

An introduction to the semantic web technologies

And their use within the @Web platform

Leandro Lovisolo

leandro.lovisolo@supagro.inra.fr

INRA SupAgro and INRIA GraphiK
Montpellier, France

September 23, 2015

Outline of the presentation

- ▶ What's an ontology?
- ▶ RDF
- ▶ RDFS
- ▶ OWL
- ▶ SKOS
- ▶ SPARQL
- ▶ The n-ary relationship pattern used in **@Web**
- ▶ Examples of tables in scientific documents annotated using n-ary relationships in **@Web**

What's an ontology?

What's an ontology?

It's a formal description of a domain of interest based on:

What's an ontology?

It's a formal description of a domain of interest based on:

- ▶ a set of *individuals* (also called entities or objects),

What's an ontology?

It's a formal description of a domain of interest based on:

- ▶ a set of *individuals* (also called entities or objects),
- ▶ a set of *classes* of individuals, and

What's an ontology?

It's a formal description of a domain of interest based on:

- ▶ a set of *individuals* (also called entities or objects),
- ▶ a set of *classes* of individuals, and
- ▶ a set of *relationships* (sometimes called properties) between these individuals;

What's an ontology?

It's a formal description of a domain of interest based on:

- ▶ a set of *individuals* (also called entities or objects),
- ▶ a set of *classes* of individuals, and
- ▶ a set of *relationships* (sometimes called properties) between these individuals;

and a set of logical constraints to specify, among other things:

What's an ontology?

It's a formal description of a domain of interest based on:

- ▶ a set of *individuals* (also called entities or objects),
- ▶ a set of *classes* of individuals, and
- ▶ a set of *relationships* (sometimes called properties) between these individuals;

and a set of logical constraints to specify, among other things:

- ▶ class membership,

What's an ontology?

It's a formal description of a domain of interest based on:

- ▶ a set of *individuals* (also called entities or objects),
- ▶ a set of *classes* of individuals, and
- ▶ a set of *relationships* (sometimes called properties) between these individuals;

and a set of logical constraints to specify, among other things:

- ▶ class membership,
- ▶ subclass/subproperty relationships,

What's an ontology?

It's a formal description of a domain of interest based on:

- ▶ a set of *individuals* (also called entities or objects),
- ▶ a set of *classes* of individuals, and
- ▶ a set of *relationships* (sometimes called properties) between these individuals;

and a set of logical constraints to specify, among other things:

- ▶ class membership,
- ▶ subclass/subproperty relationships,
- ▶ domain/range restrictions on properties,

What's an ontology?

It's a formal description of a domain of interest based on:

- ▶ a set of *individuals* (also called entities or objects),
- ▶ a set of *classes* of individuals, and
- ▶ a set of *relationships* (sometimes called properties) between these individuals;

and a set of logical constraints to specify, among other things:

- ▶ class membership,
- ▶ subclass/subproperty relationships,
- ▶ domain/range restrictions on properties,
- ▶ cardinality constraints,

What's an ontology?

It's a formal description of a domain of interest based on:

- ▶ a set of *individuals* (also called entities or objects),
- ▶ a set of *classes* of individuals, and
- ▶ a set of *relationships* (sometimes called properties) between these individuals;

and a set of logical constraints to specify, among other things:

- ▶ class membership,
- ▶ subclass/subproperty relationships,
- ▶ domain/range restrictions on properties,
- ▶ cardinality constraints,
- ▶ class union/intersection/disjointness constraints,

What's an ontology?

It's a formal description of a domain of interest based on:

- ▶ a set of *individuals* (also called entities or objects),
- ▶ a set of *classes* of individuals, and
- ▶ a set of *relationships* (sometimes called properties) between these individuals;

and a set of logical constraints to specify, among other things:

- ▶ class membership,
- ▶ subclass/subproperty relationships,
- ▶ domain/range restrictions on properties,
- ▶ cardinality constraints,
- ▶ class union/intersection/disjointness constraints,
- ▶ etc.

Web resources, URI, namespaces

A *resource* is anything that can be referred to: a web page, a person, a city, a university course, etc.

Web resources, URI, namespaces

A *resource* is anything that can be referred to: a web page, a person, a city, a university course, etc.

Resources are identified by *URIs*, for example:

- ▶ `http://example.com/MyOntology`,
- ▶ `http://example.com/MyOntology#Leandro`,
- ▶ `http://example.com/MyOntology#Pizza`,
- ▶ etc.

Web resources, URI, namespaces

A *resource* is anything that can be referred to: a web page, a person, a city, a university course, etc.

Resources are identified by *URIs*, for example:

- ▶ `http://example.com/MyOntology`,
- ▶ `http://example.com/MyOntology#Leandro`,
- ▶ `http://example.com/MyOntology#Pizza`,
- ▶ etc.

To avoid carrying long URIs, *namespaces* are used.

Web resources, URI, namespaces

A *resource* is anything that can be referred to: a web page, a person, a city, a university course, etc.

Resources are identified by *URIs*, for example:

- ▶ `http://example.com/MyOntology`,
- ▶ `http://example.com/MyOntology#Leandro`,
- ▶ `http://example.com/MyOntology#Pizza`,
- ▶ etc.

To avoid carrying long URIs, *namespaces* are used. Thus,

Web resources, URI, namespaces

A *resource* is anything that can be referred to: a web page, a person, a city, a university course, etc.

Resources are identified by *URIs*, for example:

- ▶ `http://example.com/MyOntology`,
- ▶ `http://example.com/MyOntology#Leandro`,
- ▶ `http://example.com/MyOntology#Pizza`,
- ▶ etc.

To avoid carrying long URIs, *namespaces* are used. Thus,

- ▶ `http://example.com/MyOntology`

Web resources, URI, namespaces

A *resource* is anything that can be referred to: a web page, a person, a city, a university course, etc.

Resources are identified by *URIs*, for example:

- ▶ `http://example.com/MyOntology`,
- ▶ `http://example.com/MyOntology#Leandro`,
- ▶ `http://example.com/MyOntology#Pizza`,
- ▶ etc.

To avoid carrying long URIs, *namespaces* are used. Thus,

- ▶ `http://example.com/MyOntology` becomes

Web resources, URI, namespaces

A *resource* is anything that can be referred to: a web page, a person, a city, a university course, etc.

Resources are identified by *URIs*, for example:

- ▶ `http://example.com/MyOntology`,
- ▶ `http://example.com/MyOntology#Leandro`,
- ▶ `http://example.com/MyOntology#Pizza`,
- ▶ etc.

To avoid carrying long URIs, *namespaces* are used. Thus,

- ▶ `http://example.com/MyOntology` becomes
- ▶ `example:MyOntology`

Web resources, URI, namespaces

A *resource* is anything that can be referred to: a web page, a person, a city, a university course, etc.

Resources are identified by *URIs*, for example:

- ▶ `http://example.com/MyOntology`,
- ▶ `http://example.com/MyOntology#Leandro`,
- ▶ `http://example.com/MyOntology#Pizza`,
- ▶ etc.

To avoid carrying long URIs, *namespaces* are used. Thus,

- ▶ `http://example.com/MyOntology` becomes
- ▶ `example:MyOntology` abbreviated as

Web resources, URI, namespaces

A *resource* is anything that can be referred to: a web page, a person, a city, a university course, etc.

Resources are identified by *URIs*, for example:

- ▶ `http://example.com/MyOntology`,
- ▶ `http://example.com/MyOntology#Leandro`,
- ▶ `http://example.com/MyOntology#Pizza`,
- ▶ etc.

To avoid carrying long URIs, *namespaces* are used. Thus,

- ▶ `http://example.com/MyOntology` becomes
- ▶ `example:MyOntology` abbreviated as
- ▶ `:MyOntology`

Web resources, URI, namespaces

A *resource* is anything that can be referred to: a web page, a person, a city, a university course, etc.

Resources are identified by *URIs*, for example:

- ▶ `http://example.com/MyOntology`,
- ▶ `http://example.com/MyOntology#Leandro`,
- ▶ `http://example.com/MyOntology#Pizza`,
- ▶ etc.

To avoid carrying long URIs, *namespaces* are used. Thus,

- ▶ `http://example.com/MyOntology` becomes
- ▶ `example:MyOntology` abbreviated as
- ▶ `:MyOntology`

if `example` is the default namespace.

RDF

A simple language for describing *annotations* about Web resources identified by URIs, from now on referred to as **facts**.

RDF

Triplets

Facts are stated as *RDF triplets*.

RDF

Triplets

Facts are stated as *RDF triplets*.

A triplet is made of a *subject*, a *predicate* and an *object*.

RDF

Triplets

Facts are stated as *RDF triplets*.

A triplet is made of a *subject*, a *predicate* and an *object*.

Some examples:

RDF

Triplets

Facts are stated as *RDF triplets*.

A triplet is made of a *subject*, a *predicate* and an *object*.

Some examples:

- ▶ `<:Dupond :Leads :InfoDept>`

RDF

Triplets

Facts are stated as *RDF triplets*.

A triplet is made of a *subject*, a *predicate* and an *object*.

Some examples:

- ▶ $\langle \text{:Dupond} \text{ :Leads } \text{:InfoDept} \rangle$
- ▶ $\langle \text{:Dupond} \text{ :TeachesIn } \text{:UE111} \rangle$

RDF

Triplets

Facts are stated as *RDF triplets*.

A triplet is made of a *subject*, a *predicate* and an *object*.

Some examples:

- ▶ `<:Dupond :Leads :InfoDept>`
- ▶ `<:Dupond :TeachesIn :UE111>`
- ▶ `<:Dupond :TeachesTo :Pierre>`

RDF

Triplets

Facts are stated as *RDF triplets*.

A triplet is made of a *subject*, a *predicate* and an *object*.

Some examples:

- ▶ `<:Dupond :Leads :InfoDept>`
- ▶ `<:Dupond :TeachesIn :UE111>`
- ▶ `<:Dupond :TeachesTo :Pierre>`
- ▶ `<:Pierre :EnrolledIn :InfoDept>`

RDF

Triplets

Facts are stated as *RDF triplets*.

A triplet is made of a *subject*, a *predicate* and an *object*.

Some examples:

- ▶ `<:Dupond :Leads :InfoDept>`
- ▶ `<:Dupond :TeachesIn :UE111>`
- ▶ `<:Dupond :TeachesTo :Pierre>`
- ▶ `<:Pierre :EnrolledIn :InfoDept>`
- ▶ `<:Pierre :RegisteredTo :UE111>`

RDF

Triplets

Facts are stated as *RDF triplets*.

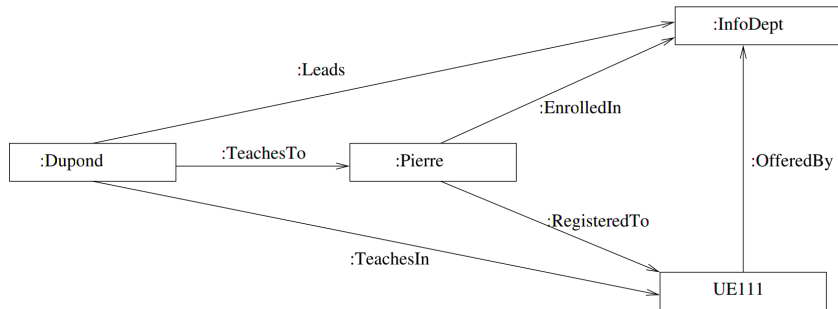
A triplet is made of a *subject*, a *predicate* and an *object*.

Some examples:

- ▶ `<:Dupond :Leads :InfoDept>`
- ▶ `<:Dupond :TeachesIn :UE111>`
- ▶ `<:Dupond :TeachesTo :Pierre>`
- ▶ `<:Pierre :EnrolledIn :InfoDept>`
- ▶ `<:Pierre :RegisteredTo :UE111>`
- ▶ `<:UE111 :OfferedBy :InfoDept>`

RDF

Graph representation



```
<:Dupond :Leads :InfoDept>  
<:Dupond :TeachesIn :UE111>  
<:Dupond :TeachesTo :Pierre>  
<:Pierre :EnrolledIn :InfoDept>  
<:Pierre :RegisteredTo :UE111>  
<:UE111 :OfferedBy :InfoDept>
```

RDF

Syntax

There are many different syntaxes for writing RDF triplets, including:

RDF

Syntax

There are many different syntaxes for writing RDF triplets, including:

- ▶ XML (as used in **@Web**),

RDF

Syntax

There are many different syntaxes for writing RDF triplets, including:

- ▶ XML (as used in **@Web**),
- ▶ Turtle,
- ▶ N-Triples,
- ▶ N-Quads,
- ▶ etc.

RDF

Syntax

There are many different syntaxes for writing RDF triplets, including:

- ▶ XML (as used in **@Web**),
- ▶ Turtle,
- ▶ N-Triples,
- ▶ N-Quads,
- ▶ etc.

However, we're going to focus on the abstract `<subject, predicate, object>` syntax during this presentation.

RDFS

The *schema language* for RDF. Allows specifying constraints on individuals and relationships used in RDF.

RDFS

The *schema language* for RDF. Allows specifying constraints on individuals and relationships used in RDF.

Such constraints are:

RDFS

The *schema language* for RDF. Allows specifying constraints on individuals and relationships used in RDF.

Such constraints are:

- ▶ `rdf:type` (used to specify class membership of an individual),
- ▶ `rdfs:subClassOf` (subclass relationship between classes),
- ▶ `rdfs:subPropertyOf` (subproperty relationship between properties),
- ▶ `rdfs:domain` (domain of a property) and
- ▶ `rdfs:range` (range of a property).

RDFS

`rdf:type`

Used to specify class membership.

RDFS

`rdf:type`

Used to specify class membership.

Syntax: `<i rdf:type C>`.

RDFS

`rdf:type`

Used to specify class membership.

Syntax: $\langle i \text{ rdf:type } C \rangle$.

First-order logic translation: $C(i)$.

RDFS

`rdf:type`

Used to specify class membership.

Syntax: $\langle i \text{ rdf:type } C \rangle$.

First-order logic translation: $C(i)$.

Examples:

- ▶ $\langle \text{:Dupond rdf:type :AcademicStaff} \rangle$
- ▶ $\langle \text{:Pierre rdf:type :MasterStudent} \rangle$

RDFS

`rdfs:subClassOf`

Used to specify subclass relationships between classes.

RDFS

`rdfs:subClassOf`

Used to specify subclass relationships between classes.

Syntax: $\langle C \text{ rdfs:subClassOf } D \rangle$.

RDFS

`rdfs:subClassOf`

Used to specify subclass relationships between classes.

Syntax: $\langle C \text{ rdfs:subClassOf } D \rangle$.

First-order logic translation: $\forall X (C(X) \implies D(X))$.

RDFS

`rdfs:subClassOf`

Used to specify subclass relationships between classes.

Syntax: $\langle C \text{ rdfs:subClassOf } D \rangle$.

First-order logic translation: $\forall X (C(X) \implies D(X))$.

Example:

► $\langle \text{:MasterStudent rdfs:subClassOf :Student} \rangle$

RDFS

`rdfs:subClassOf`

Used to specify subclass relationships between classes.

Syntax: $\langle C \text{ rdfs:subClassOf } D \rangle$.

First-order logic translation: $\forall X (C(X) \implies D(X))$.

Example:

- ▶ $\langle \text{:MasterStudent rdfs:subClassOf :Student} \rangle$

Usage example:

- ▶ $\langle \text{:Pierre rdf:type :MasterStudent} \rangle$

RDFS

`rdfs:subClassOf`

Used to specify subclass relationships between classes.

Syntax: $\langle C \text{ rdfs:subClassOf } D \rangle$.

First-order logic translation: $\forall X (C(X) \implies D(X))$.

Example:

- ▶ $\langle \text{:MasterStudent rdfs:subClassOf :Student} \rangle$

Usage example:

- ▶ $\langle \text{:Pierre rdf:type :MasterStudent} \rangle$

Which implies:

- ▶ $\langle \text{:Pierre rdf:type :Student} \rangle$

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

Syntax: $\langle P \text{ rdfs:subPropertyOf } R \rangle$.

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

Syntax: $\langle P \text{ rdfs:subPropertyOf } R \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies R(X, Y))$.

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

Syntax: $\langle P \text{ rdfs:subPropertyOf } R \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies R(X, Y))$.

Example:

- ▶ $\langle \text{:LateRegisteredTo} \text{ rdfs:subPropertyOf} \text{:RegisteredTo} \rangle$

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

Syntax: $\langle P \text{ rdfs:subPropertyOf } R \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies R(X, Y))$.

Example:

- ▶ $\langle \text{:LateRegisteredTo} \text{ rdfs:subPropertyOf } \text{:RegisteredTo} \rangle$

Usage example:

- ▶ $\langle \text{:Alice} \text{ :LateRegisteredTo } \text{:UE111} \rangle$

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

Syntax: $\langle P \text{ rdfs:subPropertyOf } R \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies R(X, Y))$.

Example:

- ▶ $\langle \text{:LateRegisteredTo} \text{ rdfs:subPropertyOf } \text{:RegisteredTo} \rangle$

Usage example:

- ▶ $\langle \text{:Alice} \text{ :LateRegisteredTo } \text{:UE111} \rangle$

Which implies:

- ▶ $\langle \text{:Alice} \text{ :RegisteredTo } \text{:UE111} \rangle$

RDFS

`rdfs:domain`

Used to specify the domain of a property.

RDFS

`rdfs:domain`

Used to specify the domain of a property.

Syntax: $\langle P \text{ rdfs:domain } C \rangle$.

RDFS

`rdfs:domain`

Used to specify the domain of a property.

Syntax: $\langle P \text{ rdfs:domain } C \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies C(X))$.

RDFS

`rdfs:domain`

Used to specify the domain of a property.

Syntax: $\langle P \text{ rdfs:domain } C \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies C(X))$.

Example:

► $\langle \text{:TeachesTo rdfs:domain :AcademicStaff} \rangle$

RDFS

`rdfs:domain`

Used to specify the domain of a property.

Syntax: $\langle P \text{ rdfs:domain } C \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies C(X))$.

Example:

- ▶ $\langle \text{:TeachesTo rdfs:domain :AcademicStaff} \rangle$

Usage example:

- ▶ $\langle \text{:Dupond :TeachesTo :Pierre} \rangle$

RDFS

`rdfs:domain`

Used to specify the domain of a property.

Syntax: $\langle P \text{ rdfs:domain } C \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies C(X))$.

Example:

- ▶ $\langle \text{:TeachesTo rdfs:domain :AcademicStaff} \rangle$

Usage example:

- ▶ $\langle \text{:Dupond :TeachesTo :Pierre} \rangle$

Which implies:

- ▶ $\langle \text{:Dupond rdf:type :AcademicStaff} \rangle$

RDFS

`rdfs:range`

Used to specify the range of a property.

RDFS

`rdfs:range`

Used to specify the range of a property.

Syntax: $\langle P \text{ rdfs:range } D \rangle$.

RDFS

`rdfs:range`

Used to specify the range of a property.

Syntax: $\langle P \text{ rdfs:range } D \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies D(Y))$.

RDFS

`rdfs:range`

Used to specify the range of a property.

Syntax: $\langle P \text{ rdfs:range } D \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies D(Y))$.

Example:

► $\langle \text{:TeachesTo rdfs:range :Student} \rangle$

RDFS

`rdfs:range`

Used to specify the range of a property.

Syntax: $\langle P \text{ rdfs:range } D \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies D(Y))$.

Example:

- ▶ $\langle \text{:TeachesTo rdfs:range :Student} \rangle$

Usage example:

- ▶ $\langle \text{:Dupond :TeachesTo :Pierre} \rangle$

RDFS

`rdfs:range`

Used to specify the range of a property.

Syntax: $\langle P \text{ rdfs:range } D \rangle$.

First-order logic translation: $\forall X \forall Y (P(X, Y) \implies D(Y))$.

Example:

- ▶ $\langle \text{:TeachesTo rdfs:range :Student} \rangle$

Usage example:

- ▶ $\langle \text{:Dupond :TeachesTo :Pierre} \rangle$

Which implies:

- ▶ $\langle \text{:Pierre rdf:type :Student} \rangle$

Thanks!