Unit 1 – The Semantic Web - Lecture Overview and Bird's View on RDF(S), OWL & SPARQL

Axel Polleres

DERI, National University of Ireland, Galway

VU 184.268 Technologien für das Semantische Web

Unit Outline

1. Organisation
2. Motivation – Aggregating Linked Open Data by Rules & Ontologies
3. How can I publish data? RDF
4. How can I query that data? SPARQL
5. What does that data mean? Ontologies described in RDFS + OWL
6. What's next?

Organisation – This lecture will be blocked in the next two weeks!

- Lecture: 5 Blocks á 3 hours. Possible slots (green preferred):
  - 12/01/2009, 9:00–12:00 Vortmann HS
  - 13/01/2009, 9:00–12:00 **or** 13:00–16:00 Sem. Room 184/3
  - 15/01/2009, 9:00–12:00 Zemanek HS
  - 16/01/2009 – 9:00–12:00 **or** 13:00–16:00 Sem. Room 184/2(!)
  - 22/01/2009 – 16:00–19:00 Sem. Room 184/3
- Small assignments for the first three blocks, collected solutions to be submitted per e-mail to `axel.polleres@deri.org`
- Question & Answers and discussion of assignments: 21/01/2009, 15:00 – 18:00
- Written exam: 23/01/2009, 15:00 – 17:00 (assignment/exam count50:50).

Prerequisites

- Some basic knowledge about first-order logics.
- Some basic knowledge about databases (SQL).
- Some basic knowledge about HTML.
- Some basic knowledge about XML would be nice.
- Who knows RDF, OWL, SPARQL already?
- Who knows Description Logics?
- Who knows Logic Programming(Datalog, Prolog, Answer Set Programming?
- Who knows XQuery, RIF, FOAF, SIOC?

Ideally: Who attended **188.399 VU "Einführung in Semantic Web"** or a similar lecture already?

Unit Outline

1. Organisation
2. Motivation – Aggregating Linked Open Data by Rules & Ontologies
3. How can I publish data? RDF
4. How can I query that data? SPARQL
5. What does that data mean? Ontologies described in RDFS + OWL
6. What's next?

Finding reviewers for a scientific Journal

Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, Jim Hendler: **N3Logic: A logical framework for the World Wide Web**. Theory and Practice of Logic Programming (TPLP), Volume 8, p249-269.

Assume you are the editor of a scientific journal:

- Who are the right reviewers?

Finding reviewers for a scientific Journal

Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, Jim Hendler: **N3Logic: A logical framework for the World Wide Web**. Theory and Practice of Logic Programming (TPLP), Volume 8, p249-269.

Assume you are the editor of a scientific journal:

- Who are the right reviewers?
- Which qualified people do I know?

Finding reviewers for a scientific Journal

Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, Jim Hendler: **N3Logic: A logical framework for the World Wide Web**. Theory and Practice of Logic Programming (TPLP), Volume 8, p249-269.

Assume you are the editor of a scientific journal:

- Who are the right reviewers?
- Which qualified people do I know?
- How can I assess their expertise?

Finding reviewers for a scientific Journal

Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, Jim Hendler: **N3Logic: A logical framework for the World Wide Web**. Theory and Practice of Logic Programming (TPLP), Volume 8, p249-269.

Assume you are the editor of a scientific journal:

- Who are the right reviewers?
- Which qualified people do I know?
- How can I assess their expertise?
- Which reviewers are in conflict?

Finding reviewers for a scientific Journal

Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, Jim Hendler: **N3Logic: A logical framework for the World Wide Web**. Theory and Practice of Logic Programming (TPLP), Volume 8, p249-269.

Assume you are the editor of a scientific journal:

- Who are the right reviewers?
- Which qualified people do I know?
- How can I assess their expertise?
- Which reviewers are in conflict?
- Observation: Much of the necessary data is available on the Web!

Finding reviewers for a scientific Journal

Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, Jim Hendler: **N3Logic: A logical framework for the World Wide Web**. Theory and Practice of Logic Programming (TPLP), Volume 8, p249-269.

Assume you are the editor of a scientific journal:

- Who are the right reviewers?
- Which qualified people do I know?
- How can I assess their expertise?
- Which reviewers are in conflict?
- Observation: Much of the necessary data is available on the Web!

Finding reviewers for a scientific Journal

Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, Jim Hendler: **N3Logic: A logical framework for the World Wide Web**. Theory and Practice of Logic Programming (TPLP), Volume 8, p249-269.

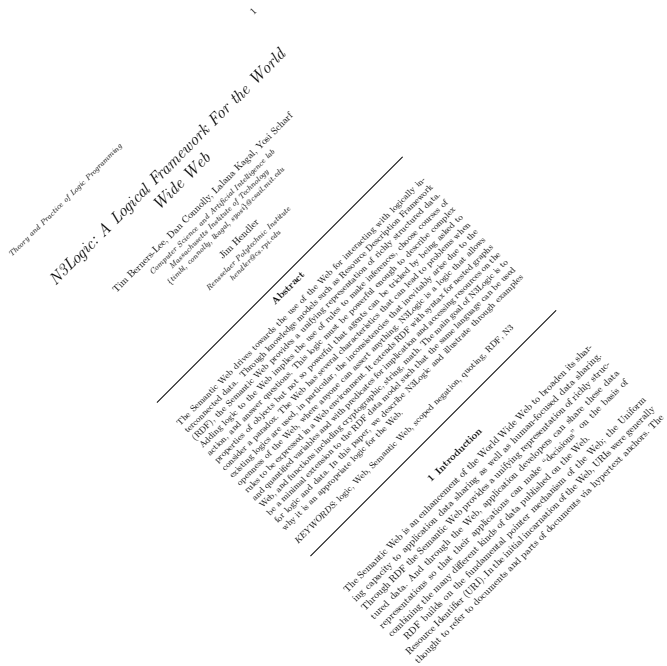
Assume you are the editor of a scientific journal:

- Who are the right reviewers?
- Which qualified people do I know?
- How can I assess their expertise?
- Which reviewers are in conflict?
- Observation: Much of the necessary data is available on the Web!

Questions:

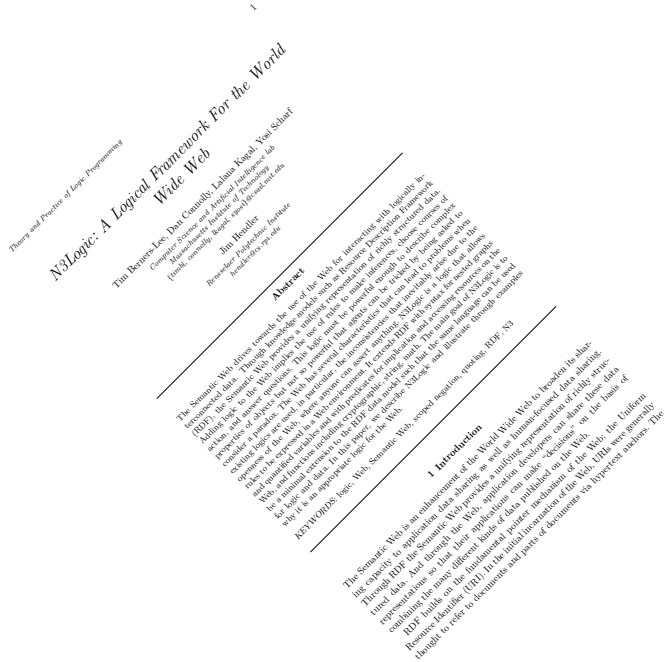
- Where do I get the right data?
- What is the format & structure (schema) of this data?
- Which rules and query languages do I use to aggregate this data?
- Which systems are out there to support me?

Where is the data? 1/4



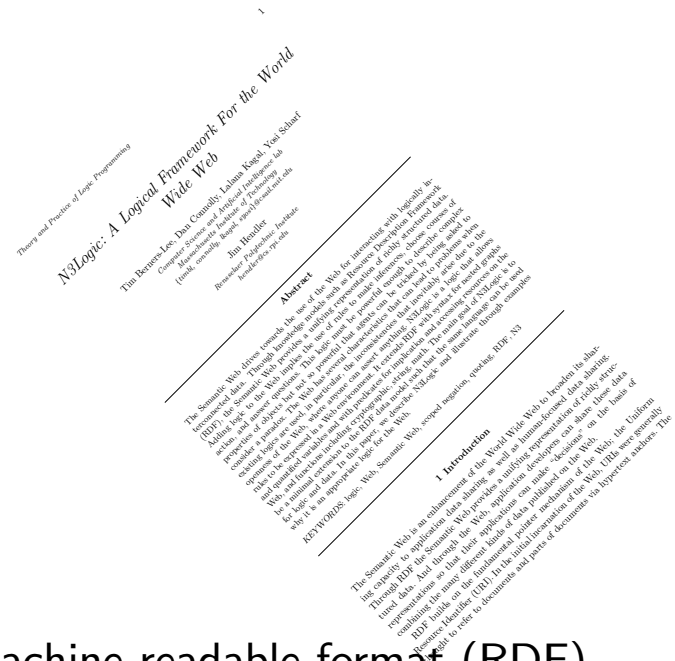
[1] Excellent tutorial here: `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`

Where is the data? 1/4



[1] Excellent tutorial here: `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`

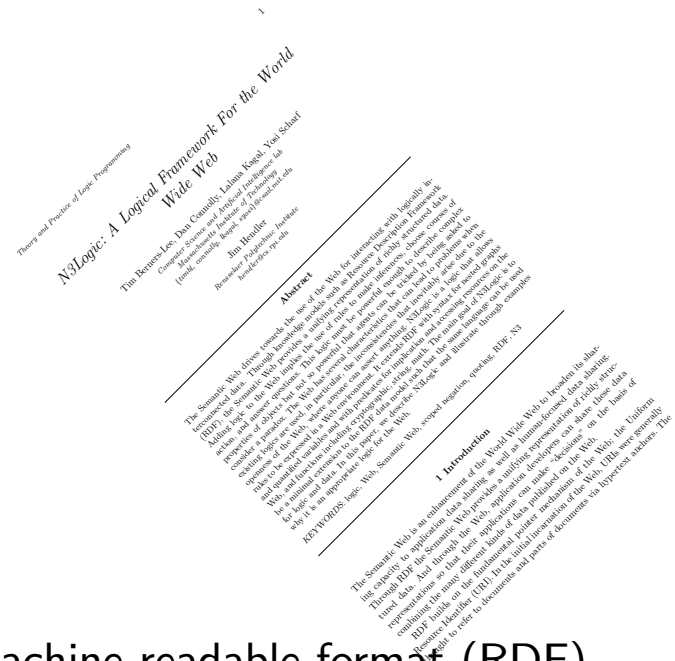
Where is the data? 1/4



- A lot of Web data already available "out there" in a machine-readable format (RDF)

[1] Excellent tutorial here: `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`

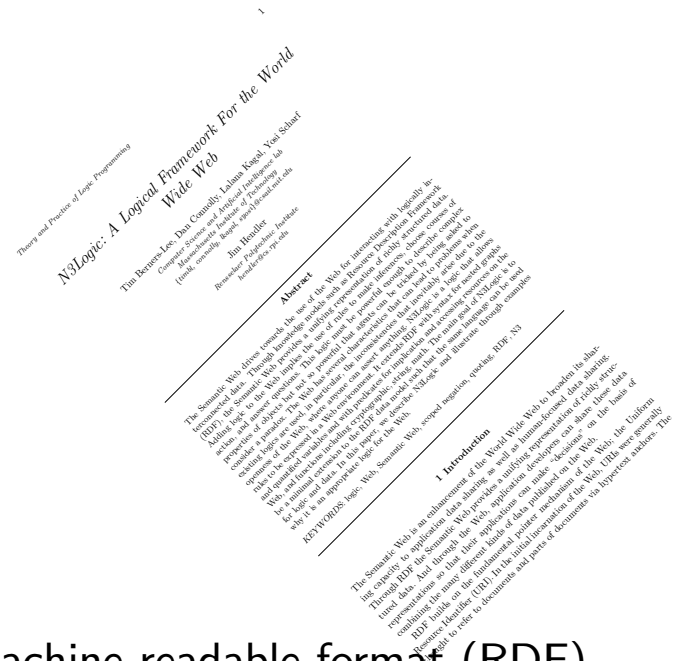
Where is the data? 1/4



- A lot of Web data already available "out there" in a machine-readable format (RDF)
- More and more of it follows the *Linked Data* principles[1], i.e.:

[1] Excellent tutorial here: `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`

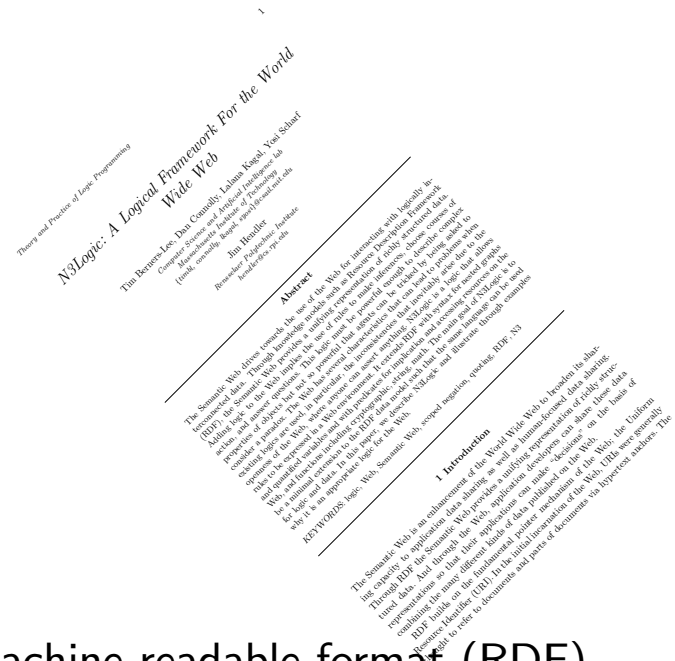
Where is the data? 1/4



- A lot of Web data already available "out there" in a machine-readable format (RDF)
- More and more of it follows the *Linked Data* principles[1], i.e.:
  1. Use URIs as names for things

[1] Excellent tutorial here: `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`

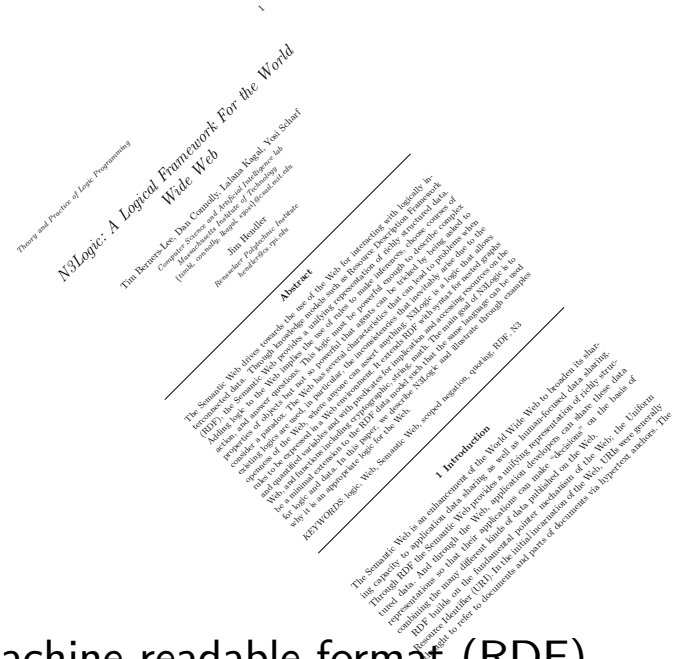
Where is the data? 1/4



- A lot of Web data already available "out there" in a machine-readable format (RDF)
- More and more of it follows the *Linked Data* principles[1], i.e.:
  1. Use URIs as names for things
  2. Use HTTP dereferenceable URIs so that people can look up those names.

[1] Excellent tutorial here: `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`

Where is the data? 1/4



- A lot of Web data already available "out there" in a machine-readable format (RDF)
- More and more of it follows the *Linked Data* principles[1], i.e.:
  1. Use URIs as names for things
  2. Use HTTP dereferenceable URIs so that people can look up those names.
  3. When someone looks up a URI, provide useful information.

[1] Excellent tutorial here: `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`

Where is the data? 1/4

- A lot of Web data already available "out there" in a machine-readable format (RDF)
- More and more of it follows the *Linked Data* principles[1], i.e.:
  1. Use URIs as names for things
  2. Use HTTP dereferenceable URIs so that people can look up those names.
  3. When someone looks up a URI, provide useful information.
  4. Include links to other URIs so that they can discover more things.

[1] Excellent tutorial here: `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`

Where is the data? 2/4

Obtaining Machine-Readable RDF data
(i) directly by the publishers, (ii) by GRDDL transformations, or (iii) by 3rd-party wrappers:

Where is the data? 2/4

Obtaining Machine-Readable RDF data

(i) directly by the publishers, (ii) by GRDDL transformations, or (iii) by 3rd-party wrappers:

FOAF/RDF linked from a home page: personal data (`foaf:name`, `foaf:phone`, etc.), relationships `foaf:knows`, `rdfs:seeAlso`)

Different Options:

RDFa [Adida *et al.*, 2008][Hausenblas *et al.*, 2008],

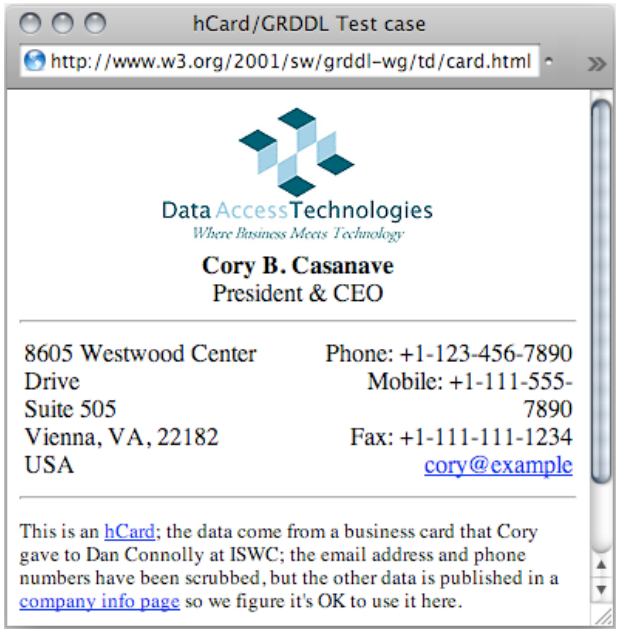
linking RDF/XML [Beckett and McBride (eds.), 2004] from (X)HTML, etc. Let's check,

e.g.`http://www.w3.org/People/Berners-Lee/`, or `http://www.cs.rpi.edu/~hendler/`

Where is the data? 3/4

Obtaining Machine-Readable RDF data
(i) directly by the publishers, (ii) by GRDDL transformations, or (iii) by 3rd-party wrappers:
GRDDL (**G**leaning **R**esource **D**escriptions from **D**ialects of **L**anguages.) [Connolly (ed.), 2007]
Simple principle:

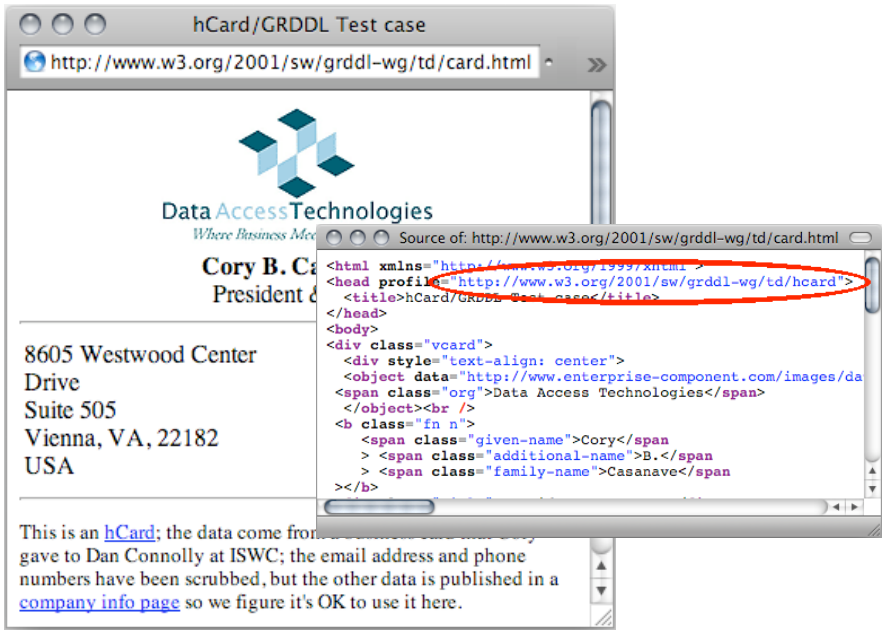
- extract RDF directly from HTML or XML files
- typically using XSLT transformations (other languages: XQuery, XSPARQL, etc.)
- useful for common Microformats, e.g. hCard, hCal:



Where is the data? 3/4

Obtaining Machine-Readable RDF data
(i) directly by the publishers, (ii) by GRDDL transformations, or (iii) by 3rd-party wrappers:
GRDDL (**G**leaning **R**esource **D**escriptions from **D**ialects of **L**anguages.) [Connolly (ed.), 2007]
Simple principle:

- extract RDF directly from HTML or XML files
- typically using XSLT transformations (other languages: XQuery, XSPARQL, etc.)
- useful for common Microformats, e.g. hCard, hCal:

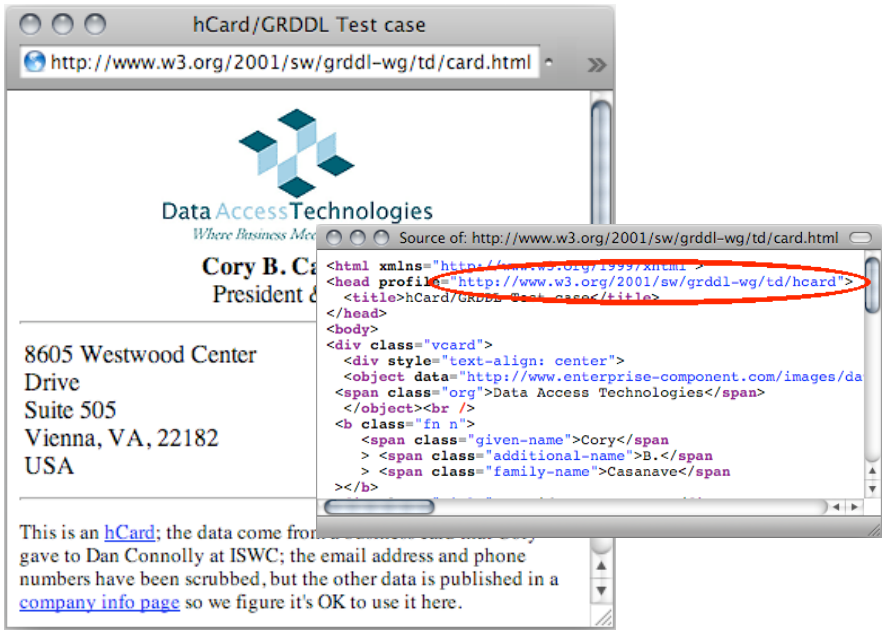


- **profile** `http://www.w3.org/2001/sw/grddl-wg/td/hcard` ...

Where is the data? 3/4

Obtaining Machine-Readable RDF data
(i) directly by the publishers, (ii) by GRDDL transformations, or (iii) by 3rd-party wrappers:
GRDDL (**G**leaning **R**esource **D**escriptions from **D**ialects of **L**anguages.) [Connolly (ed.), 2007]
Simple principle:

- extract RDF directly from HTML or XML files
- typically using XSLT transformations (other languages: XQuery, XSPARQL, etc.)
- useful for common Microformats, e.g. hCard, hCal:

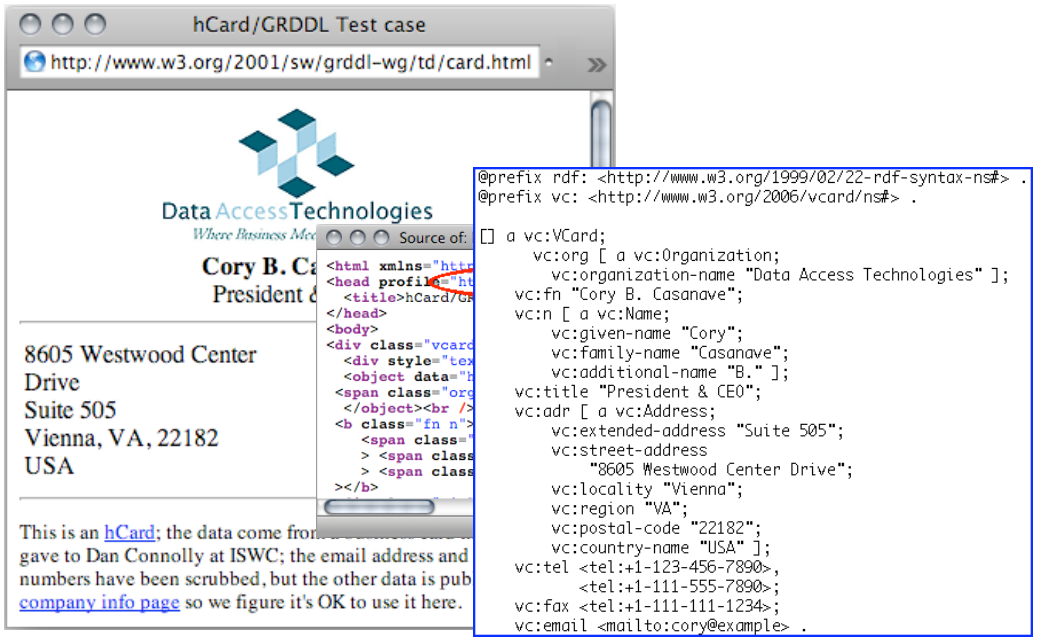


- profile `http://www.w3.org/2001/sw/grddl-wg/td/hcard` ...
- ... points to XSL transformation `http://www.w3.org/2006/vcard/hcard2rdf.xsl`

Where is the data? 3/4

Obtaining Machine-Readable RDF data
(i) directly by the publishers, (ii) by GRDDL transformations, or (iii) by 3rd-party wrappers:
GRDDL (**G**leaning **R**esource **D**escriptions from **D**ialects of **L**anguages.) [Connolly (ed.), 2007]
Simple principle:

- extract RDF directly from HTML or XML files
- typically using XSLT transformations (other languages: XQuery, XSPARQL, etc.)
- useful for common Microformats, e.g. hCard, hCal:



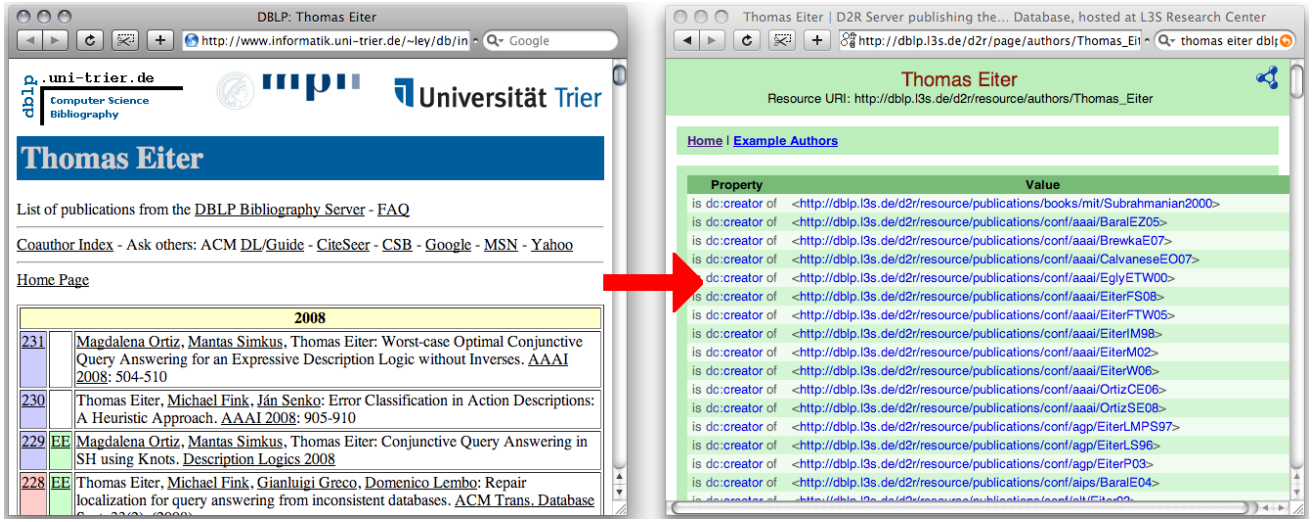
- profile `http://www.w3.org/2001/sw/grddl-wg/td/hcard` ...
- ... points to XSL transformation `http://www.w3.org/2006/vcard/hcard2rdf.xsl`
- ... which – executed on the original HTML file – extracts RDF.

Where is the data? 4/4

Obtaining Machine-Readable RDF data
(i) directly by the publishers, (ii) by GRDDL transformations, or (iii) by 3rd-party wrappers:

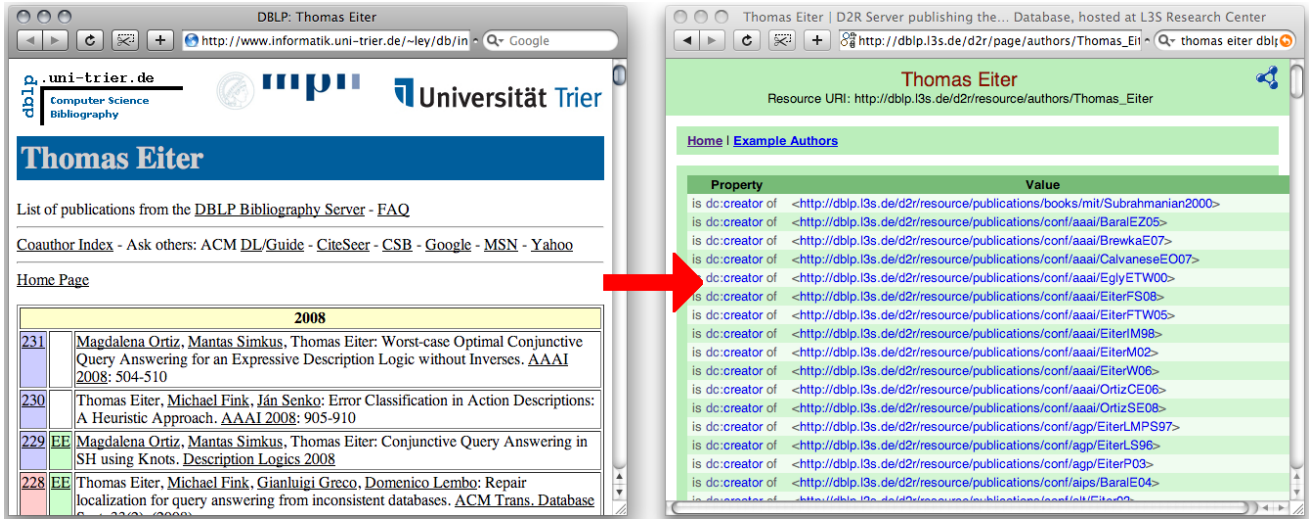
Where is the data? 4/4

Obtaining Machine-Readable RDF data
(i) directly by the publishers, (ii) by GRDDL transformations, or (iii) by 3rd-party wrappers:
L3S' RDF export of the DBLP citation index, see http://dblp.l3s.de/d2r/



Where is the data? 4/4

Obtaining Machine-Readable RDF data
(i) directly by the publishers, (ii) by GRDDL transformations, or (iii) by 3rd-party wrappers:
L3S' RDF export of the DBLP citation index, see http://dblp.l3s.de/d2r/



- Gives unique URIs to authors, documents, etc. on DBLP! E.g.,
  http://dblp.l3s.de/d2r/resource/authors/Thomas_Eiter,
  http://dblp.l3s.de/d2r/resource/authors/Tim_Berners-Lee,
  http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08, etc.
- Provides RDF version of all DBLP data + query interface!
- Other nice example: RDF+query interface for large parts of wikipedia:
  http://dbpedia.org/

How can I query that data? SPARQL

SPARQL – W3C approved standardized query language for RDF:

- look-and-feel of "SQL for the Web"
- allows to ask queries like
- "*All documents created by Thomas Eiter*"
- "*Names of all persons who co-authored with authors of the present paper*"
- "*Names of persons who know Tim Berners-Lee or who are known by Tim Berners-Lee*"
- "*All people who have published in TPLP but have not co-authored with any of the authors of the present paper*"

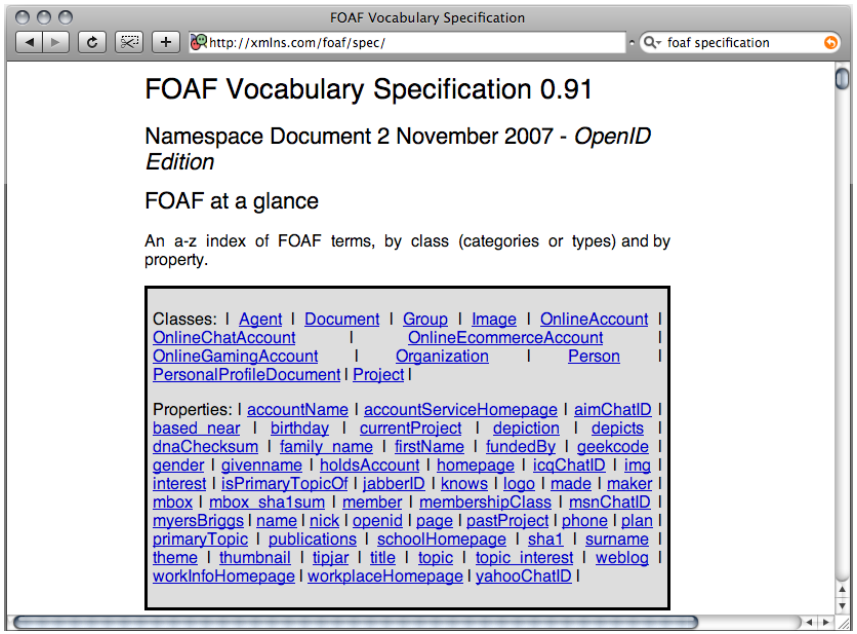
Example (**query1.sparql**):

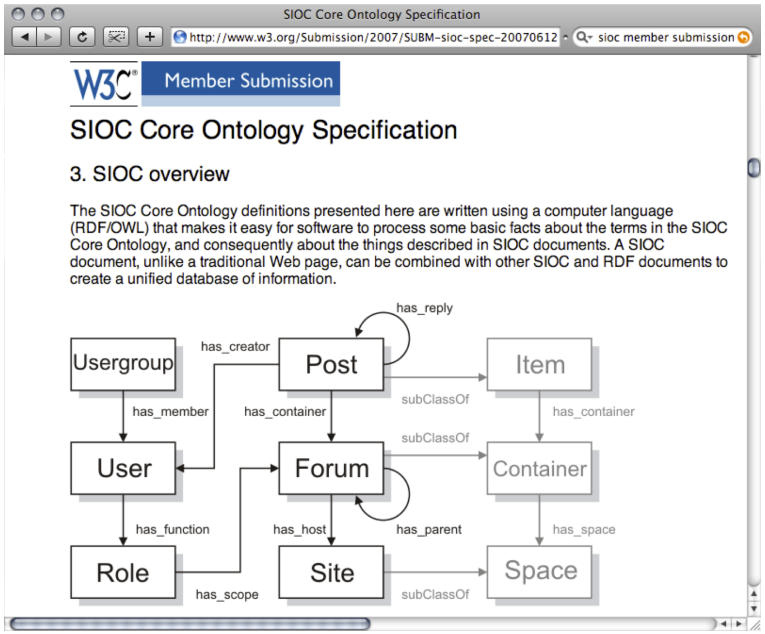
```
SELECT ?D
FROM <http://dblp.l3s.de/d2r/data/authors/Thomas_Eiter>
WHERE {?D dc:creator <http://dblp.l3s.de/d2r/resource/authors/Thomas_Eiter>}
```

What does the data mean?

Data, i.e. the used *vocabulary* to write down RDF is described by *ontologies*,themselves published in RDF, e.g.:

- Friend-of-a-Friend (FOAF) [Brickley and Miller, 2007]
- Socially-Interlinked-Online-Communities (SIOC) [Bojārs *et al.*, 2007]
- Dublin Core [Nilsson *et al.*, 2008]





Unit Outline

Semantic Web Data: The Resource Description Framework (RDF)

- RDF is describing *resources* per triples/statements

  Subject **P**redicate **O**bject.

Semantic Web Data: The Resource Description Framework (RDF)

- RDF is describing *resources* per triples/statements

  Subject **P**redicate **O**bject.

- "simplest possible database schema", data just a set of triples:

Semantic Web Data: The Resource Description Framework (RDF)

- RDF is describing *resources* per triples/statements

  Subject **P**redicate **O**bject.

- "simplest possible database schema", data just a set of triples:

  axel isA Person .
  axel hasName "Axel Polleres".
  axel knows gb .
  axel knows thomas.
  thomas hasCreated an Article
  titled "Rules and Ontologies for the Semantic Web".

Semantic Web Data: The Resource Description Framework (RDF)

- RDF is describing *resources* per triples/statements

  Subject **P**redicate **O**bject.

- "simplest possible database schema", data just a set of triples:

  axel isA Person .
  axel hasName "Axel Polleres".
  axel knows gb .
  axel knows thomas.
  $\exists X$ *thomas hasCreated* X . X *isA Article .*
  X *hasTitle "Rules and Ontologies for the Semantic Web".*

Semantic Web Data: The Resource Description Framework (RDF)

- RDF is describing *resources* per triples/statements

  Subject **P**redicate **O**bject.

- "simplest possible database schema", data just a set of triples:

  axel isA Person .
  axel hasName "Axel Polleres".
  axel knows gb .
  axel knows thomas.
  $\exists X$ *thomas hasCreated* X . X *isA Article .*
  X *hasTitle "Rules and Ontologies for the Semantic Web".*

- abstracts away from tables (RDBMS) or tree-like (XML) schemas

Semantic Web Data: The Resource Description Framework (RDF)

- RDF is describing *resources* per triples/statements

  Subject **P**redicate **O**bject.

- "simplest possible database schema", data just a set of triples:

  axel isA Person .
  axel hasName "Axel Polleres".
  axel knows gb .
  axel knows thomas.
  $\exists X$ *thomas hasCreated* X . X *isA Article .*
  X *hasTitle "Rules and Ontologies for the Semantic Web".*

- abstracts away from tables (RDBMS) or tree-like (XML) schemas
- triples can be viewed as edges of a labeled,directed graph.

Semantic Web Data: The Resource Description Framework (RDF)

- RDF is describing *resources* per triples/statements

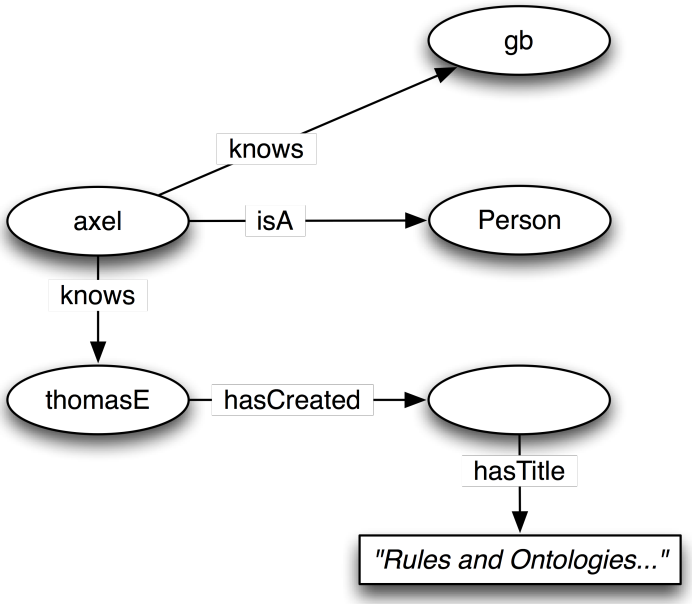
  Subject **P**redicate **O**bject.

- "simplest possible database schema", data just a set of triples:

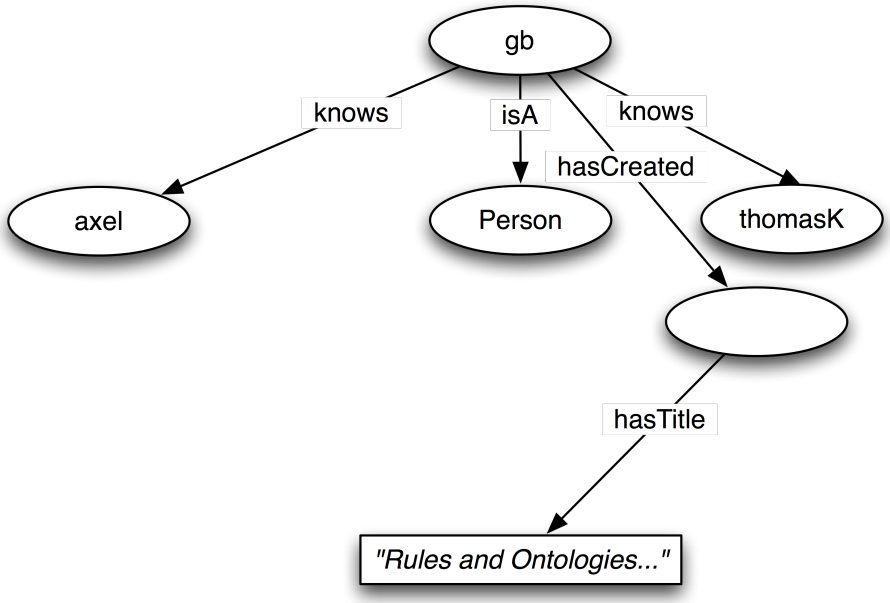
  axel isA Person .
  axel hasName "Axel Polleres".
  axel knows gb .
  axel knows thomas.
  $\exists X$ *thomas hasCreated* X . X *isA Article .*
  X *hasTitle "Rules and Ontologies for the Semantic Web".*

- abstracts away from tables (RDBMS) or tree-like (XML) schemas
- triples can be viewed as edges of a labeled,directed graph.
- main advantage: Graphs are easy to merge! (Trees,Tables aren't)

axel isA Person .
axel knows gb .
axel knows thomasE.
thomasE hasCreated X . X isA Article .
X hasTitle "Rules and Ontologies. . .".



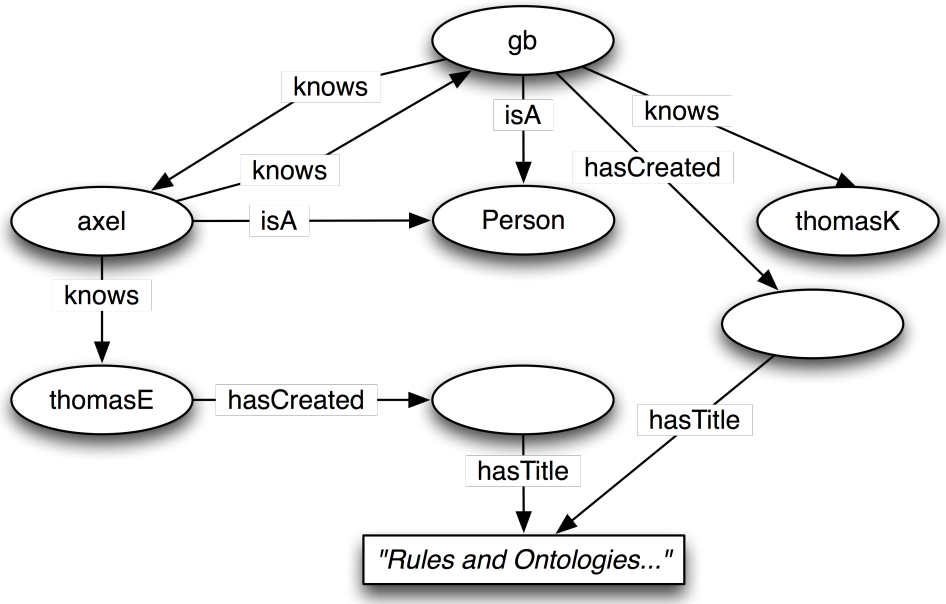
gb isA Person .
gb knows axel .
gb knows thomasK.
gb hasCreated Y . Y isA Article .
Y hasTitle "Rules and Ontologies. . .".



Observe: the "existential variables" became "blank" nodes in the Graph.

axel isA Person .
axel knows gb .
axel knows thomasE.
thomasE hasCreated X . X isA Article .
X hasTitle "Rules and Ontologies. . .".

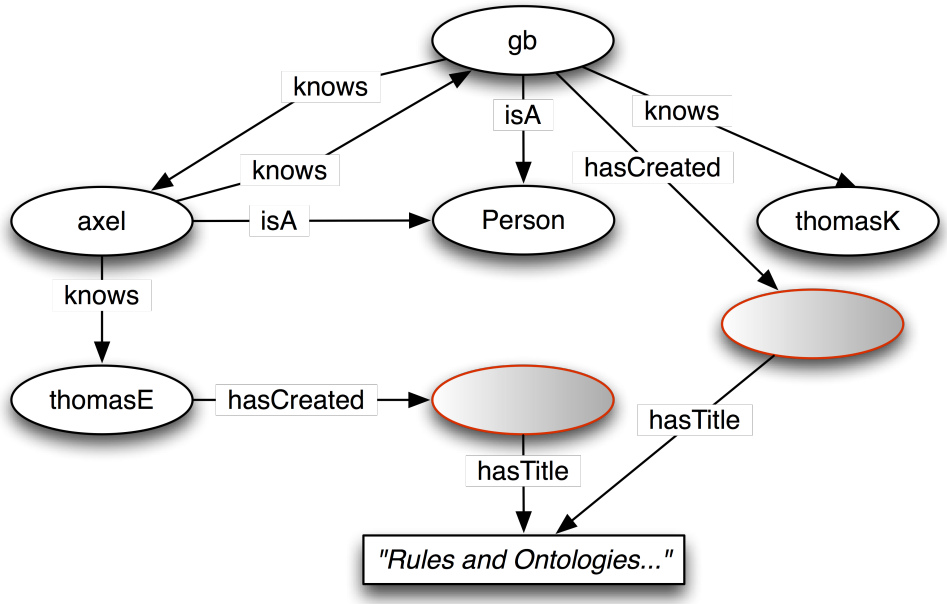
gb isA Person .
gb knows axel .
gb knows thomasK.
gb hasCreated Y . Y isA Article .
Y hasTitle "Rules and Ontologies. . .".



Observe: the "existential variables" became "blank" nodes in the Graph.

axel isA Person .
axel knows gb .
axel knows thomasE.
thomasE hasCreated X . X isA Article .
X hasTitle "Rules and Ontologies. . .".

gb isA Person .
gb knows axel .
gb knows thomasK.
gb hasCreated Y . Y isA Article .
Y hasTitle "Rules and Ontologies. . .".



Observe: the "existential variables" became "blank" nodes in the Graph. Note that we have no reason to assume that the two blank nodes are the same.

A Syntax for RDF: Turtle

There are different syntaxes for RDF

- RDF/XML [Beckett and McBride (eds.), 2004]
- Turtle [Beckett and Berners-Lee, 2008], N3 [Berners-Lee and Connolly, 2008]
- RDFa [Adida *et al.*, 2008] (i.e., RDF "embedded" in (X)HTML)

We'll use Turtle syntax in this lecture:

- it is a subset of Notation 3 [Berners-Lee and Connolly, 2008]
- sufficient to write all RDF
- almost human-readable
- also the basis for SPARQL
- tools and APIs to convert from one syntax into the other, such as `rapper` (part of the Redland API, cf. `http://librdf.org/`), e.g.

```
rapper http://polleres.net/teaching/SemWebTech_2009/testdata/foaf.ttl -i turtle -o rdfxml
```

Resources in RDF, Turtle Syntax

- Resources are identified by URIs (to encourage web-wide unique identifiers)
- There are special URIs, defined in vocabularies (FOAF, SIOC, RDF, etc.)
- Objects can be literals,

axel isA Person .
axel hasName "Axel Polleres".

Resources in RDF, Turtle Syntax

- Resources are identified by URIs (to encourage web-wide unique identifiers)
- There are special URIs, defined in vocabularies (FOAF, SIOC, RDF, etc.)
- Objects can be literals,

axel isA Person .
axel hasName "Axel Polleres".

becomes:

```
<http://polleres.net/foaf.rdf#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
                    <http://xmlns.com/foaf/0.1/Person>.
<http://polleres.net/foaf.rdf#me> <http://xmlns.com/foaf/0.1/name>
                    "Axel Polleres".
```

Resources in RDF, Turtle Syntax

- Resources are identified by URIs (to encourage web-wide unique identifiers)
- There are special URIs, defined in vocabularies (FOAF, SIOC, RDF, etc.)
- Objects can be literals, occasionally with a datatype

axel isA Person .
axel hasName "Axel Polleres".

becomes:

```
<http://polleres.net/foaf.rdf#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
                    <http://xmlns.com/foaf/0.1/Person>.
<http://polleres.net/foaf.rdf#me> <http://xmlns.com/foaf/0.1/name>
                    "Axel Polleres"^^<http://www.w3.org/2001/XMLSchema#string>.
```

Resources in RDF, Turtle Syntax

- Resources are identified by URIs (to encourage web-wide unique identifiers)
- There are special URIs, defined in vocabularies (FOAF, SIOC, RDF, etc.)
- Objects can be literals, occasionally with a datatype

axel isA Person .
axel hasName "Axel Polleres".

becomes:

```
<http://polleres.net/foaf.rdf#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
                  <http://xmlns.com/foaf/0.1/Person>.
<http://polleres.net/foaf.rdf#me> <http://xmlns.com/foaf/0.1/name>
                  "Axel Polleres"^^<http://www.w3.org/2001/XMLSchema#string>.
```

Ugly to read. . . more compact syntaxes like Turtle [Beckett and Berners-Lee, 2008] allow prefix definitions á la XML:

```
@prefix :  <http://polleres.net/foaf.rdf#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix xs:  <http://www.w3.org/2001/XMLSchema#> .
:me rdf:type foaf:Person .
:me foaf:name "Axel Polleres"^^xs:string.
```

More on RDF – Shortcuts in Turtle Syntax

```
@prefix :  <http://polleres.net/foaf.rdf#>
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
:me rdf:type foaf:Person .
:me foaf:name "Axel Polleres" .
:me foaf:knows <http://dblp.l3s.de/d2r/data/authors/Giovambattista_Ianni> .
:me foaf:knows <http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> .
<http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> dc:creator X .
X rdf:type foaf:Document .
X dc:title "Rules and Ontologies for the Semantic Web".
```

More on RDF – Shortcuts in Turtle Syntax

```
@prefix :  <http://polleres.net/foaf.rdf#>
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
:me rdf:type foaf:Person .
:me foaf:name "Axel Polleres" .
:me foaf:knows <http://dblp.l3s.de/d2r/data/authors/Giovambattista_Ianni> .
:me foaf:knows <http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> .
<http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> dc:creator _:X .
_:X rdf:type foaf:Document .
_:X dc:title "Rules and Ontologies for the Semantic Web".
```

- Blank nodes in Turtle are written as _ : *Varname*

More on RDF – Shortcuts in Turtle Syntax

```
@prefix :  <http://polleres.net/foaf.rdf#>
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
:me rdf:type foaf:Person ;
    foaf:name "Axel Polleres" ;
    foaf:knows <http://dblp.l3s.de/d2r/data/authors/Giovambattista_Ianni> ,
               <http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> .
<http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> dc:creator _:X .
_:X rdf:type foaf:Document ;
             dc:title "Rules and Ontologies for the Semantic Web" .
```

- Blank nodes in Turtle are written as _:*Varname*
- Turtle allows shortcuts:
  - Same subject triples can be grouped together with ';' , ','

More on RDF – Shortcuts in Turtle Syntax

```
@prefix :  <http://polleres.net/foaf.rdf#>
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
:me rdf:type foaf:Person;
    foaf:name "Axel Polleres";
    foaf:knows <http://dblp.l3s.de/d2r/data/authors/Giovambattista_Ianni> ,
               <http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> .
<http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> dc:creator [
               rdf:type foaf:Document ;
               dc:title "Rules and Ontologies for the Semantic Web" ] .
```

- Blank nodes in Turtle are written as _:*Varname*
- Turtle allows shortcuts:
  - Same subject triples can be grouped together with ';' , ','
  - Blank nodes can be grouped/replaced using "bracket syntax" '[', ']'

More on RDF – Shortcuts in Turtle Syntax

```
@prefix :  <http://polleres.net/foaf.rdf#>
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
:me a foaf:Person;
    foaf:name "Axel Polleres";
    foaf:knows <http://dblp.l3s.de/d2r/data/authors/Giovambattista_Ianni> ,
               <http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> .
<http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> dc:creator [
               a foaf:Document ;
               dc:title "Rules and Ontologies for the Semantic Web" ] .
```

- Blank nodes in Turtle are written as `_:`*Varname*
- Turtle allows shortcuts:
  - Same subject triples can be grouped together with ';' , ','
  - Blank nodes can be grouped/replaced using "bracket syntax" '[', ']'
  - `rdf:type` is often abbreviated with `a`.

More on RDF – Shortcuts in Turtle Syntax

```
@prefix :  <http://polleres.net/foaf.rdf#>
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
:me  foaf:Person;
     foaf:name "Axel Polleres"^^xs:string;
     foaf:knows <http://dblp.l3s.de/d2r/data/authors/Giovambattista_Ianni> ,
                <http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> .
<http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> dc:creator [
                a foaf:Document ;
                dc:title "Rules and Ontologies for the Semantic Web" ] .
```

- Blank nodes in Turtle are written as _:*Varname*
- Turtle allows shortcuts:
  - Same subject triples can be grouped together with ';' , ','
  - Blank nodes can be grouped/replaced using "bracket syntax" '[', ']'
  - `rdf:type` is often abbreviated with `a`.
  - typed literals l of type dt are written as $l^{\wedge\wedge}dt$.

More on RDF – Shortcuts in Turtle Syntax

```
@prefix :  <http://polleres.net/foaf.rdf#>
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
:me  foaf:Person;
     foaf:name "Axel Polleres"^^xs:string;
     foaf:knows <http://dblp.l3s.de/d2r/data/authors/Giovambattista_Ianni> ,
              <http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> .
<http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> dc:creator [
              a foaf:Document ;
              dc:title "Rules and Ontologies for the Semantic Web"@en ] .
```

- Blank nodes in Turtle are written as _:*Varname*
- Turtle allows shortcuts:
  - Same subject triples can be grouped together with ';' , ','
  - Blank nodes can be grouped/replaced using "bracket syntax" '[', ']'
  - `rdf:type` is often abbreviated with `a`.
  - typed literals l of type dt are written as $l^{\wedge\wedge}dt$.
  - untyped literals can have a language tag [BCP-47, 2006].

More on RDF – Shortcuts in Turtle Syntax

```
@prefix :  <http://polleres.net/foaf.rdf#>
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dc:  <http://purl.org/dc/elements/1.1/> .
:me  foaf:Person;
     foaf:name "Axel Polleres"^^xs:string;
     foaf:knows <http://dblp.l3s.de/d2r/data/authors/Giovambattista_Ianni> ,
                <http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> .
<http://dblp.l3s.de/d2r/page/authors/Thomas_Eiter> dc:creator [
                a foaf:Document ;
                dc:title "Rules and Ontologies for the Semantic Web"@en ] .
```

- Blank nodes in Turtle are written as _ : *Varname*
- Turtle allows shortcuts:
  - Same subject triples can be grouped together with ';' , ','
  - Blank nodes can be grouped/replaced using "bracket syntax" '[', ']'
  - `rdf:type` is often abbreviated with `a`.
  - typed literals l of type dt are written as $l^{\wedge\wedge}dt$.
  - untyped literals can have a language tag [BCP-47, 2006].
  - (untyped literals without a language tag are also called "plain" literals.)

Collecting RDF from the Web

- For us this is enough so far to "read" RDF on the Web.

[2] http://librdf.org/
[3] http://jena.sourceforge.net/

Collecting RDF from the Web

- For us this is enough so far to "read" RDF on the Web.
- For published RDF data there exists a machine-readable XML syntax. Lots of tools and APIs to read/process/convert this data (Redland (C++),[2] Jena (Java),[3] etc.)

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
<http://www.mat.unical.it/~ianni/foaf.rdf> a foaf:PersonalProfileDocument.
<http://www.mat.unical.it/~ianni/foaf.rdf> foaf:maker _:me .
<http://www.mat.unical.it/~ianni/foaf.rdf> foaf:primaryTopic _:me .
_:me a foaf:Person .
_:me foaf:name "Giovambattista Ianni" .
_:me foaf:homepage <http://www.gibbi.com> .
_:me foaf:phone <tel:+39-0984-496430> .
_:me foaf:knows [ a foaf:Person ;
        foaf:name "Wolfgang Faber" ;
        rdfs:seeAlso <http://www.kr.tuwien.ac.at/staff/faber/foaf.rdf>].
_:me foaf:knows [ a foaf:Person .
        foaf:name "Axel Polleres" ;
        rdfs:seeAlso <http://www.polleres.net/foaf.rdf> ].
_:me foaf:knows [ a foaf:Person .
        foaf:name "Thomas Eiter" ] .
_:me foaf:knows [ a foaf:Person .
        foaf:name "Alessandra Martello" ] .
```

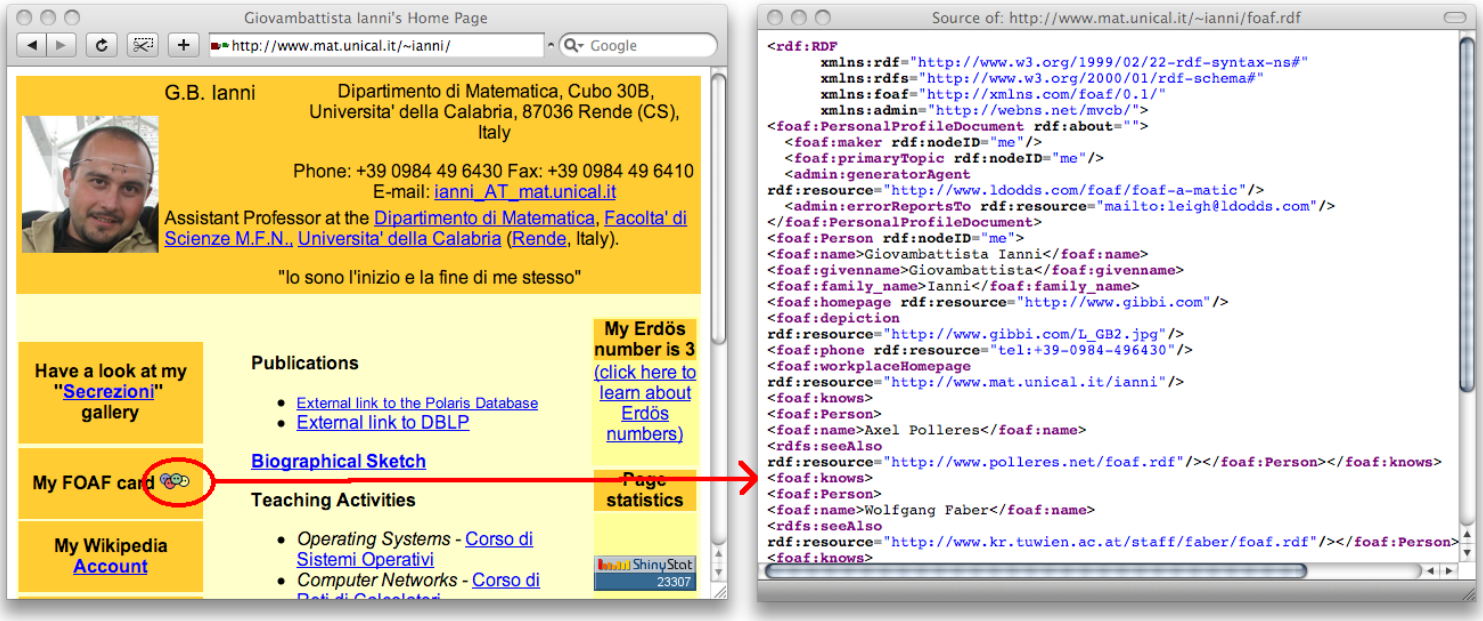
[2] http://librdf.org/

[3] http://jena.sourceforge.net/

Collecting RDF from the Web

- For us this is enough so far to "read" RDF on the Web.
- For published RDF data there exists a machine-readable XML syntax. Lots of tools and APIs to read/process/convert this data (Redland (C++),[2] Jena (Java),[3] etc.)

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
```




```
        foaf:name "Alessandra Martello" ] .
```

[2] http://librdf.org/

[3] http://jena.sourceforge.net/

Unit Outline

How can I query/aggregate RDF data? SPARQL

- First "ingredient": a standardized query language – SPARQL [Prud'hommeaux and Seaborne, 2007] – based on graph pattern matching

Prologue:	P	`PREFIX` *prefix*: <*namespace-URI*>	
Head:	C or	`CONSTRUCT { template }`	. . . construct a new RDF graph
	S or	`SELECT` *variable list*	. . . select matching resources/literals in a graph
	A	`ASK`	. . . boolean query
Body:	D	`FROM` / `FROM NAMED` <*dataset-URI*>	
	W	`WHERE { pattern }`	
	M	`ORDER BY` *expression*	
		`LIMIT` *integer* > 0	
		`OFFSET` *integer* > 0	

How can I query/aggregate RDF data? SPARQL

- First "ingredient": a standardized query language – SPARQL [Prud'hommeaux and Seaborne, 2007] – based on graph pattern matching

Prologue:	P	`PREFIX` *prefix*: <*namespace-URI*>	
Head:	C or	`CONSTRUCT { template }`	. . . construct a new RDF graph
	S or	`SELECT` *variable list*	. . . select matching resources/literals in a graph
	A	`ASK`	. . . boolean query
Body:	D	`FROM` / `FROM NAMED` <*dataset-URI*>	
	W	`WHERE { pattern }`	
	M	`ORDER BY` *expression*	
		`LIMIT` *integer* > 0	
		`OFFSET` *integer* > 0	

- Let us start with `SELECT` queries and focus on the different patterns:
  - basic graph patterns (Conjunctive queries)
  - `FILTER`s
  - `UNION`s of patterns
  - `OPTIONAL` Patterns
  - `GRAPH` Patterns

Basic Graph Patterns (Conjunctive queries)

"select all names of persons known by G.B. from his FOAF file"

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

SELECT ?N
FROM <http://www.mat.unical.it/~ianni/foaf.rdf>
WHERE {
       <http://www.mat.unical.it/~ianni/foaf.rdf#me> foaf:knows ?X .
       ?X a foaf:Person .  ?X foaf:name ?N .
      }
```

- graph patterns (`WHERE` part) allow Turtle syntax

[4] We assume here that the only people declared "known" in G.B.'s FOAF file are those known by him.

Basic Graph Patterns (Conjunctive queries)

"select all names of persons known by G.B. from his FOAF file"

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

SELECT ?N
FROM <http://www.mat.unical.it/~ianni/foaf.rdf>
WHERE {
        [ foaf:knows
          [ a foaf:Person; foaf:name ?N ]]
       }
```

- graph patterns (`WHERE` part) allow Turtle syntax
- all Turtle shortcuts allowed[4]

[4] We assume here that the only people declared "known" in G.B.'s FOAF file are those known by him.

Basic Graph Patterns (Conjunctive queries)

"select all names of persons known by G.B., Axel, and Thomas K. from their FOAF files"

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

SELECT ?N
FROM <http://www.mat.unical.it/~ianni/foaf.rdf>
FROM <http://www.polleres.net/foaf.rdf>
FROM <http://www.postsubmeta.net/foaf>
WHERE {
       [ foaf:knows
         [ a foaf:Person; foaf:name ?N ]]
      }
```

- graph patterns (`WHERE` part) allow Turtle syntax
- all Turtle shortcuts allowed[4]
- merge of several graphs can be queried at once

[4] We assume here that the only people declared "known" in G.B.'s FOAF file are those known by him.

Basic Graph Patterns (Conjunctive queries)

"select all names of persons known by G.B., Axel, and Thomas K. from their FOAF files"

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

SELECT ?N
WHERE {
       [ foaf:knows
         [ a foaf:Person; foaf:name ?N ]
       }
```

- graph patterns (`WHERE` part) allow Turtle syntax
- all Turtle shortcuts allowed[4]
- merge of several graphs can be queried at once
- Try it! E.g. using ARQ (`http://jena.sourceforge.net/ARQ/`)

```
arq -query
http://www.polleres.net/teaching/SemWebTech_2009/testdata/query2.sparql
```

[4] We assume here that the only people declared "known" in G.B.'s FOAF file are those known by him.

FILTERs in Basic Graph Patterns

"select all names of persons known by GB, Thomas, and Axel from their FOAF files" (`query3.sparql`)

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

SELECT ?N
WHERE {
       [ foaf:knows
         [a foaf:Person ; foaf:name ?N] ]
      }
```

- graph patterns (`WHERE` part) allow Turtle syntax
- all Turtle shortcuts allowed
- Dataset can also be implicit, depending on the implementation...
  so, let's assume we have a Web crawl of FOAF data ...

FILTERs in Basic Graph Patterns

"select all names of persons known by GB, Thomas, and Axel from their FOAF files" (`query3.sparql`)

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

SELECT ?N
WHERE {
       [ foaf:knows
         [a foaf:Person ; foaf:name ?N] ]
        FILTER ( ?N != "Giovambattista Ianni" &&
                 ?N != "Thomas Krennwallner" && ?N != "Axel Polleres")
      }
```

- graph patterns (`WHERE` part) allow Turtle syntax
- all Turtle shortcuts allowed
- Dataset can also be implicit, depending on the implementation. . .
  so, let's assume we have a Web crawl of FOAF data . . .
- . . . i.e., we have to filter out the authors' names from the result.

UNIONs

"Names of persons who know Axel Polleres or who are known by Axel Polleres"

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

SELECT ?N
FROM ...
WHERE {
       { [ foaf:name "Axel Polleres" ] foaf:knows [foaf:name ?N ] }
       UNION
       { [ foaf:name ?N ] foaf:knows [foaf:name "Axel Polleres" ] }
      }
```

UNIONs

"Names of persons who know Axel Polleres or who are known by Axel Polleres"

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

SELECT ?N
FROM ...
WHERE {
       { [ foaf:name "Axel Polleres" ] foaf:knows [foaf:name ?N ] }
       UNION
       { [ foaf:name ?N ] foaf:knows [foaf:name "Axel Polleres" ] }
      }
```

- UNION s allow alternative matching of several patterns, similar to UNIONs in SQL.

OPTIONALs 1/2 – Partial Matching

"Select all names of persons known by Axel from his FOAF file and – if available – their `rdfs:seeAlso` links" *`query4.sparql`*

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?N ?L
FROM <http://www.polleres.net/foaf.rdf>
WHERE {<http://www.www.polleres.net/foaf.rdf#me> foaf:knows ?X .
       ?X foaf:name ?N . ?X rdfs:seeAlso ?L
      }
```

- "Normal" basic graph pattern doesn't work here, returns only those `?X` with a `rdfs:seeAlso` link.

?N	?L
"Dan Brickley" "Ruben Lara Hernandez" ...	<http://danbri.org/foaf.rdf> <http://nets.ii.uam.es/rlara/foaf.rdf>

OPTIONALs 1/2 – Partial Matching

"Select all names of persons known by Axel from his FOAF file and – if available – their `rdfs:seeAlso` links" *`query4.sparql`*

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?N ?L
FROM <http://www.polleres.net/foaf.rdf>
WHERE {<http://www.www.polleres.net/foaf.rdf#me> foaf:knows ?X .
       ?X foaf:name ?N .  OPTIONAL  { ?X rdfs:seeAlso ?L }
      }
```

- "Normal" basic graph pattern doesn't work here, returns only those `?X` with a `rdfs:seeAlso` link.
- `OPTIONAL` allows partial variable bindings in the solutions.

?N	?L
"Dan Brickley" "Ruben Lara Hernandez" ... "Thomas Eiter" ...	<http://danbri.org/foaf.rdf> <http://nets.ii.uam.es/rlara/foaf.rdf>

OPTIONALs 2/2 – Set difference

"Select all names of persons known by Axel from his FOAF file who don't have a `rdfs:seeAlso` links" *`query5.sparql`*

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?N
FROM <http://www.polleres.net/foaf.rdf>
WHERE {<http://www.polleres.net/foaf.rdf#me> foaf:knows ?X .
       ?X foaf:name ?N . OPTIONAL { ?X rdfs:seeAlso ?L }
       FILTER ( !  bound(?L) )
      }
```

- `OPTIONAL` allows partial variable bindings in the solutions.

OPTIONALs 2/2 – Set difference

"Select all names of persons known by Axel from his FOAF file who don't have a `rdfs:seeAlso` links" *`query5.sparql`*

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?N
FROM <http://www.polleres.net/foaf.rdf>
WHERE {<http://www.polleres.net/foaf.rdf#me> foaf:knows ?X .
       ?X foaf:name ?N . OPTIONAL { ?X rdfs:seeAlso ?L }
       FILTER ( !  bound(?L) )
      }
```

- `OPTIONAL` allows partial variable bindings in the solutions.
- The negated bound() function in the `FILTER` allows to suppress unbound values.

?N
"Alexandre Passant"
"Manfred Pfeiffenberger"
"Thomas Eiter"

OPTIONALs 2/2 – Set difference

"Select all names of persons known by Axel from his FOAF file who don't have a `rdfs:seeAlso` links" *`query5.sparql`*

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?N
FROM <http://www.polleres.net/foaf.rdf>
WHERE {<http://www.polleres.net/foaf.rdf#me> foaf:knows ?X .
       ?X foaf:name ?N . OPTIONAL { ?X rdfs:seeAlso ?L }
       FILTER ( !  bound(?L) )
      }
```

- `OPTIONAL` allows partial variable bindings in the solutions.
- The negated bound() function in the `FILTER` allows to suppress unbound values.
- This is similar to set difference (NOT EXISTS) in SQL or "negation as failure" in Logic Programming rules.

?N
"Alexandre Passant"
"Manfred Pfeiffenberger"
"Thomas Eiter"

OPTIONALs 2/2 – Set difference

"Select all names of persons known by Axel from his FOAF file who don't have a `rdfs:seeAlso` links" *`query5.sparql`*

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?N
FROM <http://www.polleres.net/foaf.rdf>
WHERE {<http://www.polleres.net/foaf.rdf#me> foaf:knows ?X .
       ?X foaf:name ?N . OPTIONAL { ?X rdfs:seeAlso ?L }
       FILTER ( !  bound(?L) )
      }
```

- `OPTIONAL` allows partial variable bindings in the solutions.
- The negated bound() function in the `FILTER` allows to suppress unbound values.
- This is similar to set difference (NOT EXISTS) in SQL or "negation as failure" in Logic Programming rules.
- Many more useful `FILTER` functions available in SPARQL

?N
"Alexandre Passant"
"Manfred Pfeiffenberger"
"Thomas Eiter"

GRAPH patterns

"Select all names of persons directly known by Axel or the names of persons appearing in the files linked by `rdfs:seeAlso` links."

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?N
FROM <http://www.polleres.net/foaf.rdf>
WHERE {<http://www.polleres.net/foaf.rdf#me> foaf:knows ?X .
       { { ?X foaf:name ?N .}
         UNION
         { ?X rdfs:seeAlso ?L . GRAPH  ?L{ [a foaf:Person ] foaf:name ?N } }
      }
```

- named GRAPH patterns allow to match pattern in remote graphs

GRAPH patterns

"Select all names of persons directly known by Axel or the names of persons appearing in the files linked by `rdfs:seeAlso` links."

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?N
FROM <http://www.polleres.net/foaf.rdf>
FROM NAMED???
WHERE {<http://www.polleres.net/foaf.rdf#me> foaf:knows ?X .
       { { ?X foaf:name ?N .}
         UNION
          { ?X rdfs:seeAlso ?L . GRAPH  ?L{ [a foaf:Person ] foaf:name ?N } }
      }
```

- named `GRAPH` patterns allow to match pattern in remote graphs
- the set of named graphs [Carroll *et al.*, 2005] typically needs to be statically declared in the dataset in current SPARQL implementations (`FROM NAMED` clause), details see [Prud'hommeaux and Seaborne, 2007], i.e. most SPARQL engines will not deliver the "expected" result here.

GRAPH patterns

"Select all names of persons directly known by Axel or the names of persons appearing in the files linked by `rdfs:seeAlso` links."

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>

SELECT ?N
FROM <http://www.polleres.net/foaf.rdf>
WHERE {<http://www.polleres.net/foaf.rdf#me> foaf:knows ?X .
       { { ?X foaf:name ?N .}
         UNION
         { ?X rdfs:seeAlso ?L . GRAPH  ?L{ [a foaf:Person ] foaf:name ?N } }
      }
```

- named `GRAPH` patterns allow to match pattern in remote graphs
- the set of named graphs [Carroll *et al.*, 2005] typically needs to be statically declared in the dataset in current SPARQL implementations (`FROM NAMED` clause), details see [Prud'hommeaux and Seaborne, 2007], i.e. most SPARQL engines will not deliver the "expected" result here.
- version with "explicit" listing of named graphs, cf. `query6.sparql`, shows some limits of SPARQL on real Web data...

CONSTRUCT

CONSTRUCT queries in SPARQL allow to generate new RDF graphs from the results of a query, e.g.

> *"Create a graph which establishes 'foaf:knows relations for all persons who I have co-authored with according to DBLP." (query7.sparql)*

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
PREFIX: <http://dblp.l3s.de/d2r/resource/authors/>

CONSTRUCT  { <http://polleres.net/foaf.rdf#me> foaf:knows ?Y }
WHERE { ?D dc:creator :Axel_Polleres;
        dc:creator ?Y . FILTER( ?Y != :Axel_Polleres )
      }
```

CONSTRUCT

CONSTRUCT queries in SPARQL allow to generate new RDF graphs from the results of a query, e.g.

> *"Create a graph which establishes 'foaf:knows relations for all persons who I have co-authored with according to DBLP." (query7.sparql)*

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
PREFIX: <http://dblp.l3s.de/d2r/resource/authors/>

CONSTRUCT  { <http://polleres.net/foaf.rdf#me> foaf:knows ?Y }
WHERE { ?D dc:creator :Axel_Polleres;
        dc:creator ?Y . FILTER( ?Y != :Axel_Polleres )
      }
```

- "Output pattern" is a basic graph pattern
- similar to "views" in SQL
- May be viewed as a "rules language" itself.

ASK

ASK queries are "yes/no" queries without explicit output, e.g.

> *"Does Axel know one of the co-authors of <http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08>?"*

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>

ASK
FROM <http://polleres.net/foaf.rdf>
FROM <http://dblp.l3s.de/d2r/data/publications/journals/tplp/Berners-LeeCKSH08>
WHERE { <http://polleres.net/foaf.rdf#me> foaf:knows ?A .
        <http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08>
                                                  dc:creator ?A }
```

ASK

ASK queries are "yes/no" queries without explicit output, e.g.

> *"Does Axel know one of the co-authors of <http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08>?"*

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>

ASK
FROM <http://polleres.net/foaf.rdf>
FROM <http://dblp.l3s.de/d2r/data/publications/journals/tplp/Berners-LeeCKSH08>
WHERE { <http://polleres.net/foaf.rdf#me> foaf:knows ?A .
        <http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08>
                                                    dc:creator ?A }
```

Interestingly, this query returns "no"... why? Because SPARQL doesn't know that

- <http://dblp.l3s.de/d2r/resource/authors/Jim_Hendler> = <http://www.cs.rpi.edu/ hendler/foaf.rdf#jhendler>

ASK

ASK queries are "yes/no" queries without explicit output, e.g.

> *"Does Axel know one of the co-authors of*
> *<http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08>?"*

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>

ASK
FROM <http://polleres.net/foaf.rdf>
FROM <http://dblp.l3s.de/d2r/data/publications/journals/tplp/Berners-LeeCKSH08>
WHERE { <http://polleres.net/foaf.rdf#me> foaf:knows ?A .
        <http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08>
                                                    dc:creator ?A }
```

Interestingly, this query returns "no"... why? Because SPARQL doesn't know that

- `<http://dblp.l3s.de/d2r/resource/authors/Jim_Hendler>` = `<http://www.cs.rpi.edu/ hendler/foaf.rdf#jhendler>`

although, in `http://polleres.net/foaf.rdf` there is a triple:

`http://polleres.net/foaf.rdf#me> foaf:knows <http://www.cs.rpi.edu/ hendler/foaf.rdf#jhendler>`

ASK

ASK queries are "yes/no" queries without explicit output, e.g.

> *"Does Axel know one of the co-authors of* *<http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08>?"*

```
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>

ASK
FROM <http://polleres.net/foaf.rdf>
FROM <http://dblp.l3s.de/d2r/data/publications/journals/tplp/Berners-LeeCKSH08>
WHERE { <http://polleres.net/foaf.rdf#me> foaf:knows ?A .
        <http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08>
                                                    dc:creator ?A }
```

Interestingly, this query returns "no"... why? Because SPARQL doesn't know that

- `<http://dblp.l3s.de/d2r/resource/authors/Jim_Hendler>` = `<http://www.cs.rpi.edu/ hendler/foaf.rdf#jhendler>`

although, in `http://polleres.net/foaf.rdf` there is a triple:

`http://polleres.net/foaf.rdf#me> foaf:knows <http://www.cs.rpi.edu/ hendler/foaf.rdf#jhendler>`

More on that later...

Exercise

Using the SPARQL interface to DBLP at `http://dblp.l3s.de/d2r/snorql/` write a query that outputs the following:

Task

Names of people who have published in TPLP but have not co-authored with any of the authors of

`http://dblp.l3s.de/d2r/resource/publications/journals/tplp/Berners-LeeCKSH08`

- Can you do it in one query?
- Which of the constructs discussed do you need?

SPARQL summary

- We have only "scratched the surface" here
- Extensions of SPARQL (updates (DELETE, INSERT, ...), aggregate functions (SUM, MAX, COUNT,...), etc.) currently being discussed in W3C, e.g. [esw-wiki,]
- Rigid investigation of SPARQL's semantics and complexity [Pérez *et al.*, 2006; Gutiérrez *et al.*, 2004]
- Peculiarities in SPARQL's semantics (multiset semantics, joins over unbound variables, etc. [Prud'hommeaux and Seaborne, 2007])
- SPARQL itself may be viewed as a "rules language" (`CONSTRUCT`): Translation of SPARQL to rules [Schenk and Staab, 2008][Polleres, 2007]
- SPARQL only does RDF graph pattern matching, what about ontologies? . . . Let's take a look at this next!

Unit Outline

What does RDF data mean?

- *Ontologies* are formal descriptions of what the *vocabulary* used in an RDF document means.

[5] "data" rather than "ontology", in DL terminology this distinction is often called ABox vs. TBox.

What does RDF data mean?

- *Ontologies* are formal descriptions of what the *vocabulary* used in an RDF document means.
- By vocabulary, we mean here mostly:
  - *properties*,i.e., predicates
  - *classes*, i.e., objects of `rdf:type` triples
  - (*individuals*, i.e., concrete objects)[5]

[5] "data" rather than "ontology", in DL terminology this distinction is often called ABox vs. TBox.

What does RDF data mean?

- *Ontologies* are formal descriptions of what the *vocabulary* used in an RDF document means.
- By vocabulary, we mean here mostly:
  - *properties*,i.e., predicates
  - *classes*, i.e., objects of `rdf:type` triples
  - (*individuals*, i.e., concrete objects)[5]
- Ontologies describe ***relations*** among properties, classes and individuals (subclasses, subproperties, equivalence, domain, range, etc.)

[5] "data" rather than "ontology", in DL terminology this distinction is often called ABox vs. TBox.

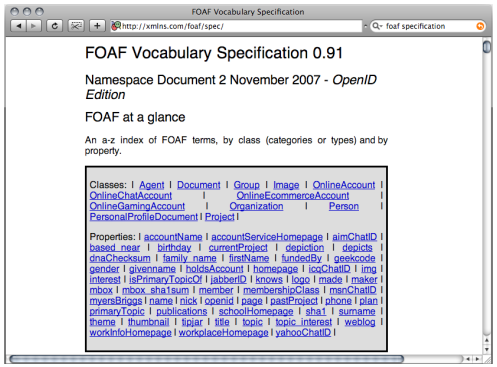
What does RDF data mean?

- *Ontologies* are formal descriptions of what the *vocabulary* used in an RDF document means.
- By vocabulary, we mean here mostly:
  - *properties*,i.e., predicates
  - *classes*, i.e., objects of `rdf:type` triples
  - (*individuals*, i.e., concrete objects)[5]
- Ontologies describe ***relations*** among properties, classes and individuals (subclasses, subproperties, equivalence, domain, range, etc.)
- The W3C has published two standards to describe ontologies, namely *RDF Schema (RDFS)* [Brickley and Guha (eds.), 2004] and the *Web Ontology language (OWL)* [Patel-Schneider *et al.*, 2004]
  - RDFS . . . simple schema language with minimal expressivity, mostly expressible in simple forward chaining inference rules (*Horn Rules*)
  - OWL . . . higher expressivity, foundations in *Description Logics*
  - both RDFS and OWL ontologies are RDF graphs themselves, i.e., OWL and RDFS provide "an RDF vocabulary to describe RDF vocabularies"

[5] "data" rather than "ontology", in DL terminology this distinction is often called ABox vs. TBox.

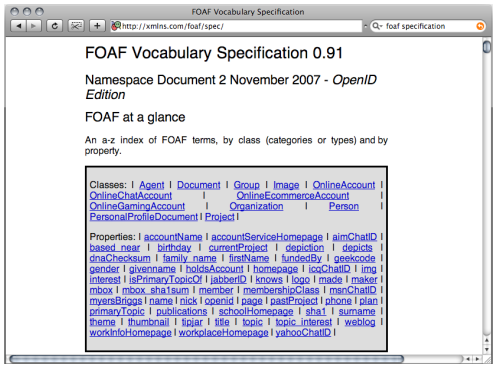
Example Vocabulary 1 – The FOAF ontology:

- **Properties:** `name`, `knows`, `homepage`, `primaryTopic` etc.
- **Classes:** `Person`, `Agent`, `Document`, `Organisation`, etc.
- **Relations:** e.g.
  - *Each `Person` is a `Agent`* (subclass)



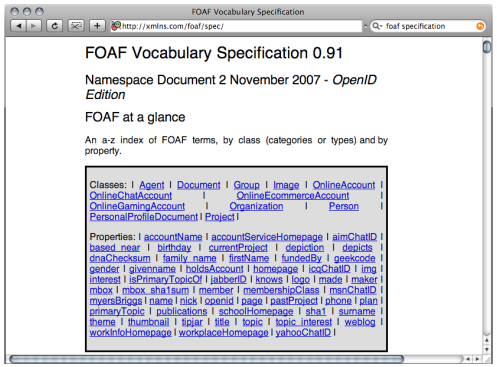
Example Vocabulary 1 – The FOAF ontology:

- **Properties:** `name`, `knows`, `homepage`, `primaryTopic` etc.
- **Classes:** `Person`, `Agent`, `Document`, `Organisation`, etc.
- **Relations:** e.g.
  - *Each `Person` is a `Agent`* (subclass)
  - *The `img` property is more specific than `depiction`* (subproperty)



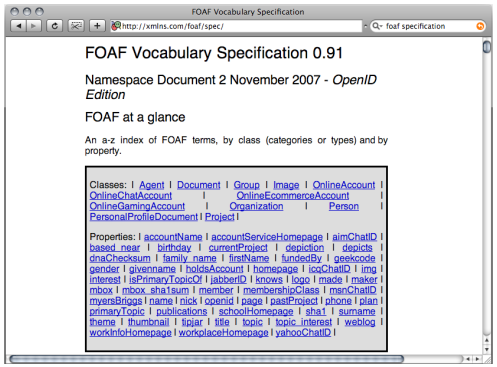
Example Vocabulary 1 – The FOAF ontology:

- **Properties:** `name`, `knows`, `homepage`, `primaryTopic` etc.
- **Classes:** `Person`, `Agent`, `Document`, `Organisation`, etc.
- **Relations:** e.g.
  - *Each `Person` is a `Agent`* (subclass)
  - *The `img` property is more specific than `depiction`* (subproperty)
  - *`img` is a relation between `Persons` and `Imgages`* (domain/range)



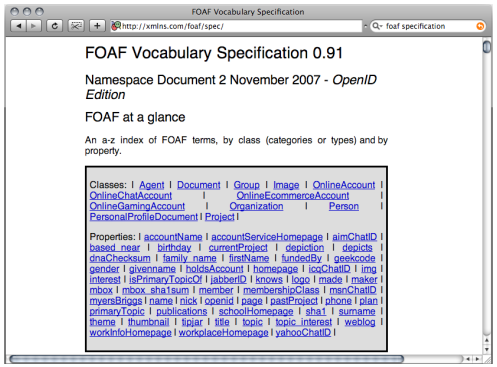
Example Vocabulary 1 – The FOAF ontology:

- **Properties:** `name`, `knows`, `homepage`, `primaryTopic` etc.
- **Classes:** `Person`, `Agent`, `Document`, `Organisation`, etc.
- **Relations:** e.g.
  - *Each `Person` is a `Agent`* (subclass)
  - *The `img` property is more specific than `depiction`* (subproperty)
  - *`img` is a relation between `Persons` and `Imgages`* (domain/range)
  - *`knows` is a relation between two `Persons`* (domain/range)



Example Vocabulary 1 – The FOAF ontology:

- **Properties:** `name`, `knows`, `homepage`, `primaryTopic` etc.
- **Classes:** `Person`, `Agent`, `Document`, `Organisation`, etc.
- **Relations:** e.g.
  - *Each `Person` is a `Agent`* (subclass)
  - *The `img` property is more specific than `depiction`* (subproperty)
  - *`img` is a relation between `Persons` and `Imgages`* (domain/range)
  - *`knows` is a relation between two `Persons`* (domain/range)
  - *`homepage` denotes **unique** homepage of an `Agent`* (uniquely identifying property)



⋮

Examples 2 – A simple ontology about reviewers:

- **Properties:** `title`, `isAuthorOf`, `publishedIn`, etc.
- **Classes:** `Senior`, `Paper`, `Publication`, etc.
- **Relations:**
  - *A `Publication` is a `Paper` which has been `published`* (subclass + existential condition on property)

[6] reuse of external ontologies!

Examples 2 – A simple ontology about reviewers:

- **Properties:** `title`, `isAuthorOf`, `publishedIn`, etc.
- **Classes:** `Senior`, `Paper`, `Publication`, etc.
- **Relations:**
  - *A `Publication` is a `Paper` which has been `published`* (subclass + existential condition on property)
  - *`isAuthorOf` is the opposite of Dublin Core's `dc:creator` Property*[6]

[6] reuse of external ontologies!

Examples 2 – A simple ontology about reviewers:

- **Properties:** `title`, `isAuthorOf`, `publishedIn`, etc.
- **Classes:** `Senior`, `Paper`, `Publication`, etc.
- **Relations:**
  - *A `Publication` is a `Paper` which has been `published`* (subclass + existential condition on property)
  - *`isAuthorOf` is the opposite of Dublin Core's `dc:creator` Property*[6]
  - *A `Senior` researcher is a `foaf:Person` who `isAuthorOf` 10+ `Publications`* (subclass + condition on cardinality)

[6] reuse of external ontologies!

Examples 2 – A simple ontology about reviewers:

- **Properties:** `title`, `isAuthorOf`, `publishedIn`, etc.
- **Classes:** `Senior`, `Paper`, `Publication`, etc.
- **Relations:**
  - *A `Publication` is a `Paper` which has been `published`* (subclass + existential condition on property)
  - *`isAuthorOf` is the opposite of Dublin Core's `dc:creator` Property*[6]
  - *A `Senior` researcher is a `foaf:Person` who `isAuthorOf` 10+ `Publications`* (subclass + condition on cardinality)
  - *Each item can be `publishedIn` at most one venue* (functional property)

⋮

[6] reuse of external ontologies!

RDF(S) vocabulary: RDF and RDFS themselves are vocabularies!

- **Properties:** `rdf:type`, `rdfs:domain`, `rdfs:range`, `rdf:subClassOf`, `rdf:subPropertyOf`, `rdf:first`, `rdf:rest` etc.
- **Classes:** `rdf:XMLLiteral`, `rdf:Literal`, `rdfs:Resource`, `rdfs:Property`, `rdfs:Class`, `rdf:List`, etc.
- **Relations:**

RDF(S) vocabulary: RDF and RDFS themselves are vocabularies!

- **Properties:** `rdf:type`, `rdfs:domain`, `rdfs:range`, `rdf:subClassOf`, `rdf:subPropertyOf`, `rdf:first`, `rdf:rest` etc.
- **Classes:** `rdf:XMLLiteral`, `rdf:Literal`, `rdfs:Resource`, `rdfs:Property`, `rdfs:Class`, `rdf:List`, etc.
- **Relations:** The semantics of the RDFs vocabulary is defined in [Hayes, 2004]; it is a "meta vocabulary" used to define the semantics of other vocabularies

The Semantics of RDF graphs:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://www.mat.unical.it/~ianni/foaf.rdf> a foaf:PersonalProfileDocument.
<http://www.mat.unical.it/~ianni/foaf.rdf> foaf:maker _:me .
<http://www.mat.unical.it/~ianni/foaf.rdf> foaf:primaryTopic _:me .
_:me a foaf:Person .
_:me foaf:name "Giovambattista Ianni" .
_:me foaf:homepage <http://www.gibbi.com> .
_:me foaf:phone <tel:+39-0984-496430> .
_:me foaf:knows [ a foaf:Person ;
                  foaf:name "Wolfgang Faber" ;
                  rdfs:seeAlso <http://www.kr.tuwien.ac.at/staff/faber/foaf.rdf>].
_:me foaf:knows [ a foaf:Person .
                  foaf:name "Axel Polleres" ;
                  rdfs:seeAlso <http://www.polleres.net/foaf.rdf>].
_:me foaf:knows [ a foaf:Person .
                  foaf:name "Thomas Eiter" ] .
_:me foaf:knows [ a foaf:Person .
                  foaf:name "Alessandra Martello" ] .
```

The Semantics of RDF graphs:

As we will see in the next Units, each RDF graph can "roughly" be viewed as a first-order formula:

$$
\begin{aligned}
&\exists me, b1, b2, b3, b4 \\
&(triple(\texttt{foaf.rdf}, \texttt{rdf:type}, \texttt{PersonalProfileDocument}) \\
&\wedge\ triple(\texttt{foaf.rdf}, \texttt{maker}, me) \\
&\wedge\ triple(\texttt{foaf.rdf}, \texttt{primaryTopic}, me) \\
&\wedge\ triple(me, \texttt{rdf:type}, \texttt{Person}) \\
&\wedge\ triple(me, \texttt{name}, \texttt{"Giovambattista Ianni"}) \\
&\wedge\ triple(me, \texttt{homepage}, \texttt{http://www.gibbi.com}) \\
&\wedge\ triple(me, \texttt{phone}, \texttt{tel:+39-0984-496430}) \\
&\wedge\ triple(me, \texttt{knows}, b2) \wedge\ triple(b1, \texttt{type}, \texttt{Person}) \\
&\wedge\ triple(b1, \texttt{name}, \texttt{"Wolfgang Faber"}) \\
&\wedge\ triple(b1, \texttt{rdfs:seeAlso}, \texttt{http://www.kr.tuwien...}) \\
&\wedge\ triple(me, \texttt{knows}, b1) \wedge\ triple(b1, \texttt{rdf:type}, \texttt{Person}) \\
&\wedge\ triple(b2, \texttt{name}, \texttt{"Axel Polleres"}) \\
&\wedge\ triple(b2, \texttt{rdfs:seeAlso}, \texttt{http://www.polleres...}) \\
&\wedge\ triple(me, \texttt{knows}, b3) \wedge\ triple(b1, \texttt{rdf:type}, \texttt{Person}) \\
&\wedge\ triple(b3, \texttt{name}, \texttt{"Thomas Eiter"}) \\
&\wedge\ triple(me, \texttt{knows}, b4) \wedge\ triple(b1, \texttt{type}, \texttt{Person}) \\
&\wedge\ triple(b4, \texttt{name}, \texttt{"Alessandra Martello"}))
\end{aligned}
$$

The Semantics of RDF graphs:

Alternatively, especially the OWL favors unary/binary predicate representation:

$$\exists me, b1, b2, b3, b4 \ (\texttt{PersonalProfileDocument}(\texttt{foaf.rdf}) \\ \wedge \texttt{maker}(\texttt{foaf.rdf}, me) \\ \wedge \texttt{primaryTopic}(\texttt{foaf.rdf}, me) \\ \wedge \texttt{Person}(me) \wedge \cdots)$$

- unary predicates for `rdf:type` predicates
- binary predicates for all other predicates

The Semantics of the RDFS vocabulary:

The formal semantics of RDF(S) [Hayes, 2004] is accompanied by a set of (informative) entailment rules . . . can be written down roughly as the following first-order formulas:

$$\forall S, P, O\ (triple(S, P, O) \supset triple(S, \texttt{rdf:type}, \texttt{rdfs:Resource}))$$

$$\forall S, P, O\ (triple(S, P, O) \supset triple(P, \texttt{rdf:type}, \texttt{rdf:Property}))$$

$$\forall S, P, O\ (triple(S, P, O) \supset triple(O, \texttt{rdf:type}, \texttt{rdfs:Resource}))$$

$$\forall S, P, O\ (triple(S, P, O) \wedge triple(P, \texttt{rdfs:domain}, C) \supset triple(S, \texttt{rdf:type}, C))$$

$$\forall S, P, O, C\ (triple(S, P, O) \wedge triple(P, \texttt{rdfs:range}, C) \supset triple(O, \texttt{rdf:type}, C))$$

$$\forall C\ (triple(C, \texttt{rdf:type}, \texttt{rdfs:Class}) \supset triple(C, \texttt{rdfs:subClassOf}, \texttt{rdfs:Resource}))$$

$$\forall C_1, C_2, C_3\ (triple(C_1, \texttt{rdfs:subClassOf}, C_2) \wedge triple(C_2, \texttt{rdfs:subClassOf}, C_3) \supset triple(C_1, \texttt{rdfs:subClassOf}, C_3))$$

$$\forall S, C_1, C_2\ (triple(S, \texttt{rdf:type}, C_1) \wedge triple(C_1, \texttt{rdfs:subClassOf}, C_2) \supset triple(S, \texttt{rdf:type}, C_2))$$

$$\forall S, C\ (triple(S, \texttt{rdf:type}, C) \supset triple(C, \texttt{rdf:type}, \texttt{rdfs:Class}))$$

$$\forall C\ (triple(C, \texttt{rdf:type}, \texttt{rdfs:Class}) \supset triple(C, \texttt{rdfs:subClassOf}, C))$$

$$\forall P_1, P_2, P_3\ (triple(P_1, \texttt{rdfs:subPropertyOf}, P_2) \wedge triple(P_2, \texttt{rdfs:subPropertyOf}, P_3) \supset triple(P_1, \texttt{rdfs:subPropertyOf}, P_3))$$

$$\forall S, P_1, P_2, O\ (triple(S, P_1, O) \wedge triple(P_1, \texttt{rdfs:subPropertyOf}, P_2) \supset triple(S, P_2, O))$$

$$\forall P\ (triple(P, \texttt{rdf:type}, \texttt{rdf:Property}) \supset triple(P, \texttt{rdfs:subPropertyOf}, P))$$

---

plus the axiomatic triples from [Hayes, 2004, Sections 3.1 and 4.1].

The Semantics of the RDFS vocabulary:

The formal semantics of RDF(S) [Hayes, 2004] is accompanied by a set of (informative) entailment rules . . . can be written down roughly as the following first-order formulas:

$$\forall S, P, O\ (triple(S, P, O) \supset triple(S, \texttt{rdf:type}, \texttt{rdfs:Resource}))$$

$$\forall S, P, O\ (triple(S, P, O) \supset triple(P, \texttt{rdf:type}, \texttt{rdf:Property}))$$

$$\forall S, P, O\ (triple(S, P, O) \supset triple(O, \texttt{rdf:type}, \texttt{rdfs:Resource}))$$

$$\forall S, P, O\ (triple(S, P, O) \wedge triple(P, \texttt{rdfs:domain}, C) \supset triple(S, \texttt{rdf:type}, C))$$

$$\forall S, P, O, C\ (triple(S, P, O) \wedge triple(P, \texttt{rdfs:range}, C) \supset triple(O, \texttt{rdf:type}, C))$$

$$\forall C\ (triple(C, \texttt{rdf:type}, \texttt{rdfs:Class}) \supset triple(C, \texttt{rdfs:subClassOf}, \texttt{rdfs:Resource}))$$

$$\forall C_1, C_2, C_3\ (triple(C_1, \texttt{rdfs:subClassOf}, C_2) \wedge triple(C_2, \texttt{rdfs:subClassOf}, C_3) \supset triple(C_1, \texttt{rdfs:subClassOf}, C_3))$$

$$\forall S, C_1, C_2\ (triple(S, \texttt{rdf:type}, C_1) \wedge triple(C_1, \texttt{rdfs:subClassOf}, C_2) \supset triple(S, \texttt{rdf:type}, C_2))$$

$$\forall S, C\ (triple(S, \texttt{rdf:type}, C) \supset triple(C, \texttt{rdf:type}, \texttt{rdfs:Class}))$$

$$\forall C\ (triple(C, \texttt{rdf:type}, \texttt{rdfs:Class}) \supset triple(C, \texttt{rdfs:subClassOf}, C))$$

$$\forall P_1, P_2, P_3\ (triple(P_1, \texttt{rdfs:subPropertyOf}, P_2) \wedge triple(P_2, \texttt{rdfs:subPropertyOf}, P_3) \supset triple(P_1, \texttt{rdfs:subPropertyOf}, P_3))$$

$$\forall S, P_1, P_2, O\ (triple(S, P_1, O) \wedge triple(P_1, \texttt{rdfs:subPropertyOf}, P_2) \supset triple(S, P_2, O))$$

$$\forall P\ (triple(P, \texttt{rdf:type}, \texttt{rdf:Property}) \supset triple(P, \texttt{rdfs:subPropertyOf}, P))$$

---

plus the axiomatic triples from [Hayes, 2004, Sections 3.1 and 4.1].

The Semantics of the RDFS vocabulary:

Note 1:
All those rules were Datalog expressible, i.e. no negation, no function symbols.

The Semantics of the RDFS vocabulary:

Note 1:
All those rules were Datalog expressible, i.e. no negation, no function symbols.

Note 2:
Writing entailment rules in unary/binary representation would yield second order, e.g.:

$$\forall S, C_1, C_2 \; (triple(S, \texttt{rdf:type}, C_1) \; \wedge triple(C_1, \texttt{rdfs:subClassOf}, C_2) \supset triple(S, \texttt{rdf:type}, C_2))$$

The Semantics of the RDFS vocabulary:

Note 1:
All those rules were Datalog expressible, i.e. no negation, no function symbols.

Note 2:
Writing entailment rules in unary/binary representation would yield second order, e.g.:

$$\forall S, C_1, C_2 \, (C_1(S) \ \wedge \texttt{rdfs:subClassOf}(C_1, C_2) \supset C_2(S))$$

RDFS Semantics Example: The FOAF ontology

FOAF Ontology:

- *Each `Person` is a `Agent`* (subclass)
- *The `img` property is more specific than `depiction`* (subproperty)
- *`img` is a relation between `Persons` and `Imgages`* (domain/range)
- *`knows` is a relation between two `Persons`* (domain/range)
- *`homepage` denotes **unique** homepage of an `Agent`* (uniquely identifying property)

⋮

RDFS: Semantics

⋮

$$\forall S, C_1, C_2\ (triple(S, \texttt{rdf:type}, C_1)\ \wedge triple(C_1, \texttt{rdfs:subClassOf}, C_2) \supset triple(S, \texttt{rdf:type}, C_2))$$

⋮

Data:

```
:me rdf:type foaf:Person .
```

RDFS Semantics Example: The FOAF ontology

FOAF Ontology in RDF:

- `foaf:Person rdfs:subClassOf foaf:Agent .`
- `foaf:img rdfs:subPropertyOf foaf:depiction .`
- `foaf:img rdfs:domain foaf:Person ; rdfs:range foaf:Image .`
- `foaf:knows rdfs:domain foaf:Person ; rdfs:range foaf:Person .`
- `???`

⋮

RDFS: Semantics

⋮

$$\forall S, C_1, C_2\ (triple(S, \texttt{rdf:type}, C_1)\ \wedge triple(C_1, \texttt{rdfs:subClassOf}, C_2) \supset triple(S, \texttt{rdf:type}, C_2))$$

⋮

Data:

```
:me rdf:type foaf:Person .
:me rdf:type foaf:Agent .
```

RDFS Semantics Example: The FOAF ontology

FOAF Ontology in RDF:

- `foaf:Person rdfs:subClassOf foaf:Agent .`
- `foaf:img rdfs:subPropertyOf foaf:depiction .`
- `foaf:img rdfs:domain foaf:Person ; rdfs:range foaf:Image .`
- `foaf:knows rdfs:domain foaf:Person ; rdfs:range foaf:Person .`
- *homepage denotes unique homepage of an Agent* ???

⋮

RDFS: Semantics

⋮

$$\forall S, C_1, C_2\ (triple(S, \texttt{rdf:type}, C_1)\ \wedge triple(C_1, \texttt{rdfs:subClassOf}, C_2) \supset triple(S, \texttt{rdf:type}, C_2))$$

⋮

Data:

```
:me rdf:type foaf:Person .
:me rdf:type foaf:Agent .
```

The OWL vocabulary:

- *homepage denotes* ***unique*** *homepage of an* `Agent` (uniquely identifying property)

For expressing this, we need more than the RDFS vocabulary. OWL is again an RDF vocabulary, extending RDF(S), fixed semantics that adds more expressivity on top of RDFS:

- **Properties:** `owl:sameAs`, `owl:differentFrom`, `owl:inverseOf`, `owl:onProperty`, `owl:allValuesFrom`, `owl:someValuesFrom`, `owl:minCardinality`, `owl:maxCardinality` etc.
- **Classes:** `owl:Restriction`, `owl:DatatypeProperty`, `owl:ObjectProperty`, `owl:FunctionalProperty`, `owl:InverseFuncionalProperty`, `owl:SymmetricProperty` etc.
- **Relations:** The semantics of OWL is defined in [Patel-Schneider *et al.*, 2004]
  - in terms of its RDF reading (OWL Full semantics), and

[7] OWL DL puts restrictions on the use of the OWL and RDF vocabulary, e.g. classes may not be used as instances, etc., for instance `one rdf:type integer . integer rdf:type simpleDatatype .` would not be allowed.

The OWL vocabulary:

- *homepage denotes* ***unique*** *homepage of an* `Agent` (uniquely identifying property)

For expressing this, we need more than the RDFS vocabulary. OWL is again an RDF vocabulary, extending RDF(S), fixed semantics that adds more expressivity on top of RDFS:

- **Properties:** `owl:sameAs`, `owl:differentFrom`, `owl:inverseOf`, `owl:onProperty`, `owl:allValuesFrom`, `owl:someValuesFrom`, `owl:minCardinality`, `owl:maxCardinality` etc.
- **Classes:** `owl:Restriction`, `owl:DatatypeProperty`, `owl:ObjectProperty`, `owl:FunctionalProperty`, `owl:InverseFuncionalProperty`, `owl:SymmetricProperty` etc.
- **Relations:** The semantics of OWL is defined in [Patel-Schneider *et al.*, 2004]
  - in terms of its RDF reading (OWL Full semantics), and
  - in terms of its Description Logics reading (OWL DL semantics)[7]

[7] OWL DL puts restrictions on the use of the OWL and RDF vocabulary, e.g. classes may not be used as instances, etc., for instance `one rdf:type integer . integer rdf:type simpleDatatype .` would not be allowed.

The Semantics of the OWL vocabulary (DL reading):

Description Logics:

- syntactic variant of first-order logic with equality
- especially tailored for talking about concepts (classes, sets) and roles (properties)
- dedicated symbols for class membership and subclass/subproperty relation:

```
foaf:Person rdfs:subClassOf foaf:Agent
```
$Person \sqsubseteq Agent$

```
:me rdf:type  foaf:Person
```
$me \in Person$

OWL DL in two slides: 1/2

Expressing property characteristics:

OWL property axioms as RDF triples	DL syntax	FOL short representation
P `rdfs:domain` C .	$\top \sqsubseteq \forall P^-.C$	$\forall x, y.P(x, y) \supset C(x)$
P `rdfs:range` C .	$\top \sqsubseteq \forall P.C$	$\forall x, y.P(x, y) \supset C(y)$
P `owl:inverseOf` P_0 .	$P \equiv P_0^-$	$\forall x, y.P(x, y) \equiv P_0(y, x)$
P `rdf:type owl:SymmetricProperty.`	$P \equiv P^-$	$\forall x, y.P(x, y) \equiv P(y, x)$
P `rdf:type owl:FunctionalProperty.`	$\top \sqsubseteq \leqslant 1P$	$\forall x, y, z.P(x, y) \wedge P(x, z) \supset y = z$
P `rdf:type owl:InverseFunctionalProperty.`	$\top \sqsubseteq \leqslant 1P^-$	$\forall x, y, z.P(x, y) \wedge P(z, y) \supset x = z$
P `rdf:type owl:TransitiveProperty.`	$P^+ \sqsubseteq P$	$\forall x, y, z.P(x, y) \wedge P(y, z) \supset P(x, z)$

OWL DL in two slides: 1/2

Expressing property characteristics:

OWL property axioms as RDF triples	DL syntax	FOL short representation
P `rdfs:domain` C .	$\top \sqsubseteq \forall P^-.C$	$\forall x,y.P(x,y) \supset C(x)$
P `rdfs:range` C .	$\top \sqsubseteq \forall P.C$	$\forall x,y.P(x,y) \supset C(y)$
P `owl:inverseOf` P_0 .	$P \equiv P_0^-$	$\forall x,y.P(x,y) \equiv P_0(y,x)$
P `rdf:type owl:SymmetricProperty.`	$P \equiv P^-$	$\forall x,y.P(x,y) \equiv P(y,x)$
P `rdf:type owl:FunctionalProperty.`	$\top \sqsubseteq \leqslant 1P$	$\forall x,y,z.P(x,y) \wedge P(x,z) \supset y = z$
P `rdf:type owl:InverseFunctionalProperty.`	$\top \sqsubseteq \leqslant 1P^-$	$\forall x,y,z.P(x,y) \wedge P(z,y) \supset x = z$
P `rdf:type owl:TransitiveProperty.`	$P^+ \sqsubseteq P$	$\forall x,y,z.P(x,y) \wedge P(y,z) \supset P(x,z)$

Expressing complex class descriptions:

OWL complex class descriptions*	DL syntax	FOL short representation
`owl:Thing`	$\top$	$x = x$
`owl:Nothing`	$\bot$	$\neg x = x$
`owl:intersectionOf` $(C_1 \ldots C_n)$	$C_1 \sqcap \cdots \sqcap C_n$	$C_1(x) \wedge \cdots \wedge C_n(x)$
`owl:unionOf` $(C_1 \ldots C_n)$	$C_1 \sqcup \cdots \sqcup C_n$	$C_1(x) \vee \cdots \vee C_n(x)$
`owl:complementOf` (C)	$\neg C$	$\neg C(x)$
`owl:oneOf` $(o_1 \ldots o_n)$	$\{o_1, \ldots, o_n\}$	$x = o_1 \vee \cdots \vee x = o_n$
`owl:restriction` $(P$ `owl:someValuesFrom` $(C))$	$\exists P.C$	$\exists y.P(x,y) \wedge C(y)$
`owl:restriction` $(P$ `owl:allValuesFrom` $(C))$	$\forall P.C$	$\forall y.P(x,y) \supset C(y)$
`owl:restriction` $(P$ `owl:value` $(o))$	$\exists P.\{o\}$	$P(x,o)$
`owl:restriction` $(P$ `owl:minCardinality` $(n))$	$\geqslant nP$	$\exists y_1 \ldots y_n . \bigwedge_{k=1}^{n} P(x,y_k) \wedge \bigwedge_{i<j} y_j \neq y_j$
`owl:restriction` $(P$ `owl:maxCardinality` $(n))$	$\leqslant nP$	$\forall y_1 \ldots y_{n+1} . \bigwedge_{k=1}^{n+1} P(x,y_k) \supset \bigvee_{i<j} y_i = y_j$

*For reasons of legibility, we use a variant of the OWL abstract syntax [Patel-Schneider *et al.*, 2004] in this table.

OWL DL in two slides: 2/2

Relating Class descriptions:

C_1 `rdfs:subClassOf` C_1	$C_1 \sqsubseteq C_2$
C_1 `owl:equivalentClass` C_2	$C_1 \equiv C_2$
C_1 `owl:disjointWith` C_2	$C_1 \sqcap C_2 \sqsubseteq \bot$

Relating individuals:

o_1 `owl:sameAs` o_1	$o_1 = o_2$
o_1 `owl:differentFrom` o_2	$o_1 \neq o_2$

OWL DL in two slides: 2/2

Relating Class descriptions:

C_1 `rdfs:subClassOf` C_1	$C_1 \sqsubseteq C_2$
C_1 `owl:equivalentClass` C_2	$C_1 \equiv C_2$
C_1 `owl:disjointWith` C_2	$C_1 \sqcap C_2 \sqsubseteq \bot$

Relating individuals:

o_1 `owl:sameAs` o_1	$o_1 = o_2$
o_1 `owl:differentFrom` o_2	$o_1 \neq o_2$

Examples:

```
<http://www.polleres.net/foaf.rdf#me> owl:sameAs
        <http://dblp.l3s.de/d2r/resource/authors/Axel_Polleres> .
<http://polleres.net/foaf.rdf#me> owl:differentFrom
        <http://www.mat.unical.it/~ianni/foaf.rdf#me> .
```

OWL Example: The FOAF ontology

- *homepage denotes* ***unique*** *homepage of an* *Agent* (uniquely identifying property)

. . . in OWL/RDF syntax:

```
foaf:homepage rdf:type owl:InverseFunctionalProperty .
```

OWL Example: The FOAF ontology

- *homepage denotes* ***unique*** *homepage of an* `Agent` (uniquely identifying property)

. . . in OWL/RDF syntax:

```
foaf:homepage rdf:type owl:InverseFunctionalProperty .
```

. . . in DL syntax:

$$\top \sqsubseteq \leqslant 1 homepage^{-}$$

OWL Example: The FOAF ontology

- *homepage denotes* ***unique*** *homepage of an* `Agent` (uniquely identifying property)

. . . in OWL/RDF syntax:

```
foaf:homepage rdf:type owl:InverseFunctionalProperty .
```

. . . in DL syntax:

$$\top \sqsubseteq \leqslant 1 homepage^{-}$$

Example inference:

```
<http://www.polleres.net/foaf.rdf#me> foaf:homepage
        <http://www.polleres.net/> .
<http://dblp.l3s.de/d2r/resource/authors/Axel_Polleres> foaf:homepage
        <http://www.polleres.net/> .
```

OWL Example: The FOAF ontology

- *homepage denotes* ***unique*** *homepage of an* `Agent` (uniquely identifying property)

. . . in OWL/RDF syntax:

```
foaf:homepage rdf:type owl:InverseFunctionalProperty .
```

. . . in DL syntax:

$$\top \sqsubseteq \leqslant 1 homepage^{-}$$

Example inference:

```
<http://www.polleres.net/foaf.rdf#me> foaf:homepage
        <http://www.polleres.net/> .
<http://dblp.l3s.de/d2r/resource/authors/Axel_Polleres> foaf:homepage
        <http://www.polleres.net/> .
```

$\models$

```
 <http://www.polleres.net/foaf.rdf#me> owl:sameAs
        <http://dblp.l3s.de/d2r/resource/authors/Axel_Polleres> .
```

OWL Example: A simple ontology about reviewers

$$\exists ex{:}title.\top \sqsubseteq ex{:}Paper \quad \text{(i)}$$

$$\exists ex{:}title^{-}.\top \sqsubseteq xsd{:}string \quad \text{(ii)}$$

$$ex{:}isAuthorOf^{-} \equiv dc{:}creator \quad \text{(iii)}$$

$$ex{:}Publication \equiv ex{:}Paper \sqcap \exists ex{:}publishedIn.\top \quad \text{(iv)}$$

$$\top \sqsubseteq \leqslant 1\ ex{:}publishedIn^{-} \quad \text{(v)}$$

$$ex{:}Senior \equiv foaf{:}Person \sqcap \geqslant 10\ ex{:}isAuthorOf \sqcap \exists ex{:}isAuthorOf.ex{:}Publication \quad \text{(vi)}$$

$$ex{:}Club100 \equiv foaf{:}Person \sqcap \geqslant 100\ ex{:}isAuthorOf \quad \text{(vii)}$$

[8] (...) is a shortcut for `rdf:Lists`, expand to `rdf:first`, `rdf:rest`, `rdf:nil` triples.

[9] (vi) is not exactly that, qualified number restrictions will likely be in OWL 2, though.

OWL Example: A simple ontology about reviewers

$$\exists ex{:}title.\top \sqsubseteq ex{:}Paper \quad \text{(i)}$$

$$\exists ex{:}title^{-}.\top \sqsubseteq xsd{:}string \quad \text{(ii)}$$

$$ex{:}isAuthorOf^{-} \equiv dc{:}creator \quad \text{(iii)}$$

$$ex{:}Publication \equiv ex{:}Paper \sqcap \exists ex{:}publishedIn.\top \quad \text{(iv)}$$

$$\top \sqsubseteq \leqslant 1\ ex{:}publishedIn^{-} \quad \text{(v)}$$

$$ex{:}Senior \equiv foaf{:}Person \sqcap \geqslant 10\ ex{:}isAuthorOf \sqcap \exists ex{:}isAuthorOf.ex{:}Publication \quad \text{(vi)}$$

$$ex{:}Club100 \equiv foaf{:}Person \sqcap \geqslant 100\ ex{:}isAuthorOf \quad \text{(vii)}$$

- *A `Publication` is a `Paper` which has been `published`* (iv)

[8] (...) is a shortcut for `rdf:Lists`, expand to `rdf:first`, `rdf:rest`, `rdf:nil` triples.

[9] (vi) is not exactly that, qualified number restrictions will likely be in OWL 2, though.

OWL Example: A simple ontology about reviewers

$$\exists ex{:}title.\top \sqsubseteq ex{:}Paper \quad \text{(i)}$$

$$\exists ex{:}title^{-}.\top \sqsubseteq xsd{:}string \quad \text{(ii)}$$

$$ex{:}isAuthorOf^{-} \equiv dc{:}creator \quad \text{(iii)}$$

$$ex{:}Publication \equiv ex{:}Paper \sqcap \exists ex{:}publishedIn.\top \quad \text{(iv)}$$

$$\top \sqsubseteq \leqslant 1\ ex{:}publishedIn^{-} \quad \text{(v)}$$

$$ex{:}Senior \equiv foaf{:}Person \sqcap \geqslant 10\ ex{:}isAuthorOf \sqcap \exists ex{:}isAuthorOf.ex{:}Publication \quad \text{(vi)}$$

$$ex{:}Club100 \equiv foaf{:}Person \sqcap \geqslant 100\ ex{:}isAuthorOf \quad \text{(vii)}$$

- *A `Publication` is a `Paper` which has been `published`* (iv)

```
ex:Publication owl:intersectionOf ( ex:Paper [ a owl:Restriction; owl:onProperty ex:publishedIn ;
owl:minCardinality 1 ] ) .
```
[8]

[8] `( ... )` is a shortcut for `rdf:Lists`, expand to `rdf:first`, `rdf:rest`, `rdf:nil` triples.

[9] (vi) is not exactly that, qualified number restrictions will likely be in OWL 2, though.

OWL Example: A simple ontology about reviewers

$$\exists ex{:}title.\top \sqsubseteq ex{:}Paper \quad \text{(i)}$$

$$\exists ex{:}title^{-}.\top \sqsubseteq xsd{:}string \quad \text{(ii)}$$

$$ex{:}isAuthorOf^{-} \equiv dc{:}creator \quad \text{(iii)}$$

$$ex{:}Publication \equiv ex{:}Paper \sqcap \exists ex{:}publishedIn.\top \quad \text{(iv)}$$

$$\top \sqsubseteq \leqslant 1\ ex{:}publishedIn^{-} \quad \text{(v)}$$

$$ex{:}Senior \equiv foaf{:}Person \sqcap \geqslant 10\ ex{:}isAuthorOf \sqcap \exists ex{:}isAuthorOf.ex{:}Publication \quad \text{(vi)}$$

$$ex{:}Club100 \equiv foaf{:}Person \sqcap \geqslant 100\ ex{:}isAuthorOf \quad \text{(vii)}$$

- *A `Publication` is a `Paper` which has been `published`* (iv)

```
ex:Publication owl:intersectionOf ( ex:Paper [ a owl:Restriction; owl:onProperty ex:publishedIn ;
owl:minCardinality 1 ] ) .
```
[8]

- *`isAuthorOf` is the opposite of Dublin Core's `dc:creator` Property* (iii)

[8] `( ... )` is a shortcut for `rdf:Lists`, expand to `rdf:first`, `rdf:rest`, `rdf:nil` triples.

[9] (vi) is not exactly that, qualified number restrictions will likely be in OWL 2, though.

OWL Example: A simple ontology about reviewers

$$\exists ex{:}title.\top \sqsubseteq ex{:}Paper \quad \text{(i)}$$

$$\exists ex{:}title^{-}.\top \sqsubseteq xsd{:}string \quad \text{(ii)}$$

$$ex{:}isAuthorOf^{-} \equiv dc{:}creator \quad \text{(iii)}$$

$$ex{:}Publication \equiv ex{:}Paper \sqcap \exists ex{:}publishedIn.\top \quad \text{(iv)}$$

$$\top \sqsubseteq \leqslant 1\ ex{:}publishedIn^{-} \quad \text{(v)}$$

$$ex{:}Senior \equiv foaf{:}Person \sqcap \geqslant 10\ ex{:}isAuthorOf \sqcap \exists ex{:}isAuthorOf.ex{:}Publication \quad \text{(vi)}$$

$$ex{:}Club100 \equiv foaf{:}Person \sqcap \geqslant 100\ ex{:}isAuthorOf \quad \text{(vii)}$$

- *A `Publication` is a `Paper` which has been `published`* (iv)

```
ex:Publication owl:intersectionOf ( ex:Paper [ a owl:Restriction; owl:onProperty ex:publishedIn ;
owl:minCardinality 1 ] ) .
```
[8]

- *`isAuthorOf` is the opposite of Dublin Core's `dc:creator` Property* (iii)

```
ex:isAuthorOf owl:inverseOf dc:creator .
```

[8] `( ... )` is a shortcut for `rdf:Lists`, expand to `rdf:first`, `rdf:rest`, `rdf:nil` triples.

[9] (vi) is not exactly that, qualified number restrictions will likely be in OWL 2, though.

OWL Example: A simple ontology about reviewers

$$\exists ex{:}title.\top \sqsubseteq ex{:}Paper \quad \text{(i)}$$

$$\exists ex{:}title^{-}.\top \sqsubseteq xsd{:}string \quad \text{(ii)}$$

$$ex{:}isAuthorOf^{-} \equiv dc{:}creator \quad \text{(iii)}$$

$$ex{:}Publication \equiv ex{:}Paper \sqcap \exists ex{:}publishedIn.\top \quad \text{(iv)}$$

$$\top \sqsubseteq \leqslant 1\ ex{:}publishedIn^{-} \quad \text{(v)}$$

$$ex{:}Senior \equiv foaf{:}Person \sqcap \geqslant 10\ ex{:}isAuthorOf \sqcap \exists ex{:}isAuthorOf.ex{:}Publication \quad \text{(vi)}$$

$$ex{:}Club100 \equiv foaf{:}Person \sqcap \geqslant 100\ ex{:}isAuthorOf \quad \text{(vii)}$$

- *A `Publication` is a `Paper` which has been `published`* (iv)

```
ex:Publication owl:intersectionOf ( ex:Paper [ a owl:Restriction; owl:onProperty ex:publishedIn ;
owl:minCardinality 1 ] ) .
```
[8]

- *`isAuthorOf` is the opposite of Dublin Core's `dc:creator` Property* (iii)

```
ex:isAuthorOf owl:inverseOf dc:creator .
```

- *A `Senior` researcher is a `foaf:Person` who `isAuthorOf` 10+ `Publications`* (vi)[9]

[8] `( ... )` is a shortcut for `rdf:Lists`, expand to `rdf:first`, `rdf:rest`, `rdf:nil` triples.

[9] (vi) is not exactly that, qualified number restrictions will likely be in OWL 2, though.

OWL Example: A simple ontology about reviewers

$$\exists ex{:}title.\top \sqsubseteq ex{:}Paper \quad \text{(i)}$$

$$\exists ex{:}title^{-}.\top \sqsubseteq xsd{:}string \quad \text{(ii)}$$

$$ex{:}isAuthorOf^{-} \equiv dc{:}creator \quad \text{(iii)}$$

$$ex{:}Publication \equiv ex{:}Paper \sqcap \exists ex{:}publishedIn.\top \quad \text{(iv)}$$

$$\top \sqsubseteq \leqslant 1\ ex{:}publishedIn^{-} \quad \text{(v)}$$

$$ex{:}Senior \equiv foaf{:}Person \sqcap \geqslant 10\ ex{:}isAuthorOf \sqcap \exists ex{:}isAuthorOf.ex{:}Publication \quad \text{(vi)}$$

$$ex{:}Club100 \equiv foaf{:}Person \sqcap \geqslant 100\ ex{:}isAuthorOf \quad \text{(vii)}$$

- *A `Publication` is a `Paper` which has been `published`* (iv)

```
ex:Publication owl:intersectionOf ( ex:Paper [ a owl:Restriction; owl:onProperty ex:publishedIn ;
owl:minCardinality 1 ] ) .
```
[8]

- *`isAuthorOf` is the opposite of Dublin Core's `dc:creator` Property* (iii)

```
ex:isAuthorOf owl:inverseOf dc:creator .
```

- *A `Senior` researcher is a `foaf:Person` who `isAuthorOf` 10+ `Publications`* (vi)[9]

```
ex:Senior owl:intersectionOf ( foaf:Person [ a owl:Restriction; owl:onProperty ex:isAuthorOf ;
owl:minCardinality 10 ] [ a owl:Restriction; owl:onProperty ex:isAuthorOf ; owl:someValuesFrom
ex:Publication ] ) .
```

[8] `( ... )` is a shortcut for `rdf:Lists`, expand to `rdf:first`, `rdf:rest`, `rdf:nil` triples.

[9] (vi) is not exactly that, qualified number restrictions will likely be in OWL 2, though.

OWL Example: A simple ontology about reviewers

$$\exists ex{:}title.\top \sqsubseteq ex{:}Paper \quad \text{(i)}$$

$$\exists ex{:}title^{-}.\top \sqsubseteq xsd{:}string \quad \text{(ii)}$$

$$ex{:}isAuthorOf^{-} \equiv dc{:}creator \quad \text{(iii)}$$

$$ex{:}Publication \equiv ex{:}Paper \sqcap \exists ex{:}publishedIn.\top \quad \text{(iv)}$$

$$\top \sqsubseteq \leqslant 1\ ex{:}publishedIn^{-} \quad \text{(v)}$$

$$ex{:}Senior \equiv foaf{:}Person \sqcap \geqslant 10\ ex{:}isAuthorOf \sqcap \exists ex{:}isAuthorOf.ex{:}Publication \quad \text{(vi)}$$

$$ex{:}Club100 \equiv foaf{:}Person \sqcap \geqslant 100\ ex{:}isAuthorOf \quad \text{(vii)}$$

- *A `Publication` is a `Paper` which has been `published`* (iv)

```
ex:Publication owl:intersectionOf ( ex:Paper [ a owl:Restriction; owl:onProperty ex:publishedIn ;
owl:minCardinality 1 ] ) .
```
[8]

- *`isAuthorOf` is the opposite of Dublin Core's `dc:creator` Property* (iii)

```
ex:isAuthorOf owl:inverseOf dc:creator .
```

- *A `Senior` researcher is a `foaf:Person` who `isAuthorOf` 10+ `Publications`* (vi)[9]

```
ex:Senior owl:intersectionOf ( foaf:Person [ a owl:Restriction; owl:onProperty ex:isAuthorOf ;
owl:minCardinality 10 ] [ a owl:Restriction; owl:onProperty ex:isAuthorOf ; owl:someValuesFrom
ex:Publication ] ) .
```

- *Each item can be `publishedIn` at most one venue* (v)

[8] `( ... )` is a shortcut for `rdf:Lists`, expand to `rdf:first`, `rdf:rest`, `rdf:nil` triples.

[9] (vi) is not exactly that, qualified number restrictions will likely be in OWL 2, though.

OWL Example: A simple ontology about reviewers

$$\exists ex{:}title.\top \sqsubseteq ex{:}Paper \quad \text{(i)}$$

$$\exists ex{:}title^{-}.\top \sqsubseteq xsd{:}string \quad \text{(ii)}$$

$$ex{:}isAuthorOf^{-} \equiv dc{:}creator \quad \text{(iii)}$$

$$ex{:}Publication \equiv ex{:}Paper \sqcap \exists ex{:}publishedIn.\top \quad \text{(iv)}$$

$$\top \sqsubseteq \leqslant 1\ ex{:}publishedIn^{-} \quad \text{(v)}$$

$$ex{:}Senior \equiv foaf{:}Person \sqcap \geqslant 10\ ex{:}isAuthorOf \sqcap \exists ex{:}isAuthorOf.ex{:}Publication \quad \text{(vi)}$$

$$ex{:}Club100 \equiv foaf{:}Person \sqcap \geqslant 100\ ex{:}isAuthorOf \quad \text{(vii)}$$

- *A `Publication` is a `Paper` which has been `published`* (iv)

```
ex:Publication owl:intersectionOf ( ex:Paper [ a owl:Restriction; owl:onProperty ex:publishedIn ;
owl:minCardinality 1 ] ) .
```
[8]

- *`isAuthorOf` is the opposite of Dublin Core's `dc:creator` Property* (iii)

```
ex:isAuthorOf owl:inverseOf dc:creator .
```

- *A `Senior` researcher is a `foaf:Person` who `isAuthorOf` 10+ `Publications`* (vi)[9]

```
ex:Senior owl:intersectionOf ( foaf:Person [ a owl:Restriction; owl:onProperty ex:isAuthorOf ;
owl:minCardinality 10 ] [ a owl:Restriction; owl:onProperty ex:isAuthorOf ; owl:someValuesFrom
ex:Publication ] ) .
```

- *Each item can be `publishedIn` at most one venue* (v)

```
ex:publishedIn a owl:FunctionalProperty .
```

[8] `( ... )` is a shortcut for `rdf:Lists`, expand to `rdf:first`, `rdf:rest`, `rdf:nil` triples.

[9] (vi) is not exactly that, qualified number restrictions will likely be in OWL 2, though.

Reasoning with Ontologies

Tools:

- Special purpose DL reasoners:
  Pellet [Sirin *et al.*, 2005], Racer [Haarslev and Möller, 2001], Fact++ [Tsarkov and Horrocks, 2006]

Reasoning with Ontologies

Tools:

- Special purpose DL reasoners:
  Pellet [Sirin *et al.*, 2005], Racer [Haarslev and Möller, 2001], Fact++ [Tsarkov and Horrocks, 2006]
- General purpose FOL theorem provers:
  SNARK [Stickel *et al.*,], SPASS [SPASS,], Vampire [Riazanov and Voronkov, 2002]

Reasoning with Ontologies

Tools:

- Special purpose DL reasoners:
  Pellet [Sirin *et al.*, 2005], Racer [Haarslev and Möller, 2001], Fact++ [Tsarkov and Horrocks, 2006]
- General purpose FOL theorem provers:
  SNARK [Stickel *et al.*,], SPASS [SPASS,], Vampire [Riazanov and Voronkov, 2002]
- **Rule/LP engines**: cf. Unit3

SPARQL & Ontologies

SPARQL on top of ontologies not (yet) entirely clear (cf. [Arenas *et al.*, , Unit 5b]):

SPARQL & Ontologies

SPARQL on top of ontologies not (yet) entirely clear (cf. [Arenas *et al.*, , Unit 5b]):

- Problem 1: infinite RDFS/OWL inferences on a finite graph

SPARQL & Ontologies

SPARQL on top of ontologies not (yet) entirely clear (cf. [Arenas *et al.*, , Unit 5b]):

- Problem 1: infinite RDFS/OWL inferences on a finite graph
- Problem 2: co-reference of blank nodes in SPARQL solutions

SPARQL & Ontologies

SPARQL on top of ontologies not (yet) entirely clear (cf. [Arenas *et al.*, , Unit 5b]):

- Problem 1: infinite RDFS/OWL inferences on a finite graph
- Problem 2: co-reference of blank nodes in SPARQL solutions
- Problem 3: SPARQL is SQL inspired (CWA), OWL/RDFS are (OWA):
  e.g., OPTIONAL patterns are non-monotonic, RDFS+OWL inference are both monotonic, that can lead to query answers valid under simple RDF, but not under OWL entailment, etc.

SPARQL & Ontologies

SPARQL on top of ontologies not (yet) entirely clear (cf. [Arenas *et al.*, , Unit 5b]):

- Problem 1: infinite RDFS/OWL inferences on a finite graph
- Problem 2: co-reference of blank nodes in SPARQL solutions
- Problem 3: SPARQL is SQL inspired (CWA), OWL/RDFS are (OWA):
  e.g., OPTIONAL patterns are non-monotonic, RDFS+OWL inference are both monotonic, that can lead to query answers valid under simple RDF, but not under OWL entailment, etc.

SPARQL & Ontologies

SPARQL on top of ontologies not (yet) entirely clear (cf. [Arenas *et al.*, , Unit 5b]):

- Problem 1: infinite RDFS/OWL inferences on a finite graph
- Problem 2: co-reference of blank nodes in SPARQL solutions
- Problem 3: SPARQL is SQL inspired (CWA), OWL/RDFS are (OWA):
  e.g., OPTIONAL patterns are non-monotonic, RDFS+OWL inference are both monotonic, that can lead to query answers valid under simple RDF, but not under OWL entailment, etc.

W3C's OWL2 WG tries to define SPARQL DL . . . stay tuned!

Unit Outline

Summary

- We should all have a rough idea about where to find RDF now.
- We should all have a rough idea about how to read RDF now.
- We should all have a rough idea of how to query RDF (SPARQL).
- We should all have an idea of how the semantics of RDF vocabularies and data can be descibed (RDFS + OWL)

Details to come!

What's next?

- Details on the semantics of RDF+RDFS
- Details on the semantics of SPARQL, and why SPARQL+RDFS is not trivial.
- Details on OWL, and sneak-preview on OWL2
- Time allowed: sneak-preview on RIF (Rule Interchange Format)
- Towards the end of the lecture: practical applications on Reasoning about Web Data.


Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton (eds.).
Rdfa in xhtml: Syntax and processing, October 2008.
W3C Recommendation, available at http://www.w3.org/TR/rdfa-syntax/.


Marcelo Arenas, Claudio Gutierrez, Bijan Parsia, Jorge Pérez, Axel Polleres, and Andy Seaborne.


Tags for identifying languages, September 2006.
available at http://www.rfc-editor.org/rfc/bcp/bcp47.txt.


Dave Beckett and Tim Berners-Lee.
Turtle - Terse RDF Triple Language, January 2008.
W3C Team Submission, http://www.w3.org/TeamSubmission/turtle/.


Dave Beckett and Brian McBride (eds.).
Rdf/xml syntax specification (revised), February 2004.
W3C Recommendation, available at http://www.w3.org/TR/rdf-syntax-grammar/.


Tim Berners-Lee and Dan Connolly.
Notation3 (N3): A readable RDF Syntax, January 2008.
W3C Team Submission, http://www.w3.org/TeamSubmission/n3/.


Uldis Bojārs, John G. Breslin, Diego Berrueta, Dan Brickley, Stefan Decker, Sergio Fernández, Christoph Görn, Andreas Harth, Tom Heath, Kingsley Idehen, Kjetil Kjernsmo, Alistair Miles, Alexandre Passant, Axel Polleres, Luis Polo, and Michael Sintek.
SIOC Core Ontology Specification, June 2007.
W3C member submission.


Dan Brickley and R.V. Guha (eds.).
RDF vocabulary description language 1.0: RDF Schema, February 2004.
W3C Recommendation, available at `http://www.w3.org/TR/rdf-schema/`.


Dan Brickley and Libby Miller.
FOAF Vocabulary Specification 0.91, November 2007.
`http://xmlns.com/foaf/spec/`.


Jeremy Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler.
Named graphs.
Journal of Web Semantics, 3(4), 2005.


Dan Connolly (ed.).
Gleaning Resource Descriptions from Dialects of Languages (GRDDL), September 2007.


ESW Wiki: SPARQL.
List of useful links for SPARQL implementations and extensions,
`http://esw.w3.org/topic/SPARQL/`.


Claudio Gutiérrez, Carlos A. Hurtado, and Alberto O. Mendelzon.
Foundations of semantic web databases.
In *PODS*, pages 95–106, 2004.


V. Haarslev and R. Möller.
RACER System Description.
In *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Computer Science (LNCS)*, pages 701–705. Springer Verlag, 2001.


Michael Hausenblas, Ivan Hermann, and Ben Adida.
Rdfa - bridging the web of documents and the web of data, 2008.
Tutorial given at ISWC2008, available at
http://www.w3.org/2008/Talks/1026-ISCW-RDFa/RDFa-ISWC08.html.


P. Hayes.
RDF semantics, 2004.
http://www.w3.org/TR/rdf-mt/.


Mikael Nilsson, Andy Powell, Pete Johnston, and Ambjörn Naeve.
Expressing dublin core metadata using the resource description framework (rdf), January 2008.
DCMI Recommendation.


Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks.
OWL Web Ontology Language Semantics and Abstract Syntax, February 2004.
W3C Recommendation.


Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez.
Semantics and complexity of sparql.
In *International Semantic Web Conference (ISWC 2006)*, pages 30–43, 2006.


Axel Polleres.
From SPARQL to rules (and back).
In *Proceedings of the 16th World Wide Web Conference (WWW2007)*, Banff, Canada, May 2007.


SPARQL Query Language for RDF, January 2007.

W3C Recommendation, available at
http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.


Alexandre Riazanov and Andrei Voronkov.
The design and implementation of vampire.
AI Communications, 15(2-3):91–110, 2002.


Simon Schenk and Steffen Staab.
Networked graphs: A declarative mechanism for sparql rules, sparql views and rdf data integration on the web.
In *Proceedings WWW-2008*, pages 585–594, 2008.


Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz.
Pellet: A practical OWL-DL reasoner.
Technical Report 68, UMIACS, University of Maryland, 2005.


Spass: An automated theorem prover for first-order logic with equality.
Available at http://www.spass-prover.org/.


Mark E. Stickel, Richard J. Waldinger, and Vinay K. Chaudhr.
A guide to snark.
Available at http://www.ai.sri.com/snark/tutorial/tutorial.html.


Dmitry Tsarkov and Ian Horrocks.
Fact++ Description Logic Reasoner: System Description.
In *Proceedings of the Third International Joint Conference on Automated Reasoning (IJCAR 2006)*, 2006.