

Chapter 4

Web Ontology Language: OWL

Grigoris Antoniou
Frank van Harmelen

1

Chapter 4

A Semantic Web Primer

Lecture Outline

1. Basic Ideas of OWL
2. The OWL Language
3. Examples
4. The OWL Namespace
5. Future Extensions

2

Chapter 4

A Semantic Web Primer

Requirements for Ontology Languages

- Ontology languages allow users to write explicit, formal conceptualizations of domain models
- The main requirements are:
 - a well-defined syntax
 - efficient reasoning support
 - a formal semantics
 - sufficient expressive power
 - convenience of expression

3

Chapter 4

A Semantic Web Primer

Tradeoff between Expressive Power and Efficient Reasoning Support

- The richer the language is, the more inefficient the reasoning support becomes
- Sometimes it crosses the border of *noncomputability*
- We need a compromise:
 - A language supported by reasonably efficient reasoners
 - A language that can express large classes of ontologies and knowledge.

4

Chapter 4

A Semantic Web Primer

Reasoning About Knowledge in Ontology Languages

- **Class membership**
 - If x is an instance of a class C , and C is a subclass of D , then we can infer that x is an instance of D
- **Equivalence of classes**
 - If class A is equivalent to class B , and class B is equivalent to class C , then A is equivalent to C , too

5

Chapter 4

A Semantic Web Primer

Reasoning About Knowledge in Ontology Languages (2)

- **Consistency**
 - X instance of classes A and B , but A and B are disjoint
 - This is an indication of an error in the ontology
- **Classification**
 - Certain property-value pairs are a sufficient condition for membership in a class A ; if an individual x satisfies such conditions, we can conclude that x must be an instance of A

6

Chapter 4

A Semantic Web Primer

Uses for Reasoning

- Reasoning support is important for
 - checking the consistency of the ontology and the knowledge
 - checking for unintended relationships between classes
 - automatically classifying instances in classes
- Checks like the preceding ones are valuable for
 - designing large ontologies, where multiple authors are involved
 - integrating and sharing ontologies from various sources

7

Chapter 4

A Semantic Web Primer

Reasoning Support for OWL

- Semantics is a prerequisite for reasoning support
- Formal semantics and reasoning support are usually provided by
 - mapping an ontology language to a known logical formalism
 - using automated reasoners that already exist for those formalisms
- OWL is (partially) mapped on a description logic, and makes use of reasoners such as FaCT and RACER
- Description logics are a subset of predicate logic for which efficient reasoning support is possible

8

Chapter 4

A Semantic Web Primer

Limitations of the Expressive Power of RDF Schema

- **Local scope of properties**
 - **rdfs:range** defines the range of a property (e.g. eats) for all classes
 - In RDF Schema we cannot declare range restrictions that apply to some classes only
 - E.g. we cannot say that cows eat only plants, while other animals may eat meat, too

9

Chapter 4

A Semantic Web Primer

Limitations of the Expressive Power of RDF Schema (2)

- **Disjointness of classes**
 - Sometimes we wish to say that classes are disjoint (e.g. **male** and **female**)
- **Boolean combinations of classes**
 - Sometimes we wish to build new classes by combining other classes using union, intersection, and complement
 - E.g. **person** is the disjoint union of the classes **male** and **female**

10

Chapter 4

A Semantic Web Primer

Limitations of the Expressive Power of RDF Schema (3)

- **Cardinality restrictions**
 - E.g. a person has exactly two parents, a course is taught by at least one lecturer
- **Special characteristics of properties**
 - Transitive property (like “greater than”)
 - Unique property (like “is mother of”)
 - A property is the inverse of another property (like “eats” and “is eaten by”)

11

Chapter 4

A Semantic Web Primer

Combining OWL with RDF Schema

- Ideally, OWL would extend RDF Schema
 - Consistent with the layered architecture of the Semantic Web
- **But** simply extending RDF Schema would work against obtaining expressive power and efficient reasoning
 - Combining RDF Schema with logic leads to uncontrollable computational properties

12

Chapter 4

A Semantic Web Primer

Three Species of OWL

- W3C's Web Ontology Working Group defined OWL as three different sublanguages:
 - OWL Full
 - OWL DL
 - OWL Lite
- Each sublanguage geared toward fulfilling different aspects of requirements

13

Chapter 4

A Semantic Web Primer

OWL Full

- It uses all the OWL languages primitives
- It allows the combination of these primitives in arbitrary ways with RDF and RDF Schema
- OWL Full is fully upward-compatible with RDF, both syntactically and semantically
- OWL Full is so powerful that it is undecidable
 - No complete (or efficient) reasoning support

14

Chapter 4

A Semantic Web Primer

OWL DL

- OWL DL (Description Logic) is a sublanguage of OWL Full that restricts application of the constructors from OWL and RDF
 - Application of OWL's constructors' to each other is disallowed
 - Therefore it corresponds to a well studied description logic
- OWL DL permits efficient reasoning support
- **But** we lose full compatibility with RDF:
 - Not every RDF document is a legal OWL DL document.
 - Every legal OWL DL document is a legal RDF document.

15

Chapter 4

A Semantic Web Primer

OWL Lite

- An even further restriction limits OWL DL to a subset of the language constructors
 - E.g., OWL Lite excludes enumerated classes, disjointness statements, and arbitrary cardinality.
- The advantage of this is a language that is easier to
 - grasp, for users
 - implement, for tool builders
- The disadvantage is restricted expressivity

16

Chapter 4

A Semantic Web Primer

Upward Compatibility between OWL Species

- Every legal OWL Lite ontology is a legal OWL DL ontology
- Every legal OWL DL ontology is a legal OWL Full ontology
- Every valid OWL Lite conclusion is a valid OWL DL conclusion
- Every valid OWL DL conclusion is a valid OWL Full conclusion

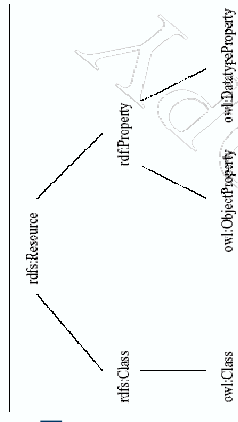
17

Chapter 4

A Semantic Web Primer

OWL Compatibility with RDF Schema

- All varieties of OWL use RDF for their syntax
- Instances are declared as in RDF, using RDF descriptions
- and typing information OWL constructors are specialisations of their RDF counterparts



18

Chapter 4

A Semantic Web Primer

OWL Compatibility with RDF Schema (2)

- Semantic Web design aims at **downward compatibility** with corresponding reuse of software across the various layers
- The advantage of full downward compatibility for OWL is only achieved for OWL Full, at the cost of computational intractability

19

Chapter 4

A Semantic Web Primer

Lecture Outline

1. Basic Ideas of OWL
2. **The OWL Language**
3. Examples
4. The OWL Namespace
5. Future Extensions

20

Chapter 4

A Semantic Web Primer

OWL Syntactic Varieties

- OWL builds on RDF and uses RDF's XML-based syntax
- Other syntactic forms for OWL have also been defined:
 - An alternative, more readable XML-based syntax
 - An abstract syntax, that is much more compact and readable than the XML languages
 - A graphic syntax based on the conventions of UML

21

OWL XML/RDF Syntax: Header

<rdf:RDF

```
xmlns:owl ="http://www.w3.org/2002/07/owl#"
xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

- An OWL ontology may start with a collection of assertions for housekeeping purposes using **owl:Ontology** element

22

owl:Ontology

```
<owl:Ontology rdf:about="">
<rdfs:comment>An example OWL ontology
</rdfs:comment>
<owl:priorVersion
  rdf:resource="http://www.mydomain.org/uni-ns-old"/>
<owl:imports
  rdf:resource="http://www.mydomain.org/persons"/>
<rdfs:label>University Ontology</rdfs:label>
</owl:Ontology>
```

- **owl:imports** is a transitive property

23

Classes

- Classes are defined using **owl:Class**
 - **owl:Class** is a subclass of **rdfs:Class**
- Disjointness is defined using **owl:disjointWith**

```
<owl:Class rdf:about="#associateProfessor">
<owl:disjointWith rdf:resource="#professor"/>
<owl:disjointWith
  rdf:resource="#assistantProfessor"/>
</owl:Class>
```

24

Classes (2)

- **owl:equivalentClass** defines equivalence of classes

```
<owl:Class rdf:ID="faculty">  
  <owl:equivalentClass rdf:resource=  
    "#academicStaffMember"/>  
</owl:Class>
```

- **owl:Thing** is the most general class, which contains everything
- **owl:Nothing** is the empty class

25

Properties

- In OWL there are two kinds of properties
 - **Object properties**, which relate objects to other objects
 - E.g. is-TaughtBy, supervises
 - **Data type properties**, which relate objects to datatype values
 - E.g. phone, title, age, etc.

26

Datatype Properties

- OWL makes use of XML Schema data types, using the layered architecture of the SW

```
<owl:DatatypeProperty rdf:ID="age">  
  <rdfs:range rdf:resource=  
    "http://www.w3.org/2001/XMLSchema  
    #nonNegativeInteger"/>  
</owl:DatatypeProperty>
```

27

Object Properties

- User-defined data types

```
<owl:ObjectProperty rdf:ID="isTaughtBy">  
  <owl:domain rdf:resource="#course"/>  
  <owl:range rdf:resource=  
    "#academicStaffMember"/>  
  <rdfs:subPropertyOf rdf:resource="#involves"/>  
</owl:ObjectProperty>
```

28

Inverse Properties

```
<owl:ObjectProperty rdf:ID="teaches">
  <rdfs:range rdf:resource="#course"/>
  <rdfs:domain rdf:resource=
    "#academicStaffMember"/>
  <owl:inverseOf rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
```

29

Chapter 4

A Semantic Web Primer

Equivalent Properties

```
owl:equivalentProperty
  <owl:ObjectProperty rdf:ID="lecturesIn">
  <owl:equivalentProperty
    rdf:resource="#teaches"/>
</owl:ObjectProperty>
```

30

Chapter 4

A Semantic Web Primer

Property Restrictions

- In OWL we can declare that the class C satisfies certain conditions
 - All instances of C satisfy the conditions
- This is equivalent to saying that C is subclass of a class C', where C' collects all objects that satisfy the conditions
 - C' can remain anonymous

31

Chapter 4

A Semantic Web Primer

Property Restrictions (2)

- A (restriction) class is achieved through an **owl:Restriction** element
- This element contains an **owl:onProperty** element and one or more **restriction declarations**
- One type defines **cardinality restrictions** (at least one, at most 3,...)

32

Chapter 4

A Semantic Web Primer

Property Restrictions (3)

- The other type defines restrictions on the kinds of values the property may take
 - **owl:allValuesFrom** specifies universal quantification
 - **owl:hasValue** specifies a specific value
 - **owl:someValuesFrom** specifies existential quantification

33

owl:allValuesFrom

```
<owl:Class rdf:about="#firstYearCourse">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isTaughtBy"/>
      <owl:allValuesFrom
        rdf:resource="#Professor"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

34

owl:hasValue

```
<owl:Class rdf:about="#mathCourse">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource=
        "#isTaughtBy"/>
      <owl:hasValue rdf:resource=
        "#949352"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

35

owl:someValuesFrom

```
<owl:Class rdf:about="#academicStaffMember">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#teaches"/>
      <owl:someValuesFrom rdf:resource=
        "#undergraduateCourse"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

36

Cardinality Restrictions

- We can specify minimum and maximum number using **owl:minCardinality** and **owl:maxCardinality**
- It is possible to specify a precise number by using the same minimum and maximum number
- For convenience, OWL offers also **owl:cardinality**

37

Cardinality Restrictions (2)

```
<owl:Class rdf:about="#course">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#isTaughtBy"/>
      <owl:minCardinality rdf:datatype=
        "&xsd;nonNegativeInteger">
        1
      </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

38

Special Properties

- **owl:TransitiveProperty** (transitive property)
 - E.g. “has better grade than”, “is ancestor of”
- **owl:SymmetricProperty** (symmetry)
 - E.g. “has same grade as”, “is sibling of”
- **owl:FunctionalProperty** defines a property that has at most one value for each object
 - E.g. “age”, “height”, “directSupervisor”
- **owl:InverseFunctionalProperty** defines a property for which two different objects cannot have the same value

39

Special Properties (2)

```
<owl:ObjectProperty rdf:ID="hasSameGradeAs">
  <rdf:type
    rdf:resource="&owl;TransitiveProperty"/>
  <rdf:type
    rdf:resource="&owl;SymmetricProperty"/>
  <rdfs:domain rdf:resource="#student"/>
  <rdfs:range rdf:resource="#student"/>
</owl:ObjectProperty>
```

40

Boolean Combinations

- We can combine classes using Boolean operations (union, intersection, complement)

```
<owl:Class rdf:about="#course">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:complementOf rdf:resource=
"#staffMember"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

41

Chapter 4

A Semantic Web Primer

Boolean Combinations (2)

```
<owl:Class rdf:ID="peopleAtUni">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#staffMember"/>
    <owl:Class rdf:about="#student"/>
  </owl:unionOf>
</owl:Class>
```

- The new class is not a subclass of the union, but rather equal to the union
 - We have stated an equivalence of classes

42

Chapter 4

A Semantic Web Primer

Boolean Combinations (3)

```
<owl:Class rdf:ID="facultyInCS">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#faculty">
      <owl:Restriction>
        <owl:onProperty rdf:resource="#belongsTo"/>
        <owl:hasValue rdf:resource=
"#CSDepartment"/>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
```

43

Chapter 4

A Semantic Web Primer

Nesting of Boolean Operators

```
<owl:Class rdf:ID="adminStaff">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#staffMember"/>
    <owl:complementOf>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#faculty"/>
        <owl:Class rdf:about=
"#techSupportStaff"/>
      </owl:unionOf>
    </owl:complementOf>
  </owl:intersectionOf>
</owl:Class>
```

44

Chapter 4

A Semantic Web Primer

Enumerations with owl:oneOf

```
<owl:oneOf rdf:parseType="Collection">
  <owl:Thing rdf:about="#Monday"/>
  <owl:Thing rdf:about="#Tuesday"/>
  <owl:Thing rdf:about="#Wednesday"/>
  <owl:Thing rdf:about="#Thursday"/>
  <owl:Thing rdf:about="#Friday"/>
  <owl:Thing rdf:about="#Saturday"/>
  <owl:Thing rdf:about="#Sunday"/>
</owl:oneOf>
```

45

Declaring Instances

- Instances of classes are declared as in RDF:

```
<rdf:Description rdf:ID="949352">
  <rdf:type rdf:resource=
    "#academicStaffMember"/>
</rdf:Description>
<academicStaffMember rdf:ID="949352">
  <uni:age rdf:datatype="&xsd:integer">
    39<uni:age>
</academicStaffMember>
```

46

No Unique-Names Assumption

- OWL does not adopt the unique-names assumption of database systems
 - If two instances have a different name or ID does not imply that they are different individuals
- Suppose we state that each course is taught by at most one staff member, and that a given course is taught by two staff members
 - An OWL reasoner does not flag an error
 - Instead it infers that the two resources are equal

47

Distinct Objects

- To ensure that different individuals are indeed recognized as such, we must explicitly assert their inequality:

```
<lecturer rdf:about="949318">
  <owl:differentFrom rdf:resource="949352"/>
</lecturer>
```

48

Distinct Objects (2)

- OWL provides a shorthand notation to assert the pairwise inequality of all individuals in a given list

```
<owl:allDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <lecturer rdf:about="949318"/>
    <lecturer rdf:about="949352"/>
    <lecturer rdf:about="949111"/>
  </owl:distinctMembers>
</owl:allDifferent>
```

49

Chapter 4

A Semantic Web Primer

Data Types in OWL

- XML Schema provides a mechanism to construct user-defined data types
 - E.g., the data type of **adultAge** includes all integers greater than 18
- Such derived data types cannot be used in OWL
 - The OWL reference document lists all the XML Schema data types that can be used
 - These include the most frequently used types such as **string**, **integer**, **Boolean**, **time**, and **date**.

50

Chapter 4

A Semantic Web Primer

Versioning Information

- **owl:priorVersion** indicates earlier versions of the current ontology
 - No formal meaning, can be exploited for ontology management
- **owl:versionInfo** generally contains a string giving information about the current version, e.g. keywords

51

Chapter 4

A Semantic Web Primer

Versioning Information (2)

- **owl:backwardCompatibleWith** contains a reference to another ontology
 - All identifiers from the previous version have the same intended interpretations in the new version
 - Thus documents can be safely changed to commit to the new version
- **owl:incompatibleWith** indicates that the containing ontology is a later version of the referenced ontology but is not backward compatible with it

52

Chapter 4

A Semantic Web Primer

Combination of Features

- In different OWL languages there are different sets of restrictions regarding the application of features
- In **OWL Full**, all the language constructors may be used in any combination as long as the result is legal RDF

53

Chapter 4

A Semantic Web Primer

Restriction of Features in OWL DL

- **Vocabulary partitioning**
 - Any resource is allowed to be only a class, a data type, a data type property, an object property, an individual, a data value, or part of the built-in vocabulary, and not more than one of these
- **Explicit typing**
 - The partitioning of all resources must be stated explicitly (e.g. a class must be declared if used in conjunction with **rdfs:subClassOf**)

54

Chapter 4

A Semantic Web Primer

Restriction of Features in OWL DL (2)

- **Property Separation**
 - The set of object properties and data type properties are disjoint
 - Therefore the following can never be specified for data type properties:
 - owl:inverseOf**
 - owl:FunctionalProperty**
 - owl:InverseFunctionalProperty**
 - owl:SymmetricProperty**

55

Chapter 4

A Semantic Web Primer

Restriction of Features in OWL DL (3)

- **No transitive cardinality restrictions**
 - No cardinality restrictions may be placed on transitive properties
- **Restricted anonymous classes: Anonymous classes are only allowed to occur as:**
 - the domain and range of either **owl:equivalentClass** or **owl:disjointWith**
 - the range (but not the domain) of **rdfs:subClassOf**

56

Chapter 4

A Semantic Web Primer

Restriction of Features in OWL Lite

- Restrictions of OWL DL and more
- **owl:oneOf**, **owl:disjointWith**, **owl:unionOf**, **owl:complementOf** and **owl:hasValue** are not allowed
- Cardinality statements (minimal, maximal, and exact cardinality) can only be made on the values 0 or 1
- **owl:equivalentClass** statements can no longer be made between anonymous classes but only between class identifiers

57

Chapter 4

A Semantic Web Primer

Lecture Outline

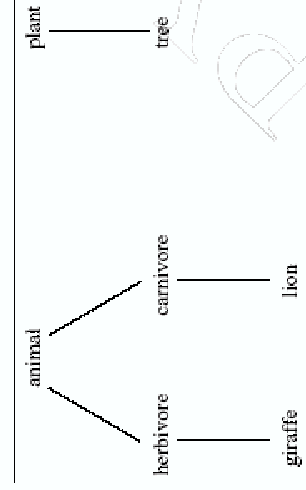
1. Basic Ideas of OWL
2. The OWL Language
3. Examples
4. The OWL Namespace
5. Future Extensions

58

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Class Hierarchy



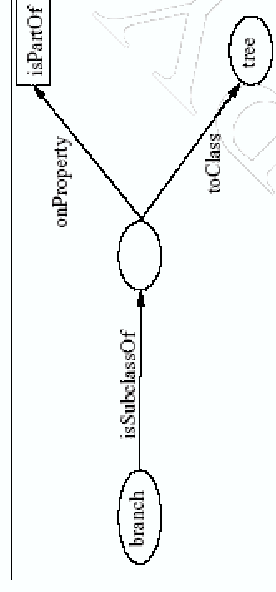
59

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Schematic Representation

Branches are parts of trees



60

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Properties

```
<owl:TransitiveProperty rdf:ID="is-part-of"/>
<owl:ObjectProperty rdf:ID="eats">
  <rdfs:domain rdf:resource="#animal"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="eaten-by">
  <owl:inverseOf rdf:resource="#eats"/>
</owl:ObjectProperty>
```

61

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Plants and Trees

```
<owl:Class rdf:ID="plant">
  <rdfs:comment>Plants are disjoint from animals.
</rdfs:comment>
<owl:disjointWith="#animal"/>
</owl:Class>
<owl:Class rdf:ID="tree">
  <rdfs:comment>Trees are a type of plant.
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#plant"/>
</owl:Class>
```

62

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Branches

```
<owl:Class rdf:ID="branch">
  <rdfs:comment>Branches are parts of trees.
</rdfs:comment>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#is-part-of"/>
    <owl:allValuesFrom rdf:resource="#tree"/>
  </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

63

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Leaves

```
<owl:Class rdf:ID="leaf">
  <rdfs:comment>Leaves are parts of branches.
</rdfs:comment>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#is-part-of"/>
    <owl:allValuesFrom rdf:resource="#branch"/>
  </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
```

64

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Carnivores

```
<owl:Class rdf:ID="carnivore">
  <rdfs:comment>Carnivores are exactly those animals
  that eat also animals.</rdfs:comment>
  <owl:intersectionOf rdf:parsetype="Collection">
    <owl:Class rdf:about="#animal"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eats"/>
      <owl:someValuesFrom rdf:resource="#animal"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

65

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Herbivores

```
<owl:Class rdf:ID="herbivore">
  <rdfs:comment>
    Herbivores are exactly those animals
    that eat only plants or parts of plants.
  </rdfs:comment>
  <rdfs:comment>
    Try it out! See book for code.
  </rdfs:comment>
</owl:Class>
```

66

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Giraffes

```
<owl:Class rdf:ID="giraffe">
  <rdfs:comment>Giraffes are herbivores, and they
  eat only leaves.</rdfs:comment>
  <rdfs:subClassOf rdf:type="#herbivore"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eats"/>
      <owl:allValuesFrom rdf:resource="#leaf"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

67

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Lions

```
<owl:Class rdf:ID="lion">
  <rdfs:comment>Lions are animals that eat
  only herbivores.</rdfs:comment>
  <rdfs:subClassOf rdf:type="#carnivore"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#eats"/>
      <owl:allValuesFrom rdf:resource="#herbivore"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

68

Chapter 4

A Semantic Web Primer

An African Wildlife Ontology – Tasty Plants

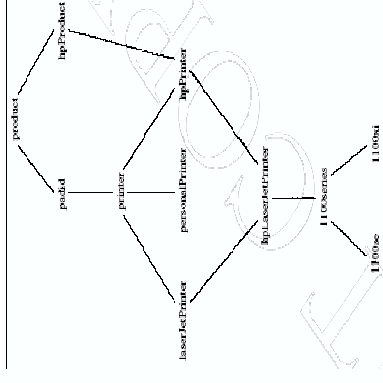
```
owl:Class rdf:ID="tasty-plant">
  <rdfs:comment>Plants eaten both by herbivores
  and carnivores </rdfs:comment>
  <rdfs:comment>
    Try it out! See book for code.
  </rdfs:comment>
</owl:Class>
```

69

Chapter 4

A Semantic Web Primer

A Printer Ontology – Class Hierarchy



70

Chapter 4

A Semantic Web Primer

A Printer Ontology – Products and Devices

```
<owl:Class rdf:ID="product">
  <rdfs:comment>Products form a class. </rdfs:comment>
</owl:Class>

<owl:Class rdf:ID="padid">
  <rdfs:comment>Printing and digital imaging devices
  form a subclass of products.</rdfs:comment>
  <rdfs:label>Device</rdfs:label>
  <rdfs:subClassOf rdf:resource="#product"/>
</owl:Class>
```

71

Chapter 4

A Semantic Web Primer

A Printer Ontology – HP Products

```
<owl:Class rdf:ID="hpProduct">
  <owl:intersectionOf>
    <owl:Class rdf:about="#product"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#manufactured-by"/>
      <owl:hasValue>
        <xsd:string rdf:value="Hewlett Packard"/>
      </owl:hasValue>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

72

Chapter 4

A Semantic Web Primer

A Printer Ontology – Printers and Personal Printers

```
<owl:Class rdf:ID="printer">
  <rdfs:comment>Printers are printing and digital imaging
  devices.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#padid"/>
</owl:Class>

<owl:Class rdf:ID="personalPrinter">
  <rdfs:comment>Printers for personal use form
  a subclass of printers.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#printer"/>
</owl:Class>
```

73

Chapter 4

A Semantic Web Primer

A Printer Ontology – HP LaserJet 1100se Printers

```
<owl:Class rdf:ID="1100se">
  <rdfs:comment>1100se printers belong to the 1100 series
  and cost $450.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#1100series"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#price"/>
      <owl:hasValue><xsd:integer rdf:value="450"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

74

Chapter 4

A Semantic Web Primer

A Printer Ontology – Properties

```
<owl:DatatypeProperty rdf:ID="manufactured-by">
  <rdfs:domain rdf:resource="#product"/>
  <rdfs:range rdf:resource="<xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:ID="printingTechnology">
  <rdfs:domain rdf:resource="#printer"/>
  <rdfs:range rdf:resource="<xsd:string"/>
</owl:DatatypeProperty>
```

75

Chapter 4

A Semantic Web Primer

Lecture Outline

1. Basic Ideas of OWL
2. The OWL Language
3. Examples
4. [The OWL Namespace](#)
5. Future Extensions

76

Chapter 4

A Semantic Web Primer

OWL in OWL

- We present a part of the definition of OWL in terms of itself
- The following captures some of OWL's meaning in OWL
 - It does **not** capture the entire semantics
 - A separate semantic specification is necessary
- The URI of the OWL definition is defined as the default namespace

77

Classes of Classes (Metaclasses)

- The class of all OWL classes is itself a subclass of the class of all RDF Schema classes:

```
<rdfs:Class rdf:ID="Class">  
<rdfs:label>Class</rdfs:label>  
<rdfs:subClassOf rdf:resource="&rdfs;Class"/>  
</rdfs:Class>
```

78

Classes of Classes (Metaclasses) – Thing and Nothing

- **Thing** is most general object class in OWL
- **Nothing** is most specific class: the empty object class
- The following relationships hold:

$$\textit{Thing} = \textit{Nothing} \cup \overline{\textit{Nothing}}$$

$$\overline{\textit{Nothing}} = \textit{Thing} = \overline{\textit{Nothing} \cup \overline{\textit{Nothing}} = \textit{Nothing} \cap \overline{\textit{Nothing}}} = \emptyset$$

79

Classes of Classes (Metaclasses) – Thing and Nothing (2)

```
<Class rdf:ID="Thing">  
<rdfs:label>Thing</rdfs:label>  
<unionOf rdf:parseType="Collection">  
<Class rdf:about="#Nothing"/>  
<Class>  
</unionOf>  
</Class>  
<Class rdf:ID="Nothing">  
<rdfs:label>Nothing</rdfs:label>  
<complementOf rdf:resource="#Thing"/>  
</Class>
```

80

Class and Property Equivalences

```
<rdf:Property rdf:ID="EquivalentClass">
  <rdfs:label>EquivalentClass</rdfs:label>
  <rdfs:subPropertyOf rdf:resource="#&rdfs;subClassOf"/>
  <rdfs:domain rdf:resource="#Class"/>
  <rdfs:range rdf:resource="#Class"/>
</rdf:Property>

<rdf:Property rdf:ID="EquivalentProperty">
  <rdfs:label>EquivalentProperty</rdfs:label>
  <rdfs:subPropertyOf
    rdf:resource="#&rdfs;subPropertyOf"/>
</rdf:Property>
```

81

Class Disjointness

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

82

Equality and Inequality

- Equality and inequality can be stated between arbitrary things
 - In OWL Full this statement can also be applied to classes
- Properties **sameIndividualAs**, **sameAs** and **differentFrom**

83

Equality and Inequality (2)

```
<rdf:Property rdf:ID="sameIndividualAs">
  <rdfs:domain rdf:resource="#Thing"/>
  <rdfs:range rdf:resource="#Thing"/>
</rdf:Property>

<rdf:Property rdf:ID="sameAs">
  <EquivalentProperty rdf:resource=
    "#sameIndividualAs"/>
</rdf:Property>
```

84

Union and Intersection of Classes

- Build a class from a list, assumed to be a list of other class expressions

```
<rdf:Property rdf:ID="unionOf">
  <rdfs:domain rdf:resource="#Class"/>
  <rdfs:range rdf:resource="#&rdf;List"/>
</rdf:Property>
```

85

Chapter 4

A Semantic Web Primer

Restriction Classes

- Restrictions in OWL define the class of those objects that satisfy some attached conditions

```
<rdfs:Class rdf:ID="Restriction">
  <rdfs:label>Restriction</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Class"/>
</rdfs:Class>
```

86

Chapter 4

A Semantic Web Primer

Restriction Properties

- All the following properties (**onProperty**, **allValuesFrom**, **minCardinality**, etc.) are only allowed to occur within a restriction definition
 - Their domain is **owl:Restriction**, but they differ with respect to their range

87

Chapter 4

A Semantic Web Primer

Restriction Properties (2)

```
<rdf:Property rdf:ID="onProperty">
  <rdfs:label>onProperty</rdfs:label>
  <rdfs:domain rdf:resource="#Restriction"/>
  <rdfs:range rdf:resource="#&rdf;Property"/>
</rdf:Property>

<rdf:Property rdf:ID="allValuesFrom">
  <rdfs:label>allValuesFrom</rdfs:label>
  <rdfs:domain rdf:resource="#Restriction"/>
  <rdfs:range rdf:resource="#&rdf;Class"/>
</rdf:Property>
```

88

Chapter 4

A Semantic Web Primer

Restriction Properties (3)

```
<rdf:Property rdf:ID="hasValue">
  <rdfs:label>hasValue</rdfs:label>
  <rdfs:domain rdf:resource="#Restriction"/>
</rdf:Property>
<rdf:Property rdf:ID="minCardinality">
  <rdfs:label>minCardinality</rdfs:label>
  <rdfs:domain rdf:resource="#Restriction"/>
  <rdfs:range rdf:resource=
    "&xsd;nonNegativeInteger"/>
</rdf:Property>
```

89

Properties

- owl:ObjectProperty and owl:DatatypeProperty are special cases of rdf:Property

```
<rdfs:Class rdf:ID="ObjectProperty">
  <rdfs:label>ObjectProperty</rdfs:label>
  <rdfs:subClassOf rdf:resource="&rdfs:Property"/>
</rdfs:Class>
```

90

Properties (2)

- Symmetric, functional and inverse functional properties can only be applied to object properties

```
<rdfs:Class rdf:ID="TransitiveProperty">
  <rdfs:label>TransitiveProperty</rdfs:label>
  <rdfs:subClassOf rdf:resource=
    "#ObjectProperty"/>
</rdfs:Class>
```

91

Properties (3)

- owl:inverseOf relates two object properties:

```
<rdf:Property rdf:ID="inverseOf">
  <rdfs:label>inverseOf</rdfs:label>
  <rdfs:domain rdf:resource="#ObjectProperty"/>
  <rdfs:range rdf:resource="#ObjectProperty"/>
</rdf:Property>
```

92

Lecture Outline

1. Basic Ideas of OWL
2. The OWL Language
3. Examples
4. The OWL Namespace
5. [Future Extensions](#)

93

Chapter 4

A Semantic Web Primer

Future Extensions of OWL

- Modules and Imports
- Defaults
- Closed World Assumption
- Unique Names Assumption
- Procedural Attachments
- Rules for Property Chaining

94

Chapter 4

A Semantic Web Primer

Modules and Imports

- The importing facility of OWL is very trivial:
 - It only allows importing of an entire ontology, not parts of it
- Modules in programming languages based on **information hiding**: state functionality, hide implementation details
 - Open question how to define appropriate module mechanism for Web ontology languages

95

Chapter 4

A Semantic Web Primer

Defaults

- Many practical knowledge representation systems allow inherited values to be overridden by more specific classes in the hierarchy
 - treat inherited values as defaults
- No consensus has been reached on the right formalization for the nonmonotonic behaviour of default values

96

Chapter 4

A Semantic Web Primer

Closed World Assumption

- OWL currently adopts the **open-world assumption**:
 - A statement cannot be assumed true on the basis of a failure to prove it
 - On the huge and only partially knowable WWW, this is a correct assumption
- **Closed-world assumption**: a statement is true when its negation cannot be proved
 - tied to the notion of defaults, leads to nonmonotonic behaviour

97

Unique Names Assumption

- Typical database applications assume that individuals with different names are indeed different individuals
- OWL follows the usual logical paradigm where this is not the case
 - Plausible on the WWW
- One may want to indicate portions of the ontology for which the assumption does or does not hold

98

Procedural Attachments

- A common concept in knowledge representation is to define the meaning of a term by attaching a piece of code to be executed for computing the meaning of the term
 - Not through explicit definitions in the language
- Although widely used, this concept does not lend itself very well to integration in a system with a formal semantics, and it has not been included in OWL

99

Rules for Property Chaining

- OWL does not allow the composition of properties for reasons of decidability
- In many applications this is a useful operation
- One may want to define properties as general rules (Horn or otherwise) over other properties
- Integration of rule-based knowledge representation and DL-style knowledge representation is currently an active area of research

100

Summary

- OWL is the proposed standard for Web ontologies
- OWL builds upon RDF and RDF Schema:
 - (XML-based) RDF syntax is used
 - Instances are defined using RDF descriptions
 - Most RDFS modeling primitives are used

101

Summary (2)

- Formal semantics and reasoning support is provided through the mapping of OWL on logics
 - Predicate logic and description logics have been used for this purpose
- While OWL is sufficiently rich to be used in practice, extensions are in the making
 - They will provide further logical features, including rules

102