XML and Semantic Web Technologies

III. 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

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

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).

- introducory 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

```
http://www.cgnm.de/rdf/relatives#Anne
http://www.cgnm.de/rdf/relatives#motherOf
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
`<` ⟨*absoluteURI*⟩ `>`	URI reference
`"` ⟨*string*⟩ `"`	untyped literal
`"` ⟨*string*⟩ `"@@<`⟨*absoluteURI*⟩`>`	typed literal
`_:` ⟨*NCName*⟩	anonymous node name

```
<http://www.cgnm.de/rdf/relatives#Anne> <http://www.cgnm.de/rdf/relatives#mother
 <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` ⟨*QName*⟩ ⟨*URI*⟩ `.`

  a prefix (the prefix of the ⟨*QName*⟩, its local name usually is empty) is bound to an URI.

- ⟨*QName*⟩s (with prefixes bound by a `@prefix` declaration) can be used as subjects, predicates, and objects.

syntax	meaning
⟨*QName*⟩	URI reference (prefix URI concatenated with local name)

Notation 3 Representation / example

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

Figure 4: RDF triple in N3 representation.

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

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

Anonymous Subjects and Objects

Anonymous subjects or objects model an existential quantifier:

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

```
@prefix xs:   <http://www.w3.org/2001/XMLSchema#> .
@prefix r: <http://www.cgnm.de/rdf/relatives#> .
r:Anne r:motherOf r:Clara .
r:Anne r:name "Anne" .
r:Anne r:age "32"^^xs:integer .
r:Clara r:name "Clara" .
r:Anne r:motherOf _:xyz1 .
_: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.

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

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

Figure 8: RDF triple in N3 representation.

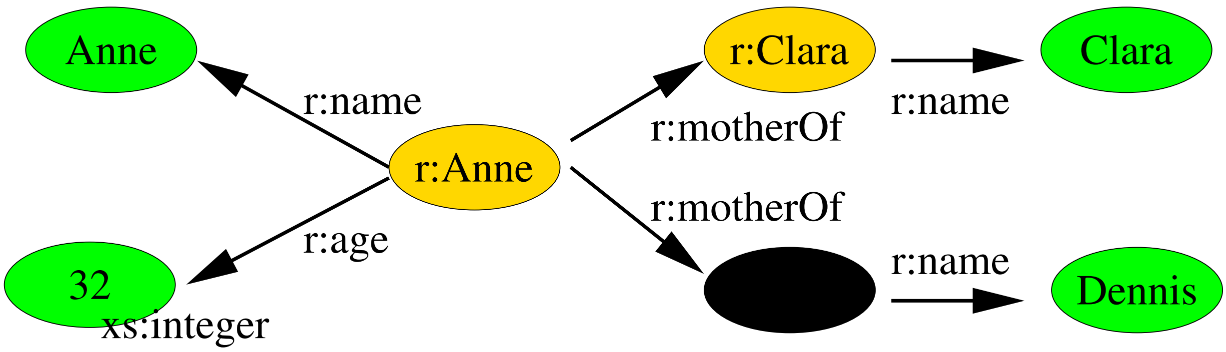

Figure 9: Graph representation of example above.

III. Semantic Web / 1. Ressource 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.,

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

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <#> .
:Mortal rdf:type rdfs:Class .
:Human rdf:type rdfs:Class .
:Human rdfs:subClassOf :Mortal .
:Sokrates rdf:type :Human .
```

Figure 10: Definition of two classes and an instance.

RDF Top-level Classes

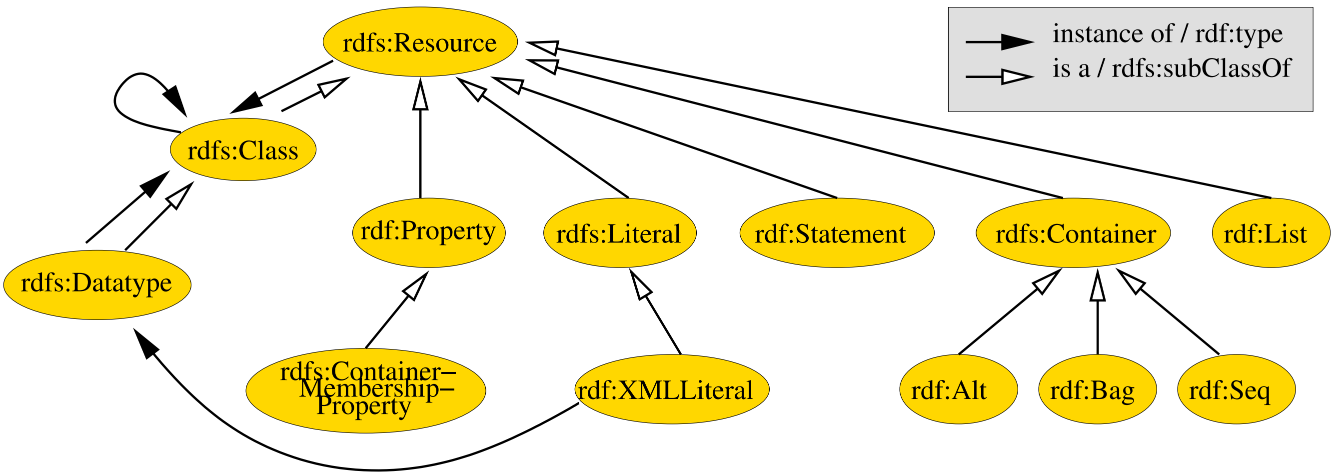

Figure 11: RDF Class Hierarchy.

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

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xs:   <http://www.w3.org/2001/XMLSchema#> .
@prefix : <#> .
:Person rdf:type rdfs:Class .
:motherOf rdf:type rdf:Property ;
          rdfs:domain :Person ;
          rdfs:range :Person .
:name rdf:type rdf:Property ;
      rdfs:domain :Person ;
      rdfs:range xs:string .
:age rdf:type rdf:Property ;
     rdfs:domain :Person ;
     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)

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <#> .
:Person rdf:type rdfs:Class .
:Mortal rdf:type rdfs:Class .
:name rdf:type rdf:Property ; rdfs:domain :Person ; rdfs:range rdfs:Literal .
:believes rdf:type rdf:Property ; rdfs:domain :Person ; rdfs:range rdf:Statement .

:Anne rdf:type rdfs:Person ; :name "Anne" ;
    :believes [ rdf:subject :Person ;
            rdf:predicate rdfs:subClassOf ;
            rdf:object :Mortal ],
          [ rdf:subject [ :name "Sokrates" ] ;
            rdf:predicate rdf:type ;
            rdf:object :Person ].
```

Figure 13: Statements about Statements: beliefs.

XML and Semantic Web Technologies / 2. RDF Built-in Resources

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

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <#> .
:Person rdf:type rdfs:Class .
:Course rdf:type rdfs:Class .
:name rdf:type rdf:Property ; rdfs:domain :Person ; rdfs:range rdfs:Literal .
:title rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdfs:Literal .
:students rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdf:Bag .

:XML :title "XML and Semantic Web Technologies" ;
     :students [ rdfs:member [ :name "Anne" ] ,
                             [ :name "Bert" ] ,
                             [ :name "Clara" ] ] .
```

Figure 14: An RDF Container.

RDF Lists (aka Collections)

`rdfs:List` is the class of all closed containers (i.e., elements cannote 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

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <#> .
:Person rdf:type rdfs:Class .
:Course rdf:type rdfs:Class .
:name rdf:type rdf:Property ; rdfs:domain :Person ; rdfs:range rdfs:Literal .
:title rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdfs:Literal .
:students rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdfs:List .

:XML :title "XML and Semantic Web Technologies" ;
    :students [ rdf:first [ :name "Anne" ] ;
            rdf:rest  [ rdf:first [ :name "Bert" ] ;
                    rdf:rest  [ rdf:first [ :name "Clara" ] ; rdf:rest rdf:nil ]
                    ]
            ] .
```

Figure 15: An RDF List.

XML and Semantic Web Technologies / 2. RDF Built-in Resources

RDF Lists

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

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <#> .
:Person rdf:type rdfs:Class .
:Course rdf:type rdfs:Class .
:name rdf:type rdf:Property ; rdfs:domain :Person ; rdfs:range rdfs:Literal .
:title rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdfs:Literal .
:students rdf:type rdf:Property ; rdfs:domain :Course ; rdfs:range rdfs:Bag .

:XML :title "XML and Semantic Web Technologies" ;
    :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.

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

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 synrax	meaning
< ⟨*absoluteURI*⟩ >	<rdf:Description rdf:about="⟨*URI*⟩"/> ⟨*QName*⟩	URI reference for predicates
" ⟨*string*⟩ "	⟨*string*⟩	untyped literal
" ⟨*string*⟩ " @@<⟨*absoluteURI*⟩>	<. . . rdf:datatype="⟨*URI*⟩">⟨*string*⟩</. . . >	typed literal
_: ⟨*NCName*⟩	<rdf:Description rdf:nodeID="⟨*NCName*⟩"/>	anonymous no

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.

```
<?xml version="1.1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:r="http://www.cgnm.de/rdf/relatives#">
  <rdf:Description rdf:about="http://www.cgnm.de/rdf/relatives#Anne">
    <r:motherOf>
      <rdf:Description rdf:about="http://www.cgnm.de/rdf/relatives#Clara"/>
    </r:motherOf>
  </rdf:Description>
</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).

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

```
<?xml version="1.1"?>
<!DOCTYPE rdf:RDF [
 <!ENTITY xs "http://www.w3.org/2001/XMLSchema#">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:r="http://www.cgnm.de/rdf/relatives#"
         xml:base="http://www.cgnm.de/rdf/relatives#">
  <rdf:Description rdf:about="#Anne">
    <r:motherOf><rdf:Description rdf:about="#Clara"/></r:motherOf>
  </rdf:Description>
  <rdf:Description rdf:about="#Anne">
    <r:age rdf:datatype="&xs;integer">32</r:age>
  </rdf:Description>
</rdf:RDF>
```

Figure 19: Abbreviation of URI attributes for additional vocabulary.

Sample RDF

```
<?xml version="1.1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:r="http://www.cgnm.de/rdf/relatives#"
        xml:base="http://www.cgnm.de/rdf/relatives#">
 <rdf:Description rdf:about="#Anne">
  <r:motherOf><rdf:Description rdf:about="#Clara"/></r:motherOf>
 </rdf:Description>
 <rdf:Description rdf:about="#Anne"><r:name>Anne</r:name></rdf:Description>
 <rdf:Description rdf:about="#Clara"><r:name>Clara</r:name></rdf:Description>
 <rdf:Description rdf:about="#Anne">
  <r:motherOf><rdf:Description rdf:nodeID="xyz1"/></r:motherOf>
 </rdf:Description>
 <rdf:Description rdf:nodeID="xyz1"><r:name>Dennis</r:name></rdf:Description>
</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.

```
<?xml version="1.1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:r="http://www.cgnm.de/rdf/relatives#"
        xml:base="http://www.cgnm.de/rdf/relatives#">
 <rdf:Description rdf:about="#Anne">
  <r:motherOf><rdf:Description rdf:about="#Clara"/></r:motherOf>
  <r:name>Anne</r:name>
  <r:motherOf><rdf:Description rdf:nodeID="xyz1"/></r:motherOf>
 </rdf:Description>
 <rdf:Description rdf:about="#Clara"><r:name>Clara</r:name></rdf:Description>
 <rdf:Description rdf:nodeID="xyz1"><r:name>Dennis</r:name></rdf:Description>
</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.

```
<rdf:Description rdf:about="#Anne">
  <r:name>Anne</r:name>
  <r:motherOf rdf:resource="#Clara"/>
  <r:motherOf>
    <rdf:Description><r:name>Dennis</r:name></rdf:Description>
  </r:motherOf>
</rdf:Description>
<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 !

```
<?xml version="1.1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:r="http://www.cgnm.de/rdf/relatives#"
        xml:base="http://www.cgnm.de/rdf/relatives#">
  <rdf:Description rdf:about="#Anne" r:name="Anne">
    <r:motherOf rdf:resource="#Clara"/>
    <r:motherOf><rdf:Description r:name="Dennis"/></r:motherOf>
  </rdf:Description>
  <rdf:Description rdf:about="#Clara" r:name="Clara"/>
</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

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY sok  "http://www.cgnm.de/rdf/sokrates.rdfs#">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:sok="http://www.cgnm.de/rdf/sokrates.rdfs#"
         xml:base="http://www.cgnm.de/rdf/sokrates.rdfs">

  <rdf:Description rdf:about="#Mortal" rdf:type="&rdfs;Class"/>
  <rdf:Description rdf:about="#Human" rdf:type="&rdfs;Class">
    <rdfs:subClassOf rdf:resource="#Mortal"/>
  </rdf:Description>

  <rdf:Description rdf:type="#Human" sok:name="Sokrates"/>
</rdf:RDF>
```

Figure 24: Another sample RDF/XML with types.

Abbreviations / Typed Resources Elements

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY sok  "http://www.cgnm.de/rdf/sokrates.rdfs#">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:sok="http://www.cgnm.de/rdf/sokrates.rdfs#"
         xml:base="http://www.cgnm.de/rdf/sokrates.rdfs">
  <rdfs:Class rdf:about="#Mortal"/>
  <rdfs:Class rdf:about="#Human">
    <rdfs:subClassOf rdf:resource="#Mortal"/>
  </rdfs:Class>

  <sok:Human sok:name="Sokrates"/>
</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

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

Figure 26: Sample RDF/XML converter to N3.

III. 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 → FOL

constants c_υ for each URI υ,
d_α for each literal α,
e_β for each language code β.

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

variables v_ν for each local name ν.

a predicate T/3 for asserting triples.

RDF x	FOL $\mathcal{T}(x)$
URI reference υ	c_υ
literal α	d_α
loc. literal α@β	pair(d_α, e_β)
typed literal α^^υ	LiteralValueOf(d_α, c_υ)
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 → 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 → 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 hierachy.

- `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 → 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`):

$$\text{rdf:type}(\text{rdfs:Resource}, \text{rdfs:Class}), \text{rdf:type}(\text{rdfs:Class}, \text{rdfs:Class})\text{etc.}$$

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

$$\text{rdf:type}(\text{rdf:type}, \text{rdf:Property}), \text{rdf:type}(\text{rdfs:domain}, \text{rdf:Property})\text{etc.}$$

  - class hierarchy (`rdfs:subClassOf`):

$$\text{rdfs:subClassOf}(\text{rdf:Alt}, \text{rdfs:Container}), \text{rdfs:subClassOf}(\text{rdf:Bag}, \text{rdfs:Contain}$$

  - property hierarchy (`rdfs:subPropertyOf`):

$$\text{rdfs:subPropertyOf}(\text{rdfs:isDefinedBy}, \text{rdfs:seeAlso})$$

  - ranges and domains (`rdfs:domain`, `rdfs:range`):

$$\text{rdfs:domain}(\text{rdf:type}, \text{rdfs:Resource}), \text{rdfs:domain}(\text{rdfs:domain}, \text{rdf:Property}), \text{e}$$

$$\text{rdfs:range}(\text{rdf:type}, \text{rdfs:Class}), \text{rdfs:range}(\text{rdfs:domain}, \text{rdfs:Class}), \text{etc.}$$

Translation RDF → FOL / Axioms

- a "datatype theory" for each type υ:

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

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

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

A Shorter Alternative ?

Mapping RDF predicates to FOL predicates, i.e.
T_v for each URI v that is used as predicate.

$T(\text{r:Anne}, \text{r:isMotherOf}, \text{r:Clara})$
$T(\text{sok:Human}, \text{rdf:type}, \text{rdfs:Class})$

$T_{\text{r:isMotherOf}}(\text{r:Anne}, \text{r:Clara})$
$T_{\text{rdf:type}}(\text{sok:Human}, \text{rdfs:Class})$

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

$\text{rdf:type}(\text{r:RelationProperties}, \text{rdfs:Class})$
$\text{rdfs:subClassOf}(\text{r:RelationProperties}, \text{rdf:Property})$
$\text{rdf:type}(\text{r:motherOf}, \text{r:RelationProperties})$

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".

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
  <!ENTITY sok  "http://www.cgnm.de/rdf/sokrates.rdfs#">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:sok="http://www.cgnm.de/rdf/sokrates.rdfs#"
        xml:base="http://www.cgnm.de/rdf/sokrates.rdfs">
  <rdfs:Class rdf:about="#Mortal"/>
  <rdfs:Class rdf:about="#Human">
    <rdfs:subClassOf rdf:resource="#Mortal"/>
    </rdfs:Class>

  <sok:Human rdf:about="#Sokrates" sok:name="Sokrates"/>
</rdf:RDF>
```

Figure 27: RDF example.

RDF Inference / Prolog (2/3)

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

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

X = 'http://www.w3.org/1999/02/22-rdf-syntax-ns#type'
Y = 'http://www.cgnm.de/rdf/sokrates.rdfs#Human' ;

X = 'http://www.cgnm.de/rdf/sokrates.rdfs#name'
Y = literal('Sokrates') ;

No

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

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

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

No
?-
```

Figure 29: Result.

RDF Inference / Sesame (1/2)

The RDF database Sesame also can do RDF inference.

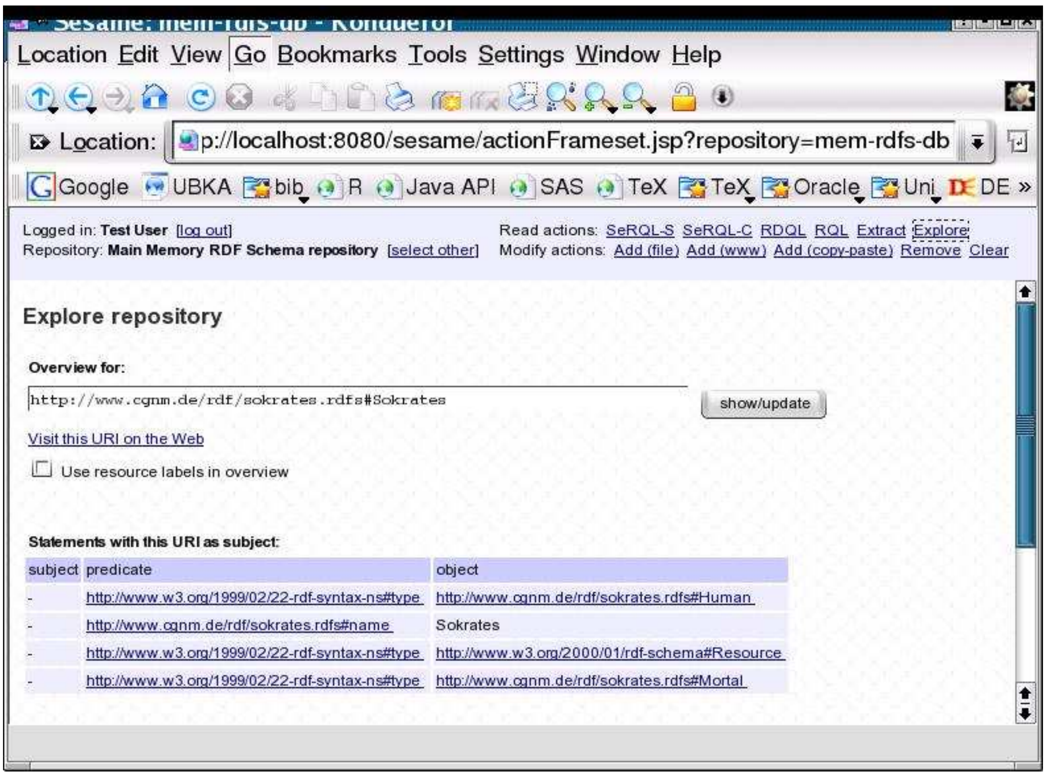

Figure 30: Sesame page for resource `sok:Sokrates`.

RDF Inference / Sesame (2/2)

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

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

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