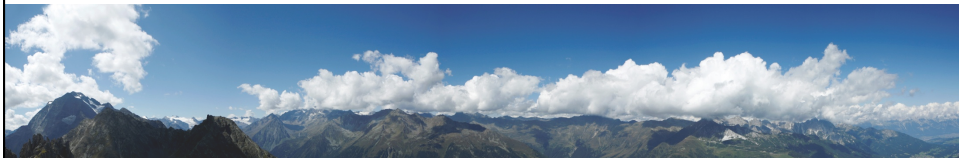


Semantic Web

Resource Description Framework (RDF)



© Copyright 2010 Dieter Fensel and Federico Facca

Where are we?



STI · INNSBRUCK

#	Title
1	Introduction
2	Semantic Web Architecture
➔ 3	Resource Description Framework (RDF)
4	Web of data
5	Generating Semantic Annotations
6	Storage and Querying
7	Web Ontology Language (OWL)
8	Rule Interchange Format (RIF)
9	Reasoning on the Web
10	Ontologies
11	Social Semantic Web
12	Semantic Web Services
13	Tools
14	Applications

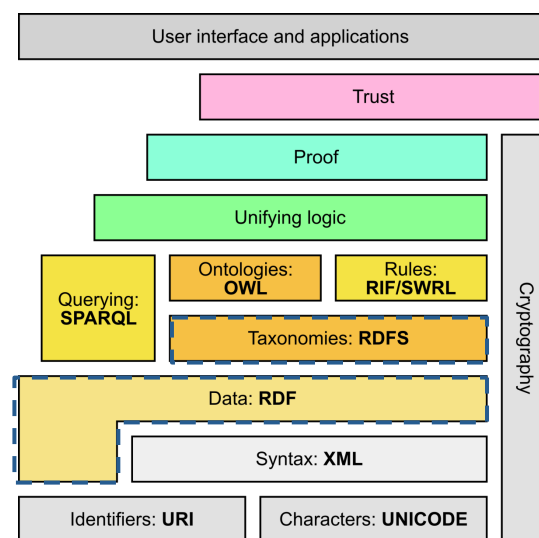
Agenda



1. Introduction and Motivation
2. Technical Solution
 1. RDF
 2. RDF Schema
 3. RDF(S) Semantics
 4. RDF(S) Serialization
 5. Tools
3. Illustration by a large example
4. Extensions
5. Summary
6. References

3

Semantic Web Stack



Adapted from http://en.wikipedia.org/wiki/Semantic_Web_Stack

4

INTRODUCTION AND MOTIVATION

5

Motivating Example 1

- Dieter Fensel is teaching the Semantic Web course.

*Examples adapted from
Grigoris Antoniou and Frank van Harmelen: A Semantic Web Primer, MIT Press 2004*

6

Motivating Example 1



- Dieter Fensel is teaching the Semantic Web course.

```
<course name="Semantic Web">  
  <lecturer>Dieter Fensel</lecturer>  
</course>
```

What's the problem you can spot in this representation?

*Examples adapted from
Grigoris Antoniou and Frank van Harmelen: A Semantic Web Primer, MIT Press 2004*

7

Motivating Example 1



- Dieter Fensel is teaching the Semantic Web course.

```
<course name="Semantic Web">  
  <lecturer>Dieter Fensel</lecturer>  
</course>  
  
<lecturer name="Dieter Fensel">  
  <teaches>Semantic Web</teaches>  
</lecturer>
```

*The first two formalizations include essentially an opposite nesting
although they represent the same information.*

There is no standard way of assigning meaning to tag nesting.

*Examples adapted from
Grigoris Antoniou and Frank van Harmelen: A Semantic Web Primer, MIT Press 2004*

8

Motivating Example 1



- Dieter Fensel is teaching the Semantic Web course.

```
<course name="Semantic Web">
  <lecturer>Dieter Fensel</lecturer>
</course>

<lecturer name="Dieter Fensel">
  <teaches>Semantic Web</teaches>
</lecturer>

<teachingOffering>
  <lecturer>Dieter Fensel</lecturer>
  <course>Semantic Web</course>
</teachingOffering>
```

*Examples adapted from
Grigoris Antoniou and Frank van Harmelen: A Semantic Web Primer, MIT Press 2004*

9

Motivating Example 2



- A lecturer is a subclass of an academic staff member.

This sentence means that all lecturers are also academic staff members. It is important to understand that there is an intended meaning associated with "is a subclass of". It is not up to the application to interpret this term; its intended meaning must be respected by all RDF processing software.

*Examples adapted from
Grigoris Antoniou and Frank van Harmelen: A Semantic Web Primer, MIT Press 2004*

10

Motivating Example 2



STI · INNSBRUCK

- A lecturer is a subclass of an academic staff member.

```
<academicStaffMember>Dieter Fensel</academicStaffMember>
```

```
<professor>Dieter Fensel</professor>
```

```
<course name="Semantic Web">  
  <isTaughtBy>Federico M. Facca</isTaughtBy>  
</course>
```

*Examples adapted from
Grigoris Antoniou and Frank van Harmelen: A Semantic Web Primer, MIT Press 2004*

11

Motivating Example 2



STI · INNSBRUCK

- Retrieve all the members of the academic staff.

*An example Xpath query to achieve the second bullet task over
presented XML is: //academicStaffMember*

*Examples adapted from
Grigoris Antoniou and Frank van Harmelen: A Semantic Web Primer, MIT Press 2004*

12

Motivating Example 2



- The result is only Dieter Fensel.

*Correct from the XML viewpoint, But semantically unsatisfactory.
Human readers would have also included Federico M. Facca.*

*This kind of information makes use of the semantic model of the
particular domain and cannot be represented in XML or in RDF but
is typical of knowledge written in RDF Schema.*

RDFS makes semantic information machine accessible

*Examples adapted from
Grigoris Antoniou and Frank van Harmelen: A Semantic Web Primer, MIT Press 2004*

13

What Are RDF and RDF Schema?



- RDF
 - Resource Description Framework
 - Data model
 - Syntax (XML)
 - Domain independent
 - Vocabulary is defined by RDF Schema
- RDF Schema
 - RDF Vocabulary Description Language
 - Captures the *semantic model* of a domain

14

RDF and RDF Schema

TECHNICAL SOLUTION

15

The power of triple representation joint with XML serialization

THE RESOURCE DESCRIPTION FRAMEWORK

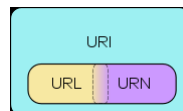
Most of the examples in the upcoming slides are taken from: <http://www.w3.org/TR/rdf-primer/>

16

- RDF is a language that enable to describe making statements on resources
 - John is father of Bill
- Statement (or triple) as a logical formula $P(x, y)$, where the binary predicate P relates the object x to the object y
- Triple data model:
 - `<subject, predicate, object>`
 - **Subject:** Resource or blank node
 - **Predicate:** Property
 - **Object:** Resource (or collection of resources), literal or blank node
- Example:
 - `<ex:john, ex:father-of, ex:bill>`
- RDF offers only binary predicates (properties)

- A resource may be:
 - Web page (e.g. `http://www.w3.org`)
 - A person (e.g. `http://www.fensel.com`)
 - A book (e.g. `urn:isbn:0-345-33971-1`)
 - Anything denoted with a URI!
- A URI is an *identifier* and **not** a location on the Web
- RDF allows making statements about resources:
 - `http://www.w3.org` has the format text/html
 - `http://www.fensel.com` has first name Dieter
 - `urn:isbn:0-345-33971-1` has author Tolkien

- A Uniform Resource Identifier (URI) is a string of characters used to identify a name or a resource on the Internet



- A URI can be a URL or a URN
- A Uniform Resource Name (URN) defines an item's identity
 - the URN *urn:isbn:0-395-36341-1* is a URI that specifies the identifier system, i.e. International Standard Book Number (ISBN), as well as the unique reference within that system and allows one to talk about a book, but doesn't suggest where and how to obtain an actual copy of it
- A Uniform Resource Locator (URL) provides a method for finding it
 - the URL *http://www.sti-innsbruck.at/* identifies a resource (STI's home page) and implies that a representation of that resource (such as the home page's current HTML code, as encoded characters) is obtainable via HTTP from a network host named *www.sti-innsbruck.at*

- Plain literals
 - E.g. *"any text"*
 - Optional language tag, e.g. *"Hello, how are you?"@en-GB*
- Typed literals
 - E.g. *"hello"^^xsd:string, "1"^^xsd:integer*
 - Recommended datatypes:
 - XML Schema datatypes
- Only as *object* of a triple, e.g.:

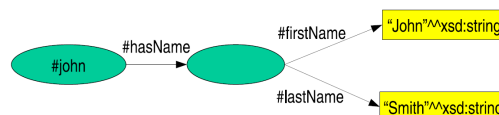

```
<http://example.org/#john>,
    <http://example.org/#hasName>,
    "John Smith"^^xsd:string>
```



- One pre-defined datatype: `rdf:XMLLiteral`
 - Used for embedding XML in RDF
- Recommended datatypes are XML Schema datatypes, e.g.:
 - `xsd:string`
 - `xsd:integer`
 - `xsd:float`
 - `xsd:anyURI`
 - `xsd:boolean`

- Blank nodes are nodes without a URI
 - Unnamed resources
 - More complex constructs
- Representation of blank nodes is *syntax-dependent*
 - *Blank node identifier*
- For example:

```
<#john>, <#hasName>, _:johnsname>
<_:johnsname, <#firstName>, "John"^^xsd:string>
<_:johnsname, <#lastName>, "Smith"^^xsd:string>
```



- **Representation of complex data**

A blank node can be used to indirectly attach to a resource a consistent set of properties which together represent a complex data

- **Anonymous classes in OWL**

The ontology language OWL uses blank nodes to represent anonymous classes such as unions or intersections of classes, or classes called restrictions, defined by a constraint on a property

- Grouping property values:

“The lecture is attended by John, Mary and Chris”

Bag

*“[RDF-Concepts] is edited by Graham and Jeremy
(in that order)”*

Seq

*“The source code for the application may be found at
ftp1.example.org,
ftp2.example.org,
ftp3.example.org”*

Alt

- Three types of containers:
 - `rdf:Bag` - unordered set of items
 - `rdf:Seq` - ordered set of items
 - `rdf:Alt` - set of alternatives
- Every container has a triple declaring the `rdf:type`
- Items in the container are denoted with
 - `rdf:_1`, `rdf:_2`, . . . , `rdf:_n`

- Three types of containers:
 - `rdf:Bag` - unordered set of items
 - `rdf:Seq` - ordered set of items
 - `rdf:Alt` - set of alternatives
- Every container has a triple declaring the `rdf:type`
- Items in the container are denoted with
 - `rdf:_1`, `rdf:_2`, . . . , `rdf:_n`
- Limitations:
 - Semantics of the container is up to the application
 - What about closed sets?
 - How do we know whether Graham and Jeremy are the only editors of [RDF-Concepts]?

-
- ```

graph TD
 A([ex:john]) -- "ex:father-of" --> B([ex:bill])

```

27

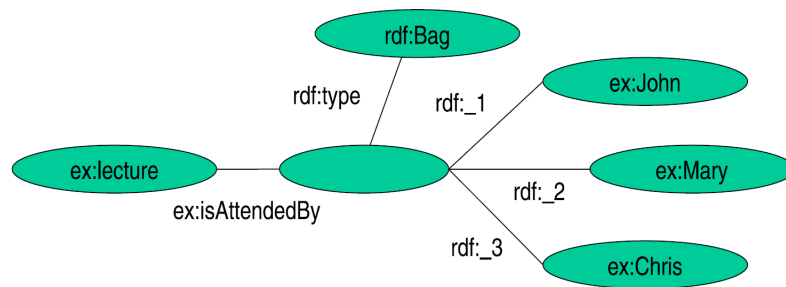
- [illegible]

28

## RDF Containers Graph Representation: Bag



*"The lecture is attended by John, Mary and Chris"*

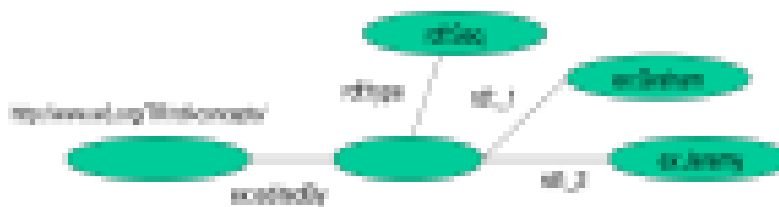


29

## RDF Containers Graph Representation: Seq

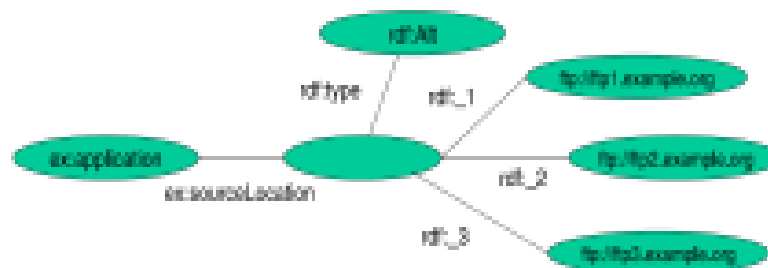


*"[RDF-Concepts] is edited by Graham and Jeremy  
(in that order)"*

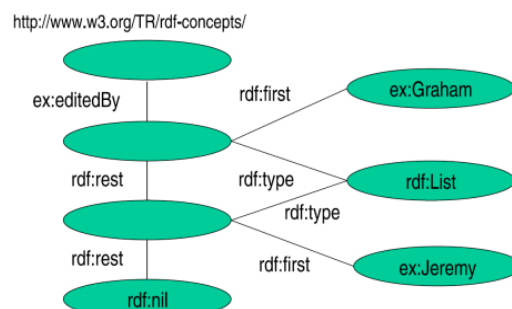


30

*“The source code for the application may be found at  
***ftp1.example.org, ftp2.example.org, ftp3.example.org***”*



*“[RDF-Concepts] is edited by Graham and Jeremy  
 (in that order) and nobody else”*



RDF provides support for describing groups containing only the specified members, in the form of RDF collections.



- Reification: statements about statements

*Mary claims that John's name is "John Smith".*

```
<#myStatement>, rdf:type, rdf:Statement>
<#myStatement>, rdf:subject, <#john>
<#myStatement>, rdf:predicate, <#hasName>
<#myStatement>, rdf:object, "John Smith"
```

This kind of statement can be used to describe belief or trust in other statements, which is important in some kinds of applications

Necessary because there are only triples in RDF: we cannot add an identifier directly to a triple (then it would be a quadruple)

- Reification: statements about statements

*Mary claims that John's name is "John Smith".*

```
<#myStatement>, rdf:type, rdf:Statement>
<#myStatement>, rdf:subject, <#john>
<#myStatement>, rdf:predicate, <#hasName>
<#myStatement>, rdf:object, "John Smith"
```



```
<#john>, <#hasName>, "John Smith"
```

In such a way we attached a label to the statement.

- Reification: statements about statements

*Mary claims that John's name is "John Smith".*

```
<#myStatement>, rdf:type, rdf:Statement>
<#myStatement>, rdf:subject, <#john>
<#myStatement>, rdf:predicate, <#hasName>
<#myStatement>, rdf:object, "John Smith">

<#mary>, <#claims>, <#myStatement>
```

RDF uses only binary properties. This restriction seems quite serious because often we use predicates with more than two arguments. Luckily, such predicates can be simulated by a number of binary predicates.

35

- RDF defines a number of resources and properties
- We have already seen: `rdf:XMLLiteral`, `rdf:type`, ...
- RDF vocabulary is defined in the namespace:  
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- Classes:
  - `rdf:Property`, `rdf:Statement`, `rdf:XMLLiteral`
  - `rdf:Seq`, `rdf:Bag`, `rdf:Alt`, `rdf:List`
- Properties:
  - `rdf:type`, `rdf:subject`, `rdf:predicate`, `rdf:object`,
  - `rdf:first`, `rdf:rest`, `rdf:_n`
  - `rdf:value`
- Resources:
  - `rdf:nil`

36

- Typing using **rdf:type**:  
`<A, rdf:type, B>`  
*"A belongs to class B"*
- All properties belong to class **rdf:Property**:  
`<P, rdf:type, rdf:Property>`  
*"P is a property"*  
  
`<rdf:type, rdf:type, rdf:Property>`  
*"rdf:type is a property"*

How to represent the semantics of data models

## THE RDF SCHEMA (RDFS)

- Types in RDF:
 

```
<#john, rdf:type, #Student>
```
- What is a “*#Student*”?
- RFD is not defining a vocabulary about the statements, but only to express statements
- We know that “*#Student*” identifies a category (a concept or a class), but this is only implicitly defined in RDF

- We need a language for defining RDF types:
  - Define classes:
    - “*#Student is a class*”
  - Relationships between classes:
    - “*#Student is a sub-class of #Person*”
  - Properties of classes:
    - “*#Person has a property hasName*”
- RDF Schema is such a language



- Classes:
  - `<#Student, rdf:type, #rdfs:Class>`
- Class hierarchies:
  - `<#Student, rdfs:subClassOf, #Person>`
- Properties:
  - `<#hasName, rdf:type, rdf:Property>`
- Property hierarchies:
  - `<#hasMother, rdfs:subPropertyOf, #hasParent>`
- Associating properties with classes (a):
  - “The property `#hasName` only applies to `#Person`”
  - `<#hasName, rdfs:domain, #Person>`
- Associating properties with classes (b):
  - “The type of the property `#hasName` is `#xsd:string`”
  - `<#hasName, rdfs:range, xsd:string>`

41



- RDFS Extends the RDF Vocabulary
- RDFS vocabulary is defined in the namespace:
  - <http://www.w3.org/2000/01/rdf-schema#>

### RDFS Classes

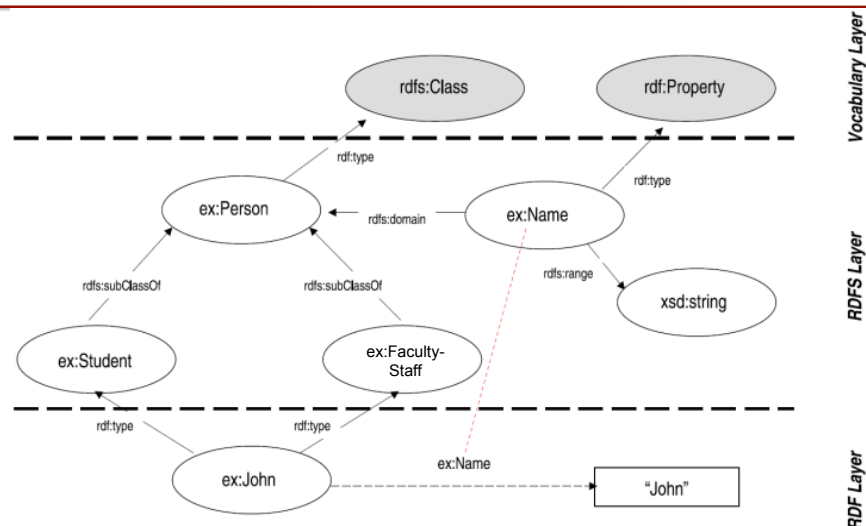
- `rdfs:Resource`
- `rdfs:Class`
- `rdfs:Literal`
- `rdfs:Datatype`
- `rdfs:Container`
- `rdfs:ContainerMembershipProperty`

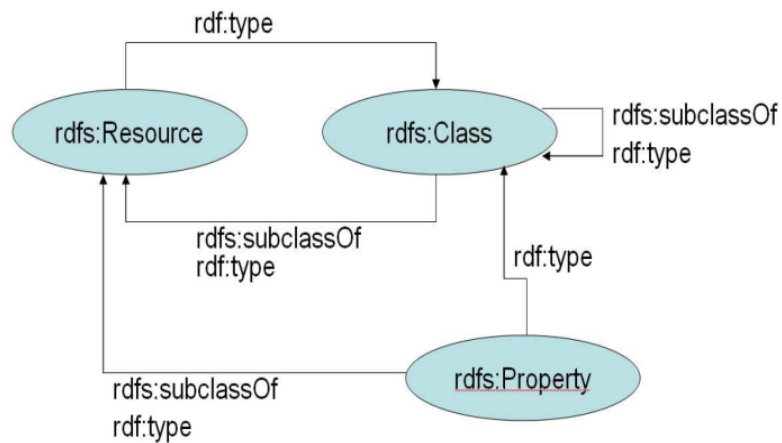
### RDFS Properties

- `rdfs:domain`
- `rdfs:range`
- `rdfs:subPropertyOf`
- `rdfs:subClassOf`
- `rdfs:member`
- `rdfs:seeAlso`
- `rdfs:isDefinedBy`
- `rdfs:comment`
- `rdfs:label`

42

- **Resource**
  - All resources are implicitly instances of `rdfs:Resource`
- **Class**
  - Describe sets of resources
  - Classes are resources themselves - e.g. Webpages, people, document types
    - Class hierarchy can be defined through `rdfs:subClassOf`
    - Every class is a member of `rdfs:Class`
- **Property**
  - Subset of RDFS Resources that are properties
    - **Domain:** class associated with property: `rdfs:domain`
    - **Range:** type of the property values: `rdfs:range`
    - Property hierarchy defined through: `rdfs:subPropertyOf`





- Metadata is “data about data”
- Any meta-data can be attached to a resource, using:
  - **`rdfs:comment`**
    - Human-readable description of the resource, e.g.
      - `<ex:Person>, rdfs:comment, "A person is any human being">`
  - **`rdfs:label`**
    - Human-readable version of the resource name, e.g.
      - `<ex:Person>, rdfs:label, "Human being">`
  - **`rdfs:seeAlso`**
    - Indicate additional information about the resource, e.g.
      - `<ex:Person>, rdfs:seeAlso, <http://xmlns.com/wordnet/1.6/Human>>`
  - **`rdfs:isDefinedBy`**
    - A special kind of `rdfs:seeAlso`, e.g.
      - `<ex:Person>, rdfs:isDefinedBy, <http://xmlns.com/wordnet/1.6/Human>>`

# RDF(S) SEMANTICS

47

## Semantics

- RDF(S) vocabulary has built-in “meaning”
- RDF(S) Semantics
  - Makes meaning explicit
  - Defines what follows from an RDF graph
- Semantic notions
  - Subgraph
  - Instance
  - Entailment

48



- $E$  is a subgraph of  $S$  if and only if  $E$  predicates are a subset of  $S$  predicates

```
- (<#john>, <#hasName>, _:johnsname)
- (<_:johnsname, <#firstName>, "John"^^xsd:string>
- (<_:johnsname, <#lastName>, "Smith"^^xsd:string>
```

- Subgraphs:

```
- (<#john>, <#hasName>, :johnsname)
- (<_:johnsname, <#firstName>, "John"^^xsd:string>

- (<_:johnsname, <#firstName>, "John"^^xsd:string>
- (<_:johnsname, <#lastName>, "Smith"^^xsd:string>

- (<#john>, <#hasName>, _:johnsname)
```

- $S'$  is an instance of  $S$  if and only if some blank nodes in  $S$  are replaced with blank nodes, literals or URIs

```
- (<#john>, <#hasName>, _:johnsname)
- (<_:johnsname, <#firstName>, "John"^^xsd:string>
- (<_:johnsname, <#lastName>, "Smith"^^xsd:string>
```

- Instances:

```
- (<#john>, <#hasName>, <#abc>)
- (<#abc>, <#firstName>, "John"^^xsd:string>
- (<#abc>, <#lastName>, "Smith"^^xsd:string>

- (<#john>, <#hasName>, _:X)
- (<_:X, <#firstName>, "John"^^xsd:string>
- (<_:X, <#lastName>, "Smith"^^xsd:string>

- (<#john>, <#hasName>, _:johnsname)
- (<_:johnsname, <#firstName>, "John"^^xsd:string>
- (<_:johnsname, <#lastName>, "Smith"^^xsd:string>
```

- Every graph is an instance of itself!

- **Entailment** or logical implication is a relation between sentences of a formal language
- $S$  entails  $E$  if  $E$  logically follows from  $S$ 
  - Written:  $S \models E$
- A graph entails all its subgraphs
  - If  $S'$  is a subgraph of  $S$ :  $S \models S'$
- All instances of a graph  $S$  entail  $S$ 
  - If  $S''$  is an instance of  $S$ :  $S'' \models S$

```
<http://example.org/#john> rdf:type <http://example.org/#Student>
<http://example.org/#Student> rdfs:subClassOf <http://example.org/#Person>
```

entails

```
<http://example.org/#john> rdf:type <http://example.org/#Person>
```

```
<http://example.org/#hasName> rdfs:domain <http://example.org/#Student>
<http://example.org/#mary> <http://example.org/#hasName> "Mary"
```

entails

```
<http://example.org/#mary> rdf:type <http://example.org/#Student>
```

```
<http://example.org/#john> <http://example.org/#hasMother> <http://example.org/#mary>
<http://example.org/#hasMother> rdfs:subPropertyOf <http://example.org/#hasParent>
```

entails

```
<http://example.org/#john> <http://example.org/#hasParent> <http://example.org/#mary>
```

```
<http://example.org/#john> rdf:type <http://example.org/#Student>
<http://example.org/#Student> rdfs:subClassOf <http://example.org/#Person>
```

entails

```
<http://example.org/#john> rdf:type <http://example.org/#Person>
```

```
<http://example.org/#hasName> rdfs:domain <http://example.org/#Student>
<http://example.org/#mary> <http://example.org/#hasName> "Mary"
```

entails

```
<http://example.org/#mary> rdf:type <http://example.org/#Student>
```

```
<http://example.org/#john> <http://example.org/#hasMother> <http://example.org/#mary>
<http://example.org/#hasMother> rdfs:subPropertyOf <http://example.org/#hasParent>
```

entails

```
<http://example.org/#john> <http://example.org/#hasParent> <http://example.org/#mary>
```

```
<http://example.org/#john> rdf:type <http://example.org/#Student>
<http://example.org/#Student> rdfs:subClassOf <http://example.org/#Person>
```

entails

```
<http://example.org/#john> rdf:type <http://example.org/#Person>
```

```
<http://example.org/#hasName> rdfs:domain <http://example.org/#Student>
<http://example.org/#mary> <http://example.org/#hasName> "Mary"
```

entails

```
<http://example.org/#mary> rdf:type <http://example.org/#Student>
```

```
<http://example.org/#john> <http://example.org/#hasMother> <http://example.org/#mary>
<http://example.org/#hasMother> rdfs:subPropertyOf <http://example.org/#hasParent>
```

entails

```
<http://example.org/#john> <http://example.org/#hasParent> <http://example.org/#mary>
```



- Semantics defined through *entailment rules*
- Rule:
  - If S contains `<triple pattern>` then add `<triple>`
- Executing all entailment rules yields *realization* of S
- S entails E if E is a subgraph of the realization of S
- Axiomatic triple are always added



- if E contains `<A, B, C>` then add  
`<B, rdf:type, rdf:Property>`
- if E contains `<A, B, 1>` (l is a valid XML literal) then add  
`<_:x, rdf:type, rdf:XMLLiteral>`  
where `_:x` identifies to blank node allocated to `1`

- everything in the subject is a resource
  - if  $E$  contains  $\langle A, B, C \rangle$  then add  $\langle A, \text{rdf:type}, \text{rdfs:Resource} \rangle$
- every non-literal in the object is a resource
  - if  $E$  contains  $\langle A, B, C \rangle$  ( $C$  is not a literal) then add  $\langle C, \text{rdf:type}, \text{rdfs:Resource} \rangle$
- every class is subclass of `rdfs:Resource`
  - if  $E$  contains  $\langle A, \text{rdf:type}, \text{rdfs:Class} \rangle$  then add  $\langle A, \text{rdfs:subClassOf}, \text{rdfs:Resource} \rangle$
- inheritance:
  - if  $E$  contains  $\langle A, \text{rdf:type}, B \rangle, \langle B, \text{rdfs:subClassOf}, C \rangle$  then add  $\langle A, \text{rdf:type}, C \rangle$
- `rdfs:subClassOf` is transitive
  - if  $E$  contains  $\langle A, \text{rdfs:subClassOf}, B \rangle, \langle B, \text{rdfs:subClassOf}, C \rangle$  then add  $\langle A, \text{rdfs:subClassOf}, C \rangle$

- `rdfs:subClassOf` is reflexive
  - if  $E$  contains  $\langle A, \text{rdf:type}, \text{rdfs:Class} \rangle$  then add  $\langle A, \text{rdfs:subClassOf}, A \rangle$
- `rdfs:subPropertyOf` is transitive
  - if  $E$  contains  $\langle A, \text{rdfs:subPropertyOf}, B \rangle, \langle B, \text{rdfs:subPropertyOf}, C \rangle$  then add  $\langle A, \text{rdfs:subPropertyOf}, C \rangle$
- `rdfs:subPropertyOf` is reflexive
  - if  $E$  contains  $\langle P, \text{rdf:type}, \text{rdf:Property} \rangle$  then add  $\langle P, \text{rdfs:subPropertyOf}, P \rangle$
- domain of properties
  - if  $E$  contains  $\langle P, \text{rdfs:domain}, C \rangle, \langle A, P, B \rangle$  then add  $\langle A, \text{rdf:type}, C \rangle$
- range of properties
  - if  $E$  contains  $\langle P, \text{rdfs:range}, C \rangle, \langle A, P, B \rangle$  then add  $\langle B, \text{rdf:type}, C \rangle$

- every literal is a member of `rdfs:Literal`
  - if  $E$  contains  $\langle A, B, l \rangle$  ( $l$  is a plain literal) then add  $\langle \_ : X, \text{rdf:type}, \text{rdfs:Literal} \rangle$
- every datatype is subclass of `rdfs:Literal`
  - if  $E$  contains  $\langle A, \text{rdf:type}, \text{rdfs:Datatype} \rangle$  then add  $\langle A, \text{rdfs:subClassOf}, \text{rdfs:Literal} \rangle$

## RDF(S) SERIALIZATION

There are several machine readable serialization formats for RDF

- RDF/XML
- Turtle
- N3

- Serializing RDF for the Web
  - XML as standardized interchange format:
    - Namespaces (e.g. `rdf:type`, `xsd:integer`, `ex:john`)
    - Encoding (e.g. UTF8, iso-8859-1)
    - XML Schema (e.g. datatypes)
- Reuse of existing XML tools:
  - Syntax checking (i.e. schema validation)
  - Transformation (via XSLT)
    - Different RDF representation
    - Layout (XHTML)
    - Different XML-based formats
- Parsing and in-memory representation/manipulation (DOM/SAX)
- ...

```
<#john, #hasName, "John">
<#john, #marriedTo, #mary>
```

```
<!ENTITY ex "http://example.org/#">

<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:ex="http://example.org#"

 <rdf:Description rdf:about="http://example.org/#john">
 <ex:hasName>John</ex:hasName>
 <ex:marriedTo rdf:resource="&ex;mary"/>
 </rdf:Description>

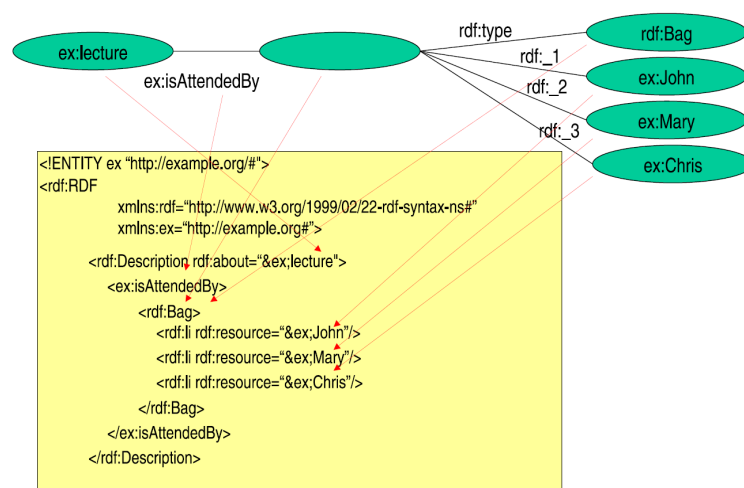
</rdf:RDF>
```

Head

Body

Foot

63



64



- Turtle stands for Terse RDF Triple Language
- An RDF serialization
- Triple representation of <Subject, Predicate, Object>
- Example:

```
@prefix person: <http://example/person/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
person: A foaf:name "Jek" .
person: A foaf:mbox <mailto:jek@example.net> .
person: B foaf:name "Yuan" .
_:b foaf:name "Jeff" .
_:b foaf:mbox <mailto:jeff@example.org> .
```

- Notation3, or N3
- A shorthand non-XML serialization of RDF models
- Designed for human-readability
  - much more compact and readable than XML RDF notation
- Example

```
{:John :Loves :Mary} :accordingTo :Bill
```

# TOOLS

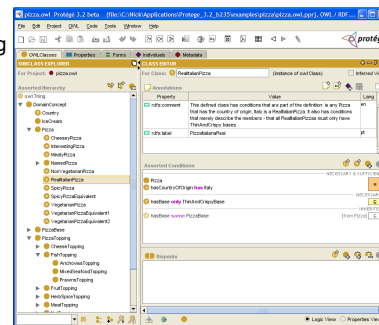
67

## Tool Support for RDF/RDFS

- Ontology editors
  - Protégé (<http://protege.stanford.edu/>)
- Browser
  - /facet (<http://slashfacet.semanticweb.org/>)
- RFD repositories
  - Sesame (<http://www.openrdf.org/>)
- APIs
  - RDF2Go – Java (<http://semanticweb.org/wiki/RDF2Go>)
  - Jena – Java (<http://jena.sourceforge.net/>)
- Validator
  - W3C Validator (<http://www.w3.org/RDF/Validator/>)

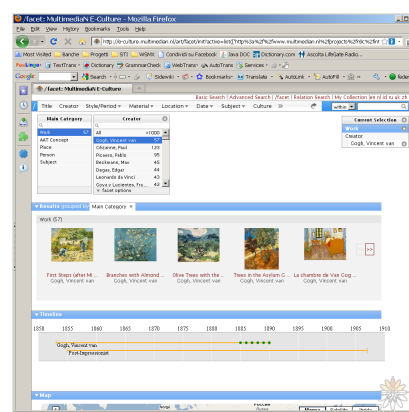
68

- Developed by Stanford Medical Informatics
- Has a large user community (approx 30k)
- Support
  - Graph view, consistency check, web, merging
- No support
  - Addition of new basic types
  - Limited multi-user support



69

- /facet is a generic browser for heterogeneous semantic web repositories
- Works on any RDFS dataset without any additional configuration
- Select and navigate facets of resources of any type
- Make selections based on properties of other, semantically related, types
- Allows the inclusion of facet-specific display options



70

- A framework for storage, querying and inferencing of RDF and RDF Schema
- A Java Library for handling RDF
- A Database Server for (remote) access to repositories of RDF data
- Features:
  - Light-weight yet powerful Java API
  - SeRQL, SPARQL
  - High scalability ( $O(10^7)$  triples on desktop hardware)
  - Various backends (Native Store, RDBMS, main memory)
  - Reasoning support
  - Transactional support
  - Context support
  - RDF/XML, Turtle, N3, N-Triples

- A Java framework for building Semantic Web applications
- Initiated by Hewlett Packard (HP) Labs Semantic Web Programme.
- Includes:
  - A RDF API
  - Reading and writing RDF in RDF/XML, N3 and N-Triples
  - An OWL API
  - In-memory and persistent storage
  - SPARQL query engine

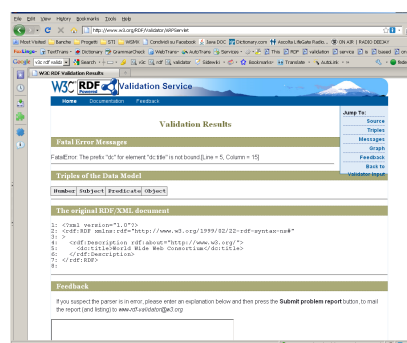


- **RDF2Go** is an abstraction over triple (and quad) stores. It allows developers to program against rdf2go interfaces and choose or change the implementation later easily
- It can be extended: you can create an adapter from any RDF Object Model to RDF2Go object model
- Directly supported implementations:
  - Jena 2.4
  - Jena 2.6
  - Sesame 2

73



- RDF Validator
- Parse RDF documents and detects errors w.r.t. the current RDF specification
- Available online service
- Downloadable code
- Based on ARP parser (the one also adopted in Jena)



74

An example of usage of RDF and RDF(S)

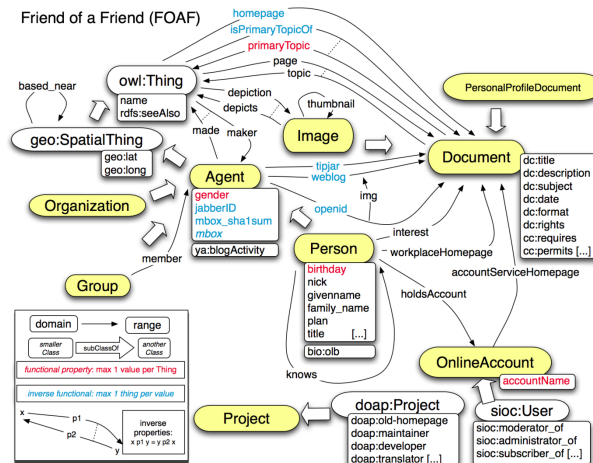
## ILLUSTRATION BY A LARGER EXAMPLE

75

### Friend of a Friend (FOAF)

- Friend of a Friend is a project that aims at providing simple ways to describe people and relations among them
- FOAF adopts RDF and RDFS
- Full specification available on: <http://xmlns.com/foaf/spec/>
- Tools based on FOAF:
  - FOAF search (<http://foaf.qdos.com/>)
  - FOAF builder (<http://foafbuilder.qdos.com/>)
  - FOAF-a-matic (<http://www.ldodds.com/foaf/foaf-a-matic>)
  - FOAF.vix (<http://foaf-visualizer.org/>)

76


[\[http://www.foaf-project.org/\]](http://www.foaf-project.org/)

77

```
<?xml version="1.0" encoding="utf-8"?>
<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:foaf="http://
 xmlns.com/foaf/0.1/">
 <foaf:Person rdf:ID="DieterFensel">
 <foaf:name>Dieter Fensel</foaf:name>
 <foaf:title>Univ.-Prof. Dr.</foaf:title>
 <foaf:givenname>Dieter</foaf:givenname>
 <foaf:family_name>Fensel</foaf:family_name>
 <foaf:mbox_sha1sum>773a221a09f1887a24853c9de06c3480e714278a</
 foaf:mbox_sha1sum>
 <foaf:homepage rdf:resource="http://www.fensel.com "/>
 <foaf:depiction rdf:resource="http://www.deri.at/fileadmin/images/photos/
 dieter_fensel.jpg"/> <foaf:phone rdf:resource="tel:+43-512-507-6488"/>
 <foaf:workplaceHomepage rdf:resource="http://www.sti-innsbruck.at"/>
 <foaf:workInfoHomepage rdf:resource="http://www.sti-innsbruck.at/
 about/team/details/?uid=40"/> </foaf:Person>
</rdf:RDF>
```

78

QDOS

BETA

FOAF

beta

Friend of a Friend

Search for a profile

Go

Simply enter an email address, or the URL of someone's homepage or blog in the box and click 'Go'.

We are currently unable to find any results for Dieter Fensel

If you know of a FOAF file for this person, please [submit it here](#)

What is FOAF?

FOAF (Friend of a Friend) is a way of describing people, their activities, and their relationships to other people and objects. FOAF allows groups of people to build a social network without the need for a centralised database.

The FOAF project (<http://www.foaf-project.org>) was started in 2000 by Libby Miller & Dan Brickley. Today there are many millions of FOAF profiles published on the Web, and it is growing daily. FOAF is a great way to build large scale, open social network that are controlled by individual users and not by any one single company.

One of the challenges with FOAF is to make it more accessible, and Gertlin's contribution is to make available a new QDOS FOAF service that allows people to search and browse millions of FOAF files in a user friendly way. Our aim is to index and make visible the entire FOAF universe and we will continue to extend, update and publish this index for free, to encourage wider use of FOAF for building social networks.

QDOS

BETA

Search for another profile

Find

What do you think?


Thoughts? Feelings?

Relationships?

Dieter Fensel

Homepage: [fensel.com](http://fensel.com)


Contact Details


 [dieter.fensel@denlog](mailto:dieter.fensel@denlog)

Connections

- choose a new relationship -

Known By 2 People

 Andreas Harth

 Jürgen Umbrich

Edit this profile

Terms & Conditions

Privacy

FAQ

Dev

Terms & Conditions

Privacy


FAQ

Developers

Applications

79

# EXTENSIONS

 STI · INNSBRUCK

80

40



- RDF(S) has a some limitations in term of representation and semantics, thus OWL was build on top of it to overcome some of them
- We have seen how to represent statements in RDF, how to query them? SPARQL is currently the standard language to query RDF data
- RDF(S) by itself is not providing any instrument to define personalized entailment rules
  - The entailment process is driven by RDF(S) Semantics
  - This is not enough in many practical contexts
- RDF can be extend to add rule support
  - RULE-ML based extensions
  - Horn Logic based extensions
  - OWL Horst (include a fragment of DL as well)
  - OWLIM (include a fragment of DL as well)

## SUMMARY



- RDF
  - Advantages:
    - Reuse existing standards/tools
    - Provides some structure for free (e.g. for containers)
    - Standard format
  - Disadvantages:
    - Verbose
    - Reconstructing RDF graph non-trivial



- RDF Schema
  - Advantages
    - A primitive ontology language
    - Offers certain modeling primitives with fixed meaning
    - Key concepts of RDF Schema
      - subclass relations, property, subproperty relations, domain and range restrictions
    - There exist query languages for RDF and RDFS
    - Allows metamodeling
  - Disadvantages
    - A quite primitive as a modeling language for the Web
    - Many desirable modeling primitives are missing
      - An ontology layer on top of RDF/RDFS is needed

- Mandatory reading
  - Semantic Web Primer
    - Chapter 3 (only Sections 3.1 to 3.6)
- Further reading
  - RDF Primer
    - <http://www.w3.org/TR/REC-rdf-syntax/>
  - RDF Vocabulary Description Language 1.0: RDF Schema
    - <http://www.w3.org/TR/rdf-schema/>

- Wikipedia
  - [http://en.wikipedia.org/wiki/Resource\\_Description\\_Framework](http://en.wikipedia.org/wiki/Resource_Description_Framework)
  - [http://en.wikipedia.org/wiki/RDF\\_schema](http://en.wikipedia.org/wiki/RDF_schema)
  - [http://en.wikipedia.org/wiki/Turtle\\_\(syntax\)](http://en.wikipedia.org/wiki/Turtle_(syntax))
  - [http://en.wikipedia.org/wiki/Notation\\_3](http://en.wikipedia.org/wiki/Notation_3)
  - <http://en.wikipedia.org/wiki/N-Triples>

## Next Lecture



| #   | Title                                |
|-----|--------------------------------------|
| 1   | Introduction                         |
| 2   | Semantic Web Architecture            |
| 3   | Resource Description Framework (RDF) |
| ➔ 4 | <b>Web of data</b>                   |
| 5   | Generating Semantic Annotations      |
| 6   | Storage and Querying                 |
| 7   | Web Ontology Language (OWL)          |
| 8   | Rule Interchange Format (RIF)        |
| 9   | Reasoning on the Web                 |
| 10  | Ontologies                           |
| 11  | Social Semantic Web                  |
| 12  | Semantic Web Services                |
| 13  | Tools                                |
| 14  | Applications                         |

87

## Questions?



88