NSWI144 – Linked Data – Lecture 3 – 22 October 2012

RDFS, OWL

Martin Svoboda

Faculty of Mathematics and Physics

Charles University in Prague

Outline

- **Schema languages**
  - RDFS
  - OWL
- **Vocabularies**
  - FOAF, DC, SKOS

Vocabularies

- Motivation
  - **Vocabulary = set of terms + usage descriptions**
    - I.e. what resources we want to use in statements…
    - … and how exactly we want to use them
- Questions
  - **Model**
    - What descriptions should we provide?
  - **Language**
    - How should we express these descriptions?

Vocabularies

- Descriptions
  - **… in theory**
    - Intended real-world meaning of terms
      - E.g. classes are used to represent types of resources
    - Intended positions of terms in triples
      - E.g. properties are usually expected to act as predicates
    - …
  - **… in practice**
    - RDFS, OWL

Vocabularies

- **Schema languages**
  - RDFS – classes and properties
  - OWL – much more expressive descriptions
- **Vocabularies**
  - RDF, RDFS, OWL
    - RDFS and OWL schema languages are based on RDF, therefore we can view them as ordinary vocabularies
    - Analogy: XML Schema language for XML documents
  - DC, FOAF, …

Vocabularies

- Schemata in general
  - **Relational databases**
    - Tables, integrity constraints, …
  - **XML documents**
    - DTD, XML Schema, Relax NG, …
  - **RDF**
    - RDFS, OWL, …
    - Different interpretation:
      - They can also provide **knowledge for new data inference** rather than standard restrictions on data only!

RDF Schema

- **RDFS**
  - **RDF Vocabulary Description Language**
    - RDFS schemata are expressed as RDF graphs
  - W3C
    - http://www.w3.org/TR/rdf-schema/
  - Constructs
    - Definition of classes and properties
    - Domains and ranges of properties
    - …

Classes

- Motivation
  - Determine type of things we want to describe
    - Do not confuse *types* and *data types* for literals
  - Resources belonging to a class are its **instances**
  - **Class** vs. **class extension** (set of its instances)
- Terms
  - Classes **rdfs:Resource**, **rdfs:Class**
  - Properties **rdf:type**, **rdfs:subClassOf**

Classes

- **Type assignment**
  - Property **rdf:Type**
  - *InstanceRef* `rdf:type` *ClassName* `.`
    - Resource can be associated with multiple types
    - *ClassName* is usually an explicitly defined class
- **Class definition**
  - Class **rdfs:Class**
  - *ClassName* `rdf:type rdfs:Class .`
    - Class names are written with an initial uppercase letter

Classes

- **Class hierarchy**
  - Class **rdfs:Resource**
    - Class for all resources = things we want to talk about
    - Therefore, any resource is an instance of rdfs:Resource
- **Class relations**
  - Property **rdfs:subClassOf**
  - *SubClass* `rdfs:subClassOf` *BaseClass* `.`
    - Any *SubClass* instance is also an instance of *BaseClass*
    - This property is transitive

Properties

- Motivation
  - Description of terms to be used as predicates
- Terms
  - Class **rdfs:Property**
  - Property **rdfs:subPropertyOf**
  - Properties **rdfs:range**, **rdfs:domain**

Properties

- **Property definition**
  - Class **rdfs:Property**
  - *propertyName* `rdf:type rdfs:Property .`
    - Properties are written with an initial lowercase letter
- **Property relations**
  - Property **rdfs:subPropertyOf**
  - *subProperty* `rdfs:subPropertyOf` *baseProperty* `.`
    - All items related by *subProperty are also related by baseProperty*
    - This property is transitive

Properties

- **Property ranges**
  - Property **rdfs:range**
  - *propertyName* `rdfs:range` *ClassName* `.`
    - Values of property *propertyName* are instances of class *ClassName*
    - There can be multiple different classes

Properties

- **Property domains**
  - Property **rdfs:domain**
  - Definition
    - *propertyName* `rdfs:domain` *ClassName* `.`
  - Meaning
    - Any resource that has property *propertyName* is an instance of class *ClassName*
    - There can be multiple different classes

Class Model

- Different idea
  - The RDFS class and property system is similar to object-oriented programming languages
  - However:
    - Instead of defining a class in terms of the properties its instances may have, **we describe properties in terms of the classes of resource to which they apply**
      - I.e. types of objects in case of property ranges and types of subjects in case of property domains

Class Model

- **Definition of classes**
  - OOP
    - Class is a collection of explicitly specified attributes
    - Attributes of different classes are different attributes
      - Despite they may have the same name
    - `class` *ClassName* `{ … }`
  - RDFS
    - **Properties are not directly bound to classes**
      - We may only associate them using domains and ranges
    - *ClassName* `rdfs:type rdfs:Class .`

Class Model

- **Scope of properties**
  - OOP
    - Attributes are directly associated with classes
    - `class` *ClassName* `{` … *type AttributeName*`;` … `}`
  - RDFS
    - **Global scope of property descriptions**
    - One property may be used across different classes
    - We also cannot define locally-different property ranges depending on types of objects
    - *PropertyName* `rdf:type rdfs:Property.`

Class Model

- **Structure of instances**
  - OOP
    - Instances must have exactly the required structure
      - All defined attributes and also of the given types
      - No additional unknown attributes are allowed
    - *ClassName Instance* `=` `new` *ClassName*`();`
  - RDFS
    - Schema descriptions are not necessarily prescriptive
    - *Instance* `rdf:type` *ClassName* `.`

Class Model

- **Conclusion** using an example
  - OOP
    - Class *Book* with an attribute *author* of type *Person*
  - RDFS
    - Property *author* with domain *Book* and range *Person*
- And there are also other tricky aspects…
  - Class can be an instance of itself
    - `ClassName rdf:type ClassName .`

RDFS Vocabulary

- Other schema information
  - Property **rdfs:comment**
    - Human-readable description of a resource
  - Property **rdfs:label**
    - Human-readable version of a resource's name
  - Property **rdfs:seeAlso**
    - Resource that might provide additional information about the subject resource

Schema Interpretation

- Different strategies
  - **Prescription of constraints**
    - We require that data follow schema descriptions
      - E.g. that instances of a given class have defined all properties introduced by descriptions of their domains
    - This conformance can be tested and errors reported
  - **Inference of information**
    - We use descriptions and data to infer new data
      - E.g. if we know that a given property has range *C*, we can infer that a particular value of this property is an instance of class *C*

Assignment 3.1

- Extend the RDF graph from Assignment 2.1 by an appropriate RDFS schema…
  - Use all introduced constructs
    - Classes and properties, type assignments, class and property relations, ranges and domains, …

OWL

- **Web Ontology Language**
  - Specifications
    - http://www.w3.org/TR/2009/ REC-owl2-overview-20091027/
    - http://www.w3.org/TR/2009/ REC-owl2-primer-20091027/
    - http://www.w3.org/TR/2009/ REC-owl2-quick-reference-20091027/
    - …
  - Ontologies
    - Classes, properties and individuals
    - Class restrictions, property cardinalities
    - …

OWL

- **Ontology**
  - Ontology is a set of precise descriptive statements about some part of the world…
    - This part is usually referred as the domain of interest
    - Descriptions…
      - … prevent misunderstandings in human communication
      - … ensure uniform and predictable software behavior
      - … are presented in a formal way with precise semantics
  - Ontology is nothing else then a vocabulary
    - But we have more mechanisms for data inference

Classes

- **Class model**
  - Analogy to RDFS
    - Individuals = instances of a class
  - Terms
    - Classes **owl:Class**, **owl:Thing**, **owl:Nothing**
    - Properties **owl:equivalentClass**, **owl:intersectionOf**, **owl:unionOf**, **owl:ComplementOf**, **owl:oneOf**
  - Class hierarchy
    - Class **owl:Thing** – class of everything
    - Class **owl:Nothing** – empty class

Classes

- Class constructors
  - … how can we define new classes?
  - **Definition**
    - *ClassName* `rdf:type owl:Class .`
  - **Hierarchy**
    - Property **rdfs:subClassOf**
    - *OneClass* `rdfs:subClassOf` *AnotherClass* `.`
  - **Equivalency**
    - Property **owl:equivalentClass**
    - *OneClass* `owl:equivalentClass` *AnotherClass* `.`

Classes

- Class constructors
  - **Intersection** using property **owl:intersectionOf**
    - Class of individuals which are instances of both classes
    - *MyClass* `owl:equivalentClass [`
      `rdf:type owl:Class ;`
      `owl:intersectionOf (`*C1 C2* `…)`
      `] .`
  - **Union** using property **owl:unionOf**
  - **Complement** using property **owl:ComplementOf**
  - **Enumeration** using property **owl:oneOf**

Classes

- Object property restrictions
  - Class **owl:Restriction**
  - **Universal** using property **owl:allValuesFrom**
    - *MyRestriction* `rdf:type owl:Restriction .`
      MyRestriction `owl:onProperty` *PropertyName* `.`
      MyRestriction `owl:allValuesFrom` *ClassName* `.`
  - **Existential** using property **owl:someValuesFrom**
  - **Individual value** using property **owl:hasValue**
  - **Cardinalities** using **owl:cardinality**, **owl:minCardinality**, **owl:maxCardinality**, **owl:qualifiedCardinality**, …
    - … and using owl:onClass in case of qualified forms

Properties

- Property types
  - **Object property**
    - Relates individuals of two classes
    - Class **owl:ObjectProperty**
  - **Datatype property**
    - Relates individuals of classes to literal values
    - Class **owl:DatatypeProperty**

Properties

- Property constructors
  - … how can we define new properties?
  - **Definition**
  - **Hierarchy**
    - Property **rdfs:subPropertyOf**
  - **Equivalency**
    - Property **owl:equivalentProperty**
    - *OneProp* `owl:equivalentProperty` *AnotherProp* `.`

Properties

- Property constructors
  - Class **owl:ReflexiveProperty**
  - Class **owl:SymmetricProperty**
  - Class **owl:AsymmetricProperty**
  - Class **owl:TransitiveProperty**
  - Class **owl:FunctionalProperty**
  - Class **owl:InverseFunctionalProperty**

Individuals

- Comparison of individuals
  - **Equality of individuals**
    - Property **owl:sameAs**
      - *Individual1* `owl:sameAs` *Individual2* `.`
  - **Inequality of individuals**
    - Property **owl:differentFrom** for two individuals
      - *Individual1* `owl:differentFrom` *Individual2* `.`

OWL

- **Semantics**
  - There are two (nearly the same) approaches
    - **Model-theoretic** – direct semantics in **Description Logic**
      - OWL 2 DL
    - **RDF-based** – ontologies are viewed as RDF graphs
      - OWL 2 Full
  - However, OWL 2 is a very expressive language!
    - … computationally – it is difficult to implement it
    - … and for users – it is difficult to work with it
    - Therefore it is useful to introduced easier profiles…

Vocabularies

- Well-known vocabularies
  - FOAF = **Friend of a friend**
  - DC = **Dublin Core**
  - SKOS = **Simple Knowledge Organization System**

Vocabularies

- **FOAF**
  - FOAF = **Friend of a friend**
  - Linking people and information using the Web
  - Specification
    - http://xmlns.com/foaf/spec/
  - Vocabulary
    - Prefix **foaf:**
    - Classes **Agent**, **Person**, **Group**, **Project**, …
    - Properties **name**, **knows**, **member**, **homepage**, …

Vocabularies

- **Dublin Core**
  - Metadata for generic description of resources
  - Specification
    - http://dublincore.org/documents/
  - Vocabulary
    - Prefixes Dublin Core Terms **dct:**, …
    - Properties **creator**, **created**, **abstract**, **description**, **subject**, **valid**, **references**, **replaces**, **hasVersion**, …
    - …

Vocabularies

- **SKOS**
  - SKOS = **Simple Knowledge Organization System**
  - Sharing and linking knowledge systems
    - Thesauri, taxonomies, classification schemes, …
  - Specification
    - http://www.w3.org/TR/2009/REC-skos-reference-20090818/

Conclusion

- **RDFS**
  - Classes **rdfs:Resource**, **rdfs:Class**, **rdfs:Property**
  - Properties **rdfs:subClassOf**, **rdfs:subPropertyOf**
  - Properties **rdfs:range**, **rdfs:domain**
  - Properties **rdfs:comment**, **rdfs:label**, **rdfs:seeAlso**
  - …

Conclusion

- **OWL**
  - Classes
    - Classes **owl:Thing**, **owl:Nothing**, **owl:Class**
    - Properties **owl:equivalentClass**, **owl:intersectionOf**, **owl:unionOf**, **owl:ComplementOf**, **owl:oneOf**
    - Properties **owl:allValuesFrom**, **owl:someValuesFrom**, **owl:hasValue**, **owl:cardinality**, **owl:minCardinality**, **owl:maxCardinality**, **owl:qualifiedCardinality**, …
    - Properties **owl:onProperty**, **owl:onClass**

Conclusion

- **OWL**
  - Properties
    - Classes **owl:ObjectProperty**, **owl:DatatypeProperty**
    - Property **owl:equivalentProperty**
    - Classes **owl:ReflexiveProperty**, **owl:SymmetricProperty**, **owl:AsymmetricProperty**, **owl:TransitiveProperty**, **owl:FunctionalProperty**, **owl:InverseFunctionalProperty**
  - Individuals
    - Properties **owl:sameAs**, **owl:differentFrom**
  - **…**