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:

- $\langle$`:Dupond :Leads :InfoDept`$\rangle$

RDF
Triplets

Facts are stated as *RDF triplets*.

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

Some examples:

- $\langle$`:Dupond :Leads :InfoDept`$\rangle$
- $\langle$`:Dupond :TeachesIn :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:

- $\langle$`:Dupond :Leads :InfoDept`$\rangle$
- $\langle$`:Dupond :TeachesIn :UE111`$\rangle$
- $\langle$`:Dupond :TeachesTo :Pierre`$\rangle$
- $\langle$`:Pierre :EnrolledIn :InfoDept`$\rangle$
- $\langle$`:Pierre :RegisteredTo :UE111`$\rangle$

RDF
Triplets

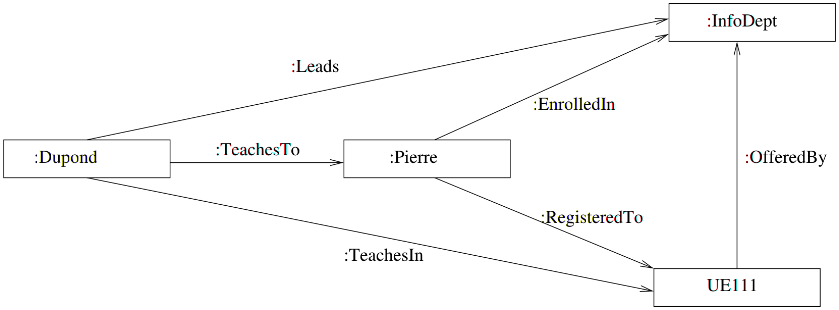
Facts are stated as *RDF triplets*.

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

Some examples:

- $\langle$`:Dupond :Leads :InfoDept`$\rangle$
- $\langle$`:Dupond :TeachesIn :UE111`$\rangle$
- $\langle$`:Dupond :TeachesTo :Pierre`$\rangle$
- $\langle$`:Pierre :EnrolledIn :InfoDept`$\rangle$
- $\langle$`:Pierre :RegisteredTo :UE111`$\rangle$
- $\langle$`:UE111 :OfferedBy :InfoDept`$\rangle$

RDF
Graph representation



$\langle$`:Dupond :Leads :InfoDept`$\rangle$
$\langle$`:Dupond :TeachesIn :UE111`$\rangle$
$\langle$`:Dupond :TeachesTo :Pierre`$\rangle$
$\langle$`:Pierre :EnrolledIn :InfoDept`$\rangle$
$\langle$`:Pierre :RegisteredTo :UE111`$\rangle$
$\langle$`:UE110 :OfferedBy :InfoDept`$\rangle$

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.

Some examples of these constraints are:

RDFS

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

Some examples of these 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),
- `rdfs:range` (range of a property),
- etc.

RDFS

`rdf:type`

Used to specify class membership.

RDFS

`rdf:type`

Used to specify class membership.

Syntax: $\langle$`i rdf:type C`$\rangle$.

RDFS

rdf:type

Used to specify class membership.

Syntax: $\langle$`i rdf:type C`$\rangle$.

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

RDFS
`rdf:type`

Used to specify class membership.

Syntax: $\langle$`i rdf:type C`$\rangle$.

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

Examples:

- $\langle$`:Dupond rdf:type :AcademicStaff`$\rangle$
- $\langle$`: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 rdfs:subClassOf D`$\rangle$.

RDFS

`rdfs:subClassOf`

Used to specify subclass relationships between classes.

Syntax: $\langle$`C 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 rdfs:subClassOf D`$\rangle$.

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

Example:

- $\langle$`:MasterStudent rdfs:subClassOf :Student`$\rangle$

RDFS

`rdfs:subClassOf`

Used to specify subclass relationships between classes.

Syntax: ⟨`C rdfs:subClassOf D`⟩.

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

Example:

- ⟨`:MasterStudent rdfs:subClassOf :Student`⟩

Usage example:

- ⟨`:Pierre rdf:type :MasterStudent`⟩

RDFS

`rdfs:subClassOf`

Used to specify subclass relationships between classes.

Syntax: ⟨`C rdfs:subClassOf D`⟩.

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

Example:

- ⟨`:MasterStudent rdfs:subClassOf :Student`⟩

Usage example:

- ⟨`:Pierre rdf:type :MasterStudent`⟩

Which implies:

- ⟨`:Pierre rdf:type :Student`⟩

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

Syntax: $\langle$`P rdfs:subPropertyOf R`$\rangle$.

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

Syntax: ⟨`P rdfs:subPropertyOf R`⟩.

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: ⟨`P rdfs:subPropertyOf R`⟩.

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

Example:

- ⟨`:LateRegisteredTo rdfs:subPropertyOf :RegisteredTo`⟩

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

Syntax: ⟨`P rdfs:subPropertyOf R`⟩.

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

Example:

- ⟨`:LateRegisteredTo rdfs:subPropertyOf :RegisteredTo`⟩

Usage example:

- ⟨`:Alice :LateRegisteredTo :UE111`⟩

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

Syntax: ⟨`P rdfs:subPropertyOf R`⟩.

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

Example:

- ⟨`:LateRegisteredTo rdfs:subPropertyOf :RegisteredTo`⟩

Usage example:

- ⟨`:Alice :LateRegisteredTo :UE111`⟩

Which implies:

- ⟨`:Alice :RegisteredTo :UE111`⟩

RDFS

`rdfs:domain`

Used to specify the domain of a property.

RDFS

`rdfs:domain`

Used to specify the domain of a property.

Syntax: ⟨`P rdfs:domain C`⟩.

RDFS

`rdfs:domain`

Used to specify the domain of a property.

Syntax: ⟨`P rdfs:domain C`⟩.

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 rdfs:domain C`$\rangle$.

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

Example:

- $\langle$`:TeachesTo rdfs:domain :AcademicStaff`$\rangle$

RDFS

`rdfs:domain`

Used to specify the domain of a property.

Syntax: ⟨`P rdfs:domain C`⟩.

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

Example:

- ⟨`:TeachesTo rdfs:domain :AcademicStaff`⟩

Usage example:

- ⟨`:Dupond :TeachesTo :Pierre`⟩

RDFS

`rdfs:domain`

Used to specify the domain of a property.

Syntax: ⟨`P rdfs:domain C`⟩.

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

Example:

- ⟨`:TeachesTo rdfs:domain :AcademicStaff`⟩

Usage example:

- ⟨`:Dupond :TeachesTo :Pierre`⟩

Which implies:

- ⟨`:Dupond rdf:type :AcademicStaff`⟩

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 rdfs:range D`$\rangle$.

RDFS

`rdfs:range`

Used to specify the range of a property.

Syntax: $\langle$`P 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 rdfs:range D`$\rangle$.

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

Example:

- $\langle$`:TeachesTo rdfs:range :Student`$\rangle$

RDFS

`rdfs:range`

Used to specify the range of a property.

Syntax: $\langle$`P rdfs:range D`$\rangle$.

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

Example:

- $\langle$`:TeachesTo rdfs:range :Student`$\rangle$

Usage example:

- $\langle$`:Dupond :TeachesTo :Pierre`$\rangle$

RDFS

`rdfs:range`

Used to specify the range of a property.

Syntax: ⟨`P rdfs:range D`⟩.

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

Example:

- ⟨`:TeachesTo rdfs:range :Student`⟩

Usage example:

- ⟨`:Dupond :TeachesTo :Pierre`⟩

Which implies:

- ⟨`:Pierre rdf:type :Student`⟩

OWL

The *Web Ontology Language*. Extends RDFS with many additional constraints.

OWL

The *Web Ontology Language*. Extends RDFS with many additional constraints.

Some examples of such constraints:

OWL

The *Web Ontology Language*. Extends RDFS with many additional constraints.

Some examples of such constraints:

- `owl:disjointWith` (specifies class disjointness),
- `owl:unionOf` (defines a class as a union of other classes),
- `owl:intersectionOf` (defines a class as an intersection of other classes),
- `owl:minCardinality` (minimum cardinality of a relationship),
- `owl:maxCardinality` (maximum cardinality of a relationship),
- `owl:functionalProperty` (a property describes a mathematical function),
- `owl:symmetricProperty` ($R(X, Y)$ implies $R(Y, X)$),
- etc.

Thanks!