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
- The n-ary relationship pattern used in **@Web**
- A sample n-ary relationship
- Example of an annotated scientific document

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

Stands for *resource description framework*.

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

RDF
Triples

Facts are stated as *RDF triples*.

RDF

Triples

Facts are stated as *RDF triples*.

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

RDF
Triples

Facts are stated as *RDF triples*.

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

Some examples:

RDF
Triples

Facts are stated as *RDF triples*.

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

Some examples:

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

RDF
Triples

Facts are stated as *RDF triples*.

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

Some examples:

- ⟨`:Dupond :Leads :InfoDept`⟩
- ⟨`:Dupond :TeachesIn :UE111`⟩

Facts are stated as *RDF triples*.

A triple 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`$\rangle$

RDF
Triples

Facts are stated as *RDF triples*.

A triple 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
Triples

Facts are stated as *RDF triples*.

A triple 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
Triples

Facts are stated as *RDF triples*.

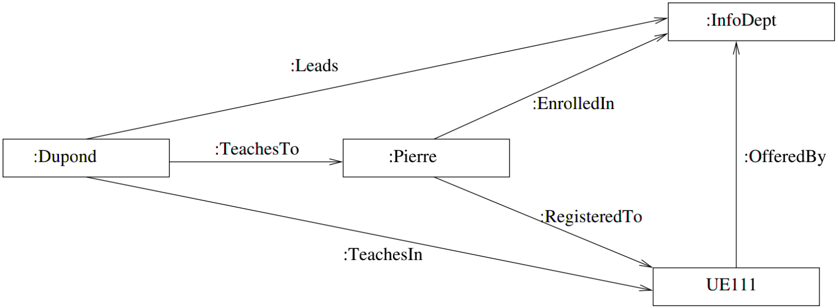
A triple is made of a *subject*, a *predicate* and an *object*.

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 triples, including:

RDF
Syntax

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

- XML (as used in **@Web**),

RDF
Syntax

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

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

RDF
Syntax

There are many different syntaxes for writing RDF triples, 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: ⟨`C rdfs:subClassOf D`⟩.

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

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

Example:

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

Usage example:

- $\langle$`:Pierre rdf:type :MasterStudent`$\rangle$

Which implies:

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

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

Example:

- $\langle$`:LateRegisteredTo rdfs:subPropertyOf :RegisteredTo`$\rangle$

Usage example:

- $\langle$`:Alice :LateRegisteredTo :UE111`$\rangle$

RDFS

`rdfs:subPropertyOf`

Used to specify subproperty relationships between properties.

Syntax: ⟨`P rdfs:subPropertyOf R`⟩.

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

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: $\langle$`P rdfs:domain C`$\rangle$.

RDFS

`rdfs:domain`

Used to specify the domain of a property.

Syntax: $\langle$`P rdfs:domain C`$\rangle$.

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

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: $\langle$`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`$\rangle$

Usage example:

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

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`$\rangle$

Usage example:

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

Which implies:

- $\langle$`: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 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: ⟨`P rdfs:range D`⟩.

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

Example:

- ⟨`:TeachesTo rdfs:range :Student`⟩

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.

SKOS

Stands for *Simple Knowledge Organization System*. It allows structuring a vocabulary in a machine readable way.

SKOS

Stands for *Simple Knowledge Organization System*. It allows structuring a vocabulary in a machine readable way.

It's a common data model for knowledge organization systems such as thesauri, classification schemes, subject heading systems and taxonomies. Typical uses include organizing large collections of objects such as books or museum artifacts.

SKOS

Stands for *Simple Knowledge Organization System*. It allows structuring a vocabulary in a machine readable way.

It's a common data model for knowledge organization systems such as thesauri, classification schemes, subject heading systems and taxonomies. Typical uses include organizing large collections of objects such as books or museum artifacts.

SKOS is not for formal ontologies. It's not meant to express formal axioms nor allowing automatic reasoning. Instead, it's meant to be a *simple* model with softer semantics that focuses on terminological information.

SKOS

Stands for *Simple Knowledge Organization System*. It allows structuring a vocabulary in a machine readable way.

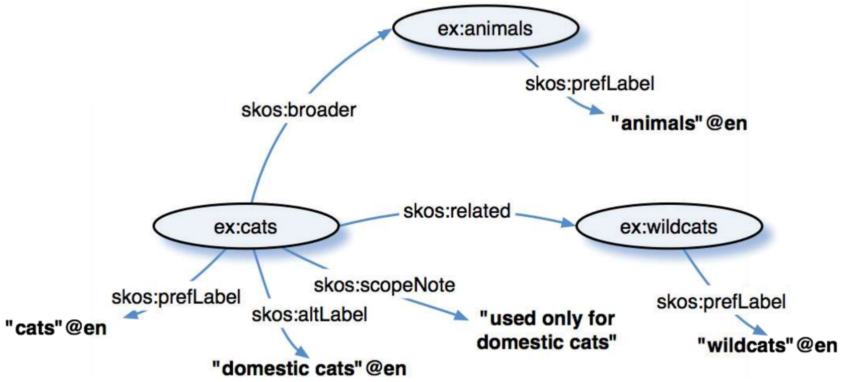
It's a common data model for knowledge organization systems such as thesauri, classification schemes, subject heading systems and taxonomies. Typical uses include organizing large collections of objects such as books or museum artifacts.

SKOS is not for formal ontologies. It's not meant to express formal axioms nor allowing automatic reasoning. Instead, it's meant to be a *simple* model with softer semantics that focuses on terminological information.

Used in **@Web** to bridge the gap between data in scientific documents and the associated domain ontology.

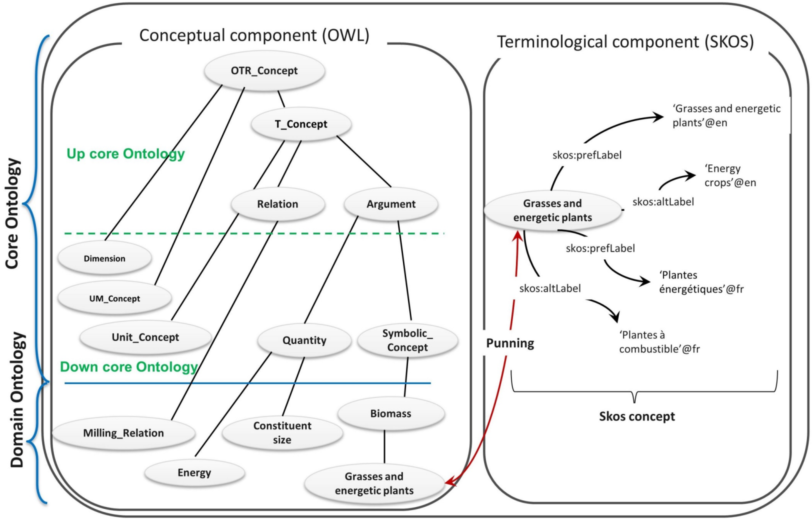
SKOS

A sample SKOS graph



SKOS

How it's used in **@Web**



SKOS

A SKOS concept taken from the biorefinery application

```
<skos:Concept rdf:ID="treated_corn_stover">
  <rdfs:subClassOf>
    <skos:Concept rdf:ID="corn_stover">
      <rdfs:subClassOf>
        <skos:Concept rdf:ID="grasses_and_energetic_plants">
          <rdfs:subClassOf rdf:resource="#biomass"/>
          <skos:prefLabel xml:lang="en">Grasses and energetic plants</skos:pref
          <skos:prefLabel xml:lang="fr">Herbes et plantes énergétiques</skos:pr
          <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
        </skos:Concept>
      </rdfs:subClassOf>
      <skos:altLabel xml:lang="en">Maize stover</skos:altLabel>
      <skos:prefLabel xml:lang="en">Corn stover</skos:prefLabel>
      <skos:prefLabel xml:lang="fr">Fourrage de maïs</skos:prefLabel>
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
    </skos:Concept>
  </rdfs:subClassOf>
  <skos:prefLabel xml:lang="en">Treated Corn stover</skos:prefLabel>
  <skos:prefLabel xml:lang="fr">Fourrage de maïs traité</skos:prefLabel>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</skos:Concept>
```

The n-ary relationship pattern used in **@Web**

Motivation

- RDF relations are binary (they only involve a subject and an object.)

The n-ary relationship pattern used in **@Web**

Motivation

- RDF relations are binary (they only involve a subject and an object.)
- Experimental data often involve more than two individuals:

The n-ary relationship pattern used in **@Web**

Motivation

- RDF relations are binary (they only involve a subject and an object.)
- Experimental data often involve more than two individuals:
  - input flow,
  - control parameters,
  - output flow

The n-ary relationship pattern used in **@Web**

Motivation

- RDF relations are binary (they only involve a subject and an object.)
- Experimental data often involve more than two individuals:
  - input flow,
  - control parameters,
  - output flow
- There are many ways to represent n-ary relations using RDF triples.

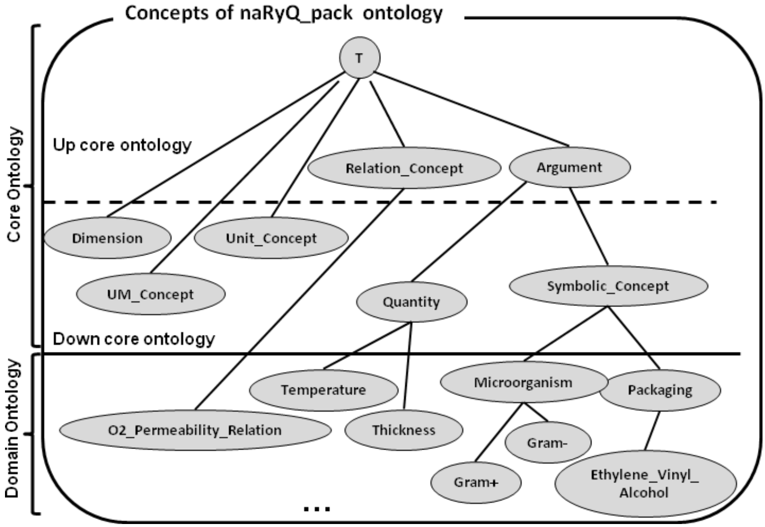
The n-ary relationship pattern used in **@Web**

Motivation

- RDF relations are binary (they only involve a subject and an object.)
- Experimental data often involve more than two individuals:
  - input flow,
  - control parameters,
  - output flow
- There are many ways to represent n-ary relations using RDF triples.
- **Proposed solution:** create an n-ary relationship design pattern specifically taylored to model experimental data.

The n-ary relationship pattern used in **@Web**

An ontology for n-ary relationships



The n-ary relationship pattern used in **@Web**

An ontology for n-ary relationships: OWL definition

```
<!-- Core ontology -->
<owl:Class rdf:ID="Relation_Concept">
<owl:Class rdf:ID="Argument"/>
<owl:Class rdf:ID="Quantity">
  <rdfs:subClassOf rdf:resource="#Argument"/>
</owl:Class>
<owl:Class rdf:ID="Symbolic_Concept">
  <rdfs:subClassOf rdf:resource="#Argument"/>
</owl:Class>
...

<!-- Domain ontology -->
<owl:Class rdf:ID="O2Permeability">
  <rdfs:subClassOf rdf:resource="#Quantity"/>
</owl:Class>
<owl:Class rdf:ID="Packaging">
  <rdfs:subClassOf rdf:resource="#Symbolic_Concept"/>
</owl:Class>
<rdf:ObjectProperty rdf:ID="hasO2Permeability">
    <rdfs:domain rdf:resource="#Relation_Concept"/>
    <rdfs:range rdf:resource="#O2Permeability"/>
</rdf:ObjectProperty>
<rdf:ObjectProperty rdf:ID="hasPackaging">
    <rdfs:domain rdf:resource="#Relation_Concept"/>
    <rdfs:range rdf:resource="#Packaging"/>
</rdf:ObjectProperty>
...
```

The n-ary relationship pattern used in **@Web**

Some observations

- This design requires experiments to have *at least* two input/control parameters.

The n-ary relationship pattern used in **@Web**

Some observations

- This design requires experiments to have *at least* two input/control parameters.
- It allows optional and mandatory parameters.

The n-ary relationship pattern used in **@Web**

Some observations

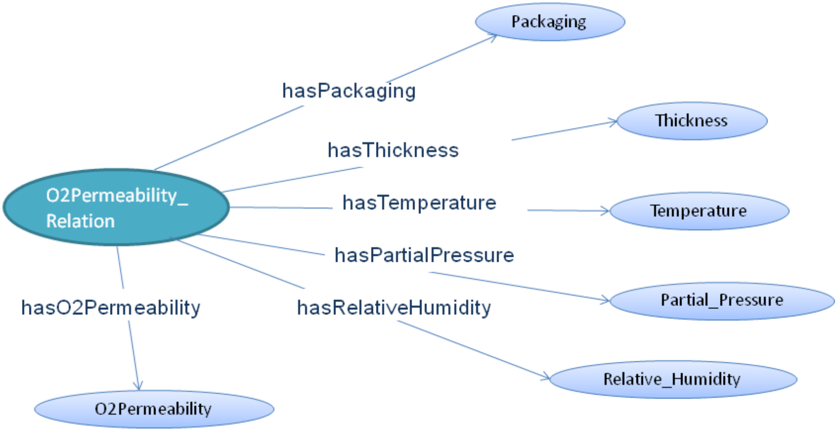
- This design requires experiments to have *at least* two input/control parameters.
- It allows optional and mandatory parameters.
- The order of the input parameters doesn't matter.

The n-ary relationship pattern used in **@Web**

Some observations

- This design requires experiments to have *at least* two input/control parameters.
- It allows optional and mandatory parameters.
- The order of the input parameters doesn't matter.
- Each instance of an n-ary relation has *exactly* one output.

A sample n-ary relationship



A sample n-ary relationship

OWL definition (I)

```
<owl:Class rdf:ID="O2Permeability_Relation">
  <rdfs:subClassOf rdf:resource="#Relation"/>

  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasO2Permeability"/>
      <owl:allValuesFrom rdf:resource="#O2Permeability"/>
    </owl:Restriction>
  </rdfs:subClassOf>

  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasO2Permeability"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
        1
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
```

A sample n-ary relationship

OWL definition (II)

```
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasPackaging"/>
    <owl:allValuesFrom rdf:resource="#Packaging"/>
  </owl:Restriction>
</rdfs:subClassOf>

<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasPackaging"/>
    <owl:mincardinality rdf:datatype="&xsd;nonNegativeInteger">
      1
    </owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>

<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#hasThickness"/>
    <owl:allValuesFrom rdf:resource="#Thickness"/>
  </owl:Restriction>
</rdfs:subClassOf>
```

A sample n-ary relationship
OWL definition (III)

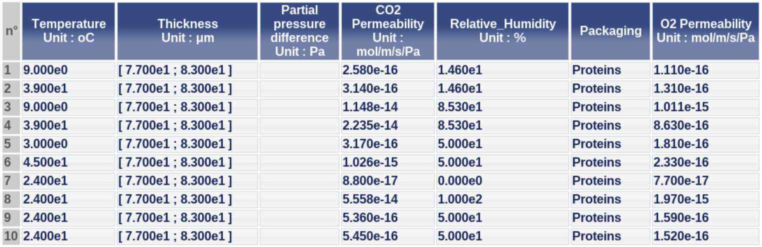
```
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasTemperature"/>
      <owl:allValuesFrom rdf:resource="#Temperature"/>
    </owl:Restriction>
  </rdfs:subClassOf>

  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasPartialPressure"/>
      <owl:allValuesFrom rdf:resource="#Partial_Pressure"/>
    </owl:Restriction>
  </rdfs:subClassOf>

  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasRelativeHumidity"/>
      <owl:allValuesFrom rdf:resource="#Relative_Humidity"/>
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Example of an annotated scientific document

Table extracted using the **@Web** platform



n°	Temperature Unit : oC	Thickness Unit : µm	Partial pressure difference Unit : Pa	CO2 Permeability Unit : mol/m/s/Pa	Relative_Humidity Unit : %	Packaging	O2 Permeability Unit : mol/m/s/Pa
1	9.000e0	[7.700e1 ; 8.300e1]		2.580e-16	1.460e1	Proteins	1.110e-16
2	3.900e1	[7.700e1 ; 8.300e1]		3.140e-16	1.460e1	Proteins	1.310e-16
3	9.000e0	[7.700e1 ; 8.300e1]		1.148e-14	8.530e1	Proteins	1.011e-15
4	3.900e1	[7.700e1 ; 8.300e1]		2.235e-14	8.530e1	Proteins	8.630e-16
5	3.000e0	[7.700e1 ; 8.300e1]		3.170e-16	5.000e1	Proteins	1.810e-16
6	4.500e1	[7.700e1 ; 8.300e1]		1.026e-15	5.000e1	Proteins	2.330e-16
7	2.400e1	[7.700e1 ; 8.300e1]		8.800e-17	0.000e0	Proteins	7.700e-17
8	2.400e1	[7.700e1 ; 8.300e1]		5.558e-14	1.000e2	Proteins	1.970e-15
9	2.400e1	[7.700e1 ; 8.300e1]		5.360e-16	5.000e1	Proteins	1.590e-16
10	2.400e1	[7.700e1 ; 8.300e1]		5.450e-16	5.000e1	Proteins	1.520e-16

Example of an annotated scientific document

RDF annotations (I)

```
<onto:hasTable>
  <onto:Table rdf:about="Table_160">
    <onto:hasForRow>
      <onto:Row rdf:about="Row-5_160">
        <onto:hasForRelation>

          <!-- O2 permeability relation instance -->
          <domain:o2_permeability_relation rdf:about="o2_permeability_relation_Row-5_160">
            <onto:hasForDegree rdf:datatype="http://www.w3.org/2001/XMLSchema#double"
            >1.0</onto:hasForDegree>

            ...
```

Example of an annotated scientific document

RDF annotations (II)

```
<!-- Experiment output -->
<core:hasResultConcept>
  <onto:Cell rdf:about="Cell-6_Row-5_160">
    <rdf:type rdf:resource="/resources/hSC9z#o2_permeability"/>
    <onto:hasForOriginalValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >233</onto:hasForOriginalValue>
    <onto:hasForColumnNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
    >6</onto:hasForColumnNumber>
    <onto:hasForFS>
      <onto:CFS rdf:about="CFS_Cell-6_Row-5_160">
        <rdf:type rdf:resource="/resources/atWeb/annotation/Scalar"/>
        <onto:hasForUnit rdf:resource="/resources/hSC9z#Mole_Per_Meter_Per_Second_Per_Pascal"
        <onto:hasForFuzzyElement>
          <onto:FuzzySet rdf:about="FS_Cell-6_Row-5_160">
            <onto:hasForMaxKernel rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#string"
            >2.330e-16</onto:hasForMaxKernel>
            <onto:hasForMinKernel rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#string"
                >2.330e-16</onto:hasForMinKernel>
                <onto:hasForMinSupport rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#string"
                >2.330e-16</onto:hasForMinSupport>
                <onto:hasForMaxSupport rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#string"
                >2.330e-16</onto:hasForMaxSupport>
          </onto:FuzzySet>
        </onto:hasForFuzzyElement>
      </onto:CFS>
    </onto:hasForFS>
  </onto:Cell>
</core:hasResultConcept>
```

Example of an annotated scientific document

RDF annotations (III)

```
<!-- Experiment input parameter: temperature -->
<core:hasAccessConcept>
  <onto:Cell rdf:about="Cell-0_Row-5_160">
    <rdf:type rdf:resource="/resources/hSC9z#temperature"/>
    <onto:hasForOriginalValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >45</onto:hasForOriginalValue>
    <onto:hasForColumnNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
    >0</onto:hasForColumnNumber>
    <onto:hasForFS>
      <onto:CFS rdf:about="CFS_Cell-0_Row-5_160">
        <rdf:type rdf:resource="/resources/atWeb/annotation/Scalar"/>
        <onto:hasForUnit rdf:resource="/resources/hSC9z#Degree_Celsius"/>
        <onto:hasForFuzzyElement>
          <onto:FuzzySet rdf:about="FS_Cell-0_Row-5_160">
            <onto:hasForMaxKernel rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#string"
            >4.500e1</onto:hasForMaxKernel>
            <onto:hasForMinKernel rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#string"
            >4.500e1</onto:hasForMinKernel>
            <onto:hasForMinSupport rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#string"
            >4.500e1</onto:hasForMinSupport>
            <onto:hasForMaxSupport rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#string"
            >4.500e1</onto:hasForMaxSupport>
          </onto:FuzzySet>
        </onto:hasForFuzzyElement>
      </onto:CFS>
    </onto:hasForFS>
  </onto:Cell>
</core:hasAccessConcept>
```

Example of an annotated scientific document

RDF annotations (IV)

```
<!-- Experiment input parameter: thickness -->
<core:hasAccessConcept>
  <onto:Cell rdf:about="Cell-1_Row-5_160">
    <rdf:type rdf:resource="/resources/hSC9z#thickness"/>
    <onto:hasForOriginalValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></onto:hasForOriginalValue>
    <onto:hasForColumnNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
    >1</onto:hasForColumnNumber>
    <onto:hasForFS>
      <onto:CFS rdf:about="CFS_Cell-1_Row-5_160">
        <rdf:type rdf:resource="/resources/atWeb/annotation/Interval"/>
        <onto:hasForUnit rdf:resource="/resources/hSC9z#Micrometer"/>
        <onto:hasForFuzzyElement>
          <onto:FuzzySet rdf:about="FS_Cell-1_Row-5_160">
            <onto:hasForMaxKernel rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#string"
            >8.300e1</onto:hasForMaxKernel>
            <onto:hasForMinKernel rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#string"
            >7.700e1</onto:hasForMinKernel>
            <onto:hasForMinSupport rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#string"
            >7.700e1</onto:hasForMinSupport>
            <onto:hasForMaxSupport rdf:datatype=
            "http://www.w3.org/2001/XMLSchema#string"
            >8.300e1</onto:hasForMaxSupport>
          </onto:FuzzySet>
        </onto:hasForFuzzyElement>
      </onto:CFS>
    </onto:hasForFS>
  </onto:Cell>
</core:hasAccessConcept>
```

Example of an annotated scientific document

RDF annotations (V)

```
    ...
    <core:hasAccessConcept rdf:resource="Cell-2_Row-5_160"/>
    <core:hasAccessConcept rdf:resource="Cell-4_Row-5_160"/>
    <core:hasAccessConcept rdf:resource="Cell-5_Row-5_160"/>
  </domain:o2_permeability_relation>
</onto:hasForRelation>
...
```

Example of an annotated scientific document

RDF annotations (VI)

```
<!-- Cell "Relative Humidity (%)" -->
<onto:hasForCell>
  <onto:Cell rdf:about="Cell-4_Row-5_160">
    <rdf:type rdf:resource="/resources/hSC9z#relative_humidity"/>
    <onto:hasForOriginalValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >50</onto:hasForOriginalValue>
    <onto:hasForColumnNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
    >4</onto:hasForColumnNumber>
    <onto:hasForFS>
      <onto:CFS rdf:about="CFS_Cell-4_Row-5_160">
        <rdf:type rdf:resource="/resources/atWeb/annotation/Scalar"/>
        <onto:hasForUnit rdf:resource="/resources/hSC9z#Percent"/>
        <onto:hasForFuzzyElement>
          <onto:FuzzySet rdf:about="FS_Cell-4_Row-5_160">
            <onto:hasForMaxKernel rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >5.000e1</onto:hasForMaxKernel>
            <onto:hasForMinKernel rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >5.000e1</onto:hasForMinKernel>
            <onto:hasForMinSupport rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >5.000e1</onto:hasForMinSupport>
            <onto:hasForMaxSupport rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >5.000e1</onto:hasForMaxSupport>
          </onto:FuzzySet>
        </onto:hasForFuzzyElement>
      </onto:CFS>
    </onto:hasForFS>
  </onto:Cell>
</onto:hasForCell>
```

Example of an annotated scientific document

RDF annotations (VII)

```
        <!-- Cell "Packaging" -->
        <onto:hasForCell>
          <onto:Cell rdf:about="Cell-5_Row-5_160">
            <rdf:type rdf:resource="/resources/hSC9z#packaging"/>
            <onto:hasForOriginalValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Wheat gluten</onto:hasForOriginalValue>
            <onto:hasForColumnNumber rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"
            >5</onto:hasForColumnNumber>
            <onto:hasForFS>
              <onto:DFS rdf:about="DFS_Cell-5_Row-5_160">
                <onto:hasForElement>
                  <domain:proteins rdf:about="proteins_Cell-5_Row-5_160">
                    <onto:hasForDegree rdf:datatype="http://www.w3.org/2001/XMLSchema#double"
                    >1.0</onto:hasForDegree>
                  </domain:proteins>
                </onto:hasForElement>
              </onto:DFS>
            </onto:hasForFS>
          </onto:Cell>
        </onto:hasForCell>

        <!-- More cells -->
        ...
      </onto:Row>
    </onto:hasForRow>

    <!-- More rows -->
    ...
  </onto:Table>
</onto:hasTable>
```

Thanks!