OWL Reasoning

Irini Fundulaki

Institute of Computer Science

FORTH
Requirements for an Ontology Language

Ontology Languages allow users to write explicit, formal conceptualizations of domain models

- Extend existing Web Standards and build upon their syntax
  - Necessary conditions for machine processing information
  - XML, RDF, RDFS

- Easy to understand and use
  - Based on known Knowledge Representation Languages

- Sufficient Expressive Power

- Formal semantics
  - describe the meaning of knowledge *precisely* without being open to different interpretations
  - Essential for automated reasoning support

Irini Fundulaki, ESWC 2015 Summer School
Limitations of RDF Schema

• Modeling primitives of RDF and RDFS concern the organization of vocabularies in typed hierarchies
  – subclass and subproperty hierarchies
  – Global domain and range definitions for properties

• Missing:
  – Disjointness of classes
  – Boolean combinations of class expressions
  – Cardinality restrictions
  – Special “characteristics” of properties
    • Transitive Properties
    • Uniqueness of property values …
A short history of Ontology Languages

• Web Ontology Working Group of W3C (2001) identified test cases that required more expressiveness than RDF/RDFS.
• Lead to a joint Initiative that produced DAML+OIL
  – http://www.daml.org/2001/03/daml+oil-index.html
  – Starting point for the W3C Web Ontology Language (OWL)
• OWL is an ontology language designed for the Semantic Web
  – Semantic Web Knowledge Representation Language for Web Resources (URIs) based on Description Logics (DLs)
  – Provides a rich collection of operators for forming concept descriptions
  – Promotes interoperation and sharing between applications
  – Designed to be compatible with existing web standards
    • Using Web-enabled syntaxes based on XML or RDF
W3C Web Ontology Language (OWL)

• Two versions of OWL:
  – OWL2 Revised Recommendation (2009)

• OWL2 is more expressive than OWL1
  – Takes advantage of developments in DL reasoning techniques
Compatibility of OWL with RDFS

• Ideally OWL would be an extension of RDF Schema
  – OWL could use the RDF meaning of classes (rdfs:Class) and properties (rdf:Property) and add language primitives to support richer expressiveness
  – Be consistent with the layered architecture of the Semantic Web
  – Extension would work against obtaining expressive power and efficient reasoning leading to uncontrollable computational properties if logic is extended with expressive primitives

• Full Set of Requirements: Unobtainable
  – Definition of three different sublanguages of OWL, each geared toward fulfilling different aspects of the set of requirements
OWL Languages

• Three sublanguages of OWL
  – OWL Full
  – OWL DL
  – OWL Lite

• Syntactic Layering

• Semantic Layering
  – OWL DL semantics exactly the OWL Full Semantics (within the DL fragment)
  – OWL Lite semantics exactly the OWL DL semantics (within the Lite fragment)

There is a tradeoff between the expressiveness of a representation language and the difficulty of reasoning over the representations built using that language.

Brachman and Levesque (1984)
OWL\textsuperscript{1}: OWL Full

- Uses all OWL Language Primitives
- Allows the combination of primitives with RDF and RDFS in arbitrary ways
  - Includes the possibility of changing the meaning of predefined primitives
    - E.g., impose a cardinality constraint on the class of all classes hence limiting the number of classes one can describe in an ontology
    - Handle classes as instances (meta-modeling)
- Advantage: fully upward compatible with RDF both syntactically and semantically
  - Any legal RDF set of statements is also a legal OWL Full set of statements
- Disadvantage: Undecidable Language, no efficient reasoning support
OWL1: OWL DL

• Sublanguage of OWL Full
  • Restricts how the constructors of OWL and RDF can be used
    – Cannot modify the semantics of predefined constructs
    – Classes cannot be used as instances
• Defined by an abstract syntax and mapping to RDF
• Direct mapping to DL/First Order Logic
• Advantages:
  – Well defined semantics
  – Well understood formal properties (complexity, decidability)
  – Efficient reasoning support
  – Highly Optimized Implemented Systems
• Disadvantage: no (direct) compatibility with RDF
  – Any legal RDF set of statements must be extended and or restricted in order to be a legal OWL DL dataset but every legal OWL DL set of statements is a legal set of RDF statements
OWL\textsubscript{1}: OWL Lite

• Sublanguage of OWL DL
  – No ability to support explicit negation or union
  – Excludes disjointness statements
  – Excludes enumerated classes as property range
  – Supports Cardinality constraints (zero or one)
    • A property of an instance can have zero or one values

• Direct mapping to DL/FOL

• Reasoning via standard RDF engines
  – Pellet, FaCT, RACER, OWLIM

• Advantage: Easiness of implementation, efficient reasoning

• Disadvantage: Limited Expressivity
OWL1: Which language for what?

- **OWL Lite**
  - Classification Hierarchy
  - Simple Constraints

- **OWL DL**
  - Maximal Expressiveness
  - Tractability is maintained
  - Standard Formalization

- **OWL Full**
  - Very high expressiveness
  - Tractability is lost
  - Non standard formalization
  - Syntactic Freedom of RDF

Syntactic and Semantic relationships between the OWL sub-languages
OWL in RDF
  – RDF/XML Syntax
XML Presentation Syntax
  – Based on an XML Schema Definition
Various syntaxes easier to be consumed by a human reader
OWL 1.0 Features and Syntax

• Ontology header for metadata

```xml
<owl:Ontology rdf:about="">
  <owl:versionOf>1.4</owl:versionOf>
  <rdfs:comment>An ontology about music</rdfs:comment>
  <owl:imports rdf:resource="http://dbpedia.org/"
</owl:Ontology>
```

• Versioning Support
  – `owl:versionInfo` (version information)
  – `owl:priorVersion` (prior version)
  – `owl:backwardsCompatibleWith`
    • Specified ontology is a prior version of current one and is compatible with it
  – `owl:incompatibleWith`
    • Specified ontology is a prior version of current one and is not compatible with it
  – Classes and properties can be declared as deprecated in the current ontology version
    • `owl:DeprecatedClass`
    • `owl:DeprecatedProperty`
OWL Classes & Properties

• **Classes**
  – owl:Class
    • Distinct from rdfs:Class
    • Needed for OWL Lite/OWL DL
  – owl:Thing
    • Everything is a member of class owl:Thing
  – owl:Nothing
    • Represents the empty class

• **Properties**
  – owl:topObjectProperty
    • A property that links every individual to every individual
  – owl:ObjectProperty
    • The class of properties whose value is a resource
  – owl:DataTypeProperty
    • The class of properties whose value is an atomic value
OWL Classes

• **A class** defines a group of individuals which share some properties

• A class is associated with a set of instances, called **class extension**
  – The individuals in the class extension are called the **instances of the class**

• A class has an **intentional meaning** (the underlying concept) which is related but not equal to its extension
  – Two classes may have the same class extension, but still be different classes.

• **Class Descriptions**
  1. a class identifier (a URI reference)
  2. an exhaustive enumeration of individuals that form the class extension
  3. a property restriction
  4. the intersection/union/complement of two or more class descriptions
1. Class Description: URI Reference

- Class http://dbpedia.org#Artist is the set of artists.
- Individual http://dbpedia.org#TomWaits is a member (or instance) of class http://dbpedia.org#Artist

**Class Definition in RDF/XML**

```xml
<owl:Class rdf:about="http://dbpedia.org#Artist">
  <rdfs:label rdf:datatype="&xsd;string" xml:lang="en">Artist</rdfs:label>
  <rdfs:label rdf:datatype="&xsd;string" xml:lang="fr">Artiste</rdfs:label>
</owl:Class>
```

**Individual Definition in RDF/XML**

```xml
<owl:NamedIndividual rdf:about="http://dbpedia.org#TomWaits">
  <rdf:type rdf:resource="http://dbpedia.org#Artist"/>
</owl:NamedIndividual>
```
OWL Classes

• Classes are organized in specialization hierarchies using built-in property rdfs:subClassOf
  – Class http://dbpedia.org/ontology/Artist is a subClassOf Class Person

• Built-in class http://www.w3.org/2002/07/owl#Thing
  – is the class of all individuals
  – is the superclass of all OWL classes

• Built-in class http://www.w3.org/2002/07/owl#Nothing
  – has no instances
  – is the subclass of all OWL classes
2. Class Description: Instance Enumeration

- Defines a class as an exhaustive enumeration of individuals that form the extension the set of instances of the class
- No new instances can be added to the class extension
- `owl:oneOf ({a1, a2, ... an})`

OWL class JazzGenre collects all types of Jazz Music

```xml
<owl:Class rdf:about="http://dbpedia.org#JazzGenre">
    <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="http://dbpedia.org#AcidJazz"/>
        <owl:Thing rdf:about="http://dbpedia.org#Avant-GardeJazz"/>
        <owl:Thing rdf:about="http://dbpedia.org#BigBand"/>
        <owl:Thing rdf:about="http://dbpedia.org#BlueNote"/>
        <owl:Thing rdf:about="http://dbpedia.org#ContemporaryJazz"/>
        <owl:Thing rdf:about="http://dbpedia.org#CrossoverJazz"/>
        <owl:Thing rdf:about="http://dbpedia.org#Dixieland"/>
        <owl:Thing rdf:about="http://dbpedia.org#Fusion"/>
        <owl:Thing rdf:about="http://dbpedia.org#MainstreamJazz"/>
        <owl:Thing rdf:about="http://dbpedia.org#SmoothJazz"/>
    </owl:oneOf>
</owl:Class>
```

Not used in OWL Lite!
3. Class Description: Property Restrictions

- Describe an anonymous class, namely a class of all individuals that satisfy the restriction
- OWL distinguishes among
  - cardinality constraints
    - Max: \( \leq nR \), Min: \( \geq nR \), Equal: \( = nR \)
  - OWL Lite: only cardinalities of `0` and `1` are allowed
  - range constraints
    - \( \exists R \) \( \forall R \)
- Local Constraints
  - they apply to the properties of the instances of concerned classes

```
<owl:Restriction>
  <owl:onProperty rdf:resource="property"/>
  Constraint Expression
</owl:Restriction>
```

**General Form of Property Restriction**
Cardinality Constraints

- Define a class based on the number of values taken by a property
  - `owl:cardinality`: property $P$ has *exactly* $n$ values ($=nR$)

A string quartet has exactly 4 members

```xml
<owl:Class rdf:about="http://dbpedia.org#StringQuartet">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://dbpedia.org#hasMembers"/>
      <owl:cardinality>4</owl:cardinality>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```
Cardinality Constraints

- Define a class based on the number of values taken by a property
  - `owl:maxCardinality`: property $P$ has *at most* $n$ values ($\leq n_R$)
  - `owl:minCardinality`: property $P$ has *at least* $n$ values ($\geq n_R$)

A full sized orchestra has at least 70 and at most 100 members

```xml
<owl:Class about="http://dbpedia.org#FullSizedOrchestra">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://dbpedia.org#hasMembers"/>
      <owl:minCardinality>70</owl:cardinality>
      <owl:maxCardinality>100</owl:cardinality>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```
(Local) Range Constraints

- Define a class based on the type of property values
- Different from global RDFS range constraints
  - owl:someValuesFrom: $\exists$ PC
    - Defines the class of individuals $x$ for which there exist at least one value $y$ (instance of class $C$ or of the specified data range) such that $(x, y)$ is an instance of property $P$
  - owl:allValuesFrom: $\forall$ PC
    - Defines a class of individuals $x$ for which it holds that if the pair $(x, y)$ is an instance of $P$, then $y$ should be an instance of class $C$ or a value in the specified data range

*Can only be used with named classes or datatypes in OWL Lite*

- owl:hasValue: $\exists$ P.$\{V\}$
  - Defines a class of the individuals $x$ that have as value for property $P$, $V$ or one that is equivalent to $V$

*Cannot be used in OWL Lite*
Members of a string quartet play one of violin, viola, cello, and double bass

```
<owl:Class rdf:about="http://dbpedia.org#StringQuartetMember">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://dbpedia.org#playsInstrument"/>
      <owl:allValuesFrom>
        <owl:Class>
          <owl:oneOf rdf:parseType="Collection">
            <owl:Thing rdf:about="http://dbpedia.org#Violin"/>
            <owl:Thing rdf:about="http://dbpedia.org#Viola"/>
            <owl:Thing rdf:about="http://dbpedia.org#Cello"/>
            <owl:Thing rdf:about="http://dbpedia.org#DoubleBass"/>
          </owl:oneOf>
        </owl:Class>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```
At least one of the members of a Jazz band plays the saxophone

```xml
<owl:Class rdf:about="http://dbpedia.org#JazzBandMember">
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://dbpedia.org#playsInstrument"/>
            <owl:someValuesFrom rdf:resource="http://dbpedia.org#Saxophone"/>
        </owl:Restriction>
    </owl:equivalentClass>
</owl:Class>
```

Violin is the instrument of a violinist

```xml
<owl:Class rdf:about="http://dbpedia.org#Violinist">
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://dbpedia.org#playsInstrument"/>
            <owl:hasValue rdf:resource="http://dbpedia.org#Violin"/>
        </owl:Restriction>
    </owl:equivalentClass>
</owl:Class>
```
4. Class Descriptions through set operations

- Set Intersection: `owl.intersectionOf`
- Set Union: `owl.unionOf`
- Set Complementation: `owl.complementOf`
- `owl:IntersectionOf`
  - links a class to a list of class descriptions
  - describes an (anonymous) class whose class extension contains the individuals that belong to the intersection of all said class descriptions
- `owl:unionOf`
  - links a class to a list of class descriptions
  - describes an (anonymous) class whose class extension contains the individuals that belong to the union of all said class descriptions
4. Class Descriptions through set operations

✓ Set Intersection: owl:intersectionOf
✓ Set Union: owl:unionOf
• Set Complementation: owl:complementOf
• owl:complementOf
  – links a class to precisely one class description
  – describes a class for which the class extension contains exactly those individuals that do not belong to the class extension of said class description
  – analogous to negation

• Only owl:intersectionOf used in OWL Lite
• Can be used with named classes and OWL restrictions only
An all-female band is a band whose members are all female musicians

```xml
<owl:Class about="http://dbpedia.org#AllFemaleBand">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://dbpedia.org#Musician"/>
        <owl:Class>
          <owl:Restriction>
            <owl:onProperty rdf:about="gender"/>
            <owl:hasValue rdf:resource="female"/>
          </owl:Restriction>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```
a class for which the class extension contains three individuals, namely Tosca, Salome, and Turandot (assuming they are all different)

```xml
<owl:Class about="http://dbpedia.org#ItalianOpera">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Tosca" />
        <owl:Thing rdf:about="#Salome" />
      </owl:oneOf>
    </owl:Class>
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Turandot" />
        <owl:Thing rdf:about="#Tosca" />
      </owl:oneOf>
    </owl:Class>
  </owl:unionOf>
</owl:Class>
```

Everything but an all-female band

```xml
<owl:Class about="http://dbpedia.org#NotAnAllFemaleBand">
  <owl:complementOf>
    <owl:Class rdf:about="http://dbpedia.org#AllFemaleBand" />
  </owl:complementOf>
</owl:Class>
```
OWL Class Axioms

• Contain additional components that state necessary and/or sufficient characteristics of a class

• OWL contains 3 language constructs for combining class descriptions into class axioms
  – C1 rdfs:subClassOf C2
  – C1 owl:equivalentClass C2
  – C1 owl:disjointWith C2
C₁ rdfs:subClassOf C₂

- Extension of class C₁ is a *subset* of the extension of class C₂
- For any class C₁ there may be any number of rdfs:subClassOf axioms.
- subclass axioms provide partial definitions
  - they represent necessary but not sufficient conditions for establishing class membership of an individual
Traditional Italian opera is defined as a subclass of a class of operas that have as opera type either Opera Seria or Opera Buffa

<owl:Class rdf:ID="http://dbpedia.org#TraditionalItalianOpera">
  <rdfs:subClassOf rdf:resource="http://dbpedia.org#Opera"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://dbpedia.org#hasOperaType"/>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:oneOf rdf:parseType="Collection">
            <owl:Thing rdf:about="http://dbpedia.org#OperaSeria"/>
            <owl:Thing rdf:about="http://dbpedia.org#OperaBuffa"/>
          </owl:oneOf>
        </owl:Class>
        <owl:Class>
          <owl:oneOf>
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

without an additional cardinality constraint, Property “hasOperaType” could actually have both values
An operetta is a musical work, that has at least one librettist and is not an opera.

Leaves open the possibility that there are other musical works that have a librettist and are not operas

Use of owl:equivalentClass to state that Operetta’s are Operas
C1 owl:equivalentClass C2

- extension of class description C1 is *exactly the same* as class extension of class description C2
  - both class extensions contain exactly the same set of individuals
- does not imply class equality
  - Use owl:sameAs construct to denote class equality
- axioms with owl:equivalentClass can also be used to define an enumerated class by linking a class identifier to an enumeration
- for any class C1 there may be *any number of owl:equivalentClass axioms*
- equivalent class axioms provide *full definitions*
  - they represent necessary and sufficient conditions for classes
C1 owl:equivalentClass C2

<owl:Class rdf:ID="http://dbpedia.org#DaPonteOperaOfMozart">
  <owl:equivalentClass>
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <Opera rdf:about="http://dbpedia.org#Nozze_di_Figaro"/>
        <Opera rdf:about="http://dbpedia.org#Don_Giovanni"/>
        <Opera rdf:about="http://dbpedia.org#Cosi_fan_tutte"/>
      </owl:oneOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>

• Operas that together represent the “Da Ponte operas of Mozart”
• Expressed using an enumeration of three instances
• State necessary and sufficient conditions for class membership through owl:equivalenceClass construct
C1 owl:equivalentClass C2

<owl:Class rdf:ID="http://dbpedia.org#DaPonteOperaOfMozart">
  <owl:equivalentClass>
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <Opera rdf:about="http://dbpedia.org#Nozze_di_Figaro"/>
        <Opera rdf:about="http://dbpedia.org#Don_Giovanni"/>
        <Opera rdf:about="http://dbpedia.org#Cosi_fan_tutte"/>
      </owl:oneOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>

Constructs owl:oneOf, owl:intersectionOf, owl:unionOf and owl:complementOf are used for defining equivalent classes
C1 owl:disjointWith C2

- **owl:disjointWith**: the class extensions of the two classes have no individuals in common.
- Any class C1 can be associated to any number of other classes through owl:disjointWith axioms.
- owl:disjointWith axioms provide partial definitions — they represent necessary but not sufficient conditions.

**Cannot be used in OWL Lite**
C1 owl:disjointWith C2

<owl:Class rdf:about="http://dbpedia.org#MusicDrama">
  <owl:equivalentClass>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="http://dbpedia.org#Opera"/>
        <owl:Class rdf:about="http://dbpedia.org#Operetta"/>
        <owl:Class rdf:about="http://dbpedia.org#Musical"/>
      </owl:unionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:about="http://dbpedia.org#Opera">
  <rdfs:subClassOf rdf:resource="http://dbpedia.org#MusicDrama"/>
</owl:Class>

<owl:Class rdf:about="http://dbpedia.org#Operetta">
  <rdfs:subClassOf rdf:resource="http://dbpedia.org#MusicDrama"/>
  <owl:disjointWith rdf:resource="http://dbpedia.org#Opera"/>
</owl:Class>

<owl:Class rdf:about="http://dbpedia.org#Musical">
  <rdfs:subClassOf rdf:resource="http://dbpedia.org#MusicDrama"/>
  <owl:disjointWith rdf:resource="http://dbpedia.org#Opera"/>
  <owl:disjointWith rdf:resource="http://dbpedia.org#Operetta"/>
</owl:Class>
OWL Properties

- Property extension a pair of (subject, object) elements
  - Not a single element: in relational terms it is a binary relation
- Properties have a direction, from domain to range
- Two types of properties
  - Object properties link individuals to individuals
  - Datatype properties link individuals to data values
- Built-in Classes:
  - `owl:ObjectProperty`: the class of properties whose value is an individual
    ```xml
    <owl:ObjectProperty rdf:ID="http://dbpedia.org#instrument">
    ```
  - `owl:DatatypeProperty`: the class of properties whose value is an atomic value
    ```xml
    <owl:DatatypeProperty rdf:ID="http://dbpedia.org#birthYear">
    ```
  - `owl:ObjectProperty` and `owl:DatatypeProperty` are subclasses of the RDF class `rdf:Property`
OWL Property Axioms

- Used for defining **additional characteristics** for OWL Properties
  - Property Hierarchies
    - rdfs:subPropertyOf
  - Domain/Range Constraints
    - rdfs:range and rdfs:domain
  - Relations to other properties
    - owl:equivalentProperty and owl:inverseOf
  - Global Cardinality Constraints
    - owl:FunctionalProperty and owl:InverseFunctionalProperty
  - Logical Property Constraints
    - owl:SymmetricProperty and owl:TransitiveProperty
Property Hierarchies

• $P_1$ rdfs:subPropertyOf $P_2$
  
  – property extension of $P_1$ should be a subset of the property extension of $P_2$

  all instances of the property “musicFusionGenre” are also members of property “overlaps”.

<owl:ObjectProperty rdf:ID="http://dbpedia.org#musicFusionGenre">
  <rdfs:subPropertyOf rdf:resource="http://dbpedia.org#overlaps"/>
</owl:ObjectProperty>
Domain/Range Constraints

- **P1 rdfs:domain C1**
  - asserts that the subjects of instances of property **P1** must belong to the extension of class **C1**
  - links a property **P1** to **one or more class descriptions C**
  - when **multiple rdfs:domain axioms for property P** exist

- **restrict** the domain of the property to those individuals that belong to the **intersection of the class descriptions**

- when **multiple classes should act as domain**, one must use **owl:unionOf**
Domain/Range Constraints

- $P_1 \text{ rdfs:range } C_1$
  - asserts that the objects of instances of property $P_1$ must belong to the extension of class $C_1$
  - links a property $P$ to one or more class descriptions $C$ or a data range
  - when multiple $\text{rdfs:range}$ axioms for property $P$ exist
- restrict the range of the property to those individuals that belong to the intersection of the class descriptions
  - when multiple classes should act as range, one must use $\text{owl:unionOf}$
Domain/Range Constraints

"musicFusionGenre" is a subproperty of "overlaps", defined in and takes its value from class "MusicGenre"

<owl:ObjectProperty rdf:ID="http://dbpedia.org#musicFusionGenre">
  <rdfs:subPropertyOf rdf:resource="http://dbpedia.org#overlaps"/>
  <rdfs:domain rdf:resource="http://dbpedia.org#MusicGenre"/>
  <rdfs:range rdf:resource="http://dbpedia.org#MusicGenre"/>
</owl:ObjectProperty>

Value Constraints vs RDFS Constraints

- owl:allValuesFrom, owl:someValuesFrom are local and enforced on the property when applied to that class
- rdfs:range and rdfs:domain constraints are global and apply to all instances of the properties irrespective to the class in which it is applied
Relations to other properties

• $P_1$ owl:equivalentProperty $P_2$
  – Properties $P_1$ and $P_2$ have the same set of instances
  – Property equivalence is not property equality
    • Equivalent properties have the same instances, but may have different intentional meaning (i.e., denote different concepts).

• $P_1$ owl:inverseOf $P_2$
  – Recall: Properties have a direction, from domain to range.
  – owl:inverseOf construct can be used to define an inverse relation between properties
Relations to other properties

```
<owl:ObjectProperty rdf:ID="http://dbpedia.org#musicComposer">
  <rdfs:subPropertyOf rdf:resource="http://dbpedia.org#coParticipatesWith"/>
  <rdfs:domain rdf:resource="http://dbpedia.org#Work"/>
  <rdfs:range rdf:resource="http://dbpedia.org#MusicalArtist"/>
  <owl:equivalentProperty rdf:resource="http://dbpedia.org#musicBy"/>
  <owl:inverseOf rdf:resource="http://dbpedia.org#composed"/>
</owl:ObjectProperty>
```

- “musicComposer” is a **subproperty** of “coParticipatesWith”
- **defined in class Work**
- **takes its values from class “MusicalArtist”**
- **equivalent to property “musicBy”**
- **inverse of property “composed”**
Global Cardinality Constraints

• $P_1$ rdf:type owl:FunctionalProperty
  – A resource $x$, can have only one (unique) value $y$ for property $P_1$
  • If $P$ is a functional property there cannot be two distinct values $y_1$ and $y_2$ such that the pairs $(x,y_1)$ and $(x,y_2)$ are both instances of this property
  – Both object and datatype properties can be functional

<owl:DataTypeProperty rdf:ID="http://dbpedia.org#birthDate">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="http://dbpedia.org#Person"/>
  <rdfs:range rdf:resource="&xsd;date"/>
  <owl:equivalentProperty rdf:resource="http://schema.org#DateOfBirth"/>
</owl:DataTypeProperty>

• a person has a unique birthdate
• property birthDate is equivalent to property DateOfBirth
Global Cardinality Constraints

• **P₁ rdf:type owl:InverseFunctionalProperty**
  – A resource $x$, is uniquely determined by the *object* $y$ of property $P₁$
  – if $P$ is an inverse functional property there cannot be two distinct instances $x₁$ and $x₂$ such that both pairs $(x₁,y)$ and $(x₂,y)$ are instances of $P$.

```
<owl:ObjectProperty rdf:ID="http://dbpedia.org#SSN">  
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
  <rdfs:domain rdf:resource="http://dbpedia.org#Person"/>
  <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
</owl:ObjectProperty>
```

• **a person is uniquely identified by her SSN number**

*Cannot be used in OWL Lite/DL*
Logical Characteristics of Properties

- \( P_1 \text{ owl:TransitiveProperty } P_2 \)
  - If \((x, y)\) and \((y, z)\) are instances of property transitive property \( P \)
    then we can infer that the pair \((x, z)\) is also an instance of property \( P \)

```
<owl:ObjectProperty rdf:ID="http://dbpedia.org#subEvent">
    <rdf:type rdf:resource="&owl;TransitiveProperty"/>
    <rdfs:domain rdf:resource="http://dbpedia.org#MusicEvent"/>
    <rdfs:range rdf:resource="http://dbpedia.org#MusicEvent"/>
</owl:ObjectProperty>
```

"subEvent" is a transitive property, whose domain and range is class "MusicEvent"
Logical Characteristics of Properties

- **P1** rdf:type owl:SymmetricProperty
  - If *P* is a *symmetric property* and if a pair *(x,y)* is an *instance of P*, and the pair *(y,x)* is *also instance of P*

```
<owl:ObjectProperty rdf:ID="http://dbpedia.org#playedWith">
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
  <rdfs:domain rdf:resource="http://dbpedia.org#MusicArtist"/>
  <rdfs:range rdf:resource="http://dbpedia.org#MusicArtist"/>
</owl:ObjectProperty>
```
OWL2

- OWL1 was based on techniques that allowed decidable, sound and complete reasoning in DL languages
- OWL1 contained 3 species of OWL
  - OWL Full: an extension of RDF to give semantics to OWL keywords
    - Intended to behave “similar” to OWL DL but applicable to all RDF documents
    - Entailment problem undecidable (if the semantics is non-contradictory)
  - OWL DL: a DL-based KR language with an RDF syntax
    - Not all RDF documents are OWL DL ontologies
  - OWL Lite: a restricted version of OWL DL
- OWL2: OWL 2 DL and OWL Full to extended OWL 1 family of languages
  - Syntactic Sugar (easiness in writing statements)
  - Constructs for increased expressivity
  - Datatype support
  - Metamodelling
  - Annotation
OWL2: Disjoint Classes/Properties

- OWL1: allows us to specify that 2 classes/properties are disjoint
- OWL2: allows us to state that classes/properties in a set of classes/properties are pairwise disjoint

```xml
<owl:AllDisjointClasses>
  <owl:members rdf:parseType="Collection">
    <owl:Class rdf:about="&example;C1"/>
    <owl:Class rdf:about="&example;C2"/>
    <owl:Class rdf:about="&example;C3"/>
  </owl:members>
</owl:AllDisjointClasses>

<owl:AllDisjointProperties>
  <owl:members rdf:parseType="Collection">
    <owl:ObjectProperty rdf:about="&example;P1"/>
    <owl:ObjectProperty rdf:about="&example;P2"/>
    <owl:ObjectProperty rdf:about="&example;P3"/>
  </owl:members>
</owl:AllDisjointProperties>
```
OWL2: Property Characteristics

• \(P_1\) rdf:type owl:ReflexiveProperty
  – For an instance \(x\), and reflexive property \(P\), then \((x,x)\) is an instance of property \(P\)

\[
<\text{owl:ReflexiveProperty} \ \text{rdf:about="&example;sameAgeAs"}>
\]

• \(P_1\) rdf:type owl:IrreflexiveProperty
  – For an instance \(x\), and irreflexive property \(P\), then \((x,x)\) is not an instance of property \(P\)

\[
<\text{owl:IrreflexiveProperty} \ \text{rdf:about="&example;strictlyTallerThan"}>
\]
OWL2: Property Characteristics

• $P_1$ rdf:type owl:AsymmetricProperty
  – For an instance $(x,y)$ of an asymmetric property $P$, then $(y,x)$ is not an instance of property $P$

```xml
<owl:AsymmetricProperty rdf:about="&example;strictlyTallerThan"/>
```

• $P_1$ owl:propertyDisjointWith $P_2$
  – For an instance $(x,y)$ of property $P_1$, then $(x,y)$ cannot be an instance of property $P_2$

```xml
<owl:ObjectProperty rdf:about="&example;connectedTo">
  <owl:propertyDisjointWith rdf:resource="&example;contiguousWith"/>
</owl:ObjectProperty>
```

```xml
<owl:AsymmetricProperty rdf:about="&example;strictlyTallerThan"/>
```
OWL2: Self Restriction

- **owl:hasSelf**: Defines a class of individuals which are related to themselves through a specific property

```
<owl:Class rdf:about="http://dbpedia.org#MusiciansCommittedSuicide">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://dbpedia.org#killed"/>
      <owl:hasSelf rdf:resource="&xsd;boolean"/>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```
OWL2: Quantified Cardinality Restrictions

- OWL1 lets us specify the local range of a property or the number of values taken by the property
- OWL2 allows us to specify both

A full sized orchestra has at least 70 and at most 100 members

```
<owl:Class about="http://dbpedia.org#FullSizedOrchestra">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://dbpedia.org#hasMembers"/>
      <owl:onClass rdf:resource="http://dbpedia.org#Person"/>
      <owl:minCardinality>70</owl:cardinality>
      <owl:maxCardinality>100</owl:cardinality>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

Similar construct can be used for datatype properties!
OWL2: Property Chain Axioms

- **Allow one to infer the existence of a property from a chain of properties**
  - If \((x,y)\) is an instance of property \(P_1\) and \((y,z)\) is an instance of property \(P_2\), then \((x,y)\) is an instance of property \(P_3\)

```
<rdf:Description rdf:about="isInfluencedBy">
  <owl:propertyChainAxiom rdf:parseType="Collection">
    <owl:ObjectProperty rdf:about="http://dbpedia.org#influencedBy"/>
    <owl:ObjectProperty rdf:about="http://dbpedia.org#influencedBy"/>
  </owl:propertyChainAxiom>
</rdf:Description>

<rdf:Description rdf:about="hasEnemy">
  <owl:propertyChainAxiom rdf:parseType="Collection">
    <owl:ObjectProperty rdf:about="http://dbpedia.org#hasEnemy"/>
    <owl:ObjectProperty rdf:about="http://dbpedia.org#hasFriend"/>
  </owl:propertyChainAxiom>
</rdf:Description>
```
OWL2: Property Chain Axioms

- *Arbitrary property chain axioms* may lead to *undecidability*
- *Restriction*: set of property chain axioms must be *regular*
  - There must be a *strict linear order on the properties*
  - Every property chain axiom has to have one of the following forms:

<table>
<thead>
<tr>
<th></th>
<th>Axiom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R(x,y) and R(y,z) then R(x,z)</td>
</tr>
<tr>
<td>2</td>
<td>R(y1,y2) and S1(y3,y3) and ... Sn (yn-1,yn) then R(y1,yn)</td>
</tr>
<tr>
<td>3</td>
<td>S1(y1,y2) and S2(y2,y3) and ... Sn(yn-1,yn) then R(y1, yn)</td>
</tr>
<tr>
<td>4</td>
<td>S1(y1,y2) and S2(y2,y3) and ... Sn(yn-1,yn) then R(y1, yn)</td>
</tr>
<tr>
<td>5</td>
<td>S(y1,y2) and S(y2,y1) then R(y1,y2)</td>
</tr>
</tbody>
</table>
OWL2: Property Chain Axioms

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R(x,y) and R(y,z) then R(x,z)</td>
</tr>
<tr>
<td>2</td>
<td>R(y1,y2) and S1(y3,y3) and ... Sn (yn-1,yn) then R(y1,yn)</td>
</tr>
<tr>
<td>3</td>
<td>S1(y1,y2) and S2(y2,y3) and ... Sn(yn-1,yn) then R(y1, yn)</td>
</tr>
<tr>
<td>4</td>
<td>S1(y1,y2) and S2(y2,y3) and ... Sn(yn-1,yn) then R(y1, yn)</td>
</tr>
<tr>
<td>5</td>
<td>S(y1,y2) and S(y2,y1) then R(y1,y2)</td>
</tr>
</tbody>
</table>

- **Example (1):**
  - R(y1,y2) and R(y2,y3) then R(y1,y3)
  - S(y1,y2) and S(y2,y3) then S(y1,y3)
  - R(y1,y2) and S(y2,y3) and R(y3,y4) then T(y1,y4)

- **Example (2):**
  - R(y1,y2) and T(y2,y3) and S(y3,y4) then T(y1,y4)

- **Example (3):**
  - R(y1,y2) and S(y2,y3) then S(y1,y3)
  - S(y1,y2) and R(y2,y3) then R(y1,y3)

Round Order: S < R < T

Does not comply to a form

No regular order exists
OWL2: Property Chain Axioms

- **Combining property chain axioms** and **cardinality constraints** may lead to **undecidability**
- **Restriction**: use only simple properties in cardinality expressions (i.e., those that cannot be directly or indirectly inferred from property chains)
- Technically:
  - For any **property chain axiom** $S_1(y_1,y_2)$ and $S_2(y_2,y_3)$ and ... $S_n(y_{n-1},y_n)$ then $R(y_1, y_n)$ then $n > 1$, then **$R$ is not a simple property**
  - For any **sub-property chain axiom** $S$, then **$R$ is not a simple property**
  - **All other properties are simple**
- Example
  - $Q(y_1,y_2)$ and $P(y_2,y_3)$ then $R(y_1,y_3)$
  - $R(y_1,y_2)$ and $P(y_2,y_3)$ then $R(y_1,y_3)$
  - $R$ rdfs:subPropertyOf $S$
  - $P$ rdfs:subPropertyOf $R$
  - $Q$ rdfs:subPropertyOf $S$
  - Non-simple $R$, $S$
  - Simple $P$, $Q$
Data Integration in OWL2

• Practical problem: given ontologies from different sources, which identifiers refer to the same individuals?

• Typical approaches in OWL:
  – Explicitly specify equality (owl:sameAs)
  – Use inverse functional properties ("same values → same individual")

• Problems:
  – equality requires explicit mappings (rare on the Web)
  – OWL DL disallows inverse functional datatype properties (complicated interplay with datatype definitions!)
  – Only one property used globally for identification, no property combinations (Example: “All participants in a music album with the same name and birthday are the same.”)
Data Integration in OWL2

• OWL2 provides a way to model keys!
  – “All participants in a music album with the same name and birthday are the same.”
  – Expressed in the form of keys \( \text{owl:hasKey} \)
• Restriction: Keys apply only to named individuals – objects of the interpretation domain to which a constant symbol refers

```xml
<owl:Class rdf:about="Person">
  <owl:hasKey rdf:parseType="Collection">
    <owl:ObjectProperty rdf:about="hasSSN">
    </owl:hasKey>
  </owl:hasKey>
</owl:Class>
```
OWL Reasoning

- A reasoner makes use of the information asserted in the ontology.
- Based on the semantics described, a reasoner can help us to discover inferences that are a consequence of the knowledge that we’ve presented that we weren’t aware of beforehand.
- Is this new knowledge?
  - What’s actually in the ontology?
OWL Reasoning

• Subsumption reasoning
  – Allows us to infer when one class is a subclass of another
  – B is a subclass of A if it is necessarily the case that (in all models), all instances of B must be instances of A.
  – This can be either due to an explicit assertion, or through some inference process based on an intentional definition.
  – Can then build concept hierarchies representing the taxonomy.
  – This is classification of classes.

• Satisfiability reasoning
  – Tells us when a concept is unsatisfiable
    • i.e. when there is no model in which the interpretation of the class is non-empty.
  – Allows us to check whether our model is consistent.
Reasoning

• Reasoning can be used as a design support tool
  – Check logical consistency of classes
  – Compute implicit class hierarchy

• May be less important in small local ontologies
  – Can still be useful tool for design and maintenance
  – Much more important with larger ontologies/multiple authors

• Valuable tool for integrating and sharing ontologies
  – Use definitions/axioms to establish inter-ontology relationships
  – Check for consistency and (unexpected) implied relationships

• For most DLs, the basic inference problems are decidable (e.g. there is some program that solves the problem in a finite number of steps)
Acknowledgments

• Content from
  – Sean Bechhofer: School of Computer Science, University of Manchester, UK Available at http://www.cs.manchester.ac.uk
  – Dr Nicholas Gibbins: Semantic Web in Depth: Web Ontology Language (OWL), University of Southampton
  – Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph: Knowledge Representation for the Semantic Web (Part I: OWL 2)
  – W3C, OWL Features: Available at http://www.w3.org/TR/owl-features/