Semantic Web

RDF and RDF Schema

Dieter Fensel and Federico Facca

Where are we?

STI · INNSBRUCK

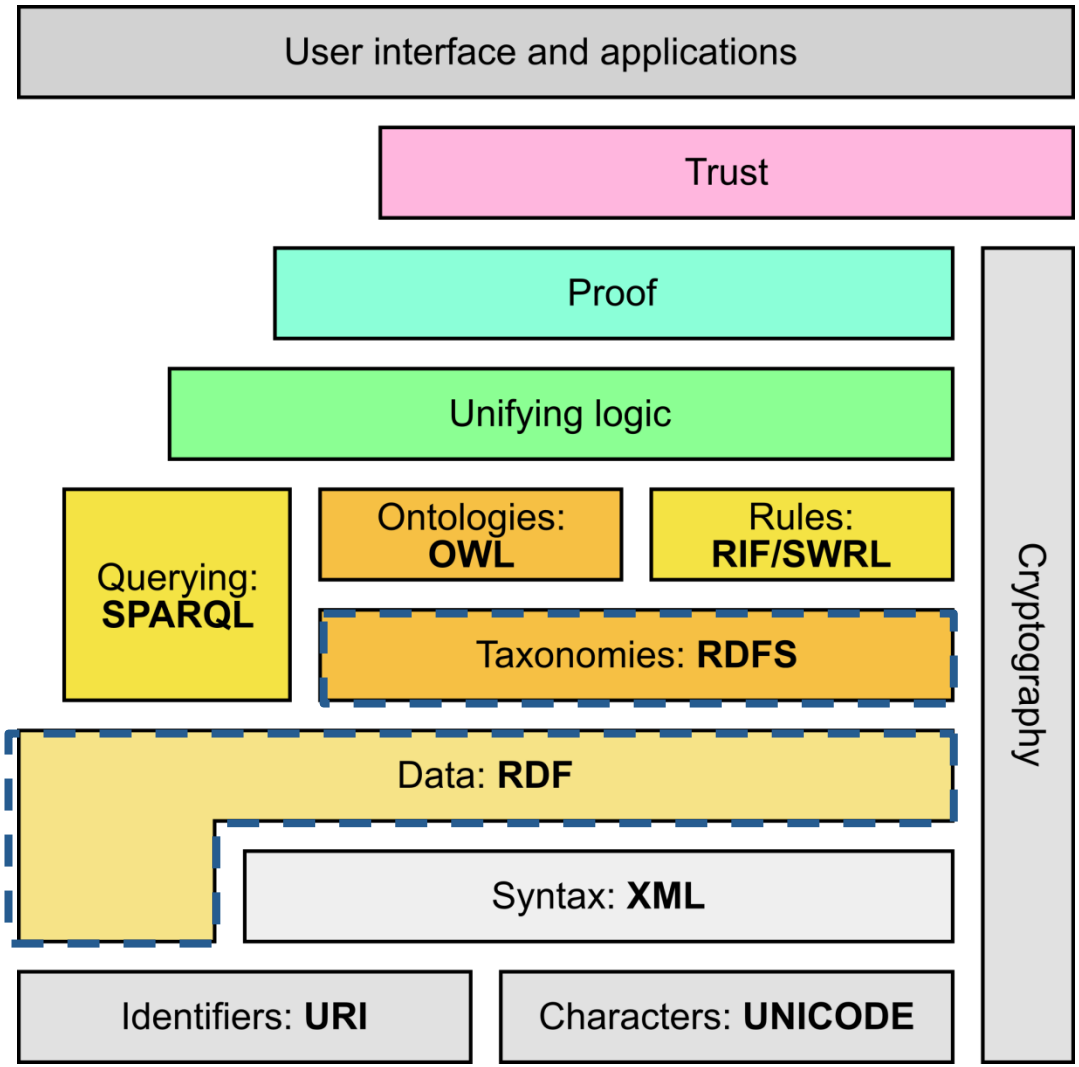
#	Title
1	Introduction
2	Semantic Web architecture
3	**RDF and RDFs**
4	**Web of hypertext (RDFa, Microformats) and Web of data**
5	Semantic annotations
6	Repositories and SPARQL
7	OWL
8	RIF
9	Web-scale reasoning
10	Social Semantic Web
11	Ontologies and the Semantic Web
12	SWS
13	Tools
14	Applications
15	Exam

Agenda

STI · INNSBRUCK

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

Semantic Web Stack



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

INTRODUCTION AND MOTIVATION

Motivating Example 1

STI · INNSBRUCK

- Dieter Fensel is teaching the Semantic Web course.

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

Motivating Example 1

STI · INNSBRUCK

- Dieter Fensel is teaching the Semantic Web course.

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

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

Motivating Example 1

STI · INNSBRUCK

- Dieter Fensel is teaching the Semantic Web course.

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

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

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

Motivating Example 1

STI · INNSBRUCK

- Dieter Fensel is teaching the Semantic Web course.

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

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

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

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

Motivating Example 2

STI · INNSBRUCK

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

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

Motivating Example 2

STI · INNSBRUCK

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

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

<professor>Dieter Fensel</professor>

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

- Retrieve all the members of the academic staff.

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

What Are RDF and RDF Schema?

STI · INNSBRUCK

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

RDF and RDF Schema

TECHNICAL SOLUTION

The power of triple representation joint with XML serialization

THE RESOURCE DESCRIPTION FRAMEWORK

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

Principles of RDF 1

STI · INNSBRUCK

- Resource (identified by URIs)
  - A URI *identifies* a resource, but does not necessarily *point* to it
  - Correspond to nodes in a graph
  - E.g.:
    `http://www.w3.org/`
    `http://example.org/#john`
    `http://www.w3.org/1999/02/22-rdf-syntax-ns#Property`
- Properties (identified by URIs)
  - Correspond to labels of edges in a graph
  - Binary relation between two resources
  - E.g.:
    `http://www.example.org/#hasName`
    `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`
- Literals
  - Concrete data values
  - E.g.:
    `"John Smith", "1", "2006-03-07"`

Principles of RDF 2

STI · INNSBRUCK

- Triple data model:

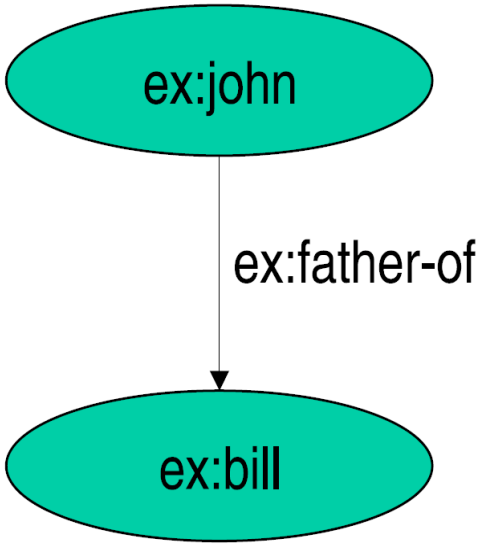
  `<subject, predicate, object>`

  - **Subject**: Resource or blank node
  - **Predicate**: Property
  - **Object**: Resource, literal or blank node

- Example:

  `<ex:john, ex:father-of, ex:bill>`

- Labeled, directed graphs
  - **Nodes**: resources, literals
  - **Labels**: properties
  - **Edges**: statements



Resources

STI · INNSBRUCK

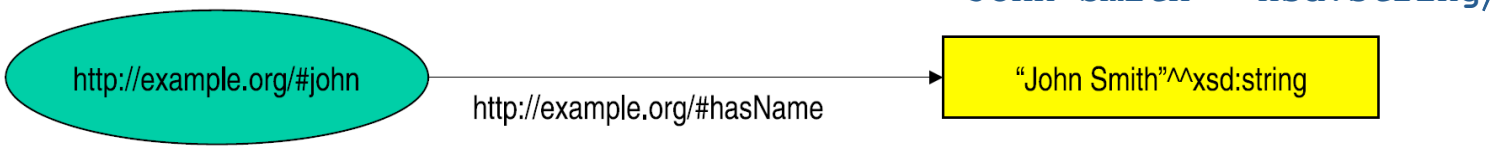
- A resource may be:
  - Web page (e.g. `http://www.w3.org`)
  - A person (e.g. `http://www.fensel.com`)
  - A book (e.g. `urn:isbn:0-345-33971-1`)
  - Anything denoted with a URI!
- A URI is an *identifier* and **not** a location on the Web
- RDF allows making statements about resources:
  - `http://www.w3.org`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

Literals

STI · INNSBRUCK

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

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



Datatypes

STI · INNSBRUCK

- One pre-defined datatype: `rdf:XMLLiteral`
  - Used for embedding XML in RDF

- Recommended datatypes are XML Schema datatypes, e.g.:
  - `xsd:string`
  - `xsd:integer`
  - `xsd:float`
  - `xsd:anyURI`
  - `xsd:boolean`

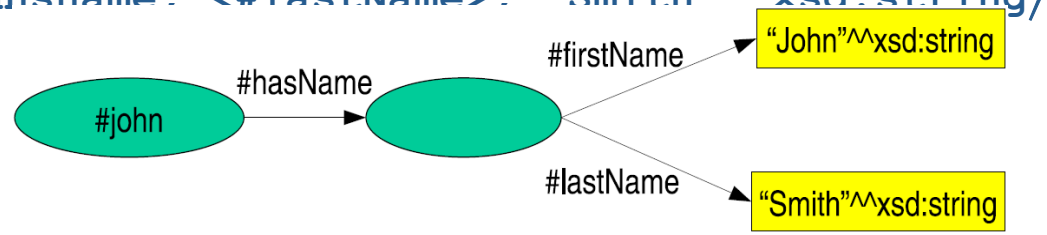
Blank Nodes

STI · INNSBRUCK

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

- For example:

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



Reification

STI · INNSBRUCK

- Reification: statements about statements

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

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

Reification

STI · INNSBRUCK

- Reification: statements about statements

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

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

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

Reification

STI · INNSBRUCK

- Reification: statements about statements

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

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

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

RDF Vocabulary

STI · INNSBRUCK

- RDF defines a number of resources and properties
- We have already seen: `rdf:XMLLiteral`, `rdf:type`, . . .
- RDF vocabulary is defined in the namespace:
  `http://www.w3.org/1999/02/22-rdf-syntax-ns#`

- Classes:
  - `rdf:Property`, `rdf:Statement`, `rdf:XMLLiteral`
  - `rdf:Seq`, `rdf:Bag`, `rdf:Alt`, `rdf:List`
- Properties:
  - `rdf:type`, `rdf:subject`, `rdf:predicate`, `rdf:object`,
  - `rdf:first`, `rdf:rest`, `rdf:_n`
  - `rdf:value`
- Resources:
  - `rdf:nil`

RDF Vocabulary

STI · INNSBRUCK

- Typing using **`rdf:type`**:

  `<A, rdf:type, B>`

  "A belongs to class B"

- All properties belong to class **`rdf:Property`**:

  `<P, rdf:type, rdf:Property>`

  "P is a property"

  `<rdf:type, rdf:type, rdf:Property>`

  "rdf:type is a property"

RDF Containers

STI · INNSBRUCK

- Grouping property values:

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

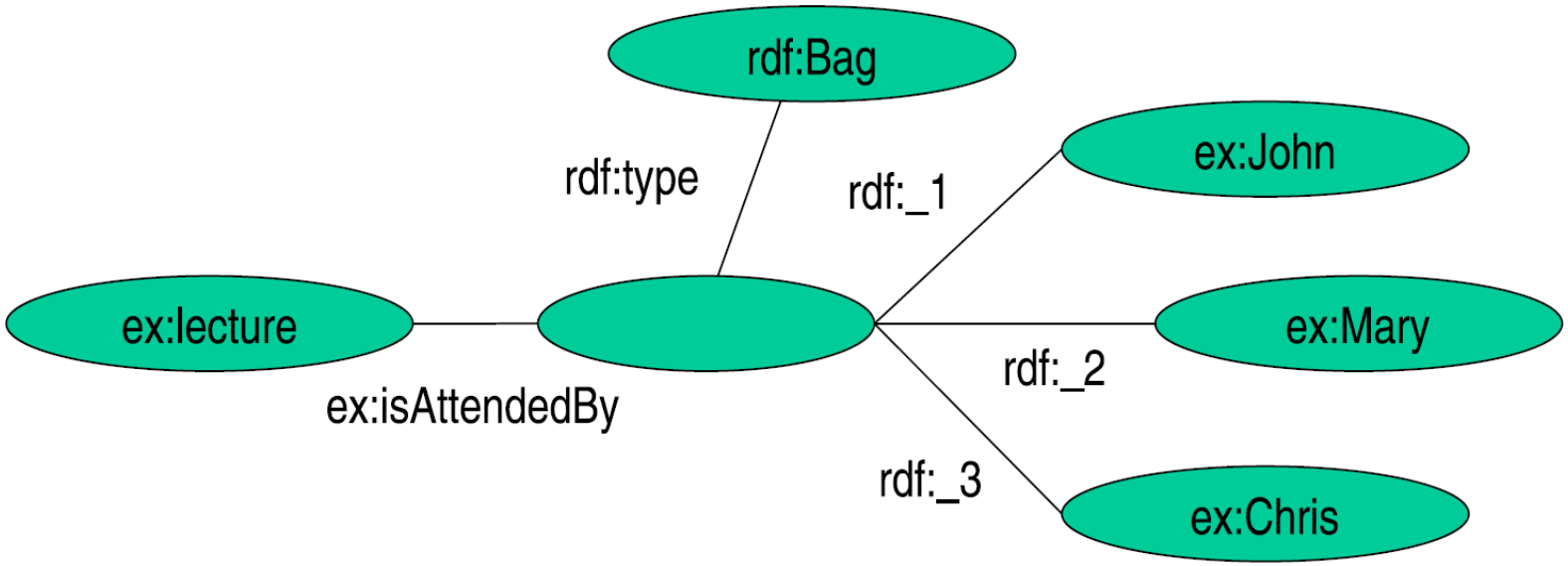
“[RDF-Concepts] is edited by Graham and Jeremy (in that order)” **`Seq`**

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

RDF Containers: Bag

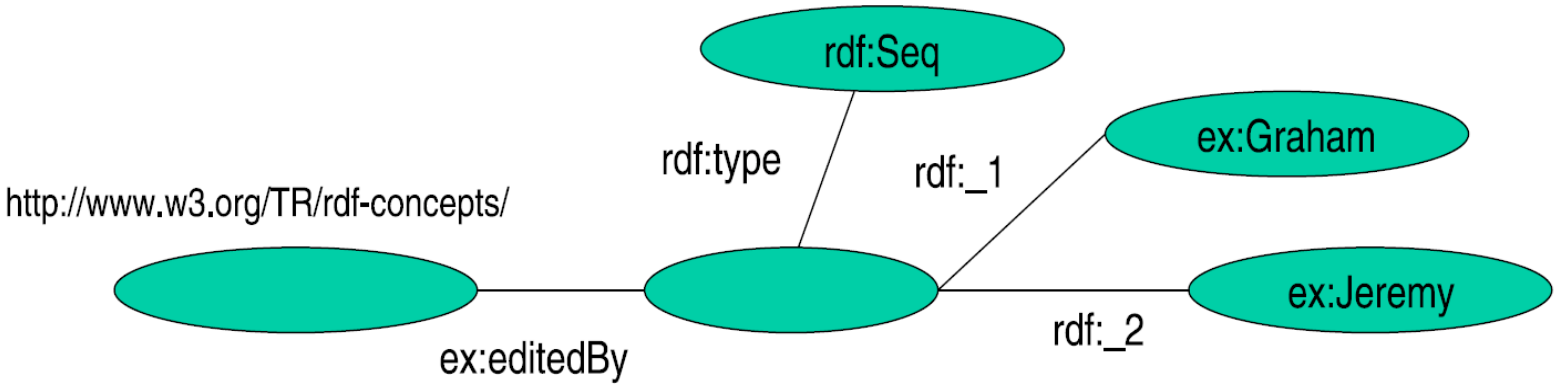
STI · INNSBRUCK

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



RDF Containers: Seq

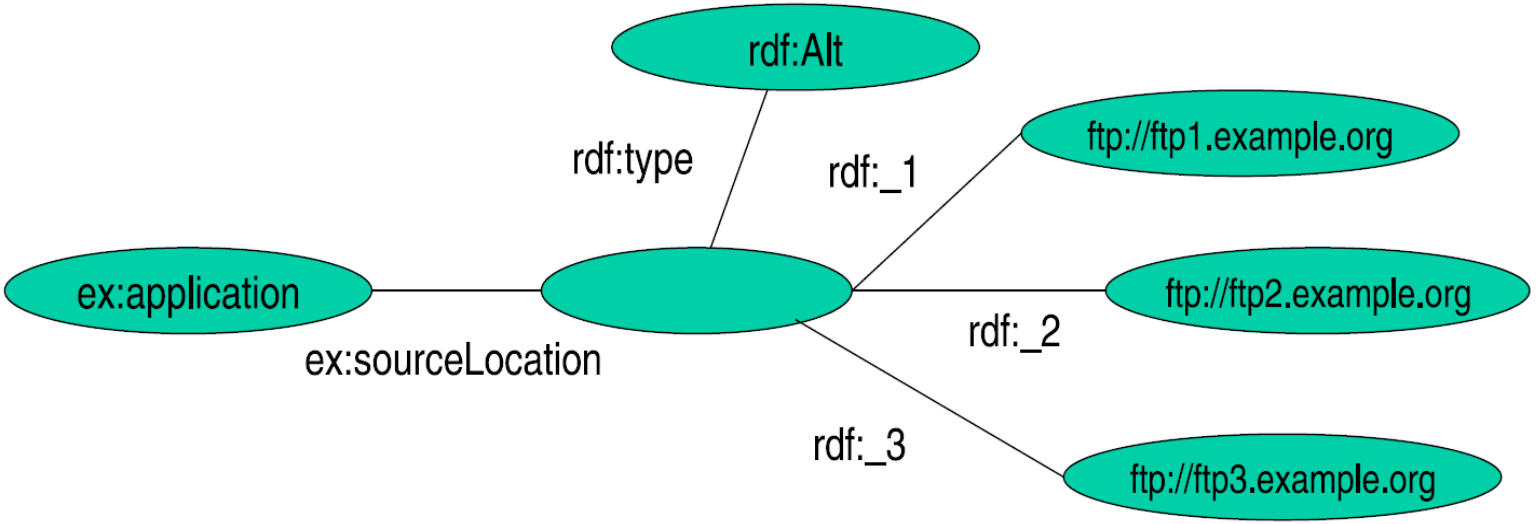
"[RDF-Concepts] is edited by Graham and Jeremy (in that order)"



RDF Containers: Alt

STI · INNSBRUCK

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



RDF Containers 2

STI · INNSBRUCK

- Three types of containers:
  - `rdf:Bag` - unordered set of items
  - `rdf:Seq` - ordered set of items
  - `rdf:Alt` - set of alternatives
- Every container has a triple declaring the `rdf:type`

- Items in the container are denoted with
  - `rdf:_1, rdf:_2, . . . ,rdf:_n`

RDF Containers 2

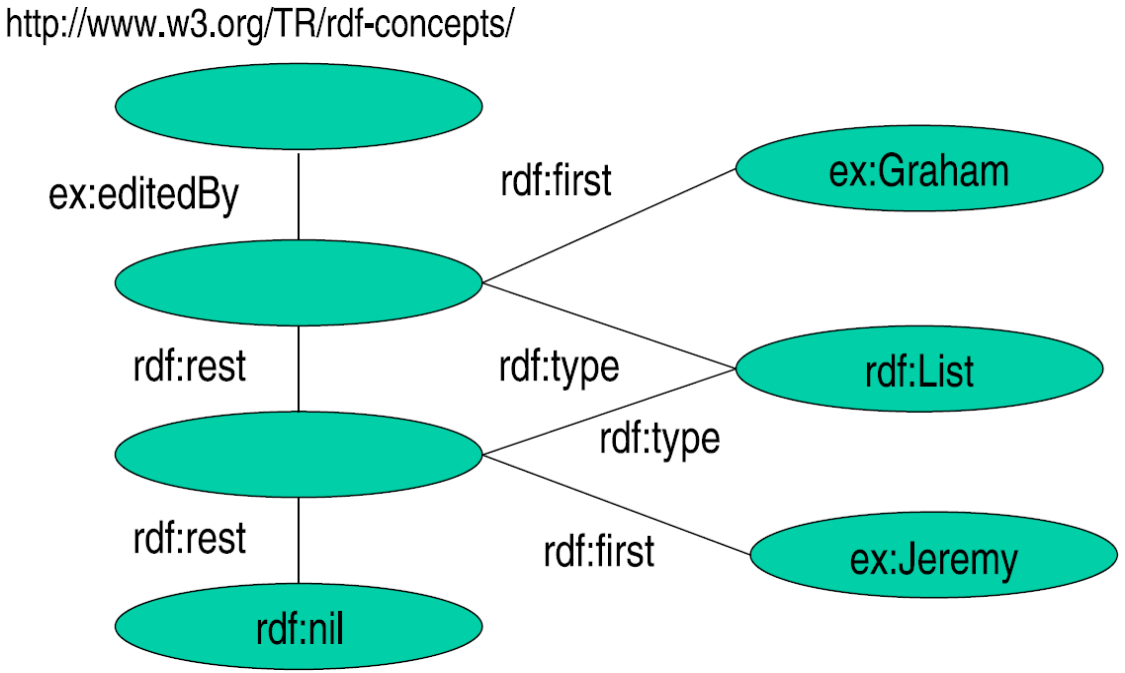
STI · INNSBRUCK

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

RDF Collections

STI · INNSBRUCK

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



RDF/XML 1

STI · INNSBRUCK

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

RDF/XML 2

STI · INNSBRUCK

<#john, #hasName, "John">

<#john, #marriedTo, #mary>

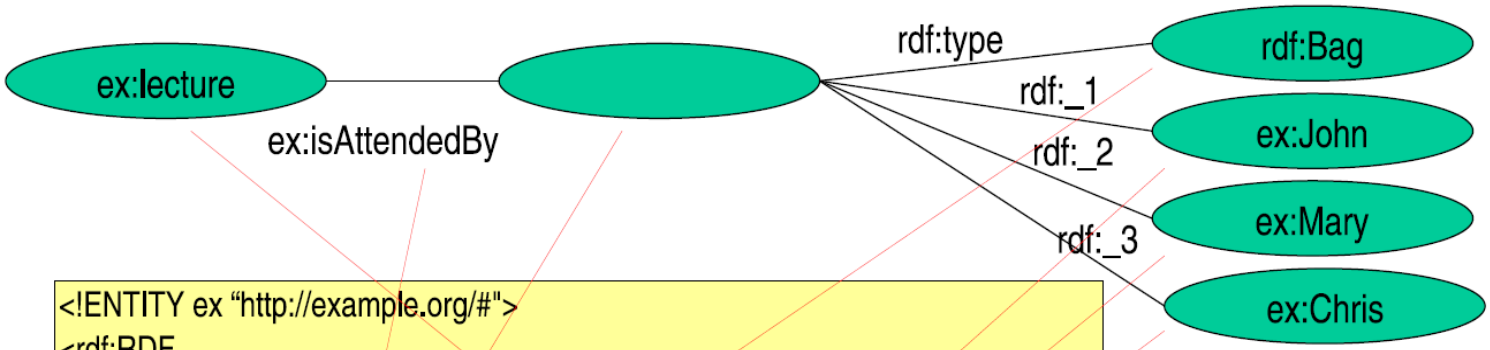
```
<!ENTITY ex "http://example.org/#">

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

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

</rdf:RDF>
```

Head

Body

Foot

RDF/XML 3




```
<!ENTITY ex "http://example.org/#">
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:ex="http://example.org#">
    <rdf:Description rdf:about="&ex;lecture">
        <ex:isAttendedBy>
            <rdf:Bag>
                <rdf:li rdf:resource="&ex;John"/>
                <rdf:li rdf:resource="&ex;Mary"/>
                <rdf:li rdf:resource="&ex;Chris"/>
            </rdf:Bag>
        </ex:isAttendedBy>
    </rdf:Description>
```

How to represent the semantics of data models

THE RDF SCHEMA (RDFS)

RDF Vocabulary Description Language 1

STI · INNSBRUCK

- Types in RDF:

  `<#john, rdf:type, #Student>`

- What is a "`#Student`"?

- A language for defining RDF types:
  - Define classes:
    - "`#Student` *is a class*"
  - Relationships between classes:
    - "`#Student` *is a sub-class of* `#Person`"
  - Properties of classes:
    - "`#Person` *has a property* `hasName`"

- RDF Schema is such a language

RDF Vocabulary Description Language 2

STI · INNSBRUCK

- Classes:
  `<#Student, rdf:type, #rdfs:Class>`
- Class hierarchies:
  `<#Student, rdfs:subClassOf, #Person>`

- Properties:
  `<#hasName, rdf:type, rdf:Property>`
- Property hierarchies:
  `<#hasMother, rdfs:subPropertyOf, #hasParent>`

- Associating properties with classes (a):
  - "The property `#hasName` only applies to `#Person`"
    `<#hasName, rdfs:domain, #Person>`

- Associating properties with classes (b):
  - "The type of the property `#hasName` is `#xsd:string`"
    `<#hasName, rdfs:range, xsd:string>`

RDF Vocabulary Revisited

STI · INNSBRUCK

- Classes:
  - `rdf:Property`, `rdf:Statement`, `rdf:XMLLiteral`
  - `rdf:Seq`, `rdf:Bag`, `rdf:Alt`, `rdf:List`
- Properties:
  - `rdf:type`, `rdf:subject`, `rdf:predicate`, `rdf:object`,
  - `rdf:first`, `rdf:rest`, `rdf:_n`
  - `rdf:value`
- Resources:
  - `rdf:nil`

RDFS Vocabulary

STI · INNSBRUCK

- RDFS Extends the RDF Vocabulary
- RDFS vocabulary is defined in the namespace:

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

RDFS Classes

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

RDFS Properties

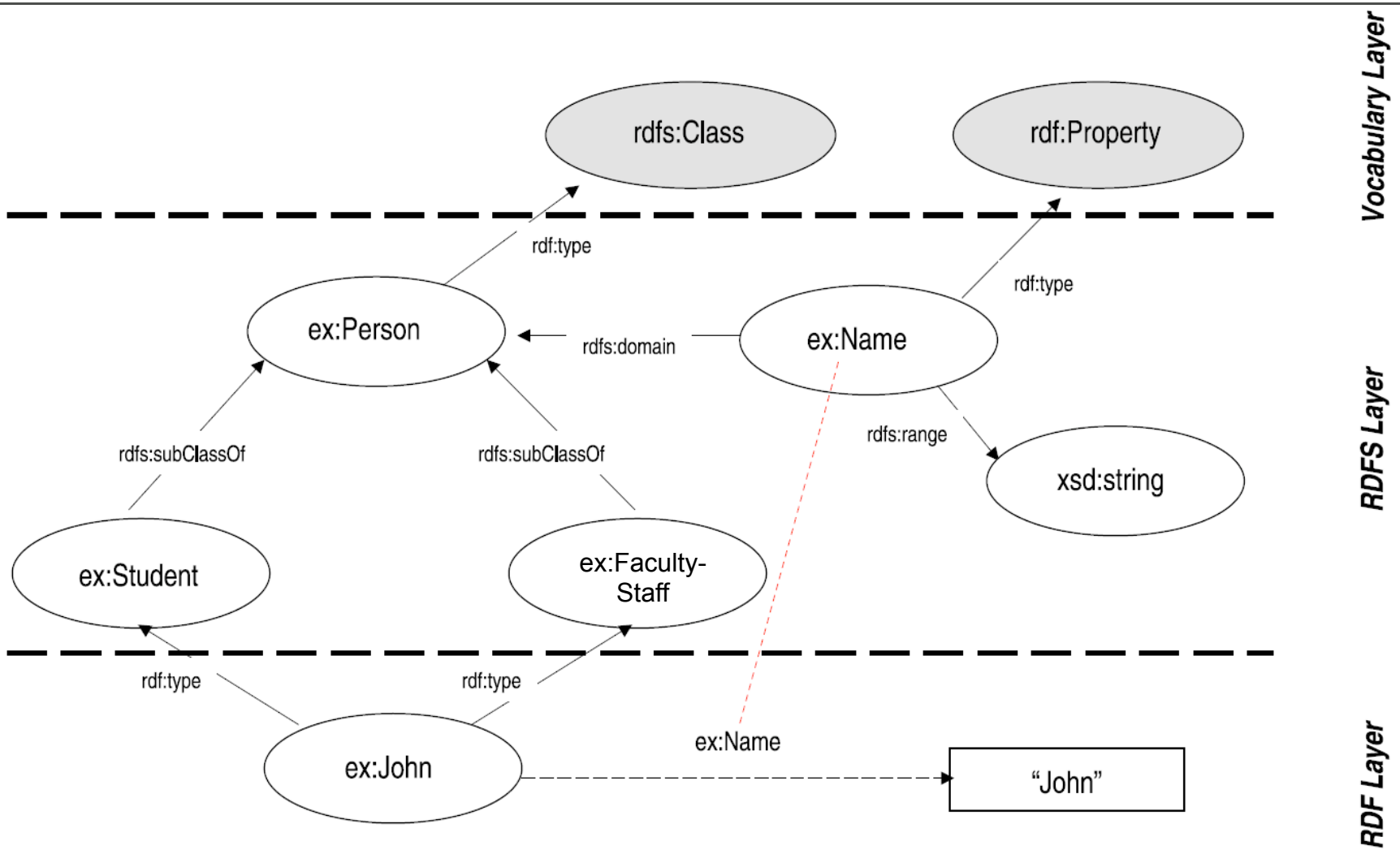
- rdfs:domain
- rdfs:range
- rdfs:subPropertyOf
- rdfs:subClassOf
- rdfs:member
- rdfs:seeAlso
- rdfs:isDefinedBy
- rdfs:comment
- rdfs:label

RDFS Principles

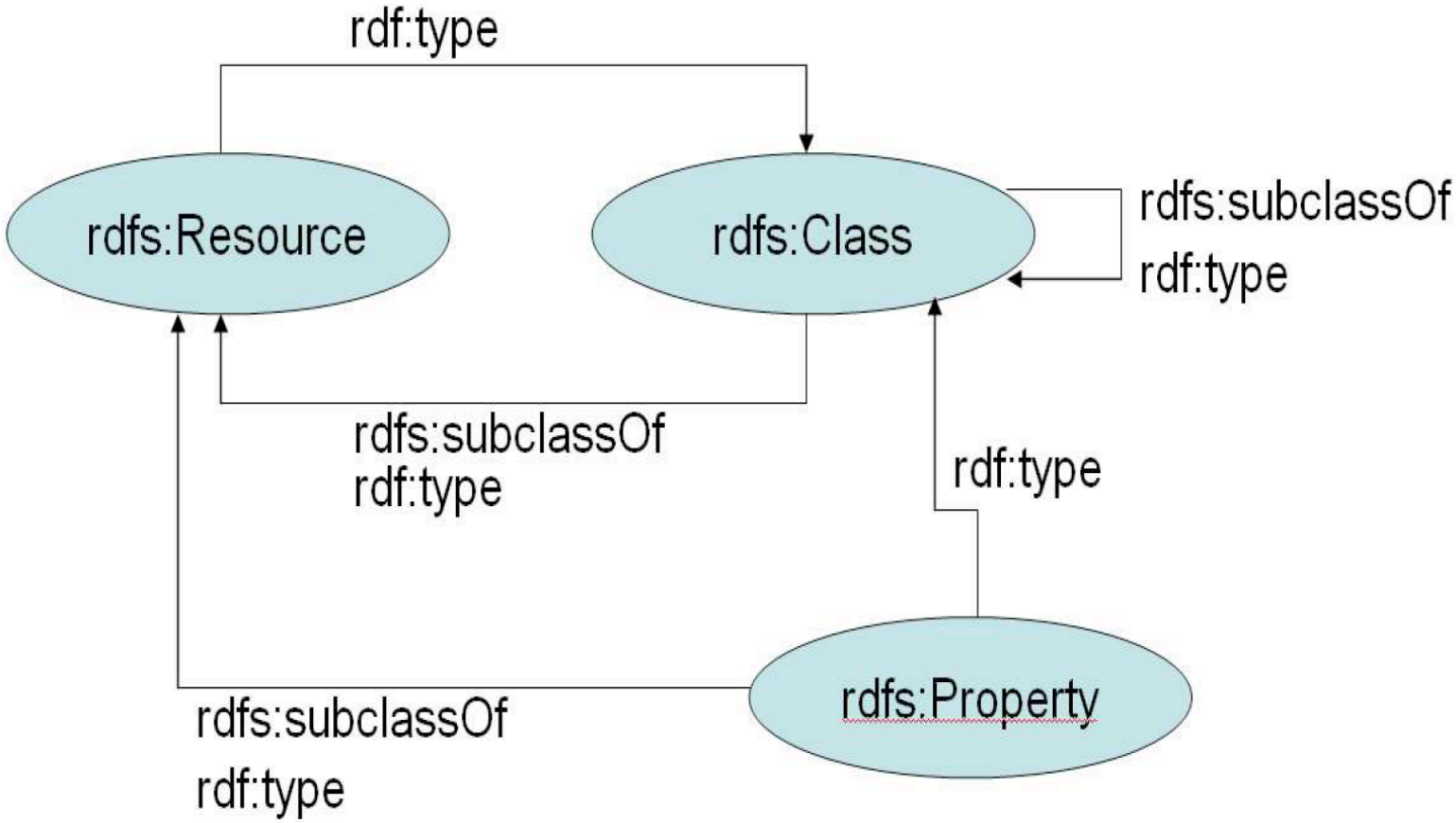
STI · INNSBRUCK

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

RDFS Example



RDFS Vocabulary Example



RDFS Metadata Properties

STI · INNSBRUCK

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

RDF Literals Revisited

STI · INNSBRUCK

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

Literals in RDFS

STI · INNSBRUCK

- Each literal is an `rdfs:Literal`
- Say, we have: `<#john, #hasName, "John">`
- Does this mean:

  `<"John", rdf:type, rdfs:Literal>`

  – No! Literals may not occur as subject
- Add:
  - `<#john, #hasName, _:X>`
  - `<_:X, rdf:type, rdfs:Literal>`

Semantics

STI · INNSBRUCK

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

Subgraph

STI · INNSBRUCK

- *E* is a subgraph of *S* if and only if *E* predicates are a subset of *S* predicates
  - ⟨<#john>, <#hasName>, _:johnsname⟩
  - ⟨<_:johnsname, <#firstName>, "John"^^xsd:string⟩
  - ⟨<_:johnsname, <#lastName>, "Smith"^^xsd:string⟩
- Subgraphs:
  - ⟨<#john>,<#hasName>, :johnsname⟩
  - ⟨_:johnsname, <#firstName>, "John"^^xsd:string⟩
  - ⟨_:johnsname, <#firstName>, "John"^^xsd:string⟩
  - ⟨_:johnsname, <#lastName>, "Smith"^^xsd:string⟩
  - ⟨<#john>, <#hasName>, _:johnsname⟩

Instance

STI · INNSBRUCK

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

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

- Instances:
  - ⟨<#john>, <#hasName>, <#abc>⟩
  - ⟨<#abc>, <#firstName>, "John"^^xsd:string⟩
  - ⟨<#abc>, <#lastName>, "Smith"^^xsd:string⟩

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

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

- Every graph is an instance of itself!

Entailment

STI · INNSBRUCK

- *S* entails *E* if *E* logically follows from *S*
  - Written: `S |= E`
- A graph entails all it subgraphs
  - If *S'* is a subgraph of *S*: `S |= S'`
- All instances of a graph *S* entail *S*
  - If *S"* is an instance of *S*: `S" |= S`

RDFS Entailment

STI · INNSBRUCK

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

entails

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

<http://example.org/#hasName> rdfs:domain <http://example.org/#Student>
<http://example.org/#mary> <http://example.org/#hasName> “Mary”

entails

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

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

entails

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

RDFS Entailment

STI · INNSBRUCK

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

entails

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

<http://example.org/#hasName> rdfs:domain <http://example.org/#Student>
<http://example.org/#mary> <http://example.org/#hasName> “Mary”

entails

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

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

entails

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

RDFS Entailment

STI · INNSBRUCK

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

entails

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

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

entails

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

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

entails

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

Entailment Rules

STI · INNSBRUCK

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

RDF Entailment

STI · INNSBRUCK

- if *E* contains `<A, B, C>` then add

  `<B, rdf:type, rdf:Property>`

- if *E* contains `<A, B, l>` (l is a valid XML literal) then add

  `<_:X, rdf:type, rdf:XMLLiteral>`

  where `_:x` identifies to blank node allocated to `l`

RDFS Entailment 1

STI · INNSBRUCK

- everything in the subject is a resource
  - if *E* contains `<A,B,C>` then add `<A, rdf:type, rdfs:Resource>`
- every non-literal in the object is a resource
  - if *E* contains `<A,B,C>` (C is not a literal) then add `<C, rdf:type, rdfs:Resource>`
- every class is subclass of `rdfs:Resource`
  - if *E* contains `<A, rdf:type, rdfs:Class>` then add `<A, rdfs:subClassOf, rdfs:Resource>`
- inheritance:
  - if *E* contains `<A, rdf:type, B>`, `<B, rdfs:subClassOf, C>` then add `<A, rdf:type, C>`
- `rdfs:subClassOf` is transitive
  - if *E* contains `<A, rdfs:subClassOf, B>`, `<B, rdfs:subClassOf, C>` then add `<A, rdfs:subClassOf, C>`

RDFS Entailment 2

STI · INNSBRUCK

- `rdfs:subClassOf` is reflexive
  - if *E* contains `<A, rdf:type, rdfs:Class>` then add `<A, rdfs:subClassOf, A>`
- `rdfs:subPropertyOf` is transitive
  - if *E* contains `<A, rdfs:subPropertyOf, B>`, `<B, rdfs:subPropertyOf, C>` then add `<A, rdfs:subPropertyOf, C>`
- `rdfs:subPropertyOf` is reflexive
  - if *E* contains `<P, rdf:type, rdf:Property>` then add `<P, rdfs:subPropertyOf, P>`
- domain of properties
  - if *E* contains `<P, rdfs:domain, C>`, `<A, P, B>` then add `<A, rdf:type, C>`
- range of properties
  - if *E* contains `<P, rdfs:range, C>`, `<A, P, B>` then add `<B, rdf:type, C>`

RDFS Entailment 3

STI · INNSBRUCK

- every literal is a member of `rdfs:Literal`
  - if *E* contains `<A, B, l>` (l is a plain literal) then add `<_:X, rdf:type, rdfs:Literal>`

- every datatype is subclass of `rdfs:Literal`
  - if *E* contains `<A, rdf:type, rdfs:Datatype>` then add `<A, rdfs:subClassOf, rdfs:Literal>`

More on Literals

STI · INNSBRUCK

Recall:

if E **contains** `<A, B, l>` (l is a valid XML literal) **then add**

`<_:X, rdf:type, rdf:XMLLiteral>`

every literal is a member of rdfs:Literal:

if E **contains** `<A, B, l>` (l is a plain literal) **then add**

`<_:X, rdf:type, rdfs:Literal>`

---

allocating blank nodes to literals:

if E **contains** `<A, B, l>` (`l` is a literal) **then add** `<A, B, _:n>`

`_:n` is allocated to `l`

"dereferencing" blank nodes:

if E **contains** `<A, B,_:n>` (`_:n` is allocated to a literal `l`) **then add** `<A, B, l>`

Tool Support for RDF/RDFS

STI · INNSBRUCK

- Ontology editors
  - Protégé (http://protege.stanford.edu/)
  - OilED (http://oiled.man.ac.uk/)
- Browser
  - /facet (http://slashfacet.semanticweb.org/)
- APIs
  - JRDF – Java (http://jrdf.sourceforge.net/)
  - Jena – Java (http://jena.sourceforge.net/)
  - RAP - PHP (http://www.seasr.org/wp-content/plugins/meandre/rdfapi-php/doc/)
  - Redland RDF – C (http://librdf.org/)
- Validator
  - W3C Validator (http://www.w3.org/RDF/Validator/)

An example of usage of RDF and RDF(S)

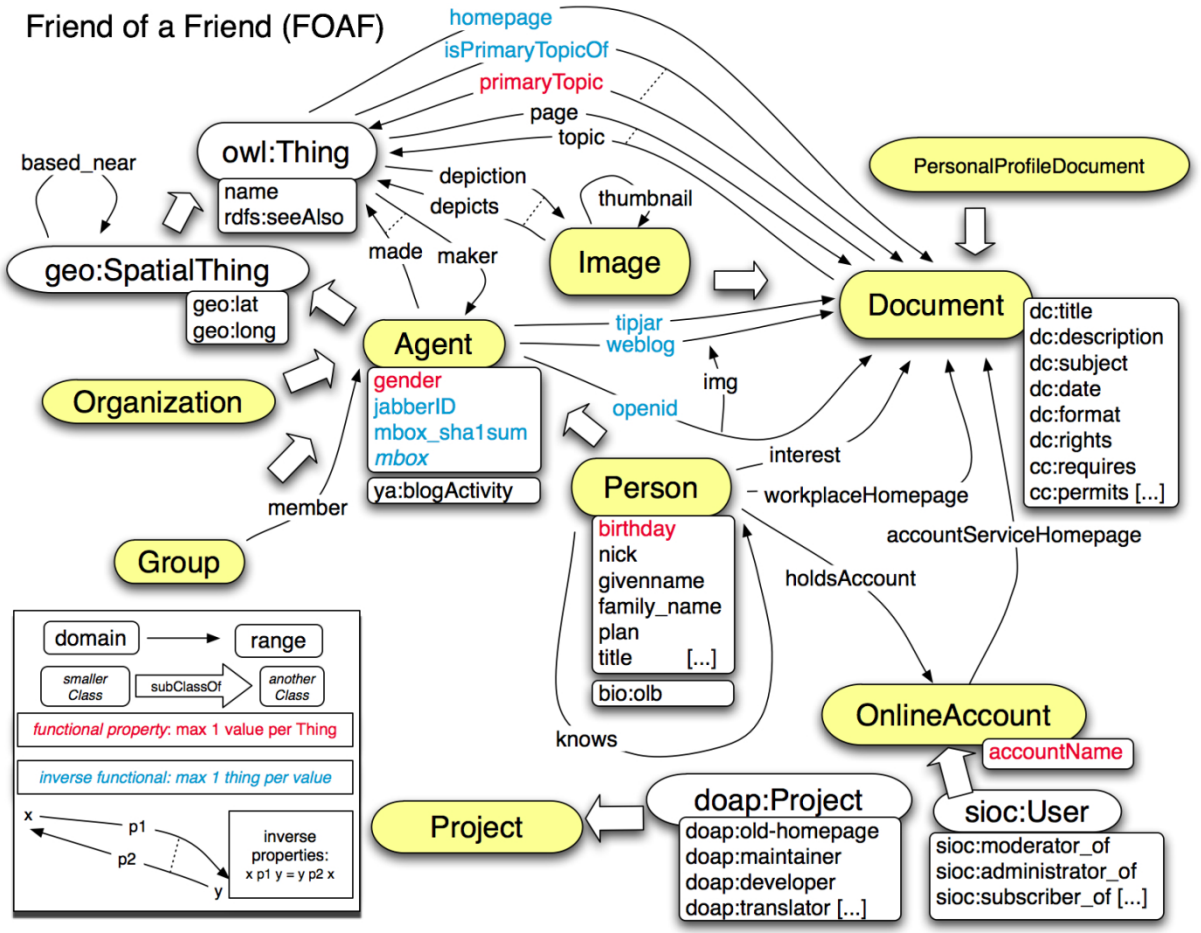
ILLUSTRATION BY A LARGER EXAMPLE

Friend of a Friend (FOAF)

STI · INNSBRUCK

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

FOAF Schema



[http://www.foaf-project.org/]

FOAF RDF Example

STI · INNSBRUCK

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:foaf="http://
    xmlns.com/foaf/0.1/">
    <foaf:Person rdf:ID="me">
        <foaf:name>Dieter Fensel</foaf:name>
        <foaf:title>Univ.-Prof. Dr.</foaf:title>
        <foaf:givenname>Dieter</foaf:givenname> <foaf:family_name>Fensel</
    foaf:family_name>
    <foaf:mbox_sha1sum>773a221a09f1887a24853c9de06c3480e714278a</
    foaf:mbox_sha1sum>
    <foaf:homepage rdf:resource="http://www.fensel.com "/>
    <foaf:depiction rdf:resource="http://www.deri.at/fileadmin/images/photos/
    dieter_fensel.jpg"/> <foaf:phone rdf:resource="tel:+43-512-507-6488"/>
    <foaf:workplaceHomepage rdf:resource="http://www.sti-innsbruck.at"/>
    <foaf:workInfoHomepage rdf:resource="http://www.sti-innsbruck.at/
    about/team/details/?uid=40"/> </foaf:Person>
</rdf:RDF>
```

EXTENSIONS

Rules and RDF(S)

STI · INNSBRUCK

- RDF(S) by itself is not providing any instrument to define personalized entailment rules
  - The entailment process is driven by RDF(S) Semantics
  - This is not enough in many practical contexts
- RDF can be extend to add rule support
  - RULE-ML based extensions
  - Horn Logic based extensions
  - OWL Horst (include a fragment of DL as well)
  - OWLIM (include a fragment of DL as well)
  - SWRL (full support to OWL, but not decidable)

SUMMARY

Summary

STI · INNSBRUCK

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

Summary

STI · INNSBRUCK

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

References

STI · INNSBRUCK

- Mandatory reading
  - Semantic Web Primer
    - Chapter 3 (only Sections 3.1 to 3.6)
- Further reading
  - RDF Primer
    - http://www.w3.org/TR/REC-rdf-syntax/
  - RDF Vocabulary Description Language 1.0: RDF Schema
    - http://www.w3.org/TR/rdf-schema/
- Wikipedia
  - http://en.wikipedia.org/wiki/Resource_Description_Framework
  - http://en.wikipedia.org/wiki/RDF_schema
  - http://en.wikipedia.org/wiki/Turtle_(syntax)
  - http://en.wikipedia.org/wiki/Notation_3
  - http://en.wikipedia.org/wiki/N-Triples

Next Lecture

STI · INNSBRUCK

#	Title
1	Introduction
2	Semantic Web architecture
3	RDF and RDFs
4	Web of hypertext (RDFa, Microformats) and Web of data
5	**Semantic annotations**
6	**Repositories and SPARQL**
7	OWL
8	RIF
9	Web-scale reasoning
10	Social Semantic Web
11	Ontologies and the Semantic Web
12	SWS
13	Tools
14	Applications
15	Exam

Questions?

STI · INNSBRUCK