Universidad Carlos III de Madrid

Knowledge Representation Languages for the Semantic Web

Luis Sánchez Fernández

Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid

Copyright Notice

- Part of the content in these slides is extracted from:
  - **RDF Primer. W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-primer/. Copyright © 2004 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.**

Review Introduction Session

- Basic idea:
  - Some web tasks cannot be automatized because require to have knowledge about Web content currently available to the humans
  - Goal: make this content understandable by computers

3

RDF

- "Resource Description Framework" (RDF)
- Goal of RDF (alternative views):
  - Language for resource description in the Web
  - Language for formal representation of (parts of) information available in a Web document (metadata)
    - Formal => machine readable
    - Vocabulary defined with ontologies
- What is a resource?
  - Web content: Web pages, images, e-mails, files, …
  - Resources *mentioned* in Web content: Persons, locations, organizations, …

4

RDF basic principles

- We want to represent a piece of information available in the Web describing a resource
- Each metadata states a property that can be modelled as a (formal) statement, composed of:
  - subject: resource being described
  - predicate: property of the resource
  - object: value of the property for the resource being described
  - "http://www.example.org has a creator whose value is John Smith"

5

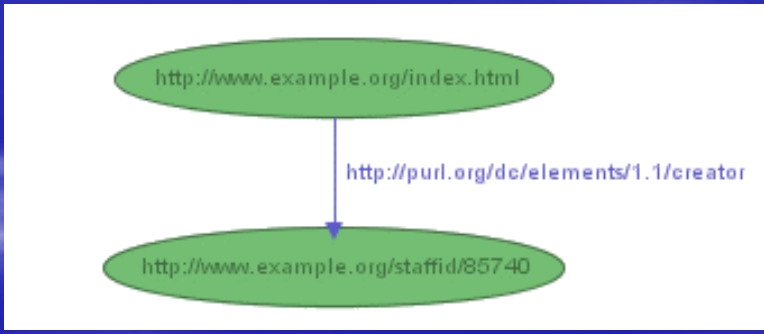
RDF Model

- An RDF model (set of RDF statements) can be represented by means of a graf
- For each statement:
  - subject is a node
  - predicate is an arc
  - object is a node
- Subject and predicate are resources
- Object can be either a resource or a literal

9

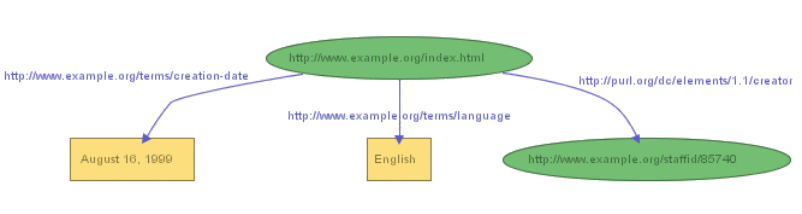
Example

- "http://www.example.org has a creator whose value is John Smith".



10

Example II



11

Textual notation (triples)

```
<http://www.example.org/index.html>
   <http://purl.org/dc/elements/1.1/creator>
   <http://www.example.org/staffid/85740> .
<http://www.example.org/index.html>
   <http://www.example.org/terms/creation-date>
  "August 16, 1999" .
<http://www.example.org/index.html>
   <http://www.example.org/terms/language>
  "English“ .
```

12

Short Representation

- With namespaces

```
ex:index.html dc:creator exstaff:85740 .
ex:index.html exterms:creation-date "August 16, 1999“ .
ex:index.html exterms:language "English" .
```

13

Typed literals

- It is possible to define a datatype for a given literal
- There is no native type model in RDF
- Datatypes should be imported
  - It is recommended to use XML Schema

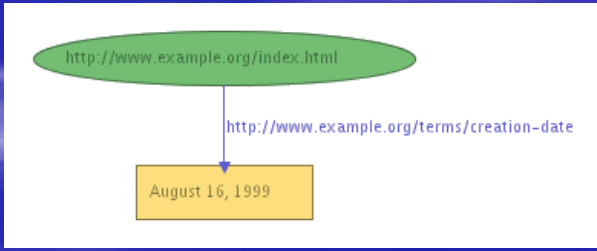
```
<http://www.example.org/staffid/85740>
<http://www.example.org/terms/age>
"27"^^<http://www.w3.org/2001/XMLSchema#integer> .
```

```
exstaff:85740 exterms:age "27"^^xsd:integer .
```

19

XML Syntax for RDF

- There exist an XML syntax for representing RDF graphs



20

XML Syntax for RDF

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3. xmlns:ex="http://www.example.org/terms/">

4. <rdf:Description rdf:about="http://www.example.org/index.html">
5. <ex:creation-date>August 16, 1999</ex:creation-date>
6. </rdf:Description>

7. </rdf:RDF>
```

21

rdf:type

- Predifined RDF property to state that a resource belongs to certain class

22

Ontology Languages for the Semantic Web

This slide has been created by Asunción Gómez Pérez (Universidad Politécnica de Madrid)

Ontologies and Metadata

Ontologies: goal

- **An ontology is a formal, explicit specification of a shared conceptualization**
- **An ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary**

30

RDF Schema

- RDF vocabulary
  - Properties definition and description of properties
  - Classes definition and description
- Can be used to define simple ontologies

32

Classes in RDF Schema

- rdfs:Class
  - Class of all RDF Schema classes
  - Can be used to define that a resource is a class
- rdfs:Resource
- rdf:Property

33

Properties in RDF Schema

- rdfs:subPropertyOf
- rdfs:range
- rdfs:domain
- rdfs:subClassOf

34

RDF and RDF Schema limitations

- No localised range and domain constraints
  - Can't say that the range of hasChild is person when applied to persons and elephant when applied to elephants
- No existence/cardinality constraints
  - Can't say that all *instances* of person have a mother that is also a person, or that persons have exactly 2 parents
- No transitive, inverse or symmetrical properties
  - Can't say that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical

35

OWL

- W3C standard to define Semantic Web related ontologies
- More powerful than RDF Schema
- 3 subsets: OWL Full, OWL DL y OWL Lite
- Case study: Pizzas ontology
  - with the Protégé ontology editor
  - http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/

36

Named Classes

- Example:
  - Pizza
  - PizzaTopping
  - PizzaBase

37

Disjoint classes

- When an instance cannot belong to two of the (disjoint) classes
  - Pizza, PizzaTopping and PizzaBase are disjoint classes
- If not stated explicitly, classes are not assumed to be disjoint

38

Use OWL Wizards

- Disjoint subclasses of PizzaBase
  - ThinAndCrispyBase
  - DeepPanBase
- Disjoint subclasses of PizzaTopping
  - MeatTopping
    - SpicyBeefTopping, PepperoniTopping, SalamiTopping, HamTopping
  - VegetableTopping
    - TomatoTopping, OliveTopping, MushroomTopping, PepperTopping, OnionTopping, CaperTopping
      - PepperTopping subclasses: RedPepperTopping, GreenPepperTopping, JalapenoPepperTopping
  - CheeseTopping
    - MozzarellaTopping, ParmezanTopping
  - SeafoodTopping
    - TunaTopping, AnchovyTopping, PrawnTopping

39

Properties

- Object properties
- Datatype properties
- In OWL-DL object properties and datatype properties should be disjoint
- Create an object property named hasIngredient, with subproperties hasTopping, hasBase
  - Set domain and range

40

Properties

- Inverse properties
  - isIngredientOf, isBaseOf, isToppingOf
- Property characteristics
  - Functional
    - hasBase
  - Inverse functional
    - ?
  - Symmetric
  - Transitive

41

Property Restrictions I

- Restrict which individuals can belong to a class
- Quantifier restrictions
  - Existential quatifier
    - ∃ property class
      - Pizza ⊆ ∃ hasBase PizzaBase
        - GINO'S pan pizza

42

Different Kinds of Pizzas

- Class NamedPizza, subclass of Pizza
- Class MargheritaPizza, subclass of NamedPizza
  - "A pizza that has only Mozzarella and Tomato toppings"
- AmericanaPizza
  - "A pizza that has only Mozzarella, Tomato and Pepperoni toppings"
- AmericanHotPizza
  - "A pizza that has only Mozzarella, Tomato, Pepperoni and Jalapeno toppings"
- SohoPizza
  - "A pizza that has only Olive, Parmezan, Mozzarella and Tomato toppings"
- Add existential restrictions

43

Using a reasoner I

- Check consistency
  - Ex. Add topping of both types vegetable and cheese

44

Defined classes

- Should have a necessary and sufficient restriction
- Example: ChessyPizza = “A Pizza that has at least one Cheese topping”
- Primitive classes = non defined classes
- A class may have several necessary restrictions

45

Reasoning II

- Asserted hierarchy: we explicitly say that class A is subclass of class B
- Inferred hierarchy: subclass relationships detected by the reasoner
- Why AmericanaPizza is not subclass of MargheritaPizza?

46

Property Restrictions II

- Universal quantifier
  - ∀ property class
- VegetarianPizza: can only have Cheese or Vegetable toppings
- Why MargheritaPizza is not subclass of VegetarianPizza?

47

Closure Axiom

- Universal quantifier
  - Restricts the values some property can have for instances of a given class.
  - Such values are the classes defined in existential restrictions for the same property
- Example: MargueritaPizza, SohoPizza

48

Cardinality Restrictions

- maxCardinality
- minCardinality
- cardinality
- Example: InterestingPizza: pizza with 3 or more toppings

49

Creating Individuals

- Create instance
- Define property over instances
- Define instances as different
- hasValue restriction
- Enumerated classes

50