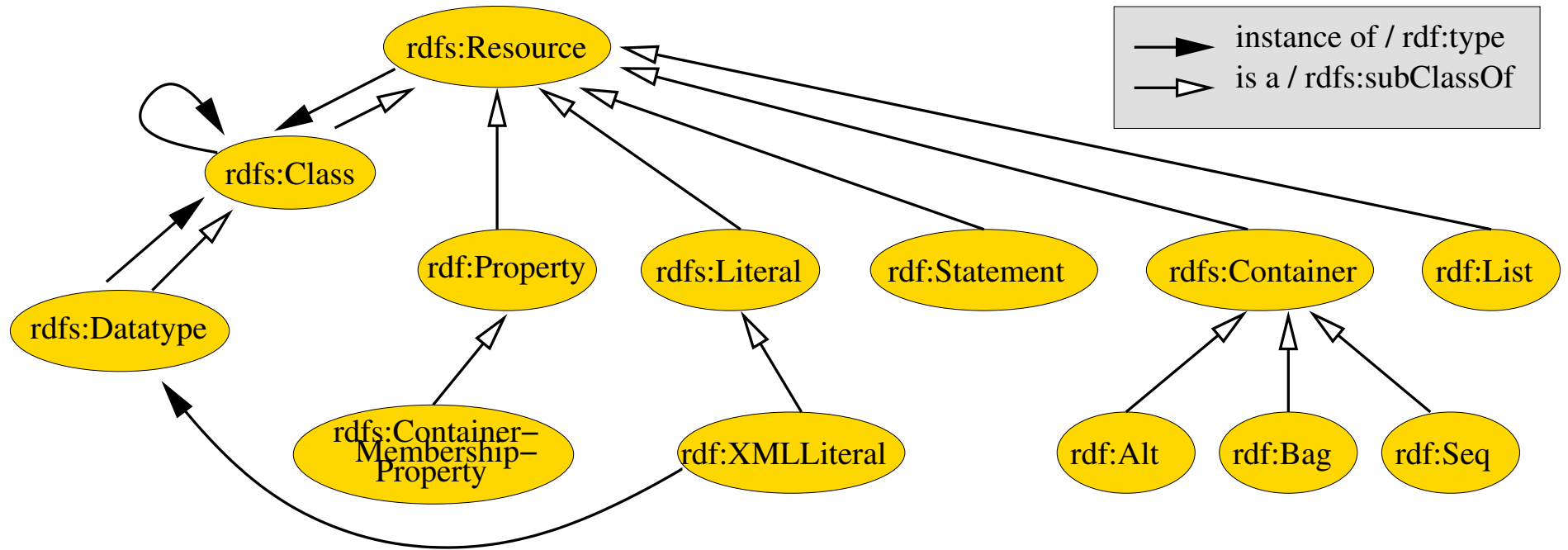


Figure 11: RDF Class Hierarchy.

RDF Predicates / Properties

`rdf:Property` is the class of predicates / properties.

- The predicate `rdfs:subPropertyOf` (`rdf:Property`, `rdf:Property`) states that the subject is a subproperty of a given property, i.e.,

$$\text{rdfs:subPropertyOf}(p, q) :\Leftrightarrow \forall x, y : p(x, y) \Rightarrow q(x, y)$$

- The property `rdfs:domain` (`rdf:Property`, `rdfs:Class`) states that all possible subjects of a predicate are of a given class, i.e.,

$$\text{rdfs:domain}(p, A) :\Leftrightarrow \forall x, y : p(x, y) \Rightarrow \text{rdf:type}(x, A)$$

- The property `rdfs:range` (`rdf:Property`, `rdfs:Class`) states that all possible objects of a predicate are of a given class, i.e.,

$$\text{rdfs:range}(p, A) :\Leftrightarrow \forall x, y : p(x, y) \Rightarrow \text{rdf:type}(y, A)$$

RDF Predicates / Properties

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix xs: <http://www.w3.org/2001/XMLSchema#> .
4 @prefix : <#> .
5 :Person rdf:type rdfs:Class .
6 :motherOf rdf:type rdf:Property ;
7     rdfs:domain :Person ;
8     rdfs:range :Person .
9 :name rdf:type rdf:Property ;
10    rdfs:domain :Person ;
11    rdfs:range xs:string .
12 :age rdf:type rdf:Property ;
13    rdfs:domain :Person ;
14    rdfs:range xs:integer .
```

Figure 12: Description of classes and predicates: an RDF Schema.

RDF Statements About Statements (Reification)

`rdf:Statement` is the class of statements.

- The property `rdf:subject` (`rdf:Statement`, `rdfs:Resource`) states that a resource is the subject of this statement.
- The property `rdf:predicate` (`rdf:Statement`, `rdfs:Resource`) states that a resource is the predicate of this statement.
- The property `rdf:object` (`rdf:Statement`, `rdfs:Resource`) states that a resource is the object of this statement.

RDF Statements About Statements (Reification)

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix : <#> .
4 :Person rdf:type rdfs:Class .
5 :Mortal rdf:type rdfs:Class .
6 :name rdf:type rdf:Property ; rdfs:domain :Person ; rdfs:range rdfs:Literal .
7 :believes rdf:type rdf:Property ; rdfs:domain :Person ; rdfs:range rdf:Statement .
8
9 :Anne rdf:type rdfs:Person ; :name "Anne" ;
10   :believes [ rdf:subject :Person ;
11             rdf:predicate rdfs:subClassOf ;
12             rdf:object :Mortal ],
13   [ rdf:subject [ :name "Sokrates" ] ;
14     rdf:predicate rdf:type ;
15     rdf:object :Person ] .
```

Figure 13: Statements about Statements: beliefs.

RDF Container

`rdfs:Container` is the class of all open containers (i.e., elements can be added at any time).

`rdf:Seq`, `rdf:Bag`, and `rdf:Alt` are subclasses, that indicate, that its elements are orderd, unordered, or mutually exclusive.

The predicates `rdf:_1`, `rdf:_2` etc. (`rdfs:Resource`, `rdfs:Resource`) state that a resource is a member of a given container at a given position.

The predicate `rdfs:member` (`rdfs:Resource`, `rdfs:Resource`) states that a resource is a member of a given container.

RDF Container

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix : <#> .
4 :Person rdf:type rdfs:Class .
5 :Course rdf:type rdfs:Class .
6 :name rdf:type rdf:Property ; rdfs:domain :Person ; rdfs:range rdfs:Literal .
7 :title rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdfs:Literal .
8 :students rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdf:Bag .
9
10 :XML :title "XML and Semantic Web Technologies" ;
11     :students [ rdfs:member [ :name "Anne" ] ,
12                 [ :name "Bert" ] ,
13                 [ :name "Clara" ] ] .
```

Figure 14: An RDF Container.

RDF Lists (aka Collections)

`rdfs:List` is the class of all closed containers (i.e., elements cannot be added later-on).

The predicate `rdfs:first` (`rdfs:List`, `rdfs:Resource`) states that a resource is the first member of a given list.

The predicate `rdfs:rest` (`rdfs:List`, `rdfs:List`) states that a resource is the sublist from second element onwards of a given list.

The resource `rdf:nil` denotes an empty list.

RDF Lists

```
1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix : <#> .
4 :Person rdf:type rdfs:Class .
5 :Course rdf:type rdfs:Class .
6 :name rdf:type rdf:Property ; rdfs:domain :Person ; rdfs:range rdfs:Literal .
7 :title rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdfs:Literal .
8 :students rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdfs:List .
9
10 :XML :title "XML and Semantic Web Technologies" ;
11     :students [ rdf:first [ :name "Anne" ] ;
12                rdf:rest [ rdf:first [ :name "Bert" ] ;
13                           rdf:rest [ rdf:first [ :name "Clara" ] ; rdf:rest rdf:nil ]
14                ]
15     ] .
```

Figure 15: An RDF List.

RDF Lists

Lists can be abbreviated by (...), where the members of the list are enumerated separated by whitespace.

1 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

3 @prefix : <#> .

4 :Person rdf:type rdfs:Class .

5 :Course rdf:type rdfs:Class .

6 :name rdf:type rdf:Property ; rdfs:domain :Person ; rdfs:range rdfs:Literal .

7 :title rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdfs:Literal .

8 :students rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdfs:Bag .

9
10 :XML :title "XML and Semantic Web Technologies" ;

11 :students ([:name "Anne"] [:name "Bert"] [:name "Clara"]) .

Figure 16: An RDF List.

RDF Representations

RDF can be expressed in various representations:

- as XML document:
 - RDF/XML – W3C default syntax.
 - several other XML representations.

- as text file with custom syntax:
 - N-triples – a very simple and verbose format.
 - Notation 3 (Berners-Lee) – an extension of N-triples.

- as graph.

- formally as triples.

II. Semantic Web / 1. Ressource Description Framework (RDF)

1. RDF Basics

2. RDF Built-in Resources

3. RDF/XML Syntax

4. RDF(S) Semantics

RDF and RDF-S use the following namespaces:

<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

<http://www.w3.org/2000/01/rdf-schema#>

RDF/XML uses different representations for

- subjects and objects (URI-attributes in `rdf:Description`-elements) and
- predicates (`<QName>`s).

N3 syntax	XML/RDF syntax	meaning
<code>< <absoluteURI> ></code>	<code><rdf:Description rdf:about="<URI>" /> <QName></code>	URI reference for predicates
<code>" <string> "</code>	<code><string></code>	untyped literal
<code>" <string> " @ <absoluteURI></code>	<code><... rdf:datatype="<URI>"><string></...></code>	typed literal
<code>_: <NCName></code>	<code><rdf:Description rdf:nodeID="<NCName>" /></code>	anonymous node

Triples

An RDF-Triple is represented as

- an `rdf:Description` element of the subject with
- a nested `<QName>`-element for the predicate with
- a nested `rdf:Description` element of the object.

```
1 <?xml version="1.1"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4     xmlns:r="http://www.cgnm.de/rdf/relatives#">
5 <rdf:Description rdf:about="http://www.cgnm.de/rdf/relatives#Anne">
6   <r:motherOf>
7     <rdf:Description rdf:about="http://www.cgnm.de/rdf/relatives#Clara"/>
8   </r:motherOf>
9 </rdf:Description>
10 </rdf:RDF>
```

Figure 17: An RDF Triple in RDF/XML representation.

Abbreviations / URIs

`<QName>`s must be "abbreviated" by declaring a namespace prefix.

URI-attributes can be abbreviated by specifying an `xml:base`-attribute (for the "main" vocabulary). Unfortunately, due to the rules for combining a base URI with a relative URI (i.e., replace last path step), base-URIs cannot contain the fragment indicator `#` (as in N3 notation).

```
1 <?xml version="1.1"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4   xmlns:r="http://www.cgnm.de/rdf/relatives#"
5   xml:base="http://www.cgnm.de/rdf/relatives#">
6   <rdf:Description rdf:about="#Anne">
7     <r:motherOf><rdf:Description rdf:about="#Clara"/></r:motherOf>
8   </rdf:Description>
9 </rdf:RDF>
```

Figure 18: Abbreviation of URI attributes for main vocabulary.

Abbreviations / URIs

URI-attributes for additional vocabularies can be abbreviated by declaring an XML entity.

```
1 <?xml version="1.1"?>
2 <!DOCTYPE rdf:RDF [
3   <!ENTITY xs "http://www.w3.org/2001/XMLSchema#">
4 ]>
5 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
7   xmlns:r="http://www.cgnm.de/rdf/relatives#"
8   xml:base="http://www.cgnm.de/rdf/relatives#">
9   <rdf:Description rdf:about="#Anne">
10    <r:motherOf><rdf:Description rdf:about="#Clara"/></r:motherOf>
11  </rdf:Description>
12  <rdf:Description rdf:about="#Anne">
13    <r:age rdf:datatype="&xs;integer">32</r:age>
14  </rdf:Description>
15 </rdf:RDF>
```

Figure 19: Abbreviation of URI attributes for additional vocabulary.

Sample RDF

```
1 <?xml version="1.1"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4   xmlns:r="http://www.cgnm.de/rdf/relatives#"
5   xml:base="http://www.cgnm.de/rdf/relatives#">
6   <rdf:Description rdf:about="#Anne">
7     <r:motherOf><rdf:Description rdf:about="#Clara"/></r:motherOf>
8   </rdf:Description>
9   <rdf:Description rdf:about="#Anne"><r:name>Anne</r:name></rdf:Description>
10  <rdf:Description rdf:about="#Clara"><r:name>Clara</r:name></rdf:Description>
11  <rdf:Description rdf:about="#Anne">
12    <r:motherOf><rdf:Description rdf:nodeID="xyz1"/></r:motherOf>
13  </rdf:Description>
14  <rdf:Description rdf:nodeID="xyz1"><r:name>Dennis</r:name></rdf:Description>
15 </rdf:RDF>
```

Figure 20: Sample RDF/XML.

Abbreviations / Repetition of Subject

If a subject is repeated (i.e., has multiple attributes), these attributes can be written as multiple child elements.

```
1 <?xml version="1.1"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4   xmlns:r="http://www.cgnm.de/rdf/relatives#"
5   xml:base="http://www.cgnm.de/rdf/relatives#">
6 <rdf:Description rdf:about="#Anne">
7   <r:motherOf><rdf:Description rdf:about="#Clara"/></r:motherOf>
8   <r:name>Anne</r:name>
9   <r:motherOf><rdf:Description rdf:nodeID="xyz1"/></r:motherOf>
10 </rdf:Description>
11 <rdf:Description rdf:about="#Clara"><r:name>Clara</r:name></rdf:Description>
12 <rdf:Description rdf:nodeID="xyz1"><r:name>Dennis</r:name></rdf:Description>
13 </rdf:RDF>
```

Figure 21: Sample RDF/XML, repeated subjects abbreviated.

Abbreviations / Nested Triples and Terminal Resources

Triples can be nested, i.e.,
attributes of an object written directly in the element content of the object.

Object URI-references with empty element content
can be replaced by an attribute `rdf:resource="⟨URI⟩"` of the predicate element.

```
6 <rdf:Description rdf:about="#Anne">
7   <r:name>Anne</r:name>
8   <r:motherOf rdf:resource="#Clara"/>
9   <r:motherOf>
10    <rdf:Description><r:name>Dennis</r:name></rdf:Description>
11  </r:motherOf>
12 </rdf:Description>
13 <rdf:Description rdf:about="#Clara"><r:name>Clara</r:name></rdf:Description>
```

Figure 22: Sample RDF/XML, nested triples and terminal resources abbreviated.

Abbreviations / Properties with String-Literal Objects

When the object of a property is a string literal
and the property is not repeated,

⇒ property can be written as attribute of the subject element.

This abbreviation can also be used for property `rdf:type` with an `rdf:resource` attribute ⇒ attribute value is interpreted as an RDF URI reference !

```
1 <?xml version="1.1"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4     xmlns:r="http://www.cgnm.de/rdf/relatives#"
5     xml:base="http://www.cgnm.de/rdf/relatives#">
6   <rdf:Description rdf:about="#Anne" r:name="Anne">
7     <r:motherOf rdf:resource="#Clara"/>
8     <r:motherOf><rdf:Description r:name="Dennis"/></r:motherOf>
9   </rdf:Description>
10  <rdf:Description rdf:about="#Clara" r:name="Clara"/>
11 </rdf:RDF>
```

Figure 23: Sample RDF/XML, properties with string literal objects abbreviated.

Abbreviations / Typed Resources Elements

rdf:Description-elements with nested rdf:type-predicate can be replaced by the *⟨QName⟩* of the class.

In case of multiple rdf:type predicates, only one can be used in this way, the others must remain as property elements or property attributes.

Another Sample RDF with Types

```
1 <?xml version="1.0"?>
2 <!DOCTYPE rdf:RDF [
3   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
4   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
5   <!ENTITY sok "http://www.cgnm.de/rdf/sokrates.rdfs#">
6 ]>
7 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
8   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
9   xmlns:sok="http://www.cgnm.de/rdf/sokrates.rdfs#"
10  xml:base="http://www.cgnm.de/rdf/sokrates.rdfs">
11
12 <rdf:Description rdf:about="#Mortal" rdf:type="&rdfs;Class"/>
13 <rdf:Description rdf:about="#Human" rdf:type="&rdfs;Class">
14   <rdfs:subClassOf rdf:resource="#Mortal"/>
15 </rdf:Description>
16
17 <rdf:Description rdf:type="#Human" sok:name="Sokrates"/>
18 </rdf:RDF>
```

Figure 24: Another sample RDF/XML with types.

Abbreviations / Typed Resources Elements

```
1 <?xml version="1.0"?>
2 <!DOCTYPE rdf:RDF [
3   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
4   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
5   <!ENTITY sok "http://www.cgnm.de/rdf/sokrates.rdfs#">
6 ]>
7 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
8     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
9     xmlns:sok="http://www.cgnm.de/rdf/sokrates.rdfs#"
10    xml:base="http://www.cgnm.de/rdf/sokrates.rdfs">
11   <rdfs:Class rdf:about="#Mortal"/>
12   <rdfs:Class rdf:about="#Human">
13     <rdfs:subClassOf rdf:resource="#Mortal"/>
14   </rdfs:Class>
15
16   <sok:Human sok:name="Sokrates"/>
17 </rdf:RDF>
```

Figure 25: Sample RDF/XML, typed resources abbreviated.

Abbreviations / Omitting rdf:Description Element of Anonymous Resources

Instead of

```
<rdf:Description nodeID="..."> ... </rdf:Description>
```

- the rdf:Description element can be replaced by any of its nested predicate elements (**property-and-node** element),
- the attribute
 rdf:parseType="Resource"
be added to this predicate element, and
- all other of its predicate elements nested in this predicate element.

Parsing RDF/XML

To parse RDF/XML, there exist special RDF/XML parsers, e.g. in the Raptor RDF Parser Toolkit (<http://librdf.org/raptor/>).

Call (after building rapper is in directory `utils`)

```
rapper rdf-triple-anne-short.rdf
```

```
1 <http://www.cgnm.de/rdf/Anne> <http://www.cgnm.de/rdf/relatives#motherOf>  
2 <http://www.cgnm.de/rdf/relatives#Clara> .
```

Figure 26: Sample RDF/XML converter to N3.

II. Semantic Web / 1. Ressource Description Framework (RDF)

1. RDF Basics

2. RDF Built-in Resources

3. RDF/XML Syntax

4. RDF(S) Semantics

RDF(S) triples form a theory (set of sentences in some logical calculus).
Can we map RDF(S) theories to theories in First Order Logics (FOL)?

Translation RDF \rightarrow FOL

constants c_v for each URI v ,
 d_α for each literal α ,
 e_β for each language code β .

variables v_ν for each local name ν .
 a predicate $T/3$ for asserting triples.

a function `pair/2` for localized literals and
`LiteralValueOf/2` for typed values.

RDF x	FOL $\mathcal{T}(x)$
URI reference v	c_v
literal α	d_α
loc. literal $\alpha@ \beta$	<code>pair</code> (d_α, e_β)
typed literal $\alpha \wedge \wedge v$	<code>LiteralValueOf</code> (d_α, c_v)
local name ν	v_ν
triple $(\sigma, \text{rdf:type}, o)$	$T(\mathcal{T}(\sigma), \mathcal{T}(\text{rdf:type}), \mathcal{T}(o))$ $\wedge T(\mathcal{T}(o), \mathcal{T}(\text{rdf:type}), \mathcal{T}(\text{rdfs:Class}))$
triple (σ, π, o)	$T(\mathcal{T}(\sigma), \mathcal{T}(\pi), \mathcal{T}(o))$ $\wedge T(\mathcal{T}(\pi), \mathcal{T}(\text{rdf:type}), \mathcal{T}(\text{rdf:Property}))$
set of triples Θ	$\exists \bigcup_{\theta \in \Theta} \text{free}(\mathcal{T}(\theta)) \bigwedge_{\theta \in \Theta} \mathcal{T}(\theta)$

Translation RDF \rightarrow FOL / Axioms

Axioms (triples (σ, π, o) written as $\pi(\sigma, o)$:

- subClassOf

- reflexivity:

$$\text{rdf:type}(x, \text{rdfs:Class}) \rightarrow \text{rdfs:subClassOf}(x, x)$$

- transitivity:

$$\text{rdfs:subClassOf}(x, y) \wedge \text{rdfs:subClassOf}(y, z) \rightarrow \text{rdfs:subClassOf}(x, z)$$

- maximal element:

$$\text{rdf:type}(x, \text{rdfs:Class}) \rightarrow \text{rdfs:subClassOf}(x, \text{rdfs:Resource})$$

- interaction with `rdf:type`:

$$\begin{aligned} \text{rdfs:subClassOf}(x, y) \rightarrow & \text{rdf:type}(x, \text{rdfs:Class}) \wedge \\ & \text{rdf:type}(y, \text{rdfs:Class}) \wedge \\ & \forall i \text{rdf:type}(i, x) \rightarrow \text{rdf:type}(i, y) \end{aligned}$$

Translation RDF \rightarrow FOL / Axioms

- `rdfs:subPropertyOf`

- reflexivity:

$$\text{rdf:type}(p, \text{rdf:Property}) \rightarrow \text{rdfs:subPropertyOf}(p, p)$$

- transitivity:

$$\text{rdfs:subPropertyOf}(p, q) \wedge \text{rdfs:subPropertyOf}(q, r) \rightarrow \text{rdfs:subPropertyOf}(p, r)$$

- interaction with property assertions:

$$\begin{aligned} \text{rdfs:subPropertyOf}(p, q) \rightarrow & \text{rdf:type}(p, \text{rdf:Property}) \wedge \\ & \text{rdf:type}(q, \text{rdf:Property}) \wedge \\ & \forall i, j p(i, j) \rightarrow q(i, j) \end{aligned}$$

`rdf:Property` is not the maximal element of the property hierarchy.

- `rdfs:domain` and `rdfs:range`:

$$\text{rdfs:domain}(p, x) \rightarrow \forall i, j p(i, j) \rightarrow \text{rdf:type}(i, x)$$

$$\text{rdfs:range}(p, x) \rightarrow \forall i, j p(i, j) \rightarrow \text{rdf:type}(j, x)$$

Translation RDF \rightarrow FOL / Axioms

- class hierarchy and domains and ranges of `rdf(s)` vocabulary, i.e.
 - maximal element of `rdf:type`:
 $\text{rdf:type}(x, \text{rdfs:Resource})$
 - classes, datatypes, properties (`rdf:type`):
 $\text{rdf:type}(\text{rdfs:Resource}, \text{rdfs:Class}), \text{rdf:type}(\text{rdfs:Class}, \text{rdfs:Class})$ etc.
 $\text{rdf:type}(\text{rdf:XMLLiteral}, \text{rdfs:Datatype})$
 $\text{rdf:type}(\text{rdf:type}, \text{rdf:Property}), \text{rdf:type}(\text{rdfs:domain}, \text{rdf:Property})$ etc.
 - class hierarchy (`rdfs:subClassOf`):
 $\text{rdfs:subClassOf}(\text{rdf:Alt}, \text{rdfs:Container}), \text{rdfs:subClassOf}(\text{rdf:Bag}, \text{rdfs:Container})$
 - property hierarchy (`rdfs:subPropertyOf`):
 $\text{rdfs:subPropertyOf}(\text{rdfs:isDefinedBy}, \text{rdfs:seeAlso})$
 - ranges and domains (`rdfs:domain`, `rdfs:range`):
 $\text{rdfs:domain}(\text{rdf:type}, \text{rdfs:Resource}), \text{rdfs:domain}(\text{rdfs:domain}, \text{rdf:Property}),$ etc.
 $\text{rdfs:range}(\text{rdf:type}, \text{rdfs:Class}), \text{rdfs:range}(\text{rdfs:domain}, \text{rdfs:Class}),$ etc.

Translation RDF \rightarrow FOL / Axioms

- a "datatype theory" for each type v :

$\text{rdf:type}(v, \text{rdfs:Datatype})$

$\text{rdf:type}(\text{LiteralValueOf}(\alpha, v), v)$ for all legal lexical literals α

$\neg \text{rdf:type}(\text{LiteralValueOf}(\alpha, v), v)$ for all illegal lexical literals α

A Shorter Alternative ?

Mapping RDF predicates to FOL predicates, i.e.

T_v for each URI v that is used as predicate.

$T(r:Anne, r:isMotherOf, r:Clara)$

$T(sok:Human, rdf:type, rdfs:Class)$

$T_{r:isMotherOf}(r:Anne, r:Clara)$

$T_{rdf:type}(sok:Human, rdfs:Class)$

But, in RDF(S) we can assert statements about predicates,

$rdf:type(r:RelationProperties, rdfs:Class)$

$rdfs:subClassOf(r:RelationProperties, rdf:Property)$

$rdf:type(r:motherOf, r:RelationProperties)$

A Shorter Alternative ?

Getting rid of built-in vocabulary?

 $T(\text{sok:Sokrates}, \text{rdf:type}, \text{sok:Human})$ $R_{\text{sok:Human}}(\text{sok:Sokrates})$

But, in RDF(S) we can assert statements about vocabulary, especially, about built-in RDF(S) vocabulary, e.g.,

 $\text{rdf:type}(\text{x:builtin}, \text{rdfs:Class})$ $\text{rdf:type}(\text{rdfs:Resource}, \text{x:builtin})$

RDF Inference / Prolog (1/3)

RDF inference can be handled, e.g., by SWI-Prologs library "semweb".

```
1 <?xml version="1.0"?>
2 <!DOCTYPE rdf:RDF [
3   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
4   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
5   <!ENTITY sok "http://www.cgnm.de/rdf/sokrates.rdfs#">
6 ]>
7 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
8   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
9   xmlns:sok="http://www.cgnm.de/rdf/sokrates.rdfs#"
10  xml:base="http://www.cgnm.de/rdf/sokrates.rdfs">
11 <rdfs:Class rdf:about="#Mortal"/>
12 <rdfs:Class rdf:about="#Human">
13   <rdfs:subClassOf rdf:resource="#Mortal"/>
14 </rdfs:Class>
15
16 <sok:Human rdf:about="#Sokrates" sok:name="Sokrates"/>
17 </rdf:RDF>
```

Figure 27: RDF example.

RDF Inference / Prolog (2/3)

```
1 [library('semweb/rdf_db')].  
2 [library('semweb/rdfs')].  
3 rdf_load('sokrates-named.rdf').  
4 rdf_register_ns(sok, 'http://www.cgnm.de/rdf/sokrates.rdfs#').  
5 rdf(sok:'Sokrates', X, Y).  
6 rdfs_individual_of(sok:'Sokrates', X).
```

Figure 28: Prolog code for querying coded triples (`rdf`) and inference (`rdfs_individual_of`).

RDF Inference / Prolog (3/3)

1 ?- rdf(sok:'Sokrates', X, Y).

2
3 X = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type'

4 Y = 'http://www.cgnm.de/rdf/sokrates.rdfs#Human' ;

5
6 X = 'http://www.cgnm.de/rdf/sokrates.rdfs#name'

7 Y = literal('Sokrates') ;

8
9 No

10
11 ?- rdfs_individual_of(sok:'Sokrates', X).

12
13 X = 'http://www.cgnm.de/rdf/sokrates.rdfs#Human' ;

14
15 X = 'http://www.cgnm.de/rdf/sokrates.rdfs#Mortal' ;

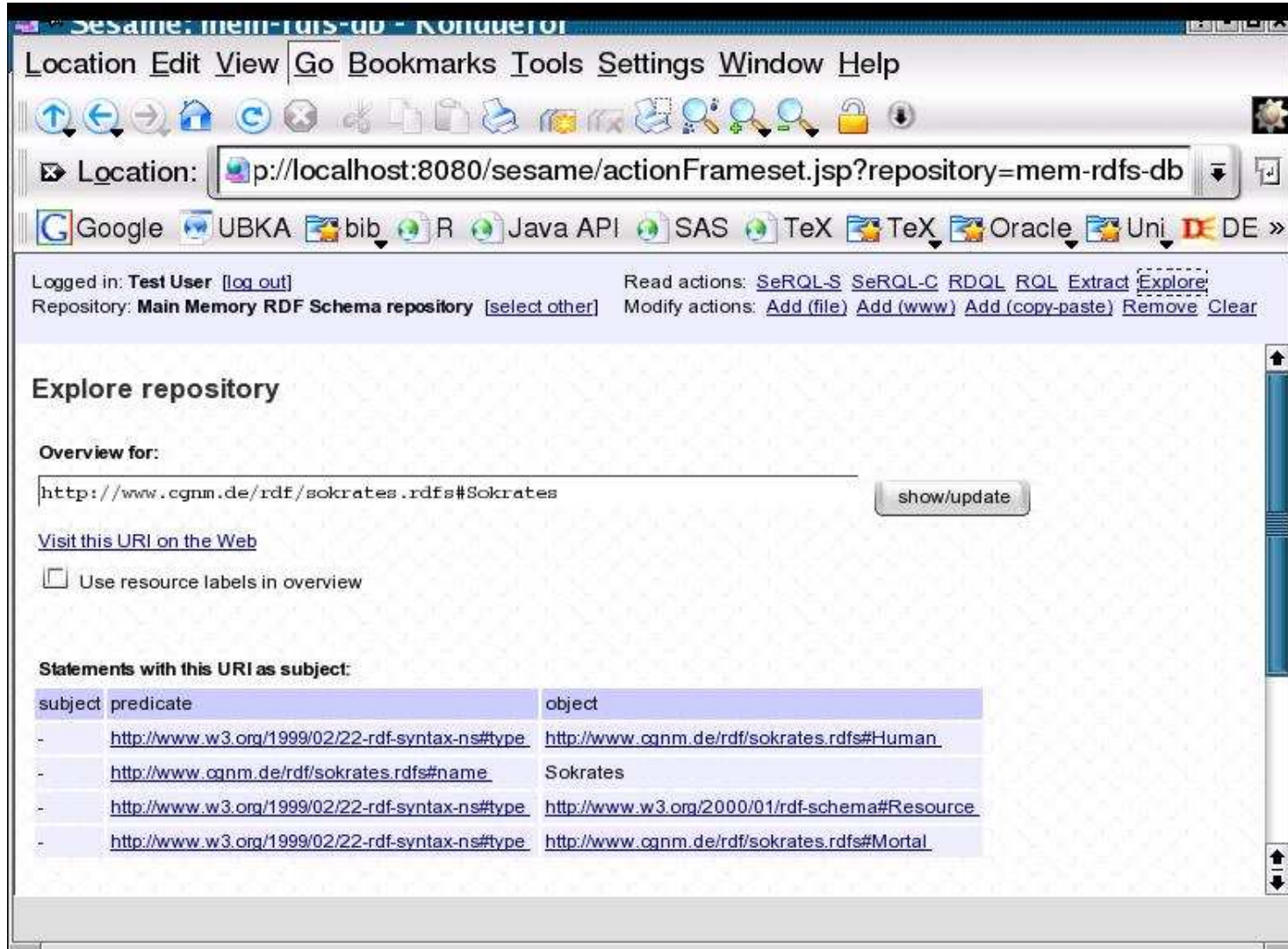
16
17 No

18 ?-

Figure 29: Result.

RDF Inference / Sesame (1/2)

The RDF database Sesame also can do RDF inference.



The screenshot shows the Sesame web interface. The browser address bar displays the URL: `http://localhost:8080/sesame/actionFrameset.jsp?repository=mem-rdfs-db`. The page title is "Sesame: mem-rdfs-db - Konqueror". The interface includes a menu bar (Location, Edit, View, Go, Bookmarks, Tools, Settings, Window, Help) and a toolbar with various icons. The main content area is titled "Explore repository" and shows the following information:

Logged in: **Test User** [log out] Read actions: [SeRQL-S](#) [SeRQL-C](#) [RDQL](#) [RQL](#) [Extract](#) [Explore](#)
 Repository: **Main Memory RDF Schema repository** [select other] Modify actions: [Add \(file\)](#) [Add \(www\)](#) [Add \(copy-paste\)](#) [Remove](#) [Clear](#)

Explore repository

Overview for:

Visit this URI on the Web
 Use resource labels in overview

Statements with this URI as subject:

subject	predicate	object
-	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.cqnm.de/rdf/sokrates.rdfs#Human
-	http://www.cqnm.de/rdf/sokrates.rdfs#name	Sokrates
-	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2000/01/rdf-schema#Resource
-	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.cqnm.de/rdf/sokrates.rdfs#Mortal

Figure 30: Sesame page for resource `sok:Sokrates`.

RDF Inference / Sesame (2/2)

Sesame also supports several RDF query languages, among them, SeRQL.

```
1 SELECT C FROM {<sok:Sokrates>} <rdf:type> {C}
2 using namespace
3   rdf = <!http://www.w3.org/1999/02/22-rdf-syntax-ns#>,
4   rdfs = <!http://www.w3.org/2000/01/rdf-schema#>,
5   sok = <!http://www.cgnm.de/rdf/sokrates.rdfs#>
```

Figure 31: SeRQL query to retrieve all classes Sokrates is an element of.

References