Introduction to Ontology Concepts and Terminology

DC-2013 Tutorial
September 2, 2013

Steven J. Miller
University of Wisconsin-Milwaukee

Content may be shared and remixed if attributed to Steven J. Miller and used for noncommercial purposes, subject to Creative Commons BY-NC License: http://creativecommons.org/licenses/by-nc/3.0/
Welcome and Introductions

- Introduce ourselves to the group

- How many participants have a general idea of what the semantic web and linked data are about?

- How many have some familiarity with the Resource Description Framework (RDF) data model?
Tutorial topics and outline

1. Tutorial Background Overview
   - The Semantic Web, Linked Data, and the Resource Description Framework

2. Ontology Basics and RDFS Tutorial
   - Semantic modeling, domain ontologies, and RDF Vocabulary Description Language (RDFS) concepts and terminology
   - Examples: domain ontologies, models, and schemas
   - Exercises

3. OWL Overview Tutorial
   - Web Ontology Language (OWL): selected concepts and terminology
   - Exercises
Tutorial audience

Information professionals

- who have *little or no prior familiarity* with ontologies, RDFS, or OWL
- who want to gain an *introductory level* understanding of basic ontology concepts and terminology
Tutorial outcomes

At the conclusion of the tutorial, participants will:

1. Understand basic RDFS ontology concepts such as classes, properties, instances, domain and range.
2. Understand how ontologies provide structure to RDF triples.
3. Be able to create a basic, beginning-level RDFS-compatible ontology.
4. Determine logical inferencing capabilities based on specific class, property, domain and range specifications.
5. Gain initial exposure to more complex OWL property and class specifications and their greater potential inferencing power.
6. Better understand: existing RDF-based ontologies such as BIBO, BIBFRAME, the BBC ontologies, and the Europeana Data Model; DCMI Metadata Terms specifications; and conceptual models such as the Dublin Core Abstract Model.
7. Be better able to understand and contribute to professional discussions about ontologies, ontology concepts, and ontology terminology on discussion lists, at conferences, and the like.
Background Sources: Books

- *Semantic Web for the Working Ontologist*
  Dean Allemang, James Hendler

- *A Semantic Web Primer*
  Grigoris Antoniou and Frank van Harmelen
Background Sources: Books

1. A Developer’s Guide to the Semantic Web
   - Liyang Yu
   - Springer

2. Learning SPARQL
   - Bob DuCharme
   - O’REILLY
Background Sources: Books (3)

Publishing and Using Cultural Heritage Linked Data on the Semantic Web

Eero Hyvönen
Background Sources: Practical Guides


  - Available online: http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf

- For further information on these and other sources, see “Selected Readings and Resources” at the end of the tutorial materials
Semantic Web, Linked Data, and RDF

Part 1: Tutorial background overview
Semantic Web (SW)

- **Current Web**: a web of linked *documents*
  - Unstructured data, connect by hyperlinks
  - Suitable for human consumption (but not for machines)
  - Queried by matching keywords in documents and using relevance ranking algorithms

- **Semantic Web**: a web of linked *data*
  - Structured data (metadata), carrying semantic meaning, connecting by semantically meaningful links
    - People, places, time periods, concepts, …
  - Making parts of the Web more like a database, able to be queried like a database
  - Suitable for machine consumption – to better serve humans
  - Full-fledged semantics also enable machines to make logical inferences not explicitly stated by humans
Linked data (LD)

- Newer idea than that of the semantic web (SW)
  - But sometimes easier to think of SW as building on the ideas behind LD.
- LD not a specification, but a set of best practices for providing a data infrastructure that makes it easier to share data across the Web.
- SW technologies such as RDFS, OWL, and SPARQL can then be used to build applications around that data.
- Tim Berners-Lee: four principles of Linked Data:
  - 1. Use URIs as names for things.  [URI = Uniform Resource Identifier]
  - 2. Use HTTP URIs so that people can look up those names.
  - 3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
  - 4. Include links to other URIs so that they can discover more things.

From data silos to distributed knowledge

- Current data in databases **closed** to one another and to the web
  - Unconnected information and knowledge **silos**

- Semantic Web and Linked Data: **distributed** information and knowledge environments
  - Publishing data in an **open** Web environment
  - Making the data **linkable** to other data
  - Creating a vast **web of linked data**
Semantic Web assumptions

- **Open World Assumption**
  - **Closed world**: databases with tightly controlled content; all relevant information about an entity is included; inferences can be made accordingly
  - **Open world**: uncontrolled open data; someone can always contribute something new about an entity
    - Machine inferencing must take this into account: “we may draw no conclusions that rely on assuming that the information available at any one point is all the information available”

- **Nonunique Naming Assumption**
  - **Unique names**: may hold in controlled databases or triple stores
  - **Nonunique names**: in an open world context, different Web authors will use different URIs for the same entity / resource
    - Machine inferencing cannot assume that two entities with different URIs are different individuals

RDF: Resource Description Framework (1)

- The Resource Description Framework (RDF) provides a graph-based data model or framework for structuring data as statements about resources
  - A “resource” may be any “thing” that exists in the world: a person, place, event, book, museum object, but also an abstract concept

- Each statement is composed of a subject, predicate, and object.

- The subject of a statement is called a resource, the predicate is called a property, and the object is called a value.

- Each statement is a triple, consisting of these three components
RDF: Resource Description Framework (2)

- In a graph diagram, “nodes” represent things; “arcs” (or “edges”) connect nodes and denote the relationship between them.
Example of tabular database DC metadata record for digital image

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Manchester Street Bridge, Sauk County, Wisconsin</td>
</tr>
<tr>
<td>Date</td>
<td>1896</td>
</tr>
<tr>
<td>Creator</td>
<td>Lassig Bridge and Iron Works</td>
</tr>
<tr>
<td>Subject</td>
<td>Truss bridges</td>
</tr>
<tr>
<td>Format</td>
<td>128.9 ft. long; 13.7 ft. deck width</td>
</tr>
<tr>
<td>Coverage</td>
<td>Sauk County</td>
</tr>
<tr>
<td>Type</td>
<td>Still Image</td>
</tr>
<tr>
<td>Creator</td>
<td>Szarkowski, John</td>
</tr>
<tr>
<td>Date</td>
<td>1955</td>
</tr>
<tr>
<td>Format</td>
<td>35 mm.</td>
</tr>
<tr>
<td>Format</td>
<td>Black &amp; white slide</td>
</tr>
<tr>
<td>Identifier</td>
<td>171, 33b-765</td>
</tr>
<tr>
<td>Relation</td>
<td>Paul J. Kramer Archival Photograph Collection</td>
</tr>
<tr>
<td>Relation</td>
<td>Bridges of Wisconsin</td>
</tr>
<tr>
<td>Rights</td>
<td>Copyright © 2009 Hagenville University</td>
</tr>
<tr>
<td>Format</td>
<td>image/jpeg</td>
</tr>
<tr>
<td>Identifier</td>
<td>WB0078736</td>
</tr>
</tbody>
</table>
Example of the same metadata in XML

<metadata>
  <oai_dc:dc xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/">
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
    <dc:title>Manchester Street Bridge, Sauk County, Wisconsin</dc:title>
    <dc:date>1896</dc:date>
    <dc:creator>Lassig Bridge and Iron Works</dc:creator>
    <dc:subject>Truss bridges</dc:subject>
    <dc:format>128.9 ft. long; 13.7 ft. deck width</dc:format>
    <dc:coverage>Sauk County</dc:coverage>
    <dc:type>Still Image</dc:type>
    <dc:creator>Szarkowski, John</dc:creator>
    <dc:date>1955</dc:date>
    <dc:format>35 mm.</dc:format>
    <dc:format>Black & white slide</dc:format>
    <dc:identifier>171, 33b-765</dc:identifier>
    <dc:relation>Paul J. Kramer Archival Photograph Collection</dc:relation>
    <dc:relation>Bridges of Wisconsin</dc:relation>
    <dc:rights>Copyright (c)2009 Hagenville University</dc:rights>
    <dc:format>image/jpeg</dc:format>
    <dc:identifier>WB0078736</dc:identifier>
  </oai_dc:dc>
</metadata>
### URIs: Uniform Resource Identifiers

#### Used as *resources* (subjects) in RDF triples:

<table>
<thead>
<tr>
<th>Resource (subject)</th>
<th>Property (predicate)</th>
<th>Value (object)</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.hdl.edu/WisBridges/WB0078736">http://www.hdl.edu/WisBridges/WB0078736</a></td>
<td>Subject</td>
<td>Truss bridges</td>
</tr>
<tr>
<td><a href="http://www.hdl.edu/WisBridges/WB0078736">http://www.hdl.edu/WisBridges/WB0078736</a></td>
<td>Creator</td>
<td>Szarkowski, John</td>
</tr>
</tbody>
</table>

#### Used as *properties* in RDF triples:

<table>
<thead>
<tr>
<th>Resource (subject)</th>
<th>Property (predicate)</th>
<th>Value (object)</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.hdl.edu/WisBridges/WB0078736">http://www.hdl.edu/WisBridges/WB0078736</a></td>
<td><a href="http://purl.org/dc/elements/1.1/subject">http://purl.org/dc/elements/1.1/subject</a></td>
<td>Truss bridges</td>
</tr>
<tr>
<td><a href="http://www.hdl.edu/WisBridges/WB0078736">http://www.hdl.edu/WisBridges/WB0078736</a></td>
<td><a href="http://purl.org/dc/elements/1.1/creator">http://purl.org/dc/elements/1.1/creator</a></td>
<td>Szarkowski, John</td>
</tr>
</tbody>
</table>

#### Used as *values* in RDF triples:

<table>
<thead>
<tr>
<th>Resource (subject)</th>
<th>Property (predicate)</th>
<th>Value (object)</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.hdl.edu/WisBridges/WB0078736">http://www.hdl.edu/WisBridges/WB0078736</a></td>
<td><a href="http://purl.org/dc/elements/1.1/creator">http://purl.org/dc/elements/1.1/creator</a></td>
<td><a href="http://www.hdl.edu/nameauthority/938475">http://www.hdl.edu/nameauthority/938475</a></td>
</tr>
</tbody>
</table>
### Statements about a digital image

<table>
<thead>
<tr>
<th>Resource (subject)</th>
<th>Property (predicate)</th>
<th>Value (object)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Image WB0078736</td>
<td>hasTitle</td>
<td>Manchester Street Bridge, Sauk County, Wisconsin</td>
</tr>
<tr>
<td>Digital Image WB0078736</td>
<td>hasSubject</td>
<td>Truss bridges</td>
</tr>
<tr>
<td>Digital Image WB0078736</td>
<td>hasCreator</td>
<td>Szarkowski, John</td>
</tr>
</tbody>
</table>

RDF statements are **directed graphs**: the property goes in one direction, from the subject to the object.
# Literals, Strings, and “Things”

<table>
<thead>
<tr>
<th>Resource (subject)</th>
<th>Property (predicate)</th>
<th>Value (object)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBook 23/23-h/23-h</td>
<td>DC Title</td>
<td>Narrative of the Life of Frederick Douglass</td>
</tr>
<tr>
<td>EBook 23/23-h/23-h</td>
<td>DC Creator</td>
<td>Douglass, Frederick, 1817-1895</td>
</tr>
<tr>
<td>EBook 23/23-h/23-h</td>
<td>DC Subject</td>
<td>African American abolitionists--Biography</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource (subject)</th>
<th>Property (predicate)</th>
<th>Value (object)</th>
</tr>
</thead>
</table>
The Narrative of the Life of Frederick Douglass

http://www.gutenberg.org/files/23/23-h/23-h.htm

http://purl.org/dc/elements/1.1/creator
http://www.viaf.org/viaf/10088/

A "String" or "Literal;" cannot be linked to anything beyond itself

"Things" or "Non-literals" expressed as URIs; can be linked to matching URIs

http://purl.org/dc/elements/1.1/subject
http://id.loc.gov/authorities/sh2007100462#concept
The Power of Linking and Querying in the Linked Data Cloud

RDF software detects matching URIs and links the two triples.
RDF: URIs and literals

- Subjects and predicates of RDF triples must be URIs
  - In the form of http:// URLs
  - May or may not be "dereferenceable" (that is, referencing an actual location on the Web)

- Objects of RDF triples may be either URIs or literals
  - A "literal" is raw text that can be used instead of a resource/thing in RDF triples
  - A literal may be a string (of characters), an integer, a date, etc.
Namespace prefixes

- XML namespace declarations in the opening element of an RDF file:
  - `xmlns:hdllibb='http://www.hdl.edu/WisBridges/'`
  - `xmlns:dc='http://purl.org/dc/elements/1.1/'`
  - `xmlns:viatif='http://www.viaf.org/viaf/'`
  - `xmlns:tgm='http://id.loc.gov/vocabulary/graphicMaterials/'`

- In the body of the RDF file, the prefix stands in place of the whole namespace URL in the triples:
  - `hdllibb:WB0078736 dc:creator viatif:110959125`
  - `hdllibb:WB0078736 dc:subject tgm:tgm011115`
Serialization syntaxes for RDF

- Machine-readable syntaxes that serialize the triples
  - That is, express them as a series of characters that can be processed in a specified order by a computer with RDF software

- RDF/XML
  - the normative syntax for writing RDF

- Notation 3 (N3)
  - a shorthand, non-XML serialization of RDF

- Turtle
  - “Terse RDF Triple Language,” a subset of Notation 3

- N-Triples
  - A subset of Turtle
Example 1 in RDF/XML syntax

<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/>

  <rdf:Description rdf:about="http://www.hdl.edu/WisBridges/WB0078736">
    <dc:title>Manchester Street Bridge, Sauk County, Wisconsin</dc:title>
  </rdf:Description>

</rdf:RDF>
Example 1 in N-Triples syntax

<http://www.hdl.edu/WisBridges/WB0078736>  
<http://purl.org/dc/elements/1.1/title>  
"Manchester Street Bridge, Sauk County, Wisconsin" .

<http://www.hdl.edu/WisBridges/WB0078736>  
<http://purl.org/dc/elements/1.1/subject>  

<http://www.hdl.edu/WisBridges/WB0078736>  
<http://purl.org/dc/elements/1.1/creator>  
Example 1 in Turtle syntax

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix hdlbr: <http://www.hdl.edu/WisBridges/> .
@prefix dc: http://purl.org/dc/elements/> .
@prefix tgm: http://id.loc.gov/vocabulary/graphicMaterials/ .
@prefix viaf: http://www.viaf.org/viaf/ .

hdlbr:WB0078736 dc:title "Manchester Street Bridge, Sauk County, Wisconsin" .
Distributed knowledge graph
(partial invented draft to convey basic idea)
RDF triple stores

- Databases (stores) of RDF triples
  - store and retrieve data in the form of triples
  - aka: graph databases

- Use a different data model than table-based flat or relational databases
  - Namely, the RDF triple / graph-based model

- Also have the ability to merge information from multiple data sources
Querying RDF: SPARQL

- Databases, including RDF triple stores, are useless unless they can be queried

- **SPARQL** is the query language for RDF, RDFS, and OWL
  - Acronym for: **SPARQL Protocol and RDF Query Language**
  - [http://www.w3.org/TR/rdf-sparql-query/](http://www.w3.org/TR/rdf-sparql-query/)
SPARQL: a one slide introduction 😊

- **Triple statements:**
  - gb:/23-h dc:title “Narrative of the Life of Frederick Douglass”
  - gb:/23-h dc:creator viaf:DouglassFrederick
  - gb:/23-h dc:subject lcsh:AfricanAmericanAbolitionists

- **SPARQL queries work like this** (conceptually; this is not the actual encoded syntax):  
  - gb:/23-h dc:title what?
  - gb:/23-h dc:creator who?
  - gb:/23-h dc:subject what?
  - what? dc:creator viaf:DouglassFrederick

- **In other words:**
  - Who is the creator of the Project Gutenberg ebook 23-h?
  - What are all of the works (within certain parameters) that were created by Frederick Douglass?
  - What are all of the works (within certain parameters) that have the LCSH subject heading African American abolitionists?

- **SPARQL enables many other, much more complex queries using various parameters, but this is the basic idea**
Questions?
Ontology Basics and RDFS
(RDF Vocabulary Description Language)

Tutorial Part 2
Tutorial part 3 objectives

- Understand an **ontology** as a **semantic model** of a specific **knowledge domain**, defining its concepts and relationships.
- Understand basic ontology building blocks, including **classes**, **subclasses**, **properties**, **subproperties**, **domain and range specifications**, and the **principle of inheritance**.
- Understand how an ontology proper plus **instances or individuals** comprise a **knowledge-base** that can enable semantic inferencing and **querying** by machines.
- Be able to create a beginning ontology using the components covered in this tutorial.
- Be aware that **RDFS** (RDF Vocabulary Description Language) is an **RDF-based language** for expressing ontologies at a basic level.
Part 1 concepts & terminology

- Semantic modeling
- Knowledge domain
- Vocabulary
- Ontology
- Knowledge base
- RDFS
  - RDF Vocabulary Description Language (formerly RDF Schema)
- Class
- Subclass
- Property (or Slot)
- Subproperty
- Instance (or Individual)
- Inheritance
- Domain (of a property)
- Range (of a property)
- Inference
Semantic modeling

- Compare the graph-based RDF data model with other data models (object oriented, entity-relationship, relational, hierarchical, etc.)

- Add *semantics* to RDF to model a **knowledge domain**
  - Semantics = meaning; in this context, machine-processable meaning
  - Also allow merging of information from different domains of knowledge

- Important terms:
  - **Vocabulary**: “a collection of terms given a well-defined meaning that is consistent across contexts”
  - **Ontology**: “allows you to define contextual relationships behind a defined vocabulary. It is the cornerstone of defining the knowledge domain.” –LinkedDataTools Tutorial 3.

- “**Ontologies, schemas, and vocabularies**, which all mean roughly the same thing, are RDF information about … other RDF information.” – Joshua Tauberer
Ontology

- Term from philosophy co-opted by computer science

- *Definitions*
  - “an explicit and formal specification of a conceptualization”
  - “defines the concepts and relationships used to describe and represent an area of knowledge” (W3C?)
  - a formal model of the things that exist in a specified knowledge domain and the relationships among those things
    - "things" may be concepts, works, persons, places, objects, events, etc.

- *Brodest sense:*
  - almost any kind of model, schema, or vocabulary; does not have to be encoded

- *More specific Semantic Web sense:*
  - a model encoded in an RDF-based ontology language (e.g., RDFS or OWL)
  - a computer-actionable model that enables logical inferencing: (e.g., OWL)
Ontologies are one way to bring structure or constraints to RDF triples

- In a crude sense similar to DTDs or XML Schemas for XML data; or MARC tag tables for MARC data

- **Ontologies model a knowledge domain.** Within that specified domain they establish:
  - What kinds of resources can we make RDF statements about?
  - What RDF properties will we use to relate these resources to each other?
  - What can be the subject of a given RDF property?
  - What can be the object of a given RDF property?
Core components of an ontology

1. Classes
   - May include subclasses (and superclasses)

2. Properties
   - May include subproperties (and superproperties)
   - Called “slots” in older terminology

3. Instances
   - Also called “individuals”
   - Specific members of a class
(1) Classes

- A class is a type of thing.
  - A type of “resource” in the RDF sense: a type of person, place, object, concept, event, etc.
- Classes and subclasses form a hierarchical taxonomy
- Members of a subclass inherit the characteristics of their parent class (superclass)
Class hierarchy example 1 (partial)

Types of resources specific to a cultural heritage knowledge domain

- Cultural Heritage Resource
  - Museum Object
    - Painting
    - Sculpture
  - Bibliographic Resource
    - Book
    - Journal
  - Archival Resource
    - Archival Collection
    - Personal Correspondence
Class hierarchy example 2 (partial)

Types of resources (things, instances, individuals) specific to a family relationships knowledge domain

- Person
  - Parent
    - Father
    - Mother
  - Child
    - Son
    - Daughter
  - Sibling
    - Brother
    - Sister
Inheritance

- Members of a subclass inherit the characteristics and properties of their parent class (superclass)
- Everything true of the parent class is true also of the child or subclass
- A member of a subclass “is a”, or “is a kind of” its parent class
- Class<-->Subclass relationships must be very strictly logical in RDFS and OWL in order to enable correct computer inferencing
# Traditional controlled vocabulary & thesaural semantic relationships

(Source: ANSI/NISO Z39.19)

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Type Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equivalency</strong></td>
<td></td>
</tr>
<tr>
<td>Synonymy</td>
<td>UN / United Nations</td>
</tr>
<tr>
<td>Lexical variants</td>
<td>pediatrics / paediatrics</td>
</tr>
<tr>
<td>Near synonymy</td>
<td>sea water / salt water</td>
</tr>
<tr>
<td><strong>Hierarchy</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Generic</strong> (or IsA) (is a kind of)</td>
<td>bird / parrot</td>
</tr>
<tr>
<td></td>
<td>![RDFS/OWL Class relationship!]</td>
</tr>
<tr>
<td><strong>Instance</strong> (or IsA) (is a specific instance of)</td>
<td>sea / Mediterranean Sea</td>
</tr>
<tr>
<td></td>
<td>![RDFS/OWL Instance]</td>
</tr>
<tr>
<td><strong>Whole/Part</strong> (actually meronomy, not hierarchy)</td>
<td>brain / brain stem</td>
</tr>
<tr>
<td></td>
<td>![NOT a RDFS/OWL Class relationship!!]</td>
</tr>
<tr>
<td><strong>Associative</strong></td>
<td></td>
</tr>
<tr>
<td>Cause / Effect</td>
<td>accident / injury</td>
</tr>
<tr>
<td>Process / Agent</td>
<td>velocity measurement / speedometer</td>
</tr>
<tr>
<td>Process / Counter-agent</td>
<td>fire / flame retardant</td>
</tr>
<tr>
<td>Action / Product</td>
<td>writing / publication</td>
</tr>
<tr>
<td>Action / Property</td>
<td>communication / communication skills</td>
</tr>
<tr>
<td>Action / Target</td>
<td>teaching / student</td>
</tr>
<tr>
<td>Concept or Object / Property</td>
<td>steel alloy / corrosion resistance</td>
</tr>
<tr>
<td>Concept or Object / Origins</td>
<td>water / well</td>
</tr>
<tr>
<td>Concept or Object / Measurement Unit or Mechanism</td>
<td>chronometer / minute</td>
</tr>
<tr>
<td>Raw material / Product</td>
<td>grapes / wine</td>
</tr>
<tr>
<td>Discipline or Field / Object or Practitioner</td>
<td>neonatology / infant</td>
</tr>
</tbody>
</table>
Class/subclass relationships

Each subclass is a (is a type of) its superclass. It inherits all of the properties of its parent class.
Classes as sets and subsets

An alternative way to view classes, subclasses, and inheritance: as sets, represented by Venn style diagrams
(2) Properties

- **Predicates** in RDF
- Ontologies define a set of properties to be used in a specific knowledge domain
- Properties (predicates) connect or relate resources to each other
  - (subject – predicate --> object)
- In an ontology context, properties relate members of one class to members of another class, or to a literal
Properties convey relationships between resources. In an ontology, they connect members of one class to members of another class (or to a literal).

- **Person**
  - hasChild: Parent
  - isChildOf: Child
  - hasFather: Father
  - hasMother: Mother
  - hasSibling: Sibling

- **Parent**
  - hasChild: Child

- **Child**
  - isChildOf: Parent

- **Sibling**
  - hasSibling: Sibling
Domain and Range

Restrictions on properties (predicates in RDF triples)

- **Domain**
  - restricts what kinds of resources or members of a class can be the *subject* of a given property in an RDF triple

- **Range**
  - restricts what kinds of resources / members of a class *or* data types (literals) can be the *object* of a given property in an RDF triple
Domain and Range

Restrict the possible values (instances) of subjects and object of a given property to members of a specific class or type.

**Domain:**
The subject of the property in the RDF triple must be a member of a specific class.

**Range:**
The object of the property in the RDF triple must be a member of a specific class [or a literal].
Directed graph: what relationship does this *child* property indicate?

A Parent **hasChild** a Child?

A Child **isChildOf** a Parent?
Domain and range for “child” property

- Property: child [property]
  - domain: Parent [class]
  - range: Child [class]

Therefore, only a member of the class Parent can be the RDF subject of the child property

And only a member of the class Child can be the RDF object of the child property
Inverse properties

Some ontologies establish property names with clear directionality, and some ontologies include all inverse properties, for example:

- **Property: hasChild** [property]
  - domain: Parent [class]
  - range: Child [class]

- **Property: isChildOf** [property]
  - domain: Child [class]
  - range: Parent [class]
Domain and range inheritance

- **Subproperties inherit the domain and range of their superproperties**
  - Unless more specific domain and range assertions are made for them

  **Example:**
  - Property: `isParentOf`
    - domain: `Parent`
    - range: `Child`
  - Property: `isFatherOf`
    - subPropertyOf `isParentOf`
  - **Result:** `isFatherOf` inherits domain `Parent` and range `Child`

But we can specify a narrower domain (and/or range when applicable):

- Property: `isFatherOf`
  - domain: `Father`
(3) Individuals

- Also called **Instances**
- The specific entities or concepts of interest to us
  - Concrete specific members or instances of classes
- For example:
  - David (by Michelangelo): member of the class Sculpture in a cultural heritage ontology
  - Maria I. Taylor: member of the class Mother in a family relationships ontology
- The actual data making up a graph database
  - Governed by the ontology proper
Ontology statement examples

- **Class definition statements:**
  - Parent isA Class
  - Mother isA Class
  - Mother subClassOf Parent
  - Child isA Class

- **Property definition statements:**
  - isMotherOf isA Property
    - isMotherOf domain Mother
    - isMotherOf range Child

- **Individual-instance statements:**
  - MariaITaylor isA Mother
  - AdamJTaylor isA Child
  - MariaITaylor isMotherOf AdamJTaylor
“Machine-readable knowledge bases store knowledge in a computer-readable form, usually for the purpose of having automated deductive reasoning applied to them. They contain a set of data, often in the form of rules that describe the knowledge in a logically consistent manner. An ontology can define the structure of stored data - what types of entities are recorded and what their relationships are. ... Such knowledge bases are also used by the semantic web.” --Wikipedia: [http://en.wikipedia.org/wiki/Knowledge_base](http://en.wikipedia.org/wiki/Knowledge_base)

For our purposes, a knowledge base is comprised of:

- **An ontology proper**
  - Defines the structure of the RDF data, the allowable classes, properties, and their characteristics

- **Individuals: the RDF instance data**
  - Statements about the actual things of interest in the knowledge domain (such as specific persons, places, things, events, concepts); must conform to the ontology
RDFS

- **RDF Vocabulary Description Language**
  - Originally stood for: RDF Schema Language
  - A simple, RDF-based language for encoding RDF ontologies
  - An RDF/XML-based encoding syntax
Key RDFS elements

- **rdfs:Resource**
  - the class of all resources

- **rdfs:Class**
  - the class of all classes

- **rdfs:Literal**
  - the class of all literals (strings)

- **rdf:Property**
  - the class of all properties

- **rdf:type**
  - relates a resource to its class

- **rdfs:subClassOf**
  - relates a class to one of its superclasses

- **rdfs:subPropertyOf**
  - relates a property to one of its superproperties

- **rdfs:domain**
  - Specifies the domain of a property
  - Any resource that is the subject of that property is an instance of the domain class

- **rdfs:range**
  - Specifies the range of a property
  - Any resource that is the object (value) of that property is an instance of the range class
RDFS notation options

- Examples on the next three slides
  1. RDFS in RDF/XML syntax
  2. A simplified notation for greater human readability
  3. A yet more simplified notation that will be used in the majority of slides in this workshop
- The emphasis is on understanding the concepts, the logical (RDF) statements, and the resulting logical inferencing capabilities rather than on reading RDF XML syntax code
RDFS statement examples in RDF/XML

<?xml version="1.0"?>

<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xml:base="http://abc.xyz/familyrelationshipsontology#">

  <rdfs:Class rdf:ID="Relative"/>

  <rdfs:Class rdf:ID="Parent">
    <rdfs:subClassOf rdf:resource="#Relative"/>
  </rdfs:Class>

  <rdfs:Property rdf:ID="hasChild">
    <rdfs:domain rdf:resource="#Parent"/>
    <rdfs:range rdf:resource="#Child"/>
  </rdfs:Property>

  <rdf:Description rdf:about="fro:MarialTaylor">
    <rdf:type rdf:resource="fro:Parent"/>
    <fro:hasChild rdf:resource="fro:AdamJTaylor"/>
  </rdf:Description>

</rdf:RDF>
RDFS statement examples
(simplified notation)

- **Namespaces**
  - xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  - xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  - xmlns:fro="http://abc.xyz/familyrelationshipsontology#"

- **Classes**
  - fro:Parent rdf:type rdfs:Class
  - fro:Parent rdfs:sublcassOf fro:Relative

- **Properties**
  - fro:hasChild rdf:type rdf:Property
  - fro:hasChild rdfs:domain fro:Parent
  - fro:hasChild rdfs:range fro:Child

- **Instances**
  - fro:MariaITaylor rdf:type fro:Parent
  - fro:MariaITaylor fro:hasChild fro:AdamJTaylor

- Notice a mixture of rdf: and rdfs: elements in RDFS
RDFS statement examples
(even more simplified notation)

- Parent isA Class
- Parent subClassOf Relative
- hasChild isA Property
- hasChild domain Parent
- hasChild domain Child
- MariaITaylor isA Parent
- MariaITaylor hasChild AdamJTaylor
Family relationships ontology example

- Advantages
  - Familiarity of family classes and relationships
  - Easier to understand the ontology and individuals
    - inheritance, domains and ranges, property types and restrictions, and logical inferences
  - Generalizable to other knowledge domains
Family Relationships Ontology

Classes
- Person
  - Female
  - Aunt
  - Daughter
  - Mother
  - Sister
  - Wife
- Male
  - Brother
  - Father
  - Husband
  - Son
  - Uncle
- Relative
  - Ancestor
    - Parent
      - Father
      - Mother
  - Aunt
  - Child
    - Daughter
    - Son
  - FirstCousin
  - Sibling
    - Brother
    - Sister
  - Spouse
    - Husband
    - Wife
    - Uncle

Properties
- hasAncestor
  - hasParent
    - hasFather
    - hasMother
  - hasAunt
  - hasChild
    - hasDaughter
    - hasSon
  - hasFirstCousin
  - hasSibling
    - hasBrother
    - hasSister
  - hasSpouse
  - hasUncle
  - isAncestorOf
    - isParentOf
      - isFatherOf
      - isMotherOf
  - isAuntOf
  - isChildOf
    - isDaughterOf
    - isSonOf
  - isFirstCousinOf
  - isSiblingOf
    - isBrotherOf
    - isSisterOf
  - isSpouseOf
  - isUncleOf

Properties (continued)
- hasBirthDate
- hasDeathDate
- hasGivenName
- hasMiddleNameOrInitial
- hasPreMarriageSurname
- hasSurname

Instances/Individuals
- AdamJTaylor
- AdinaRTaylor
- BrianCTaylor
- IsabellaAGarcia
- JosephBTaylor
- MarialTaylor
- MaryLTaylor
- RachelJCohen
- SofiaMTaylor
- TomasCGarcia
Family relationships instances
(traditional family tree)

Taylor-Garcia Family Tree

Brian C. Taylor
1946-

Rachel J. (Cohen)
Taylor 1946-

Tomás C. Garcia
1951-

Isabella A. (Reyes)
Garcia 1953-

Adina R. Taylor
1972-

Joseph B. Taylor
1975-

Maria I. (Garcia)
Taylor 1976-

Santiago C. Garcia
1974-

Sofia M. Taylor
1999-

Adam J. Taylor
2001-

Mary I. Taylor
2004-
Ontology

Instances / Individuals

Introduction to Ontology Concepts and Terminology / Steven J. Miller

DC-2013 Tutorial (Lisbon, Portugal)
Inferencing

- **Semantics** based **IF ... THEN** on inference rules
- Computers can make inferences not directly stated by a human being
- **RDF**
  - Allows some lightweight inferencing
  - Based especially on shared URIs for resources and properties, and resulting linked data graphs
- **RDFS**
  - Enables much greater inferencing based on class/subclass, property/subproperty and resulting inheritance relationships, and domain and range specifications
- **OWL**
  - Enables yet more powerful inferencing based on the use of specific types of properties
  - As we will see in the part 4 of this tutorial
Inferences expressed as statements

- **Statements:**
  - Relative isA Class
  - Relative subClassOf Person
  - Parent isA Class
  - Parent subClassOf Relative
  - Mother isA Class
  - Mother subClassOf Parent
  - isMotherOf isA Property
    - isMotherOf domain Mother
    - isMotherOf range Child
  - isMotherOf subPropertyOf isParentOf
  - MariaITaylor isMotherOf AdamJTaylor

- **Inferences:**
  - MariaITaylor isA Mother
  - MariaITaylor isA Parent
  - MariaITaylor isA Relative
  - MariaITaylor isParentOf AdamJTaylor
  - AdamJTaylor isA Child
Further inferences displayed in Protégé
Ontology editors (software)

Two of the best know and most widely used

- **TopBraid Composer** by TopQuadrant

- **Protégé** from Stanford University
  - See information in "Selected Resources" at the end of the tutorial materials
Protégé is a free, open source ontology editor and knowledge-base framework.

The Protégé platform supports modeling ontologies via a web client or a desktop client. Protégé ontologies can be developed in a variety of formats including OWL, RDF(S), and XML Schema.

Protégé is based on Java, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development.

Protégé is supported by a strong community of developers and academic, government and corporate users who are using Protégé for knowledge solutions in areas as diverse as biomedicine, intelligence gathering, and corporate modeling.

Protégé is available from this site as a free download along with plug-ins and ontologies.
Tutorial part 2 summary

- Ontologies are **models** of the entities of interest to a particular domain and the relationships among those entities.
- RDF-based ontologies are machine readable and actionable:
  - Basic-level RDF ontologies are encoded in **RDFS**: RDF Vocabulary Description Language.
- **Ontologies consist of:**
  - **Classes**: *types of entities*
    - Usually in class-subclass **hierarchies**
  - **Properties**: designating *relationships among entities* (members of classes)
    - Usually in property-subproperty **hierarchies**
    - **Domain and range** specifications about allowable subjects and object of properties
  - **Instances** or **individuals** tied to the ontology proper; must conform to the model.
- Ontologies allow **logical inferences**
  - based on class and property hierarchies (inheritance) and domain and range specifications.
Questions?
Examples

Domain ontologies, models, or schemas that use classes, properties, domain, range, class and property hierarchies
CIDOC CRM ontology

CIDOC: International Committee for Documentation

The CIDOC Conceptual Reference Model (CRM) provides definitions and a formal structure for describing the implicit and explicit concepts and relationships used in cultural heritage documentation.

The CIDOC CRM is intended to promote a shared understanding of cultural heritage information by providing a common and extensible semantic framework that any cultural heritage information can be mapped to. It is intended to be a common language for domain experts and implementers to formulate requirements for information systems and to serve as a guide for good practice of conceptual modelling. In this way, it can provide the "semantic glue" needed to mediate between different sources of cultural heritage information, such as that published by museums, libraries and archives.

The CIDOC CRM is the culmination of over 10 years work by the CIDOC Documentation Standards Working Group and CIDOC CRM SIG which are working groups of CIDOC. Since 9/12/2006 it is official standard ISO 21127:2006.

CIDOC = International Committee for Documentation
CRM = Conceptual Reference Model
CIDOC CRM Classes

Diagram showing the hierarchy of CRM classes including:
- Temporal Entity
- Time-Span
- Place
- Dimension
- Persistent Item
- Appellation
- Contact Point
- Thing
- Man-Made Thing
- Legal Object
- Conceptual Object
- Type
- Right
- Information Object
- Physical Man-Made Thing
- Physical Feature
- Physical Thing
- Man-Made Object
- Biological Object
- Information Carrier
- Person
- Group
- Service
- Legal Body
- Creator
<table>
<thead>
<tr>
<th>Property id</th>
<th>Property Name</th>
<th>Entity - Domain</th>
<th>Entity - Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>is identified by (identifies)</td>
<td>E1 CRM Entity</td>
<td>E41 Appellation</td>
</tr>
<tr>
<td>P2</td>
<td>has type (is type of)</td>
<td>E1 CRM Entity</td>
<td>E55 Type</td>
</tr>
<tr>
<td>P3</td>
<td>has note</td>
<td>E1 CRM Entity</td>
<td>E62 String</td>
</tr>
<tr>
<td>P4</td>
<td>has time-span (is time-span of)</td>
<td>E2 Temporal Entity</td>
<td>E52 Time-Span</td>
</tr>
<tr>
<td>P7</td>
<td>took place at (witnessed)</td>
<td>E4 Period</td>
<td>E53 Place</td>
</tr>
<tr>
<td>P10</td>
<td>falls within (contains)</td>
<td>E4 Period</td>
<td>E4 Period</td>
</tr>
<tr>
<td>P12</td>
<td>occurred in the presence of (was present at)</td>
<td>E5 Event</td>
<td>E77 Persistent Item</td>
</tr>
<tr>
<td>P11</td>
<td>- had participant (participated in)</td>
<td>E5 Event</td>
<td>E39 Actor</td>
</tr>
<tr>
<td>P14</td>
<td>- carried out by (performed)</td>
<td>E7 Activity</td>
<td>E39 Actor</td>
</tr>
<tr>
<td>P16</td>
<td>- used specific object (was used for)</td>
<td>E7 Activity</td>
<td>E70 Thing</td>
</tr>
<tr>
<td>P31</td>
<td>- has modified (was modified by)</td>
<td>E11 Modification</td>
<td>E24 Physical Man-Made Thing</td>
</tr>
<tr>
<td>P108</td>
<td>- has produced (was produced by)</td>
<td>E12 Production</td>
<td>E24 Physical Man-Made Thing</td>
</tr>
<tr>
<td>P92</td>
<td>- brought into existence (was brought into existence by)</td>
<td>E63 Beginning of Existence</td>
<td>E77 Persistent Item</td>
</tr>
<tr>
<td>P108</td>
<td>- has produced (was produced by)</td>
<td>E12 Production</td>
<td>E24 Physical Man-Made Thing</td>
</tr>
<tr>
<td>P94</td>
<td>- has created (was created by)</td>
<td>E65 Creation</td>
<td>E28 Conceptual Object</td>
</tr>
<tr>
<td>P93</td>
<td>- took out of existence (was taken out of existence by)</td>
<td>E64 End of Existence</td>
<td>E77 Persistent Item</td>
</tr>
<tr>
<td>P15</td>
<td>was influenced by (influenced)</td>
<td>E7 Activity</td>
<td>E70 Thing</td>
</tr>
<tr>
<td>P16</td>
<td>- used specific object (was used for)</td>
<td>E7 Activity</td>
<td>E5 Event</td>
</tr>
<tr>
<td>P20</td>
<td>had specific purpose (was purpose of)</td>
<td>E70 Thing</td>
<td>E54 Dimension</td>
</tr>
<tr>
<td>P43</td>
<td>has dimension (is dimension of)</td>
<td>E18 Physical Thing</td>
<td>E18 Physical Thing</td>
</tr>
<tr>
<td>P46</td>
<td>is composed of (forms part of)</td>
<td>E18 Physical Thing</td>
<td>E53 Place</td>
</tr>
<tr>
<td>P59</td>
<td>has section (is located on or within)</td>
<td>E89 Propositional Object</td>
<td>E1 CRM Entity</td>
</tr>
<tr>
<td>P67</td>
<td>refers to (is referred to by)</td>
<td>E39 Actor</td>
<td>E30 Right</td>
</tr>
<tr>
<td>P75</td>
<td>possesses (is possessed by)</td>
<td>E52 Time-Span</td>
<td>E61 Time Primitive</td>
</tr>
<tr>
<td>P81</td>
<td>ongoing throughout</td>
<td>E52 Time-Span</td>
<td>E61 Time Primitive</td>
</tr>
<tr>
<td>P82</td>
<td>at some time within</td>
<td>E53 Place</td>
<td>E53 Place</td>
</tr>
<tr>
<td>P89</td>
<td>falls within (contains)</td>
<td>E72 Legal Object</td>
<td>E90 Symbolic Object</td>
</tr>
<tr>
<td>P104</td>
<td>is subject to (applies to)</td>
<td>E90 Symbolic Object</td>
<td>E39 Actor</td>
</tr>
<tr>
<td>P106</td>
<td>is composed of (forms part of)</td>
<td>E74 Group</td>
<td>E55 Type</td>
</tr>
<tr>
<td>P107</td>
<td>has current or former member (is current or former member of)</td>
<td>E24 Physical Man-Made Thing</td>
<td>E90 Symbolic Object</td>
</tr>
<tr>
<td>P127</td>
<td>has broader term (has narrower term)</td>
<td>E70 Thing</td>
<td>E70 Thing</td>
</tr>
<tr>
<td>P128</td>
<td>carries (is carried by)</td>
<td>E13 Attribute Assignment</td>
<td>E1 CRM Entity</td>
</tr>
<tr>
<td>P130</td>
<td>shows features of (features are also found on)</td>
<td>E13 Attribute Assignment</td>
<td>E1 CRM Entity</td>
</tr>
<tr>
<td>P140</td>
<td>assigned attribute to (was attributed by)</td>
<td>E89 Propositional Object</td>
<td>E89 Propositional Object</td>
</tr>
<tr>
<td>P141</td>
<td>assigned (was assigned by)</td>
<td>E89 Propositional Object</td>
<td>E89 Propositional Object</td>
</tr>
<tr>
<td>P148</td>
<td>has component (is component of)</td>
<td>E89 Propositional Object</td>
<td>E89 Propositional Object</td>
</tr>
</tbody>
</table>
New exhibition: Leaving Europe

Why did 30 million Europeans emigrate to America in the 19th/20th centuries? This exhibition of rarely seen pictures tells their story.

From the blog

Betsy, Angel of Prisons
Elizabeth Gurney, or Betsy as she was known, was born on 21 May 1820 and worked at Newgate Prison from 1851 as a warden, becoming the first female prison officer in the UK.
Europeana Data Model (EDM)
Class Hierarchy

The classes introduced by EDM are shown in light blue rectangles. The classes in the white rectangles are re-used from other schemas; the schema is indicated before the colon.

Source: http://pro.europeana.eu/documents/900548/0d0f6ec3-1905-4c4f-96c8-1d817c03123c
Europeana Data Model (EDM) Property Hierarchy

The EDM property hierarchy without the properties included in ESE (Europeana Semantic Elements). The properties introduced by EDM are shown in light blue rectangles. The properties in the white rectangles are re-used from other schemas.
EDM “happened at” property

<table>
<thead>
<tr>
<th>Property name: happenedAt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace: europeana</td>
</tr>
<tr>
<td>URI: <a href="http://www.europeana.eu/schemas/edm/happenedAt">http://www.europeana.eu/schemas/edm/happenedAt</a></td>
</tr>
<tr>
<td>Label: happened at</td>
</tr>
<tr>
<td>Definition: This property associates an event with the place at which the event happened.</td>
</tr>
<tr>
<td>Subproperty of: dc:relation</td>
</tr>
<tr>
<td>Equivalent property: P7 Took place at (CIDOC CRM)</td>
</tr>
<tr>
<td>Domain: ens:Event</td>
</tr>
<tr>
<td>Range: ens:Place</td>
</tr>
<tr>
<td>Europeana note:</td>
</tr>
<tr>
<td>Obligation &amp; Occurrence: An event may have happened at 0 to 1 place, and a place may have 0 to many events that happened at it.</td>
</tr>
<tr>
<td>Example: The creation of Mona Lisa happened at Florence. The excavation of the Egyptian Amphora L2409 happened at Heraklion, Crete.</td>
</tr>
<tr>
<td>Rationale: This property is useful for supporting discoveries concerning places (where query) since it relates a place to the events which happened at that place. In addition, it can be used to browse specific events.</td>
</tr>
</tbody>
</table>
http://bibliontology.com/
BIBO: The Bibliographic Ontology

- **Thing**
  - Agent
  - Collection
  - Periodical
  - Code
  - 'Court Reporter'
  - Journal
  - Magazine
  - Newspaper
  - Series
  - Website

- **Document**
  - Article
  - 'Academic Article'
  - Book
  - Proceedings
  - 'Collected Document'
  - 'Edited Book'
  - Issue
  - Image
  - Map
  - 'Legal Document'
  - 'Legal Case Document'
  - Brief
  - Decision
  - Legislation
  - Bill
  - Statute
  - Manual
  - Manuscript
  - Note
  - Patent

- **Personal Communication**
  - Email
  - Letter
  - Reference Source
  - Report
  - Slideshow
  - Standard
  - Thesis
  - Webpage
  - 'audio document'
  - 'audio-visual document'

- **Document Status**
  - Document
  - 'Book Section'
  - Chapter
  - Excerpt
  - Quote
  - Slide

- **Event**
  - Conference
  - Hearing
  - Interview
  - Performance
  - 'Personal Communication'
  - Workshop

- **Organization**
  - Person
  - Resource
  - Seq
  - Thesis degree

- **topObjectProperty**
  - agent
  - based_near
  - 'cited by'
  - cites
  - contributor
  - director
  - editor
  - interviewee
  - interviewer
  - performer
  - translator
  - court
  - depiction
  - distributor
  - format
  - hasPart
  - homepage
  - isPartOf
  - reproducedIn
  - isReferencedBy
  - subsequentLegalDecision
  - affirmedBy
  - reversedBy
  - isVersionOf
  - 'translation of'
  - language
  - listOfContributors
  - listOfAuthors
  - listOfEditors
  - organizer
  - owner
  - place
  - produced_in
  - 'presented at'
  - producer
  - product
  - publisher

- **topDataProperty**
  - abstract
  - content
  - date
  - created
  - issued
  - 'date argued'
  - description
  - doi = doi
  - eisbn = eisbn
  - edition = edition
  - edition = edition
  - endingPage = 'page end'
  - family_name
  - givenname
  - identifier
  - asin
  - coden
  - doi = doi
  - eanuc13
  - eisbn = eisbn
  - gtin14
  - handle
  - isbn = isbn
  - issn = issn
  - icon
  - oclcnun
  - pmid
  - sici
  - upc
  - uri
  - isbn = isbn
  - issn = issn
  - localityName
  - locator = number
  - chapter
  - issue
  - 'page end' = endingPage
  - 'page start' = volumeStart
BIBO “performer” property: domain and range
BIBFRAME Model Overview

The Bibliographic Framework Transition Initiative is an undertaking by the Library of Congress and the community to better accommodate future needs of the library community. A major focus of the initiative will be to determine a transition path for the MARC 21 exchange format to more Web-based, Linked Data standards. Zepheira and The Library of Congress are working together to develop a Linked Data model, vocabulary and enabling tools/services for supporting this Initiative.

BIBFRAME.ORG is a central hub for this effort.

Getting started

- New to BIBFRAME? A good place to start is the BIBFRAME Model Primer (PDF).
- Want to see library data described in the BIBFRAME Model? Check out the demonstration area.
- You can also see your MARC data in BIBFRAME by using online tools.
- Explore the BIBFRAME vocabulary along with supporting documentation.
- If you code and you want to experiment, head over to the BIBFRAME code repository on GitHub.
- Interested in participating? See how on the contribute page.

Recent updates

- On BIBFRAME Authority - Discussion Paper (NEW - 10 May 2013)
- BIBFRAME Annotation Model - Community Draft (NEW - 2 May 2013)
- Vocabulary updates (Ongoing)
- MARC21 to BIBFRAME Transformation updates (Ongoing)

BIBFRAME.ORG is a collaborative effort of US Library of Congress, Zepheira and you!
BIBFRAME model: main classes

- **Creative Work**
  - a resource reflecting a conceptual essence of the cataloging item.

- **Instance**
  - a resource reflecting an individual, material embodiment of the Work.

- **Authority**
  - a resource reflecting key authority concepts that have defined relationships reflected in the Work and Instance. Examples of Authority Resources include People, Places, Topics, Organizations, etc.

- **Annotation**
  - a resource that decorates other BIBFRAME resources with additional information. Examples of such annotations include Library Holdings information, cover art and reviews.
Introduction to Ontology Concepts and Terminology / Steven J. Miller

DC-2013 Tutorial (Lisbon, Portugal)

Source of diagrams:

Figure 1: A graphical representation of the BIBFRAME Linked Data model defining the relation between Work and Instance resources and their contextualization to Web addressable Authority resources.
Figure 2: A graphical representation of the BIBFRAME Linked Data model in the context of a flexible annotation framework.
### Resource

#### Properties used with Resource

<table>
<thead>
<tr>
<th>Property</th>
<th>Label</th>
<th>Expected value</th>
<th>MARC Mapping</th>
<th>Related RDA Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>authorizedAccessPoint</td>
<td>Authorized access point</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>Description of resource</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>identifier</td>
<td>identifier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>label</td>
<td>Label for resource</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>relatedResource</td>
<td>Related resource</td>
<td></td>
<td>787, 700, 710, 711, with $t$ and i2 not 2, 730, 740, and i2 not 2</td>
<td></td>
</tr>
</tbody>
</table>

#### More specific Resource types

<table>
<thead>
<tr>
<th>Class</th>
<th>Label</th>
<th>MARC Mapping</th>
<th>Related RDA Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation</td>
<td>856</td>
<td></td>
</tr>
<tr>
<td>Authority</td>
<td>Authority</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instance</td>
<td>Instance</td>
<td>856... 260</td>
<td></td>
</tr>
<tr>
<td>Work</td>
<td>Work</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Class hierarchy**

- Annotation
- Authority
- Instance
- Work
Coverage of Content (Property)

**Coverage of Content**

- Used with: Work
- Expected value(s): CoverageEntity OR Literal

- Property
- Domain
- Range
DCMI Metadata Terms:
http://dublincore.org/documents/dcmi-terms/

**DCMI Metadata Terms**

- **Title:** DCMI Metadata Terms
- **Creator:** DCMI Usage Board
- **Identifier:** http://dublincore.org/documents/dc2013/06/14/dcmi-terms/
- **Date Issued:** 2012-06-14
- **Latest Version:** http://dublincore.org/documents/dc2013/06/14/dcmi-terms/
- **Translations:** http://dublincore.org/resources/translations/
- **Document Status:** This is a DCMI Recommendation.
- **Description:** This document is an up-to-date specification of all metadata terms maintained by the Dublin Core Metadata Initiative, including properties, vocabulary encoding schemes, syntax encoding schemes, and classes.

**Table of Contents**

1. Introduction and Definitions
2. Properties in the /terms/ namespace
3. Properties in the /elements/1.1/ namespace
4. Vocabulary Encoding Schemes
5. Syntax Encoding Schemes
6. Classes
7. DCMI Type Vocabulary
# DCMI Metadata Terms

## Index of Terms

| Properties in the /terms/ namespace | abstract, accessRights, accrualMethod, accrualPeriodicity, accrualPolicy, alternative, audience, available, bibliographicCitation, conformsTo, contributor, coverage, created, creator, date, dateAccepted, dateCopyrighted, dateSubmitted, description, educationLevel, extent, format, hasFormat, hasPart, hasVersion, identifier, instructionalMethod, isFormatOf, isPartOf, isReferencedBy, isReplacedBy, isRequiredBy, issued, isVersionOf, language, license, mediator, medium, modified, provenance, publisher, references, relation, replaces, requires, rights, rightsHolder, source, spatial, subject, tableOfContents, temporal, title, type, valid |
| Properties in the /elements/1.1/ namespace | contributor, coverage, creator, date, description, format, identifier, language, publisher, relation, rights, source, subject, title, type |
| Vocabulary Encoding Schemes | DCMIType, DDC, IMT, LCC, LCSH, MESH, NLM, TGN, UDC |
| Syntax Encoding Schemes | Box, ISO3166, ISO639-2, ISO639-3, Period, Point, RFC1766, RFC3066, RFC4646, RFC5646, URI, W3CDTF |
| Classes | Agent, AgentClass, BibliographicResource, FileFormat, Frequency, Jurisdiction, LicenseDocument, LinguisticSystem, Location, LocationPeriodOrJurisdiction, MediaType, MediaTypeOrExtent, MethodOfAccrual, MethodOfInstruction, PeriodOfTime, PhysicalMedium, PhysicalResource, Policy, ProvenanceStatement, RightsStatement, SizeOrDuration, Standard |
| Terms related to the DCMI Abstract Model | memberOf, VocabularyEncodingScheme |
# DC medium property

<table>
<thead>
<tr>
<th>Term Name:</th>
<th>medium</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI:</td>
<td><a href="http://purl.org/dc/terms/medium">http://purl.org/dc/terms/medium</a></td>
</tr>
<tr>
<td>Label:</td>
<td>Medium</td>
</tr>
<tr>
<td>Definition:</td>
<td>The material or physical carrier of the resource.</td>
</tr>
<tr>
<td>Type of Term:</td>
<td>Property</td>
</tr>
<tr>
<td>Refines:</td>
<td><a href="http://purl.org/dc/elements/1.1/format">http://purl.org/dc/elements/1.1/format</a></td>
</tr>
<tr>
<td>Refines:</td>
<td><a href="http://purl.org/dc/terms/format">http://purl.org/dc/terms/format</a></td>
</tr>
<tr>
<td>Has Domain:</td>
<td><a href="http://purl.org/dc/terms/PhysicalResource">http://purl.org/dc/terms/PhysicalResource</a></td>
</tr>
<tr>
<td>Has Range:</td>
<td><a href="http://purl.org/dc/terms/PhysicalMedium">http://purl.org/dc/terms/PhysicalMedium</a></td>
</tr>
<tr>
<td>Version:</td>
<td><a href="http://dublincore.org/usage/terms/history/#medium-003">http://dublincore.org/usage/terms/history/#medium-003</a></td>
</tr>
</tbody>
</table>
# DC physicalMedium & physicalResource Classes

<table>
<thead>
<tr>
<th>Term Name:</th>
<th>PhysicalMedium</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URI:</strong></td>
<td><a href="http://purl.org/dc/terms/PhysicalMedium">http://purl.org/dc/terms/PhysicalMedium</a></td>
</tr>
<tr>
<td><strong>Label:</strong></td>
<td>Physical Medium</td>
</tr>
<tr>
<td><strong>Definition:</strong></td>
<td>A physical material or carrier.</td>
</tr>
<tr>
<td><strong>Comment:</strong></td>
<td>Examples include paper, canvas, or DVD.</td>
</tr>
<tr>
<td><strong>Type of Term:</strong></td>
<td>Class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term Name:</th>
<th>PhysicalResource</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URI:</strong></td>
<td><a href="http://purl.org/dc/terms/PhysicalResource">http://purl.org/dc/terms/PhysicalResource</a></td>
</tr>
<tr>
<td><strong>Label:</strong></td>
<td>Physical Resource</td>
</tr>
<tr>
<td><strong>Definition:</strong></td>
<td>A material thing.</td>
</tr>
<tr>
<td><strong>Type of Term:</strong></td>
<td>Class</td>
</tr>
<tr>
<td><strong>Version:</strong></td>
<td><a href="http://dublincore.org/usage/terms/history/#PhysicalResource-001">http://dublincore.org/usage/terms/history/#PhysicalResource-001</a></td>
</tr>
</tbody>
</table>
Directed graph: what relationship does the DC property “creator” indicate?
DCMI Metadata Terms: 
range declaration of creator property

http://dublincore.org/documents/dcmi-terms/#terms-creator

<table>
<thead>
<tr>
<th>Term Name: creator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URI:</strong></td>
</tr>
<tr>
<td><strong>Label:</strong></td>
</tr>
<tr>
<td><strong>Definition:</strong></td>
</tr>
<tr>
<td><strong>Comment:</strong></td>
</tr>
<tr>
<td><strong>Type of Term:</strong></td>
</tr>
<tr>
<td><strong>Refines:</strong></td>
</tr>
<tr>
<td><strong>Refines:</strong></td>
</tr>
<tr>
<td><strong>Has Range:</strong></td>
</tr>
<tr>
<td><strong>Version:</strong></td>
</tr>
<tr>
<td><strong>EquivalentProperty:</strong></td>
</tr>
</tbody>
</table>

The object of the predicate dc/terms/creator must be a member of the class dc/terms/agent.
DCMI Metadata Terms: Agent class defined

http://dublincore.org/documents/dcmi-terms/#terms-Agent

<table>
<thead>
<tr>
<th>Term Name:</th>
<th>Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI:</td>
<td><a href="http://purl.org/dc/terms/Agent">http://purl.org/dc/terms/Agent</a></td>
</tr>
<tr>
<td>Label:</td>
<td>Agent</td>
</tr>
<tr>
<td>Definition:</td>
<td>A resource that acts or has the power to act.</td>
</tr>
<tr>
<td>Comment:</td>
<td>Examples of Agent include person, organization, and software agent.</td>
</tr>
<tr>
<td>Type of Term:</td>
<td>Class</td>
</tr>
<tr>
<td>Instance Of:</td>
<td><a href="http://purl.org/dc/terms/AgentClass">http://purl.org/dc/terms/AgentClass</a></td>
</tr>
<tr>
<td>Version:</td>
<td><a href="http://dublincore.org/usage/terms/history/#Agent-001">http://dublincore.org/usage/terms/history/#Agent-001</a></td>
</tr>
</tbody>
</table>
Therefore:

The **Object** of the triple must be a member of the **Agent** class. **Douglass** can be an Agent but the **Ebook** cannot.

**Correct:**

- **SUBJECT**: Ebook
  - **PREDICATE**: dc/terms/creator
  - **OBJECT**: Douglass, Frederick, 1817-1895

**Incorrect:**

- **Douglass, Frederick, 1817-1895**
  - **creator**: Ebook
  - **Ebook**: 23/23-h/23-h

**Douglass isCreatorOf ebook**
BBC Sport Ontology

Vocabulary Diagram
The following diagram illustrates the relationships between the key classes in the ontology as applied to the Olympic cycling.

Sports Ontology | Olympics 2012 Track Cycling

- cycling
  - sport:Discipline
  - sport:subDiscipline

- track cycling
  - sport:Discipline
  - sport:discipline

- Men’s Sprint gold medal
  - sport:Award
  - sport:awarded_to

- Chris Hoy
  - foaf:Agent

- Men’s Sprint
  - sport:MultiStageCompetition
  - sport:hasDivision
  - bbc_event:recurringEvent

- Summer Olympics
  - sport:RecurringCompetition

- 2012 Olympics
  - sport:DivisionalCompetition

- Men’s Sprint Semifinals
  - sport:KnockoutCompetition
  - sport:hasRound

- Men’s Sprint Semifinals - Heat 1
  - sport:Round
  - sport:hasUnitCompetition

- Event
  - bbc_event:Event
  - bbc_event:coverage
  - event:place
  - start:dateTime
  - end:dateTime

- Time

- GB Olympic Team 2012
  - sport:CompetitiveSportingGroup

- Great Britain
  - sport:CompetitiveSportingOrganisation

- Olympic Park Velodrome
  - geo:SpatialThing

- Chris Hoy
  - sport:Person

- cyclistRole
  - par:Rule
  - par:holder

- programme

- key

- subclass
  - property

- sport
  - http://www.bbc.co.uk/ontologies/sport/13
  - bbc_event
  - http://www.bbc.co.uk/ontologies/bbc-event
  - par
  - http://uri.org/schema/option/sport
  - geo
  - http://www.w3.org/2003/01/geo/wgs84_pos
  - event
  - http://uri.org/12/Tutorials/event
  - foaf
  - http://xmlns.com/foaf/0.1/
BBC Sport Ontology

Overview Of Terms
An alphabetical index of the ontology terms, divided into classes, properties and individuals. All the terms are hyperlinked to their detailed description for quick reference.

Classes:  | Competition | CompetitionType | CompetitiveSportingGroup | CompetitiveSportingOrganisation |
          | DivisionalCompetition | EventGender | FootballManagerRole | FootballPlayerRole |
          | GroupCompetition | KnockoutCompetition | LeagueCompetition |
          | Match | MultiRoundCompetition | MultiStageCompetition |
          | RecurringCompetition | Round | Session |
          | SportGoverningBody | SportingOrganisation | SportsDiscipline |
          | UnitCompetition

Properties:  | awayCompetitor | competesIn | competitionType | discipline | eventGender | firstRound | firstSession |
             | firstUnitCompetition | hasRound | hasCompetitor | hasGroup | hasMatch | hasSession | hasStage | hasUnitCompetition |
             | homeCompetitor | isCompetitiveSportingOrganisationOf | isGroupOf | isMatchOf | isRoundOf | isSessionOf | isStageOf |
             | lastRound | lastSession | lastUnitCompetition | nextSession | nextUnitCompetition | prevSession | prevUnitCompetition |
             | roundNumber | subDiscipline
BBC Sport Ontology

Property: hasMatch

<table>
<thead>
<tr>
<th>Label</th>
<th>has match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>Range</td>
<td>Match</td>
</tr>
<tr>
<td>Domain</td>
<td>Round</td>
</tr>
</tbody>
</table>

associates a round with a match.

Property: hasSession

<table>
<thead>
<tr>
<th>Label</th>
<th>has session</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>Sub-Properties</td>
<td>firstSession lastSession</td>
</tr>
<tr>
<td>Range</td>
<td>Session</td>
</tr>
<tr>
<td>Domain</td>
<td>Competition</td>
</tr>
</tbody>
</table>

associates a competition with a session.

Property: hasStage

<table>
<thead>
<tr>
<th>Label</th>
<th>has stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>Sub-Properties</td>
<td>firstStage lastStage</td>
</tr>
<tr>
<td>Range</td>
<td>Competition</td>
</tr>
<tr>
<td>Domain</td>
<td>MultiStageCompetition</td>
</tr>
</tbody>
</table>

associates a multi stage competition to the stages that it contains.

Property: hasUnitCompetition

<table>
<thead>
<tr>
<th>Label</th>
<th>has unit competition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>Sub-Properties</td>
<td>firstUnitCompetition hasMatch lastUnitCompetition</td>
</tr>
<tr>
<td>Range</td>
<td>UnitCompetition</td>
</tr>
<tr>
<td>Domain</td>
<td>Round</td>
</tr>
</tbody>
</table>

associates a round to a unit competition in that round.
Questions?
Exercise 1: Ontology Basics & RDFS

1. Identify correct and incorrect hierarchical class and property relationships and property domain and range declarations.

2. Distinguish classes from instances (individuals).

3. Determine logical inferences that can be made based on a set of statements.
Exercise 1: Ontology Basics and RDFS (Tutorial Part 2)

1. **Ontology Classes and Properties:**
Identify correct and incorrect hierarchical class and property relationships and property domain and range declarations for an imaginary cultural heritage ontology.

**Classes:**
- Cultural Heritage Resource
  - Bibliographic Resource
    - Book
    - Journal
      - Journal Article
  - Museum Object
    - Painting
    - Sculpture
- Continent
  - Country
    - Region
    - City
- Agent
  - Creator
    - Author
    - Composer
    - Painter
    - Sculptor
  - Contributor
    - Editor
    - Illustrator

**Properties:**
- isCreatorOf
  - isPainterOf
  - isSculptorOf
- isCreatorOf
domain: Creator
  range: Museum Object
- isPainterOf
domain: Painting
  range: Painter
- isAuthorOf
domain: Author
  range: Bibliographic Resource

2. **Which of the following are classes and which are instances (individuals)?**

- Addis Ababa
- Africa
- City
- Continent
- Country
- Ethiopia
- Europe
- France
- Lisbon
- Paris
- Portugal
3. Logical Inferences.

A. Based on the following statements, what inferences can we make about Person123?

- Agent IsA Class
- Creator subClassOf Agent
- Author subClassOf Creator
- Painter subClassOf Creator
- Sculptor subClassOf Creator
- Person123 isA Painter

B. Based on the following statements, what inferences can we make about WorkABC?

- CulturalHeritageResource isA Class
- MusicalComposition subClassOf CulturalHeritageResource
- Symphony subClassOf MusicalComposition
- WorkABC isA Symphony

C. Based on the following statements, what inferences can be make about Person456 and about WorkABC?

- isCreatorOf isA Property
  - domain: Creator
  - range: CulturalHeritageResource
- isComposerOf isA Property
  - subPropertyOf isCreatorOf
- Person 456 isComposerOf WorkABC

D. Based on the following new statement, in addition to what is stated in C above, what new inferences can we make about Person 456 and WorkABC?

- isComposerOf isA Property
  - domain Composer
  - range MusicalComposition
OWL Overview: Web Ontology Language

Tutorial Part 3
Tutorial part 4 objectives

- Be aware that OWL Web Ontology Language is a full-fledged ontology language with a much higher level of expressive power and logical reasoning / inferencing capabilities than RDFS.
- Be exposed to some of OWL's property characteristics and other constructs and the inferences they enable, including:
  - Inverse, symmetric, transitive, functional, and inverse functional properties, cardinality restrictions, and building anonymous equivalent classes.
- Understand the kinds of queries that an OWL ontology and knowledge-base could be able to answer for end users.
OWL goes beyond RDFS

- RDFS deals primarily with:
  - classes, subclasses, properties, subproperties, domain and range
- OWL is a full-fledged ontology language
  - Not a direct extension of RDFS, but does builds on it
  - Is usually expressed in RDF/XML
  - May be represented graphically (often based on UML)
- OWL allows for much fuller reasoning and inferencing by enabling specifications for:
  - Relations between classes (disjointness, equivalence, union, intersection)
  - Cardinality (minimum, maximum, exact number)
  - Equality (same as)
  - Richer typing of properties (object vs. datatype, specific datatypes)
  - Characteristics of properties / special properties (transitive, symmetric, functional, inverse functional)
  - Enumerated classes
OWL elements (1)

- **Class and individual elements**
  - owl:Class
  - owl:Thing
  - owl:Nothing
  - owl:NamedIndividual

- **RDFS elements used in OWL**
  - rdfs:subClassOf
  - rdfs:domain
  - rdfs:range
  - rdfs:subPropertyOf
  - rdf:Property

- **Datatype specification**
  - xsd:datatypes

- **Property characteristics**
  - owl:ObjectProperty
  - owl:InverseOf
  - owl:TransitiveProperty
  - owl:SymmetricProperty
  - owl:FunctionalProperty
  - owl:InverseFunctionalProperty

- **Cardinality restrictions**
  - owl:minCardinality
  - owl:maxCardinality
  - owl:cardinality
OWL elements (2)

- **Equality/inequality**
  - owl:equivalentClass
  - owl:equivalentProperty
  - owl:sameAs
  - owl:differentFrom
  - owl:AllDifferent
  - owl:distinctMembers

- **Property restrictions**
  - owl:Restriction
  - owl:onProperty
  - owl:allValuesFrom
  - owl:someValuesFrom

- **Class intersection**
  - owl:intersectionOf

**OWL DL & OWL Full**:

- **Class axioms**
  - owl:one of, dataRange
  - owl:disjointWith
  - owl:equivalentClass
    - (applied to class expressions)
  - rdfs:subClassOf
    - (applied to class expressions)

- **Boolean combinations of class expressions**
  - owl:unionOf
  - owl:complementOf
  - owl:intersectionOf

- **Property information**
  - owl:hasValue
OWL elements (3)

- **Ontology header information**
  - owl:Ontology
  - owl:imports

- **OWL versions**
  - OWL 1
  - OWL 2

- **OWL 1 sublanguages**
  - OWL Lite
  - OWL DL
  - OWL Full

- **Ontology versioning information**
  - owl:versionInfo
  - owl:priorVersion
  - owl:backwardCompatibleWith
  - owl:incompatibleWith
  - owl:DeprecatedClass
  - owl:DeprecatedProperty
OWL basic elements

- **owl:Class**
  - a subclass of rdfs:Class

- **owl:Thing**
  - the most general class, which contains everything
OWL properties

In OWL there are two basic kinds of properties

- **Object properties**, which relate objects to other objects
  - When the value of the RDF triple is another “resource” or “thing” represented with a linkable URI
  - E.g. isTaughtBy, supervises, hasChild, isChildOf, creates, createdBy, etc.

- **Data type properties**, which relate objects to datatype values
  - When the value of the RDF triple is a literal value
  - E.g. hasPhoneNumber, hasTitle, hasBirthdate, etc.
OWL datatype properties

OWL makes use of XML Schema data types

- xsd datatypes recommended for use with OWL: 
  http://www.w3.org/TR/owl-guide/#Datatypes1

- For example, among others:
  - xsd:string
  - xsd:integer
  - xsd:nonNegativeInteger
  - xsd:date
  - xsd:boolean

- And: rdfs:Literal

- In OWL XML they are referenced:
  <owl:DatatypeProperty rdf:ID="hasAge">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema
      #nonNegativeInteger"/>
  </owl:DatatypeProperty>
OWL element examples

- The ontology begins with `owl:Ontology`
- Classes are defined using `owl:Class`
  - `owl:Class` rdf:ID `fro:Parent`
  - `fro:Parent` rdfs:subClassOf `fro:Relative`
- Object properties
  - `owl:ObjectProperty` rdf:ID `fro:hasParent`
  - `fro:hasParent` rdfs:domain `fro:Child`
- Datatype properties and ranges
  - `owl:DatatypeProperty` rdf:ID `fro:hasBirthdate`
  - `fro:hasBirthdate` rdfs:range `XMLSchema#date`
- Notice a mixture of `rdf:`, `rdfs:`, and `owl:` elements in OWL
Simplified workshop notation
(examples from previous slide)

- Parent isA Class
- Parent subClassOf Relative
- hasParent isA ObjectProperty
- hasParent domain Child
- hasBirthdate isA DatatypeProperty
- hasBirthdate range XMLSchema#date
Inverse properties
(owl:inverseOf)

● **Statements:**
  ● isParentOf inverseOf hasParent
  ● MariaITaylor isParentOf AdamJTaylor
  ● isParentOf subPropertyOf hasAncestor

● **Inferences:**
  ● hasParent inverseOf isParentOf
  ● AdamJTaylor hasParent MariaITaylor
  ● AdamJTaylor hasAncestor MariaITaylor
Inverse property declaration:
- hasSibling isInverseOf isSiblingOf

Individual statement:
SofiaMTaylor isSiblingOf AdamJTaylor

Inference:
- AdamJTaylor isSiblingOf SofiaMTaylor
**OWL special properties** (1)
(Source: W3C OWL Web Ontology Language Overview: http://www.w3.org/TR/owl-features/)

- **OWL Symmetric Property**: If a property is symmetric, then if the pair (x,y) is an instance of the symmetric property P, then the pair (y,x) is also an instance of P.
  - For example, friend may be stated to be a symmetric property. Then a reasoner that is given that Frank is a friend of Deborah can deduce that Deborah is a friend of Frank.

- **OWL Transitive Property**: If a property is transitive, then if the pair (x,y) is an instance of the transitive property P, and the pair (y,z) is an instance of P, then the pair (x,z) is also an instance of P.
  - For example, if ancestor is stated to be transitive, and if Sara is an ancestor of Louise (i.e., (Sara,Louise) is an instance of the property ancestor) and Louise is an ancestor of Deborah (i.e., (Louise,Deborah) is an instance of the property ancestor), then a reasoner can deduce that Sara is an ancestor of Deborah (i.e., (Sara,Deborah) is an instance of the property ancestor).
  - OWL Lite (and OWL DL) impose the side condition that transitive properties (and their superproperties) cannot have a maxCardinality 1 restriction. Without this side-condition, OWL Lite and OWL DL would become undecidable languages. See the property axiom section of the OWL Semantics and Abstract Syntax document for more information.
Symmetric properties (owl:SymmetricProperty)

- **Statements:**
  - hasSibling isA SymmetricProperty
  - SofiaMTaylor hasSibling AdamJTaylor

- **Inferences:**
  - AdamJTaylor hasSibling SofiaMTaylor
Transitive properties
(owl:TransitiveProperty)

- **Statements:**
  - isAncestorOf isA TransitiveProperty
  - BrianCTaylor isAncestorOf JosephBTaylor
  - JosephBTaylor isAncestorOf SofiaMTaylor

- **Inferences:**
  - BrianCTaylor isAncestorOf SofiaMTaylor
OWL special properties
(Source: W3C OWL Web Ontology Language Overview: http://www.w3.org/TR/owl-features/)

- **OWL Functional Property**: If a property is a FunctionalProperty, then it has no more than one value for each individual (it may have no values for an individual). This characteristic has been referred to as having a unique property. FunctionalProperty is shorthand for stating that the property's minimum cardinality is zero and its maximum cardinality is 1.
  - For example, `hasPrimaryEmployer` may be stated to be a FunctionalProperty. From this a reasoner may deduce that no individual may have more than one primary employer. This does not imply that every Person must have at least one primary employer however.

- **OWL Inverse Functional Property**: If a property is inverse functional then the inverse of the property is functional. Thus the inverse of the property has at most one value for each individual. This characteristic has also been referred to as an unambiguous property.
  - For example, `hasUSSocialSecurityNumber` (a unique identifier for United States residents) may be stated to be inverse functional (or unambiguous). The inverse of this property (which may be referred to as `isTheSocialSecurityNumberFor`) has at most one value for any individual in the class of social security numbers. Thus any one person's social security number is the only value for their `isTheSocialSecurityNumberFor` property.
  - From this a reasoner can deduce that no two different individual instances of Person have the identical US Social Security Number.
  - Also, a reasoner can deduce that if two instances of Person have the same social security number, then those two instances refer to the same individual.
Functional properties
(owl:FunctionalProperty)

- **Statements:**
  - hasBirthdate isA FunctionalProperty
  - SofiaMTaylor hasBirthdate "1999-01-15"
  - SofiaMTaylor hasBirthdate "1999-01-05"

- **Inferences:**
  - Error in ontology: an individual may have only one unique value for the hasBirthdate property
Ontology Error Notification in Protégé

Your ontology is inconsistent which means that the OWL reasoner will no longer be able to provide any useful information about the ontology.

You have several options at this point:
- Click the Explain button to try the Protege reasoner.
- If you think you know what the problem is, you can double-click on the individuals listed below to see an explanation of an inconsistency.
- Some reasoners come with commands for inconsistent ontologies.

For SofiaMTaylor:

- Functional: hasBirthDate

- hasBirthDate "1999-01-15"
- hasBirthDate "1999-01-31"
Functional properties
(owl:FunctionalProperty)

- **Statements:**
  - hasMother isA FunctionalProperty
  - SofiaMTaylor hasMother MariaITaylor
  - SofiaMTaylor hasMother MariaIGarciaTaylor

- **Inferences:**
  - SofiaMTaylor may have only one individual who is her mother
  - Because of the non-unique names assumption, however, in OWL (and Protégé) there is no error because there can be no inference that MariaITaylor[URI] and MariaIGarciaTaylor[URI] are different individuals
  - In this hypothetical case, they are in fact the same individual with two different URIs
SW and OWL assumptions

- **Open World Assumption**
  - **Closed world**: databases with tightly controlled content; all relevant information about an entity is included; inferences can be made accordingly
  - **Open world**: uncontrolled open data; someone can always contribute something new about an entity
    - Machine inferencing must take this into account: “we may draw no conclusions that rely on assuming that the information available at any one point is all the information available”

- **Nonunique Naming Assumption**
  - **Unique names**: may hold in controlled databases or triple stores
  - **Nonunique names**: in an open world context, different Web authors will use different URIs for the same entity / resource
    - Machine inferencing cannot assume that two entities with different URIs are different individuals

Different individuals
(owl:differentFrom)

- **Statements:**
  - hasMother isA FunctionaProperty
  - SofiaMTaylor hasMother MariaITaylor
  - SofiaMTaylor hasMother AdinaRTaylor
  - MariaITaylor differentFrom AdinaRTaylor

- **Inferences:**
  - Error in ontology: one individual may have only one other individual as the value of the hasMother property, and these two have now been asserted to be different individuals
Inverse functional properties
(owl:InverseFunctionalProperty)

● **Statements:**
  ● hasUSPassportNumber isA InverseFunctionalProperty
  ● MariaITaylor hasUSPassportNumber “12345678”
  ● JosephBTaylor hasUSPassportNumber “12345678”

● **Inferences:**
  ● Only one individual may have any given passport number (for a specific country)
  ● No error in the inferencing, however, because the non-unique names assumption does not allow the conclusion that the URIs for MariaITaylor and JosephBTaylor are for different individuals
Different individuals (owl:differentFrom)

- **Statements:**
  - hasUSPassportNumber isA FunctionalProperty
  - MariaITaylor hasUSPassportNumber 12345678
  - JosephBTaylor hasUSPassportNumber 12345678
  - MariaITaylor differentFrom JosephBTaylor

- **Inferences:**
  - Error in ontology: different individuals cannot have the same value for the hasUSPassportNumber property
A better inverse functional property example

- A health care network assigns unique patient IDs
- Different doctors offices, clinics, and hospitals in the network are beginning to share medical records within new semantic environment

**Statements:**
- hasPatientID isA InverseFunctionalProperty
- *Doctor Lee’s office:*
  - Linda-A-Brown-URI hadAppointmentDate “2013-02-06”
  - Linda-A-Brown-URI treatedFor xyzDisease
  - Linda-A-Brown-URI hasPatientID 987654
- Hilltop Hospital
  - Linda-A-Porter-URI treatedFor abcDisease
  - Linda-A-Porter-URI hasPatientID 987654

**Inferences:**
- Linda-A-Brown-URI and Linda-A-Porter-URI are the same individual
- This individual has been treated for xyzDisease and abcDisease
- This individual had an appointment with Dr. Lee on February 6, 2013 and was in Hilltop Hospital May 10-12, 2013
- etc.
Disjoint classes (owl:disjointWith)

- An individual can be a member of at most one of any set of classes declared to be disjointWith each other
- **Statements:**
  - Father disjointWith Mother [i.e., biological parents]
  - Parent disjointWith Child $\Leftarrow$ incorrect! why?
  - JosephBTaylor isA Father
- **Inference:**
  - JosephBTaylor cannot be a Mother
- **Statements:**
  - JosephBTaylor isA Father
  - JosephBTaylor isA Mother
- **Inference:**
  - Error in ontology: the last two statements above cannot both be true. Because the Father and Mother classes are disjoint, the same individual cannot be a member of both classes
OWL restrictions, equality, etc.

- **Cardinality restrictions**
  - owl:minCardinality
  - owl:maxCardinality
  - owl:Cardinality

- **OWL property restrictions**
  - owl:Restriction
  - owl:onProperty
  - owl:allValuesFrom
    - specifies *universal* quantification
  - owl:hasValue
    - specifies a specific value
  - owl:someValuesFrom
    - specifies *existential* quantification

- **Equality/inequality**
  - equivalentClass
  - equivalentProperty
  - sameAs
  - differentFrom
  - AllDifferent
  - distinctMembers

- **Class intersection**
  - intersectionOf
Maximum cardinality

● **Statements**
  ● hasParent isA ObjectProperty
  ● hasParent range Restriction:
    ● onProperty resource hasParent
    ● onClass resource Parent
    ● maxCardinality datatype nonNegativeInteger “2”
  ● MariaITaylor hasParent TomásCGarcia
  ● MariaITaylor hasParent IsabellaAGarcia
  ● MariaITaylor hasParent SofiaGTaylor

● **Inferences**
  ● Any individual may have a maximum of two individuals as the values of the hasParent property
  ● It cannot be inferred that any of the above individuals are not the same individual, following the non-unique names assumption
  ● But if all three were declared differentFrom each other, there would be an ontology error
Maximum cardinality: OWL RDF/XML

<owl:ObjectProperty rdf:about="fro;hasParent">
  <rdfs:range>
    <owl:Restriction>
      <owl:onProperty rdf:resource="fro;hasParent"/>
      <owl:onClass rdf:resource="fro;Parent"/>
      <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:range>
</owl:ObjectProperty>
Anonymous equivalent class, class intersection, and equality (owl:equivalentClass, owl:intersectionOf, owl:Restriction, owl:someValuesFrom)

- Define the Aunt class as equivalent to the class of all Relatives which are Female and have at least one Sibling that is a member of the Parent class:

  - Aunt subClassOf Relative
  - Aunt isA equivalentClass:
    - intersectionOf:
      - isA Female
      - Restriction:
        - hasSibling
        - someValuesFrom Parent

- This class is an "anonymous class" or "unnamed class," which is the equivalent of the intersection of the characteristics asserted above. All of our other classes so far have been "named classes."
Equivalent class, class intersection, and equality: OWL RDF/XML

```xml
<owl:Class rdf:about="fro.owl#Aunt">
  <rdfs:subClassOf rdf:resource="fro.owl#Relative"/>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="fro.owl#Female"/>
        <owl: Restriction>
          <owl:onProperty rdf:resource="fro.owl#hasSibling"/>
          <owl:someValuesFrom rdf:resource="fro.owl#Parent"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```
Inferences from previous equivalent class assertions

● **Statements:**
  ● AdinaRTaylor isA Relative
  ● AdinaRTaylor isA Female
  ● AdinaRTaylor hasSibling JosephBTaylor
  ● JosephBTaylor isA Parent

● **Inferences:**
  ● AdinaRTaylor isA Aunt
Equivalent class example 2

- Define the Husband class as equivalent to the class of all Persons which are Male and have at least one Spouse that is a member of the Person class.

- \textbf{Husband subClassOf Person}
- \textbf{Husband isA equivalentClass:}
  - \textbf{intersectionOf:}
    - \textbf{isA Male}
    - \textbf{Restriction:}
      - \textbf{hasSpouse}
      - \textbf{someValuesFrom Person}

[Side note: in this particular model, the husband’s spouse may be either female or male, such that two men or two women may be spouses to each other, two men may be husbands to each other, and two women may be wives to each other]
Protégé inference examples

- Protégé OWL 4.2 includes two software reasoners: FaCT++ and HermiT

Three statements I made about JosephBTaylor
(one class membership and two property assertions):
Inferences about JosephBTaylor:
Various class and property relationship inferences made by the FaCT++ Reasoner --made on the basis of class and property hierarchies, domain and range declarations, various OWL property type declarations, and property relations stated about other individuals
Inferences about AdinaRTaylor
Class membership and individual relationships graph for JosephBTaylor, generated by OntoGraf in Protégé
Mouseover an arc in OntoGraf: displays relationship (triple statement)
Mouseover an individual in OntoGraph: displays URI and property assertions
Protégé: View of the Mother Class
Query examples

The Family Relationships Ontology and Knowledge Base will allow such queries as the following to be answered:

- Who are the parents of TomasCGarcia?
- Where were they born, what were their birthdates, who were their other children?
- Who are all of the ancestors of SofiaMTaylor?
- Give me all of the female ancestors of JosephBTaylor who were born between 1800 and 1950
- Give me all of the male relatives of SantiagoCGarcia who were not parents
- etc.
Questions?
Exercise 2: OWL Ontologies

1. Determine which properties are likely to be Object Properties and Datatype Properties in OWL.
2. Match a set of properties to relevant OWL property types.
3. Determine which OWL property would be useful for relating two resources to each other.
4. Determine logical inferences that can be made based on a set of statements.
5. Brainstorm ideas for classes, properties, inferences, and query possibilities for domain ontologies & knowledge-bases for:
   A. 2016 Summer Olympic Games
   B. Health care symptoms, diseases, and treatments
Exercise 2: OWL Ontology Overview (Tutorial Part 3)

1. Which of the following are likely to be Object Properties and which Datatype Properties in OWL? (Context: a cultural heritage resource ontology.)
   - coAuthoredWith
   - hasUniqueArtistNameID
   - isRevisionOf
   - wasBornIn

2. Match each property in the left column with a relevant OWL property type in the right column:

<table>
<thead>
<tr>
<th>Ontology Properties</th>
<th>OWL Property Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>coAuthoredWith</td>
<td>Functional</td>
</tr>
<tr>
<td></td>
<td>Inverse Functional</td>
</tr>
<tr>
<td></td>
<td>Symmetric</td>
</tr>
<tr>
<td></td>
<td>Transitive</td>
</tr>
<tr>
<td>hasUniqueArtistNameID</td>
<td>Domain: Artist</td>
</tr>
<tr>
<td></td>
<td>Range: string</td>
</tr>
<tr>
<td>isRevisionOf</td>
<td>Domain: ArtWork</td>
</tr>
<tr>
<td></td>
<td>Range: ArtWork</td>
</tr>
<tr>
<td>wasBornIn</td>
<td>Domain: Creator</td>
</tr>
<tr>
<td></td>
<td>range: City</td>
</tr>
</tbody>
</table>

3. Which OWL property would be useful for relating the following resources to each other?
   - Library of Congress Name Authority file (LCNAF) URI [http://id.loc.gov/authorities/names/n81046163](http://id.loc.gov/authorities/names/n81046163) for Aung San Suu Kyi [1945-, Burmese opposition politician].
   - Virtual International Authority File (VIAF) URI [http://viaf.org/viaf/112144921](http://viaf.org/viaf/112144921) for Aung San Suu Kyi [1945-, Burmese opposition politician].
4. Logical inferences.

A. Based on the following statements, what inferences can we make about Person456?
   - coAuthoredWith isA SymmetricProperty
   - Person123 coAuthoredWith Person456

B. Based on the following statements, what inferences can we make about WorkABC and WorkDEF?
   - isRevisionOf isA TransitiveProperty
   - WorkABC isRevisionOf WorkDEF
   - WorkDEF isRevisionOf WorkGHI
   - WorkGHI isRevisionOf WorkXYZ

C. Based on the following statements, what inferences can we make about the Artist Name ID number "1234567"?
   - hasUniqueArtistNameID isA InverseFunctionalProperty
   - Person123 hasUniqueArtistNameID "1234567"

D. Suppose that your ontology includes the classes and properties stated below. How could you create a new class of all Expressionist painters using the OWL anonymous equivalent class capabilities?
   - Painter isA Class
   - ArtisticStyle isA Class
     - Mannerism subClassOf ArtisticStyle
     - Expressionism subClassOf ArtisticStyle
     - Cubism subClassOf ArtisticStyle
   - hasArtisticStyle isA Property
     - domain: Artist
     - Range: ArtisticStyle

5. Brainstorm ideas for classes, properties, inferences, and especially query possibilities that might be used in domain ontologies and knowledge-bases for:
   A. 2016 Summer Olympic Games
   B. Health care symptoms, diseases, and treatments (for laypeople and health care providers)
Tutorial Conclusion

Final Questions, Comments, Discussion?
Selected Readings and Resources

Books:

Detailed technical introductions to RDF, ontologies, RDFS, OWL, reasoning, querying, applications, etc:


Detailed technical introduction to SPARQL:


General overview of Linked Data and Semantic Web for cultural heritage data:


Selected W3C Standards Documents:

- Semantic Web Standards: [http://www.w3.org/standards/semanticweb/](http://www.w3.org/standards/semanticweb/)
- Linked Data: [http://www.w3.org/standards/semanticweb/data](http://www.w3.org/standards/semanticweb/data)
- Ontologies/Vocabularies: [http://www.w3.org/standards/semanticweb/ontology.html](http://www.w3.org/standards/semanticweb/ontology.html)
- RDF Resource Description Framework: [http://www.w3.org/standards/techs/rdf](http://www.w3.org/standards/techs/rdf)
- OWL Web Ontology Language: [http://www.w3.org/standards/techs/owl](http://www.w3.org/standards/techs/owl)
  - OWL 2 Web Ontology Language Primer (Second Edition): [http://www.w3.org/TR/2012/REC-owl2-primer-20121211/](http://www.w3.org/TR/2012/REC-owl2-primer-20121211/)
- Query: [http://www.w3.org/standards/semanticweb/query](http://www.w3.org/standards/semanticweb/query)
- Inference: [http://www.w3.org/standards/semanticweb/inference.html](http://www.w3.org/standards/semanticweb/inference.html)
- SKOS: [http://www.w3.org/2004/02/skos/](http://www.w3.org/2004/02/skos/)

Ontology Creation:

A good, practical introduction to creating an ontology:

• It refers to some aspects of an older version or Protégé, but 95% of it remains valid and relevant.

Protégé Ontology Editor
• http://protege.stanford.edu/
• Protégé is a free, open source ontology editor and knowledge-base framework.
• The Protégé platform supports modeling ontologies via a web client or a desktop client. Protégé ontologies can be developed in a variety of formats including OWL, RDF(S), and XML Schema.
• There are several versions of Protégé available for download. I have been using Protégé 4.1 for OWL Ontologies. There is now a Protégé 4.3 available, as well as a new WebProtege which I have not used myself.

A good, full-fledged tutorial for Protégé-OWL:
• This tutorial document takes you through using Protégé OWL to build an OWL ontology step-by-step.

Other RDF and ontology editors to be aware of:
• TopBraid Composer: http://www.topquadrant.com/products/TB_Composer.html
• Altova SemanticWorks: http://www.altova.com/semanticworks.html

Domain Ontology Examples:
  o RDFS: p. 170-171; Graphical illustrations: p. 172-176
• BBC Programmes Ontology: http://purl.org/ontology/po/
• BBC Wildlife Ontology: http://purl.org/ontology/wo/
• BIBO: The Bibliographic Ontology: http://bibliontology.com/
  o OWL RDF file: http://purl.org/ontology/bibo/
• GeoNames Ontology: http://www.geonames.org/ontology/documentation.html
  o OWL RDF file: http://purl.org/ontology/geonames/ontology_v3.01.rdf
• Music Ontology Specification: http://www.musicontology.com/
  o OWL file: http://motools.sourceforge.net/doc/musicontology.rdfs
Ontology Libraries:

- Protégé Ontology Library: http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library
- Swoogle: http://swoogle.umbc.edu/
- SemWebCentral: http://www.semwebcentral.org/
- Ontology Engineering Group: http://www.oeg-upm.net/
- Open Biological and Biomedical Ontologies: http://obofoundry.org/
- W3C ontology repositories: http://www.w3.org/wiki/Ontology_repositories
- SchemaWeb: http://www.schemaweb.info/

Articles, Papers, Presentations, Tutorials, Videos, Technical Documents, Websites:

Introduction to RDF, Semantic Web and Linked Data

  - Note: RDFa is one of several methods of using RDA on the Web. The video includes a nice introduction to RDF and triples in general but also includes many specifics of the N3 encoding syntax.
- Tauberer, Joshua. [2005]. "Quick Intro to RDF." http://www.rdfabout.com/quickintro.xpd

OWL Ontologies


Libraries, RDA, Bibliographic Data, Linked Data, and the Semantic Web

- The RDA (Resource Description and Access) Vocabularies in the Open Metadata Registry: http://rdvocab.info/
BIBFRAME

- BIBFRAME.ORG website: http://bibframe.org/

Dublin Core Metadata Initiative (DCMI)

- DCMI Metadata Terms: http://dublincore.org/documents/dcmi-terms/
- DCMI Abstract Model: http://dublincore.org/documents/abstract-model/

SKOS

- SKOS Simple Knowledge Organization System website: http://www.w3.org/2004/02/skos/
- SKOS Primer: http://www.w3.org/TR/skos-primer/

Europeana


Google

- Google: "Rich snippets (microdata, microformats, and RDFa)." http://www.google.com/support/webmasters/bin/answer.py?answer=99170
How to View XML, RDF, SKOS, and OWL (.xml, .rdf, .owl) files

**Viewing files in their native XML or other serialization (encoding) formats.**

- You may click on the link to an RDF, SKOS, or other file, and in some cases it will display in your browser, other times it won’t.
- You may choose to open the file, or save the file to your computer and then open it, in a plain text editor like Notepad.
- I recommend the free [Notepad++] text editor as usually preferable for display and reading the file.

**Viewing triples and graphs**

- You may copy and paste RDF XML code into the input box in the [W3C RDF Validation Service](http://www.w3.org/RDF/Validator/).
- You may select to display the results as Triples only, Graph only, or both. Click on Parse RDF to see the results.
- An alternative below the direct input box is to check by URI, copying and pasting or manually entering the URI for the RDF XML file and making the same display selections. Click on Parse URI.
- This works for BIBFRAME and SLOS files as well as other RDF files.

**Viewing ontologies in an ontology editor**

- Besides viewing the native XML, you may also view OWL ontology files in an ontology editor.
- You may download the free [Protégé OWL ontology editor](http://protege.stanford.edu/) from [http://protege.stanford.edu/](http://protege.stanford.edu/). I currently use Protégé 4.2, but there is now a 4.3 available.
- You may open an ontology file such as the Bibliographic Ontology Specification (BIBO) in Protégé from its URL or save the file to your computer and open it from there software from also open. The URL for the BIBO ontology file is [http://purl.org/ontology/bibo/](http://purl.org/ontology/bibo/). It is also available from this page: [http://bibliontology.com/specification](http://bibliontology.com/specification).
- For anyone interested, I will make my Family Relationships Ontology file available to you after the workshop at a URL address I will give later.