Chapter 3
Describing Web Resources in RDF

Grigoris Antoniou
Frank van Harmelen

Lecture Outline

1. Basic Ideas of RDF
2. XML-based Syntax of RDF
3. Basic Concepts of RDF Schema
4. The Language of RDF Schema
5. The Namespaces of RDF and RDF Schema
6. Axiomatic Semantics for RDF and RDFS
7. Direct Semantics based on Inference Rules
8. Querying of RDF/RDFS Documents using RQL

Drawbacks of XML

- XML is a universal metalanguage for defining markup
- It provides a uniform framework for interchange of data and metadata between applications
- However, XML does not provide any means of talking about the semantics (meaning) of data
- E.g., there is no intended meaning associated with the nesting of tags
  - It is up to each application to interpret the nesting.

Nesting of Tags in XML

David Billington is a lecturer of Discrete Maths

```
<course name="Discrete Maths">
	<lecturer>David Billington</lecturer>
</course>
<lecturer name="David Billington">
	<teaches>Discrete Maths</teaches>
</lecturer>
```

Opposite nesting, same information!

Basic Ideas of RDF

- Basic building block: **object-attribute-value** triple
  - It is called a statement
  - Sentence about Billington is such a statement
- RDF has been given a syntax in XML
  - This syntax inherits the benefits of XML
  - Other syntactic representations of RDF possible

Basic Ideas of RDF (2)

- The fundamental concepts of RDF are:
  - resources
  - properties
  - statements

Resources

- We can think of a resource as an object, a “thing” we want to talk about
  - E.g. authors, books, publishers, places, people, hotels
- Every resource has a URI, a Universal Resource Identifier
- A URI can be
  - a URL (Web address) or
  - some other kind of unique identifier

Properties

- Properties are a special kind of resources
- They describe relations between resources
  - E.g. “written by”, “age”, “title”, etc.
- Properties are also identified by URIs
- Advantages of using URIs:
  - A global, worldwide, unique naming scheme
  - Reduces the homonym problem of distributed data representation

Statements

- Statements assert the properties of resources
- A statement is an object-attribute-value triple
  - It consists of a resource, a property, and a value
- Values can be resources or **literals**
  - Literals are atomic values (strings)

Three Views of a Statement

- A triple
- A piece of a graph
- A piece of XML code

Thus an RDF document can be viewed as:

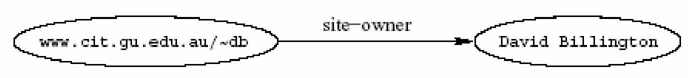
- A set of triples
- A graph (semantic net)
- An XML document

Statements as Triples

(http://www.cit.gu.edu.au/~db, http://www.mydomain.org/site-owner, "David Billington")

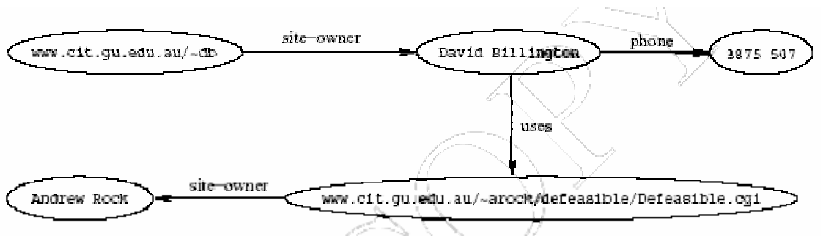
- The triple (x,P,y) can be considered as a logical formula P(x,y)
  - Binary predicate P relates object x to object y
  - RDF offers only binary predicates (properties)

XML Vocabularies



- A directed graph with labeled nodes and arcs
  - **from** the resource (the **subject** of the statement)
  - **to** the value (the **object** of the statement)
- Known in AI as a *semantic net*
- The value of a statement may be a resource
  - It may be linked to other resources

A Set of Triples as a Semantic Net



Statements in XML Syntax

- Graphs are a powerful tool for human understanding **but**
- The Semantic Web vision requires machine-accessible and machine-processable representations
- There is a 3rd representation based on XML
  - But XML is not a part of the RDF data model
  - E.g. serialisation of XML is irrelevant for RDF

Statements in XML (2)

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:mydomain="http://www.mydomain.org/my-rdf-ns">

        <rdf:Description
                rdf:about="http://www.cit.gu.edu.au/~db">
                        <mydomain:site-owner>
                        David Billington
                        </mydomain:site-owner>
        </rdf:Description>
</rdf:RDF>
```


Statements in XML (3)

- An RDF document is represented by an XML element with the tag **rdf:RDF**
- The content of this element is a number of descriptions, which use **rdf:Description** tags.
- Every description makes a statement about a resource, identified in 3 ways:
  - an **about** attribute, referencing an existing resource
  - an **ID** attribute, creating a new resource
  - without a name, creating an anonymous resource

Statements in XML (4)

- The **rdf:Description** element makes a statement about the resource **http://www.cit.gu.edu.au/~db**
- Within the description
  - the property is used as a tag
  - the content is the value of the property

Reification

- In RDF it is possible to make statements about statements
  - **Grigoris believes that David Billington is the creator of http://www.cit.gu.edu.au/~db**
- Such statements can be used to describe belief or trust in other statements
- The solution is to assign a unique identifier to each statement
  - It can be used to refer to the statement

Reification (2)

- Introduce an auxiliary object (e.g. **belief1**)
- relate it to each of the 3 parts of the original statement through the properties **subject**, **predicate** and **object**
- In the preceding example
  - subject of **belief1** is **David Billington**
  - predicate of **belief1** is creator
  - object of **belief1** is **http://www.cit.gu.edu.au/~db**

Data Types

- Data types are used in programming languages to allow interpretation
- In RDF, typed literals are used, if necessary

(“David Billington”,
http://www.mydomain.org/age,
“27”^^http://www.w3.org/2001/XMLSchema#integer)

Data Types (2)

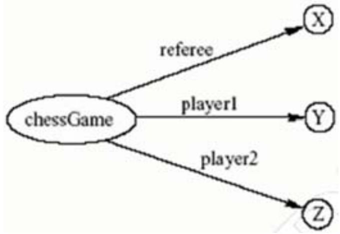
- ^^-notation indicates the type of a literal
- In practice, the most widely used data typing scheme will be the one by XML Schema
  - But the use of **any** externally defined data typing scheme is allowed in RDF documents
- XML Schema predefines a large range of data types
  - E.g. Booleans, integers, floating-point numbers, times, dates, etc.

A Critical View of RDF: Binary Predicates

- RDF uses only binary properties
  - This is a restriction because often we use predicates with more than 2 arguments
  - But binary predicates can simulate these
- Example: **referee(X,Y,Z)**
  - **X** is the referee in a chess game between players **Y** and **Z**

A Critical View of RDF: Binary Predicates (2)

- We introduce:
  - a new auxiliary resource **chessGame**
  - the binary predicates **ref**, **player1**, and **player2**
- We can represent **referee(X,Y,Z)** as:



A Critical View of RDF: Properties

- Properties are special kinds of resources
  - Properties can be used as the object in an object-attribute-value triple (statement)
  - They are defined independent of resources
- This possibility offers flexibility
- But it is unusual for modelling languages and OO programming languages
- It can be confusing for modellers

A Critical View of RDF: Reification

- The reification mechanism is quite powerful
- It appears misplaced in a simple language like RDF
- Making statements about statements introduces a level of complexity that is not necessary for a basic layer of the Semantic Web
- Instead, it would have appeared more natural to include it in more powerful layers, which provide richer representational capabilities

A Critical View of RDF: Summary

- RDF has its idiosyncrasies and is not an optimal modeling language **but**
- It is already a de facto standard
- It has sufficient expressive power
  - At least as for more layers to build on top
- Using RDF offers the benefit that information maps unambiguously to a model

Lecture Outline

1. Basic Ideas of RDF
2. XML-based Syntax of RDF
3. Basic Concepts of RDF Schema
4. The Language of RDF Schema
5. The Namespaces of RDF and RDF Schema
6. Axiomatic Semantics for RDF and RDFS
7. Direct Semantics based on Inference Rules
8. Querying of RDF/RDFS Documents using RQL

XML-Based Syntax of RDF

- An RDF document consists of an **rdf:RDF** element
  - The content of that element is a number of descriptions
- A namespace mechanism is used
  - Disambiguation
  - Namespaces are expected to be RDF documents defining resources that can be reused
  - Large, distributed collections of knowledge

Example of University Courses

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XLMSchema#"
   xmlns:uni="http://www.mydomain.org/uni-ns">

   <rdf:Description rdf:about="949318">
         <uni:name>David Billington</uni:name>
         <uni:title>Associate Professor</uni:title>
         <uni:age rdf:datatype="&xsd:integer">27<uni:age>
   </rdf:Description>
```


Example of University Courses (2)

```
   <rdf:Description rdf:about="CIT1111">
         <uni:courseName>Discrete Maths</uni:courseName>
         <uni:isTaughtBy>David Billington</uni:isTaughtBy>
   </rdf:Description>

   <rdf:Description rdf:about="CIT2112">
         <uni:courseName>Programming III</uni:courseName>
         <uni:isTaughtBy>Michael Maher</uni:isTaughtBy>
   </rdf:Description>

</rdf:RDF>
```


rdf:about versus rdf:ID

- An element **rdf:Description** has
  - an **rdf:about** attribute indicating that the resource has been "defined" elsewhere
  - An **rdf:ID** attribute indicating that the resource is defined
- Formally, there is no such thing as "defining" an object in one place and referring to it elsewhere
  - Sometimes is useful (for human readability) to have a defining location, while other locations state "**additional**" properties

Property Elements

- Content of **rdf:Description** elements

```
<rdf:Description rdf:about="CIT3116">
      <uni:courseName>Knowledge
      Representation</uni:courseName>
      <uni:isTaughtBy>Grigoris Antoniou</uni:isTaughtBy>
</rdf:Description>
```

- **uni:courseName** and **uni:isTaughtBy** define two property-value pairs for **CIT3116** (two RDF statements)
  - read conjunctively

Data Types

- The attribute rdf:datatype="&xsd:integer" is used to indicate the data type of the value of the age property

```
<rdf:Description rdf:about="949318">
   <uni:name>David Billington</uni:name>
   <uni:title>Associate Professor</uni:title>
   <uni:age rdf:datatype="&xsd:integer">27<uni:age>
</rdf:Description>
```


Data Types (2)

- The **age** property has been defined to have **"&xsd:integer"** as its *range*
  - It is still required to indicate the type of the value of this property each time it is used
  - This is to ensure that an RDF processor can assign the correct type of the property value even if it has not "seen" the corresponding RDF Schema definition before
  - This scenario is quite likely to occur in the unrestricted WWW

The rdf:resource Attribute

- The relationships between courses and lecturers (in the example) were not formally defined but existed implicitly through the use of the same name
- The use of the same name may just be a coincidence for a machine
- We can denote that two entities are the same using the **rdf:resource** attribute

The rdf:resource Attribute (2)

```
<rdf:Description rdf:about="CIT1111">
      <uni:courseName>Discrete
   Mathematics</uni:courseName>
      <uni:isTaughtBy rdf:resource="949318"/>
</rdf:Description>
<rdf:Description rdf:about="949318">
      <uni:name>David Billington</uni:name>
      <uni:title>Associate Professor</uni:title>
</rdf:Description>
```

Referencing Externally Defined Resources

- E.g., to refer the externally defined resource CIT1111: **http://www.mydomain.org/uni-ns#CIT1111** as the value of **rdf:about**
- **www.mydomain.org/uni-ns** is the URI where the definition of CIT1111 is found
- A description with an **ID** defines a fragment URI, which can be used to reference the defined description

Nested Descriptions: Example

```
<rdf:Description rdf:about="CIT1111">
    <uni:courseName>Discrete
  Maths</uni:courseName>
    <uni:isTaughtBy>
        <rdf:Description rdf:ID="949318">
            <uni:name>David Billington</uni:name>
            <uni:title>Associate Professor</uni:title>
        </rdf:Description>
    </uni:isTaughtBy>
</rdf:Description>
```


Nested Descriptions

- Descriptions may be defined within other descriptions
- Other courses, such as **CIT3112**, can still refer to the new resource with ID **949318**
- Although a description may be defined within another description, its scope is global

Introducing some Structure to RDF Documents using the rdf:type Element

```
<rdf:Description rdf:ID="CIT1111">
        <rdf:type rdf:resource="http://www.mydomain.org/uni-
                ns#course"/>
        <uni:courseName>Discrete Maths</uni:courseName>
        <uni:isTaughtBy rdf:resource="#949318"/>
</rdf:Description>

<rdf:Description rdf:ID="949318">
        <rdf:type rdf:resource="http://www.mydomain.org/uni-
                ns#lecturer"/>
        <uni:name>David Billington</uni:name>
        <uni:title>Associate Professor</uni:title>
</rdf:Description>
```

Abbreviated Syntax

- Simplification rules:
  1. Childless property elements within description elements may be replaced by XML attributes
  2. For description elements with a typing element we can use the name specified in the **rdf:type** element instead of **rdf:Description**
- These rules create syntactic variations of the same RDF statement
  - They are equivalent according to the RDF data model, although they have different XML syntax

Abbreviated Syntax: Example

```
<rdf:Description rdf:ID="CIT1111">
        <rdf:type rdf:resource="http://www.mydomain.org/uni-
                ns#course"/>
        <uni:courseName>Discrete Maths</uni:courseName>
        <uni:isTaughtBy rdf:resource="#949318"/>
</rdf:Description>
```


Application of First Simplification Rule

```
<rdf:Description rdf:ID="CIT1111"
            uni:courseName="Discrete Maths">
    <rdf:type rdf:resource="http://www.mydomain.org/uni-
            ns#course"/>
    <uni:isTaughtBy rdf:resource="#949318"/>
</rdf:Description>
```


Application of 2nd Simplification Rule

```
<uni:course rdf:ID="CIT1111"
            uni:courseName="Discrete Maths">
    <uni:isTaughtBy rdf:resource="#949318"/>
</uni:course>
```

Container Elements

- Collect a number of resources or attributes about which we want to make statements as a whole
- E.g., we may wish to talk about the courses given by a particular lecturer
- The content of container elements are named **rdf:_1**, **rdf:_2**, etc.
  - Alternatively **rdf:li**

Three Types of Container Elements

- **rdf:Bag** an unordered container, allowing multiple occurrences
  - E.g. members of the faculty board, documents in a folder
- **rdf:Seq** an ordered container, which may contain multiple occurrences
  - E.g. modules of a course, items on an agenda, an alphabetized list of staff members (order is imposed)
- **rdf:Alt** a set of alternatives
  - E.g. the document home and mirrors, translations of a document in various languages

Example for a Bag

```
<uni:lecturer rdf:ID="949352" uni:name="Grigoris
   Antoniou"
                 uni:title="Professor">
   <uni:coursesTaught>
       <rdf:Bag>
              <rdf:_1 rdf:resource="#CIT1112"/>
              <rdf:_2 rdf:resource="#CIT3116"/>
       </rdf:Bag>
   </uni:coursesTaught>
</uni:lecturer>
```


Example for Alternative

```
<uni:course rdf:ID="CIT1111"
       uni:courseName="Discrete Mathematics">
    <uni:lecturer>
       <rdf:Alt>
              <rdf:li rdf:resource="#949352"/>
              <rdf:li rdf:resource="#949318"/>
       </rdf:Alt>
    </uni:lecturer>
</uni:course>
```

Rdf:ID Attribute for Container Elements

```
<uni:lecturer rdf:ID="949318"
              uni:name="David Billington">
    <uni:coursesTaught>
       <rdf:Bag rdf:ID="DBcourses">
              <rdf:_1 rdf:resource="#CIT1111"/>
              <rdf:_2 rdf:resource="#CIT3112"/>
       </rdf:Bag>
    </uni:coursesTaught>
</uni:lecturer>
```


RDF Collections

- A limitation of these containers is that there is no way to **close** them
  - “these are **all** the members of the container”
- RDF provides support for describing groups containing **only** the specified members, in the form of RDF collections
  - **list** structure in the RDF graph
  - constructed using a predefined collection vocabulary: **rdf:List**, **rdf:first**, **rdf:rest** and **rdf:nil**

RDF Collections (2)

- Shorthand syntax:
  - **"Collection"** value for the **rdf:parseType** attribute:

```
<rdf:Description rdf:about="#CIT2112">
       <uni:isTaughtBy rdf:parseType="Collection">
              <rdf:Description rdf:about="#949111"/>
              <rdf:Description rdf:about="#949352"/>
              <rdf:Description rdf:about="#949318"/>
       </uni:isTaughtBy>
</rdf:Description>
```


Reification

- Sometimes we wish to make statements about other statements
- We must be able to refer to a statement using an identifier
- RDF allows such reference through a reification mechanism which turns a statement into a resource

Reification Example

```
<rdf:Description rdf:about="#949352">
        <uni:name>Grigoris Antoniou</uni:name>
</rdf:Description>
```

- reifies as

```
<rdf:Statement rdf:ID="StatementAbout949352">
        <rdf:subject rdf:resource="#949352"/>
        <rdf:predicate rdf:resource="http://www.mydomain.org/
                uni-ns#name"/>
        <rdf:object>Grigoris Antoniou</rdf:object>
</rdf:Statement>
```


Reification (2)

- **rdf:subject, rdf:predicate** and **rdf:object** allow us to access the parts of a statement
- The **ID** of the statement can be used to refer to it, as can be done for any description
- We write an **rdf:Description** if we don't want to talk about a statement further
- We write an **rdf:Statement** if we wish to refer to a statement

Lecture Outline

1. Basic Ideas of RDF
2. XML-based Syntax of RDF
3. Basic Concepts of RDF Schema
4. The Language of RDF Schema
5. The Namespaces of RDF and RDF Schema
6. Axiomatic Semantics for RDF and RDFS
7. Direct Semantics based on Inference Rules
8. Querying of RDF/RDFS Documents using RQL

Basic Ideas of RDF Schema

- RDF is a universal language that lets users describe resources in their own vocabularies
  - RDF does not assume, nor does it define semantics of any particular application domain
- The user can do so in RDF Schema using:
  - Classes and Properties
  - Class Hierarchies and Inheritance
  - Property Hierarchies

Classes and their Instances

- We must distinguish between
  - Concrete "things" (individual objects) in the domain: Discrete Maths, David Billington etc.
  - Sets of individuals sharing properties called classes: lecturers, students, courses etc.
- Individual objects that belong to a class are referred to as instances of that class
- The relationship between instances and classes in RDF is through **rdf:type**

Why Classes are Useful

- Impose restrictions on what can be stated in an RDF document using the schema
  - As in programming languages
  - E.g. A+1, where A is an array
  - Disallow nonsense from being stated

Nonsensical Statements disallowed through the Use of Classes

- Discrete Maths is taught by Concrete Maths
  - We want courses to be taught by lecturers only
  - Restriction on values of the property "is taught by" (**range restriction**)
- Room MZH5760 is taught by David Billington
  - Only courses can be taught
  - This imposes a restriction on the objects to which the property can be applied (**domain restriction**)

Class Hierarchies

- Classes can be organised in hierarchies
  - A is a subclass of B if every instance of A is also an instance of B
  - Then B is a superclass of A
- A subclass graph need not be a tree
- A class may have multiple superclasses

Class Hierarchy Example

Inheritance in Class Hierarchies

- Range restriction: **Courses must be taught by academic staff members only**
- Michael Maher is a professor
- He inherits the ability to teach from the class of academic staff members
- This is done in RDF Schema by fixing the semantics of “is a subclass of”
  - It is not up to an application (RDF processing software) to interpret “is a subclass of

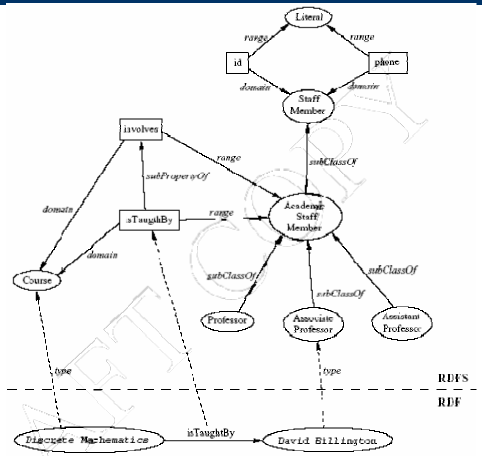
Property Hierarchies

- Hierarchical relationships for properties
  - E.g., “is taught by” is a subproperty of “involves”
  - If a course C is taught by an academic staff member A, then C also involves A
- The converse is not necessarily true
  - E.g., A may be the teacher of the course C, or
  - a tutor who marks student homework but does not teach C
- P is a subproperty of Q, if Q(x,y) is true whenever P(x,y) is true

RDF Layer vs RDF Schema Layer

- Discrete Mathematics is taught by David Billington
- The schema is itself written in a formal language, RDF Schema, that can express its ingredients:
  - subClassOf, Class, Property, subPropertyOf, Resource, etc.

RDF Layer vs RDF Schema Layer (2)



Lecture Outline

1. Basic Ideas of RDF
2. XML-based Syntax of RDF
3. Basic Concepts of RDF Schema
4. The Language of RDF Schema
5. The Namespaces of RDF and RDF Schema
6. Axiomatic Semantics for RDF and RDFS
7. Direct Semantics based on Inference Rules
8. Querying of RDF/RDFS Documents using RQL

RDF Schema in RDF

- The modeling primitives of RDF Schema are defined using resources and properties (RDF itself is used!)
- To declare that “lecturer” is a subclass of “academic staff member”
  - Define resources **lecturer**, **academicStaffMember**, and **subClassOf**
  - define property **subClassOf**
  - Write triple (**subClassOf,lecturer,academicStaffMember**)
- We use the XML-based syntax of RDF

Core Classes

- **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:Statement**, the class of all reified statements

Core Properties

- **rdf:type**, which relates a resource to its class
  - The resource is declared to be an instance of that class
- **rdfs:subClassOf**, which relates a class to one of its superclasses
  - All instances of a class are instances of its superclass
- **rdfs:subPropertyOf**, relates a property to one of its superproperties

Core Properties (2)

- **rdfs:domain**, which specifies the domain of a property P
  - The class of those resources that may appear as subjects in a triple with predicate P
  - If the domain is not specified, then any resource can be the subject
- **rdfs:range**, which specifies the range of a property P
  - The class of those resources that may appear as values in a triple with predicate P

Examples

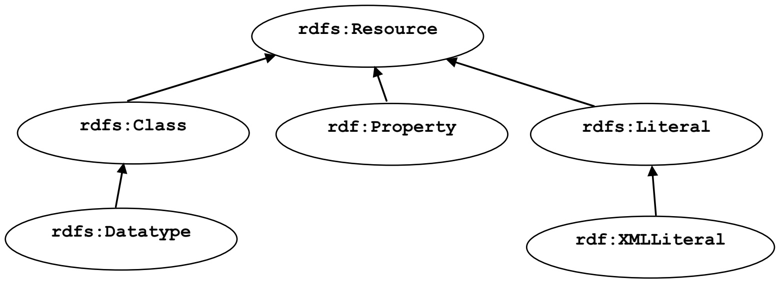
```
<rdfs:Class rdf:about="#lecturer">
   <rdfs:subClassOf rdf:resource="#staffMember"/>
</rdfs:Class>

<rdf:Property rdf:ID="phone">
        <rdfs:domain rdf:resource="#staffMember"/>
        <rdfs:range rdf:resource="http://www.w3.org/
                2000/01/rdf-schema#Literal"/>
</rdf:Property>
```

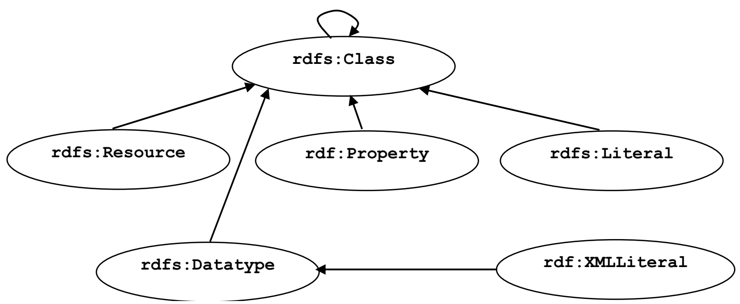

Relationships Between Core Classes and Properties

- **rdfs:subClassOf** and **rdfs:subPropertyOf** are transitive, by definition
- **rdfs:Class** is a subclass of **rdfs:Resource**
  - Because every class is a resource
- **rdfs:Resource** is an instance of **rdfs:Class**
  - **rdfs:Resource** is the class of all resources, so it is a class
- Every class is an instance of **rdfs:Class**
  - For the same reason

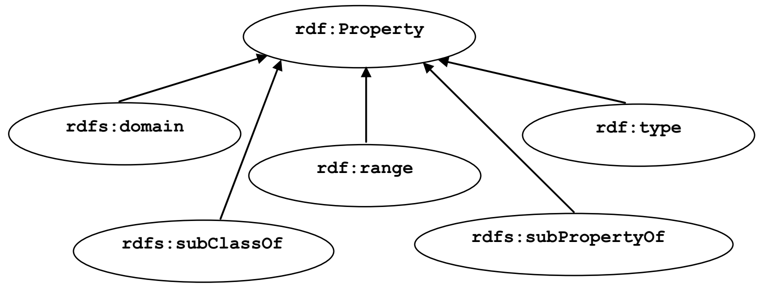
Subclass Hierarchy of Some Modeling Primitives of RDF Schema



Instance Relationships of Some Modeling Primitives of RDFS



Instance Relationships of Some Core Properties of RDF and RDF Schema



Reification and Containers

- **rdf:subject**, relates a reified statement to its subject
- **rdf:predicate**, relates a reified statement to its predicate
- **rdf:object**, relates a reified statement to its object
- **rdf:Bag**, the class of bags
- **rdf:Seq**, the class of sequences
- **rdf:Alt**, the class of alternatives
- **rdfs:Container**, which is a superclass of all container classes, including the three above

Utility Properties

- **rdfs:seeAlso** relates a resource to another resource that explains it
- **rdfs:isDefinedBy** is a subproperty of **rdfs:seeAlso** and relates a resource to the place where its definition, typically an RDF schema, is found
- **rfds:comment**. Comments, typically longer text, can be associated with a resource
- **rdfs:label**. A human-friendly label (name) is associated with a resource

Example: A University

```
<rdfs:Class rdf:ID="lecturer">
    <rdfs:comment>
        The class of lecturers. All lecturers are
        academic staff members.
    </rdfs:comment>
    <rdfs:subClassOf
    rdf:resource="#academicStaffMember"/>
</rdfs:Class>
```


Example: A University (2)

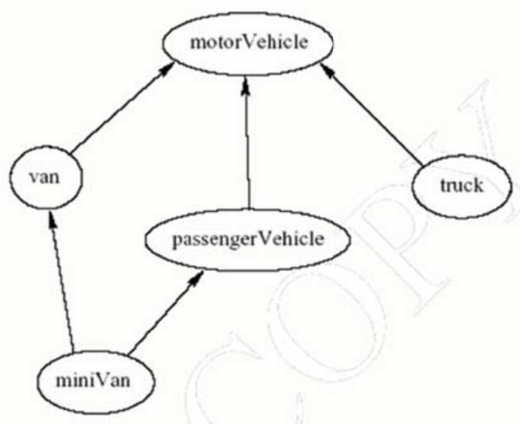
```
<rdfs:Class rdf:ID="course">
    <rdfs:comment>The class of courses</rdfs:comment>
</rdfs:Class>

<rdf:Property rdf:ID="isTaughtBy">
    <rdfs:comment>
        Inherits its domain ("course") and range ("lecturer")
        from its superproperty "involves"
    </rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="#involves"/>
</rdf:Property>
```


Example: A University (3)

```
<rdf:Property rdf:ID="phone">
    <rdfs:comment>
        It is a property of staff members
        and takes literals as values.
    </rdfs:comment>
    <rdfs:domain rdf:resource="#staffMember"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
        schema#Literal"/>
</rdf:Property>
```

Class Hierarchy for the Motor Vehicles Example



Lecture Outline

1. Basic Ideas of RDF
2. XML-based Syntax of RDF
3. Basic Concepts of RDF Schema
4. The Language of RDF Schema
5. The Namespaces of RDF and RDF Schema
6. Axiomatic Semantics for RDF and RDFS
7. Direct Semantics based on Inference Rules
8. Querying of RDF/RDFS Documents using RQL

The Namespace of RDF

```
<rdfs:Class rdf:ID="Statement"
        rdfs:comment="The class of triples consisting of a
        predicate, a subject and an object (that is, a
        reified statement)"/>

<rdfs:Class rdf:ID="Property"
        rdfs:comment="The class of properties"/>

<rdfs:Class rdf:ID="Bag"
        rdfs:comment="The class of unordered collections"/>
```


The Namespace of RDF (2)

```
<rdf:Property rdf:ID="predicate"
        rdfs:comment="Identifies the property of a
        statementin reified form"/>
        <rdfs:domain rdf:resource="#Statement"/>
        <rdfs:range rdf:resource="#Property"/>
</rdf:Property>
```

The Namespace of RDF Schema

```
<rdfs:Class rdf:ID="Resource"
        rdfs:comment="The most general class"/>

<rdfs:Class rdf:ID="Class"
        rdfs:comment="The concept of classes.
                All classes are resources"/>
        <rdfs:subClassOf rdf:resource="#Resource"/>
</rdfs:Class>
```


The Namespace of RDF Schema (2)

```
<rdf:Property rdf:ID="subPropertyOf">
   <rdfs:domain rdf:resource="http://www.w3.org/
                1999/02/22-rdf-syntax-ns#Property"/>
   <rdfs:range rdf:resource="http://www.w3.org/
                1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Property>

<rdf:Property rdf:ID="subClassOf">
   <rdfs:domain rdf:resource="#Class"/>
   <rdfs:range rdf:resource="#Class"/>
</rdf:Property>
```


Namespace versus Semantics

- Consider **rdfs:subClassOf**
  - The namespace specifies only that it applies to classes and has a class as a value
  - The meaning of being a subclass not expressed
- The meaning cannot be expressed in RDF
  - If it could RDF Schema would be unnecessary
- External definition of semantics required
  - Respected by RDF/RDFS processing software

Lecture Outline

1. Basic Ideas of RDF
2. XML-based Syntax of RDF
3. Basic Concepts of RDF Schema
4. The Language of RDF Schema
5. The Namespaces of RDF and RDF Schema
6. Axiomatic Semantics for RDF and RDFS
7. Direct Semantics based on Inference Rules
8. Querying of RDF/RDFS Documents using RQL

Axiomatic Semantics

- We formalize the meaning of the modeling primitives of RDF and RDF Schema
- By translating into first-order logic
- We make the semantics unambiguous and machine accessible
- We provide a basis for reasoning support by automated reasoners manipulating logical formulas

The Approach

- All language primitives in RDF and RDF Schema are represented by constants:
  - **Resource**, **Class**, **Property**, **subClassOf**, etc.
- A few predefined predicates are used as a foundation for expressing relationships between the constants
- We use predicate logic with equality
- Variable names begin with ?
- All axioms are implicitly universally quantified

An Auxiliary Axiomatisation of Lists

- Function symbols:
  - **nil** (empty list)
  - **cons(x,l)** (adds an element to the front of the list)
  - **first(l)** (returns the first element)
  - **rest(l)** (returns the rest of the list)
- Predicate symbols:
  - **item(x,l)** (tests if an element occurs in the list)
  - **list(l)** (tests whether l is a list)
- Lists are used to represent containers in RDF

Basic Predicates

- **PropVal(P,R,V)**
  - A predicate with 3 arguments, which is used to represent an RDF statement with resource **R**, property **P** and value **V**
  - An RDF statement (triple) **(P,R,V)** is represented as **PropVal(P,R,V)**.
- **Type(R,T)**
  - Short for **PropVal(type,R,T)**
  - Specifies that the resource **R** has the type **T**
- **Type(?r,?t) ↔ PropVal(type,?r,?t)**

RDF Classes

- Constants: **Class**, **Resource**, **Property**, **Literal**
  - All classes are instances of **Class**

Type(Class,Class)
Type(Resource,Class)
Type(Property,Class)
Type(Literal,Class)

RDF Classes (2)

- **Resource** is the most general class: every class and every property is a resource

Type(?p,Property) → Type(?p,Resource)
Type(?c,Class) → Type(?c,Resource)

- The predicate in an RDF statement must be a property
- **PropVal(?p,?r,?v) → Type(?p,Property)**

The type Property

- **type** is a property

PropVal(type,type,Property)

- **type** can be applied to resources (domain) and has a class as its value (range)

Type(?r,?c) → (Type(?r,Resource) ∧ Type(?c,Class))

The Auxiliary FuncProp Property

- **P** is a functional property if, and only if,
  - it is a property, and
  - there are no **x**, **y1** and **y2** with **P(x,y1)**, **P(x,y2)** and **y1≠y2**

Type(?p, FuncProp) ↔
(Type(?p, Property) ∧
∀?r ∀?v1 ∀?v2
(PropVal(?p,?r,?v1) ∧
PropVal(?p,?r,?v2) → ?v1 = ?v2))

Containers

- Containers are lists:

Type(?c,Container) → list(?c)

- Containers are bags or sequences or alternatives:

Type(?c,Container) ↔

(Type(?c,Bag) ∨ Type(?c,Seq) ∨ Type(?c,Alt))

- Bags and sequences are disjoint:

¬(Type(?x,Bag) ∧ Type(?x,Seq))

Containers (2)

- For every natural number n > 0, there is the selector **_n,** which selects the n^{th} element of a container
- It is a functional property:

Type(_n,FuncProp)

- It applies to containers only:

PropVal(_n,?c,?o) → Type(?c,Container)

Subclass

- **subClassOf** is a property:

Type(subClassOf,Property)

- If a class C is a subclass of a class C', then all instances of C are also instances of C':

PropVal(subClassOf,?c,?c') ↔

(Type(?c,Class) ∧ Type(?c',Class) ∧

∀?x (Type(?x,?c) → Type(?x,?c')))

Subproperty

- P is a subproperty of P', if P'(x,y) is true whenever P(x,y) is true:

Type(subPropertyOf,Property)

PropVal(subPropertyOf,?p,?p') ↔

(Type(?p,Property) ∧ Type(?p',Property) ∧

∀?r ∀?v (PropVal(?p,?r,?v) →

PropVal(?p',?r,?v)))

Domain and Range

- If the domain of P is D, then for every P(x,y), x∈D

PropVal(domain,?p,?d) →
∀?x ∀?y (PropVal(?p,?x,?y) → Type(?x,?d))

- If the range of P is R, then for every P(x,y), y∈R

PropVal(range,?p,?r) →
∀?x ∀?y (PropVal(?p,?x,?y) → Type(?y,?r))

Lecture Outline

1. Basic Ideas of RDF
2. XML-based Syntax of RDF
3. Basic Concepts of RDF Schema
4. The Language of RDF Schema
5. The Namespaces of RDF and RDF Schema
6. Axiomatic Semantics for RDF and RDFS
7. Direct Semantics based on Inference Rules
8. Querying of RDF/RDFS Documents using RQL

Semantics based on Inference Rules

- Semantics in terms of RDF triples instead of restating RDF in terms of first-order logic
- … and sound and complete inference systems
- This inference system consists of inference rules of the form:

IF E contains certain triples
THEN add to E certain additional triples

- where **E** is an arbitrary set of RDF triples

Examples of Inference Rules

IF E contains the triple (?x,?p,?y)
THEN E also contains (?p,rdf:type,rdf:property)

IF E contains the triples (?u,rdfs:subClassOf,?v) and (?v,rdfs:subclassOf,?w)
THEN E also contains the triple (?u,rdfs:subClassOf,?w)

IF E contains the triples (?x,rdf:type,?u) and (?u,rdfs:subClassOf,?v)
THEN E also contains the triple (?x,rdf:type,?v)

Examples of Inference Rules (2)

- Any resource **?y** which appears as the value of a property **?p** can be inferred to be a member of the range of **?p**
  - This shows that range definitions in RDF Schema are not used to restrict the range of a property, but rather to infer the membership of the range

IF E contains the triples (?x,?p,?y) and
(?p,rdfs:range,?u)
THEN E also contains the triple (?y,rdf:type,?u)

Lecture Outline

1. Basic Ideas of RDF
2. XML-based Syntax of RDF
3. Basic Concepts of RDF Schema
4. The Language of RDF Schema
5. The Namespaces of RDF and RDF Schema
6. Axiomatic Semantics for RDF and RDFS
7. Direct Semantics based on Inference Rules
8. Querying of RDF/RDFS Documents using RQL

Why an RDF Query Language? Different XML Representations

- XML at a lower level of abstraction than RDF
- There are various ways of syntactically representing an RDF statement in XML
- Thus we would require several XQuery queries, e.g.
  - **//uni:lecturer/uni:title** if **uni:title** element
  - **//uni:lecturer/@uni:title** if **uni:title** attribute
  - Both XML representations equivalent!

Why an RDF Query Language? Understanding the Semantics

```
<uni:lecturer rdf:ID="949352">
   <uni:name>Grigoris Antoniou</uni:name>
</uni:lecturer>
<uni:professor rdf:ID="949318">
         <uni:name>David Billington</uni:name>
</uni:professor>
<rdfs:Class rdf:about="#professor">
   <rdfs:subClassOf rdf:resource="#lecturer"/>
</rdfs:Class>
```

- A query for the names of all lecturers should return both Grigoris Antoniou and David Billington

RQL Basic Queries

- The query **Class** retrieves all classes
- The query **Property** retrieves all properties
- To retrieve the instances of a class (e.g. **course**) we write

course

- If we do not wish to retrieve inherited instances, then we have to write

^course

RQL Basic Queries (2)

- The resources and values of triples with a specific property (e.g. **involves**) are retrieved using the query

involves

- The result includes all subproperties of **involves**
- If we do not want these additional results, then we have to write

^involves

Using select-from-where

- As in SQL
  - **select** specifies the number and order of retrieved data
  - **from** is used to navigate through the data model
  - **where** imposes constraints on possible solutions
- Retrieve all phone numbers of staff members:

select X,Y

from {X}phone{Y}

- Here **X** and **Y** are variables, and **{X}phone{Y}** represents a resource-property-value triple

Implicit Join

- Retrieve all lecturers and their phone numbers:

select X,Y
from lecturer{X}.phone{Y}

- Implicit join: We restrict the second query only to those triples, the resource of which is in the variable X
  - Here we restrict the domain of phone to lecturers
  - A dot **.** denotes the implicit join

Explicit Join

- Retrieve the name of all courses taught by the lecturer with ID 949352

```
select N
from course{X}.isTaughtBy{Y}, {C}name{N}
where Y="949352" and X=C
```


Querying the Schema

- Schema variables have a name with prefix **$** (for classes) or **@** (for properties)
- Retrieve all resources and values of triples with property **phone**, or any of its subproperties, and their classes

```
select X,$X,Y,$Y
from {X:$X}phone{Y:$Y}
```


Querying the Schema (2)

- The domain and range of a property can be retrieved as follows:

```
select domain(@P),range(@P)
from @P
where @P=phone
```


Summary

- RDF provides a foundation for representing and processing metadata
- RDF has a graph-based data model
- RDF has an XML-based syntax to support syntactic interoperability.
  - XML and RDF complement each other because RDF supports semantic interoperability
- RDF has a decentralized philosophy and allows incremental building of knowledge, and its sharing and reuse

Summary (2)

- RDF is domain-independent
- RDF Schema provides a mechanism for describing specific domains
- RDF Schema is a primitive ontology language
  - It offers certain modelling primitives with fixed meaning
- Key concepts of RDF Schema are class, subclass relations, property, subproperty relations, and domain and range restrictions
- There exist query languages for RDF and RDFS

Points for Discussion in Subsequent Chapters

- RDF Schema is quite primitive as a modelling language for the Web
- Many desirable modelling primitives are missing
- Therefore we need an ontology layer on top of RDF and RDF Schema