OWL
Web Ontology Language

Mustafa Jarrar

Birzeit University

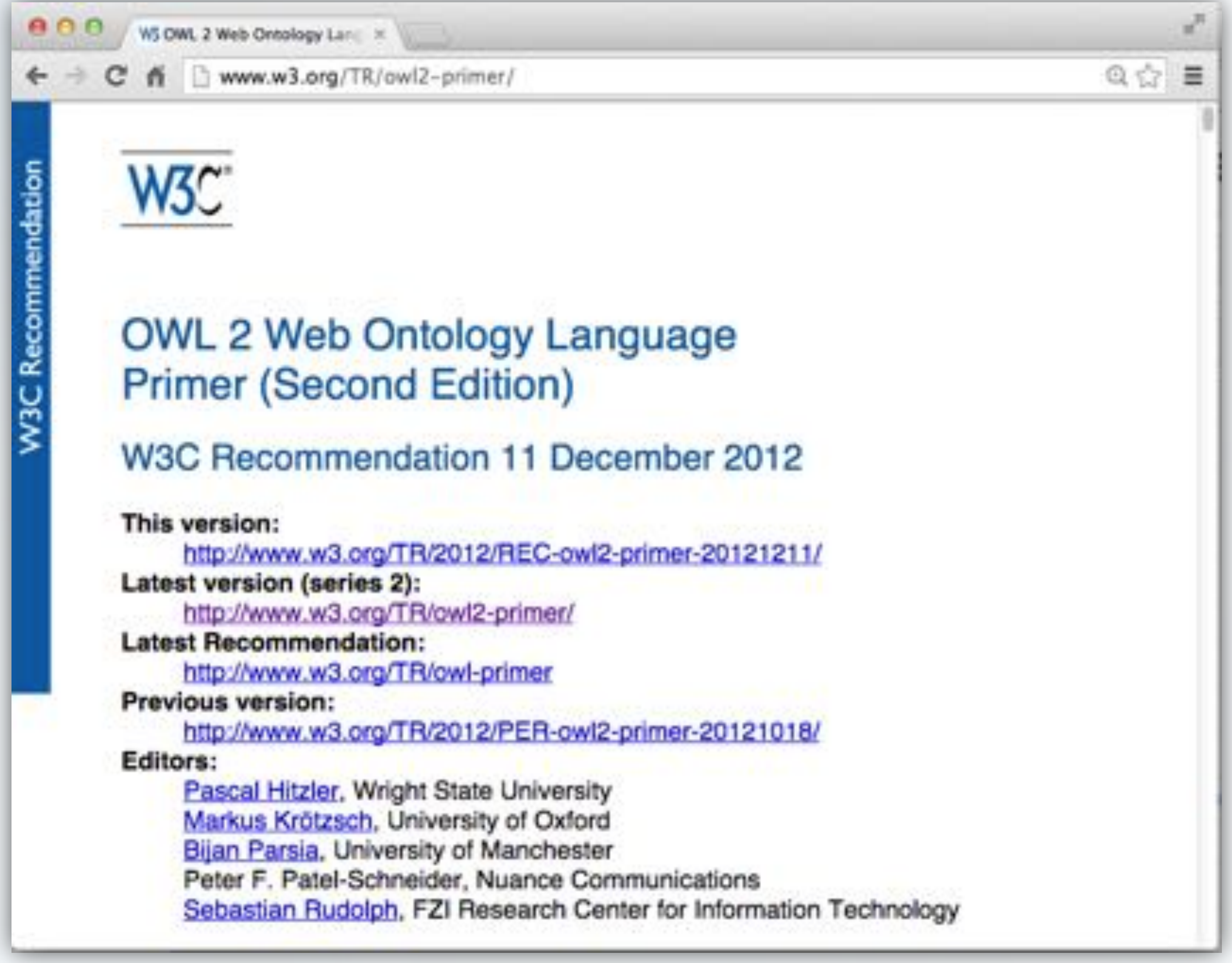
Watch this lecture and download the slides

Course Page: http://www.jarrar.info/courses/AI/

More Online Courses at: http://www.jarrar.info

Thanks to Anton Deik and Rami Hodrob for their help in preparing these slides
Most information based on [1]

Reading



OWL 2 Web Ontology Language
Primer (Second Edition)

W3C Recommendation 11 December 2012

This version:
http://www.w3.org/TR/2012/REC-owl2-primer-20121211/
Latest version (series 2):
http://www.w3.org/TR/owl2-primer/
Latest Recommendation:
http://www.w3.org/TR/owl-primer
Previous version:
http://www.w3.org/TR/2012/PER-owl2-primer-20121018/
Editors:
Pascal Hitzler, Wright State University
Markus Krötzsch, University of Oxford
Bijan Parsia, University of Manchester
Peter F. Patel-Schneider, Nuance Communications
Sebastian Rudolph, FZI Research Center for Information Technology

http://www.w3.org/TR/owl2-primer/

OWL
Web Ontology Language

In this lecture:

What is OWL?

OWL stands for Web Ontology Language.

OWL became a W3C standard in February 2004. OWL 2, and **OWL2** in 27 October 2009.

OWL is part of the "Semantic Web Vision" – The next generation of the Web (Web 3.0).

To describe the meaning/semantics of the Web data (RDF is written in terms of OWL)

OWL is called an ontology language. That is, what we specify in OWL is called an ontology (Ontology is about the exact description of things and their relationships.) For the web, ontology is about the exact description of web information and relationships between them.

OWL's underpinning description logic is SHOIQ.

OWL versus RDFS and RDF

OWL and RDFS are much of the same thing, but OWL allow constrains and rules.

OWL can be seen as an extension of RDF/RDFS, OWL comes with a larger vocabulary and stronger syntax than RDF and RDFS.

OWL has three sublanguages (OWL full, OWL DL, OWL Lite).

Variations of OWL

OWL Lite

- A subset of OWL DL
- This part of OWL is easier for reasoning

OWL DL

- The part of OWL Full that is in the Description Logic framework.
- Known to have decidable reasoning.

OWL Full

- maximum expressiveness
- no computational guarantees (not undecidable reasoning)

OWL 2

- OWL 2, became a W3C recommendation in 27 October 2009.
- An ontology language for the semantic Web which is extended from OWL 1 and empowered by new features.
- OWL 2 DL underpinning description logic is SROIQ.
- OWL 2 is supported by several semantic reasoners such as RacerPro 2, Pellet, Hermit and FaCT++.

Versions of OWL 2

OWL 2 EL

- Allows for subclass axioms with intersection, existential quantification, top, bottom and closed classes <u>with only one member</u>.
- Disallow for negation, disjunction, arbitrary universal quantification, role inverses.

OWL 2 QL

- Allows for subproperties, domain, range and subclass statements.
- Disallow for closed classes.

OWL 2 RL

- Allows for all axiom types, cardinality restrictions(only ≤1 and ≤0 on right hand side) and closed classes with only one member.
- Disallow certain constructors(universal and negation on left hand side and extensional and union on right hand side of subclass statements).

OWL

Web Ontology Language

In this lecture:

- ❑ Part 1: **Introduction to OWL**
- ❑ Part 2: **OWL Basics (Classes, Individuals, Properties)**
- ❑ Part 3: **Class Expression Axioms**
- ❑ Part 4: **Property Axioms**
- ❑ Part 5: **Assertions**
- ❑ Part 6: **Class Expressions -Propositional Connectives and Enumeration**
- ❑ Part 7: **Class Expressions -Property Restrictions**
- ❑ Part 8: **Class Expressions -Cardinality Restrictions**

Classes

Owl:Class

Unary predicates: Person(__), City(__).

User-defined classes which are subclasses of root class owl:Thing. A class may contain individuals, which are instances of the class, and other subclasses.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<owl:ontology rdf:about="http://www.example.com">
   <rdfs:comment> comments  here</rdfs:comment>
</owl:ontology>
      <owl:Class rdf:about="Person"/>
      <owl:Class rdf:about="City"/>
      <owl:Class rdf:about="Country"/>
</rdf:RDF>
```

➔ Similar to Object-Type in ORM, Entity in EER, or Class in UML

Individuals

- Constants: Edward_Said, Palestine.
- An individual is a member/instance of a class

```
<Man rdf:about=“Edward_Said" />
<Country rdf:about=“Palestine" />
<City rdf:about=“Jerusalem" />
```

→Notice that there is no way in ORM/UML/EER to add instance of concepts as art of the schema.

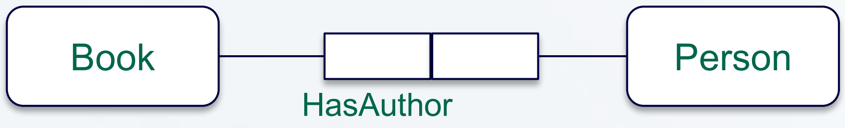
Properties (ObjectProperty)

owl:ObjectProperty

- A relation between instances of two classes.
- Binary predicates (hasAuthor(__, __)).

```
<owl:ObjectProperty rdf:about="hasAuthor">
        <rdfs:domain rdf:resource="# Book"/>
        <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
```

→ Similar to a binary relation in ORM between two entity types.



But unlike ORM, property may not have domain and range, these are used for reasoning.

Properties (DatatypeProperty)

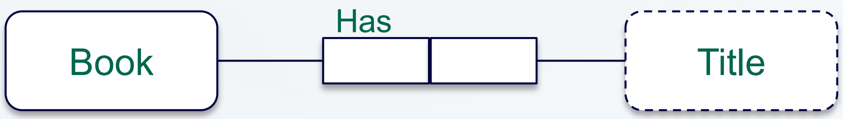
owl:DataTypeProperty

- Relation between instances of classes and values.
- The range of a literal datatype (use XML Schema datatypes).

```
<owl:DatatypeProperty rdf:about=“hasTitle">
  <rdfs:domain rdf:resource=“Book" />
  <rdfs:range rdf:resource=" xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about=“hasPrice">
  <rdfs:domain rdf:resource=“Book" />
  <rdfs:range rdf:resource="xsd:integer"/>
</owl:DatatypeProperty>
```

➔ Similar to a binary relation in ORM between an Entity and a Value.



Using Properties (Assertions)

Examples of using instances of properties

```
 <rdf:Description rdf:about=” http://en.wikipedia.org/wiki/Orientalism">
     <hasAuther rdf:resource="http://en.wikipedia.org/wiki/Edward_Said"/>
     <hasTitle rdf:datatype=”xsd:string">Orientalism</hasTitle>
     <hasPrice rdf:datatype=”xsd:integer">51</hasPrice>
</rdf:Description>
```

```
<rdf:Description rdf:about="John">
   <hasWife rdf:resource="Mary"/>
 </rdf:Description>
```

Special Classes and Properties

- **Top** class: T

  owl:Thing

  contains all individuals of the domain.

- **Bottom** class: ⊥

  owl:Nothing

  contains no individuals(empty class).

- **Universal** property: U

  owl:topObjectProperty

  links every individual to every individual.

OWL

Web Ontology Language

In this lecture:

Subclass Axioms

rdfs:subClassOf

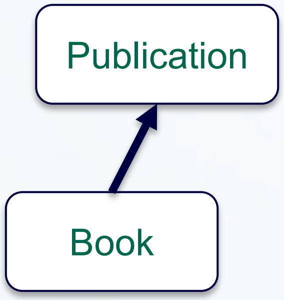
Notice that rdfs:subClassOf is part of RDFS, not OWL. This to show that OWL extends RDF/RDFS.

SubClassOf(*a:Man a:Person*) means that Each Man is a Person.

```
<owl:Class rdf:about="Man">
      <rdfs:subClassOf rdf:resource="Person"/>
</owl:Class>

<owl:Class rdf:about="Book">
      <rdfs:subClassOf rdf:resource="Publication" />
</owl:Class>
```

➔ Similar to the SubType relation ORM, UML and EER



Equivalent Classes

owl:equivalentClass

- States equivalence (i.e., class extension) of two named classes.

- Useful when integrating or mapping between two different ontologies.

```
<owl:Class rdf:about=“Palestine_President">
   <owl:equivalentClass rdf:resource=“PrincipalResidentOfPlestine"/>
</owl:Class>
```

- A class axiom may contain (multiple) owl:equivalentClass statements.
- The classes Being, Human and Person are semantically equivalent to each other.

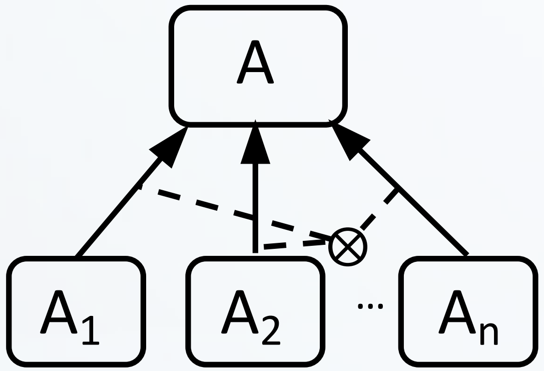
```
<owl:Class rdf:about=“Being">
    <owl:equivalentClass rdf:resource=“Human"/>
    <owl:equivalentClass rdf:resource=“Person"/>
</owl:Class>
```

Disjoint Classes

owl:AllDisjointClasses

- To state that these classes have no individuals in common.
- AllDisjointClasses (C1 ... Cn): all of the classes Ci, 1 ≤ i ≤ n, are pairwise disjoint; that is, no individual can be at the same time an instance of both Ci and Cj for i ≠ j.

```
<owl:AllDisjointClasses>
  <owl:members rdf:parseType="Collection">
    <owl:Class rdf:about="Woman"/>
    <owl:Class rdf:about="Man"/>
  </owl:members>
</owl:AllDisjointClasses>
```



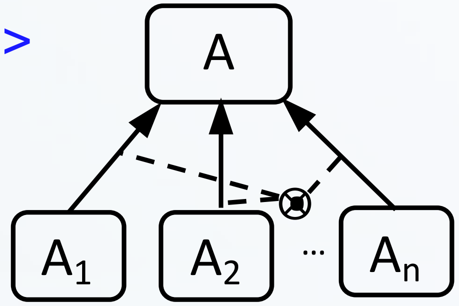
Disjoint Union of Class Expressions

owl:disjointUnionOf

- DisjointUnion (C C1 ... Cn): a class C is a disjoint union of the classes Ci, 1 ≤ i ≤ n, all of which are pairwise disjoint the extensions of all Ci exactly cover the extension of C.

- DisjointUnion (Person Female Male) means each Person is either a Male or a Female, each Female is a Person, each Male is a Person, and nothing can be both a Female and a Male.

```
<owl:Class rdf:about="#Person">
   <owl:disjointUnionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="#Female"/>
      <rdf:Description rdf:about="#Male"/>
   </owl:disjointUnionOf>
```

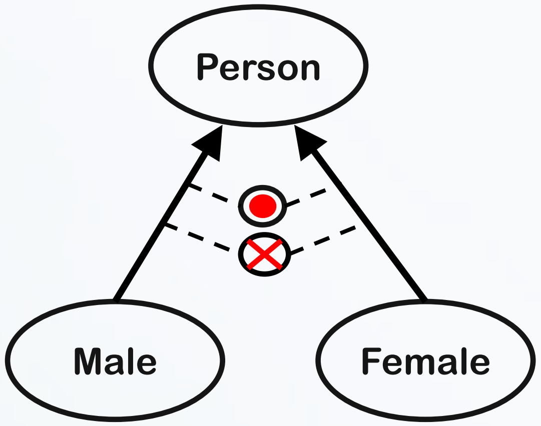


Example1

- Here we illustrate a simple example represented in ORM and OWL 2.
- In the example, the illustrated OWL 2 constructs are Class, Subclass, Union of Classes and Disjoint Classes.

```
<owl:Class rdf:about="&Example1:Person“/>
<owl:Class rdf:about="&Example1:Female">
    <rdfs:subClassOf rdf:resource="&Example1:Person"/>
</owl:Class>
 <owl:Class rdf:about="&Example1:Male">
    <rdfs:subClassOf rdf:resource="&Example1:Person"/>
 </owl:Class>

  <owl:Class rdf:about="&Example1:Person">
    <owl:equivalentClass>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="&Example1:Female"/>
                <rdf:Description rdf:about="&Example1:Male"/>
            </owl:unionOf>
        </owl:Class>
     </owl:equivalentClass>
  </owl:Class>
<owl:AllDisjointClasses>
    <owl:members rdf:parseType="Collection">
        <owl:Class rdf:about=“Female"/>
        <owl:Class rdf:about="Male"/>
    </owl:members>
```



OWL
Web Ontology Language

In this lecture:

Object Subproperties

rdfs:subPropertyOf

Properties, like classes, can be arranged in a hierarchy.

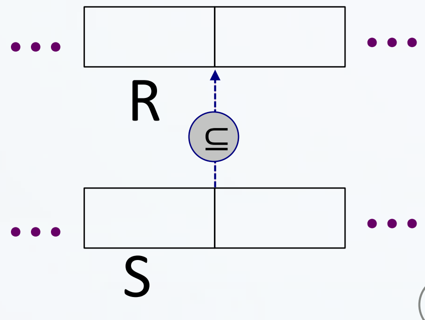
SubPropertyOf (OP1 OP2): if an individual x is connected by OP1 to an individual y, then x is also connected by OP2 to y.

The rdfs:subPropertyOf means that anything with a Borrows property with value X also has a ChecksOut property with value X.

```
<owl:ObjectProperty rdf:about="Borrows">
        <rdfs:domain rdf:resource="Person"/>
        <rdfs:range rdf:resource="Book"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="ChecksOut">
        <rdfs:subPropertyOf rdf:resource="Borrows" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="hasWife">
         <rdfs:subPropertyOf rdf:resource="hasSpouse"/>
 </owl:ObjectProperty>
```



➔Similar to the Subset constraint in ORM.

Equivalent Properties

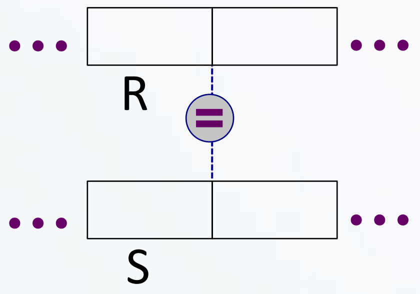
owl:equivalentProperty

- EquivalentObjectProperties (OP_1 ... OP_n): all of the object properties OP_i, $1 \leq i \leq n$, are semantically equivalent to each other.

- EquivalentObjectProperties (*a:hasBrother* *a:hasMaleSibling*) means that having a brother is the same as having a male sibling

```
<owl:ObjectProperty rdf:about="hasChild">
  <owl:equivalentProperty rdf:resource=”x:child"/>
 </owl:ObjectProperty>

 <owl:DatatypeProperty rdf:about="hasAge">
  <owl:equivalentProperty rdf:resource=”y:age"/>
 </owl:DatatypeProperty>
```

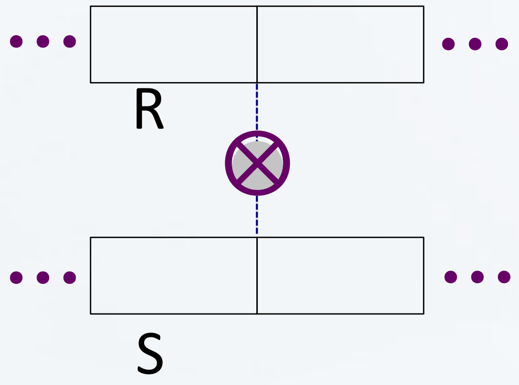


Disjoint Properties

owl:propertyDisjointWith

- DisjointObjectProperties (OP_1 ... OP_n): all of the object properties OP_i, $1 \leq i \leq n$, are pairwise disjoint; that is, no individual *x* can be connected to an individual *y* by both OP_i and OP_j for $i \neq j$.
- DisjointObjectProperties(*a:hasFather a:hasMother*) means that Fatherhood is disjoint with motherhood.

```
<rdf:Description rdf:about="hasFather">
   <owl:propertyDisjointWith rdf:resource="hasMother"/>
 </rdf:Description>
```

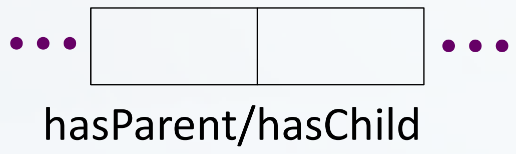


Inverse Object Properties

owl:inverseOf

- If a property P is tagged as the owl:inverseOf Q, then for all x and y: P(x,y) iff Q(y,x).
- Note that the syntax for owl:inverseOf takes a property name as an argument. A iff B means (A implies B) and (B implies A).
- InverseObjectProperties (OP_1 OP_2): the object property OP_1 is an inverse of the object property OP_2, that is if an individual x is connected by OP_1 to an individual y, then y is also connected by OP_2 to x, and vice versa.

```
<owl:ObjectProperty rdf:about="hasParent">
  <owl:inverseOf rdf:resource="hasChild"/>
 </owl:ObjectProperty>
```

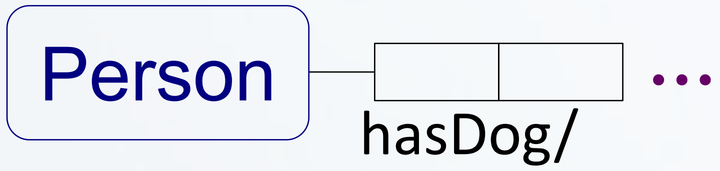


Property Domain

rdfs:domain

- ObjectPropertyDomain (OP C): the domain of the object property OP is the class C, that is, if an individual x is connected by OP with some other individual, then x is an instance of C.
- ObjectPropertyDomain(*a:hasDog a:Person*) means that only people can own dogs.

```
<owl:ObjectProperty rdf:about="hasDog">
    <rdfs:domain rdf:resource="Person"/>
    …
</owl:ObjectProperty>
```

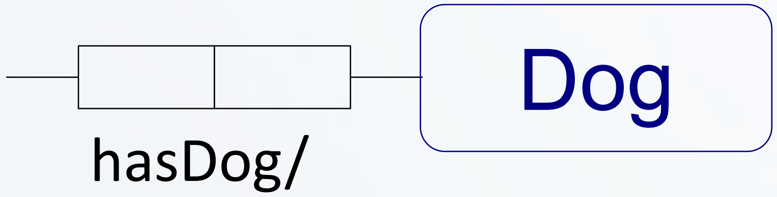


Property Range

rdfs:range

- ObjectPropertyRange(OP C): the range of the object property OP is the class C , that is, if some individual is connected by OP with an individual x, then x is an instance of C.
- ObjectPropertyRange(*a:hasDog a:Dog*) means that the range of the *a:hasDog* property is the class *a:Dog*.

```
<owl:ObjectProperty rdf:about="#hasDog">
    …
    <rdfs:range rdf:resource="#Dog"/>
 </owl:ObjectProperty>
```

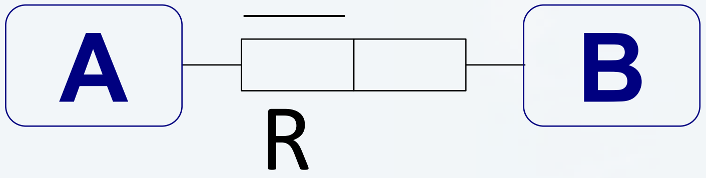


Functional Object Property

Owl:FunctionalProperty

- FunctionalObjectProperty (OP): the object property OP is functional, that is, for each individual x, there can be at most one distinct individual y such that x is connected by OP to y.
- FunctionalObjectProperty (*a:hasFather*) means that each individual can have at most one father.

```
<owl:FunctionalProperty rdf:about="#hasFather"/>
```

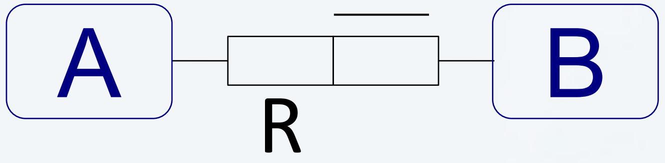


Inverse-Functional Object Property

Owl:InverseFunctionalProperty

- InverseFunctionalObjectProperty (OPI): the object property OPI is inverse-functional, that is, for each individual x, there can be at most one individual y such that y is connected by OP with x.
- InverseFunctionalObjectProperty (*a:fatherOf*) means that each object can have at most one father.

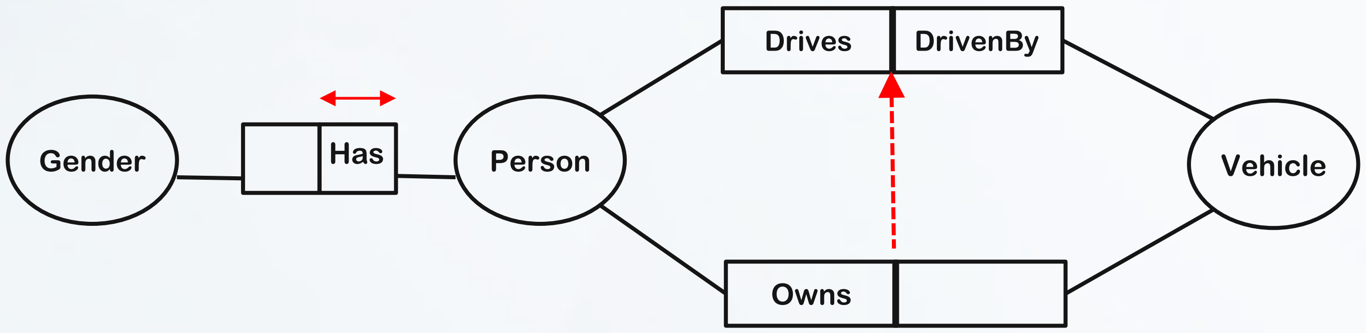
<owl:InverseFunctionalProperty rdf:about=“#FatherOf"/>



There is no inverse for Datatype properties

OWL2 Example2

- Here we illustrate a simple example represented in ORM and OWL 2.
- In the example, the illustrated OWL 2 constructs are Object Subproperties, Inverse Object Properties, Object Property Domain, Object Property Range and Functional Object Properties.



```
 <owl:Class rdf:about="&OWL2Example2:Person"/>
<owl:Class rdf:about="&OWL2Example2:Vehicle"/>
<owl:Class rdf:about="&OWL2Example2:Gender"/>

<owl:ObjectProperty rdf:about="&OWL2Example2:Drives">
       <rdfs:domain rdf:resource="&OWL2Example2:Person"/>
       <rdfs:range rdf:resource="&OWL2Example2:Vehicle"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&DrivenBy">
      <owl:inverseOf rdf:resource="OWL2Example2:Drives"/>
 </owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="&OWL2Example2:Owns">
       <rdfs:domain rdf:resource="&OWL2Example2:Person"/>
       <rdfs:range rdf:resource="&OWL2Example2:Vehicle"/>
 </owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&OWL2Example2:Owns">
     <rdfs:subPropertyOf rdf:resource="&OWL2Example2:Drives"/>
 </owl:ObjectProperty>

<owl:FunctionalProperty rdf:about="&OWL2Example2:Has/>
```

OWL
Web Ontology Language

In this lecture:

Individual Equality

owl:sameAs

- **Identity between Individuals**
- This mechanism is similar to that for classes, but declares two individuals to be identical.
- SameIndividual (a1 ... an): all of the individuals ai, 1 ≤ i ≤ n, are equal to each other(synonymes).

```
<rdf:Description rdf:about="http://www.amazon.com/Orientalism-Edward-W-Said/dp/039474067X">
 <owl:sameAs rdf:resource="http://en.wikipedia.org/wiki/Orientalism"/>
</rdf:Description>
```

```
<rdf:Description rdf:about="x:Edward_Said">
 <owl:sameAs rdf:resource="# E_Said"/>
</rdf:Description>
```

Individual Inequality

owl:differentFrom

- An **owl:differentFrom** statement indicates that two URI references refer to different individuals.

```
<rdf:Description rdf:about="http:/amazon.com/Orientalism-Edward-W-Said/dp/039474067X">
    <owl:differentFrom rdf:resource="http://www.youtube.com/watch?v=X3iA73JXIFo"/>
</rdf:Description>
```

```
<rdf:Description rdf:about="John">
    <owl:differentFrom rdf:resource="Bill"/>
</rdf:Description>
```

OWL
Web Ontology Language

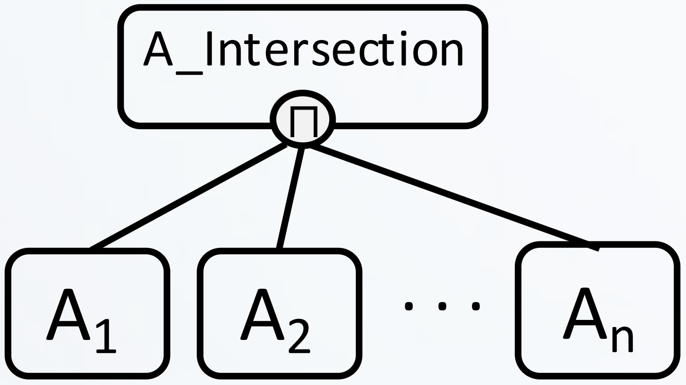
In this lecture:

Intersection of Class Expressions

owl:intersectionOf

Describes a class for which the class extension contains precisely those individuals that are members of the class extension of all class descriptions in the list.

```
<owl:Class rdf:about="Mother">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="Woman"/>
        <owl:Class rdf:about="Parent"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```



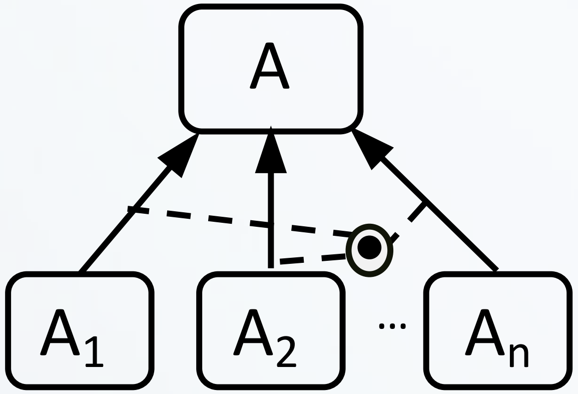
Proposed Notation

Union of Class Expressions

owl:unionOf

The union of two classes contains every individual which is contained in at least one of these classes. Therefore we could characterize the class of all parents as the union of the classes Mother and Father:

```
<owl:Class rdf:about="Parent">
   <owl:equivalentClass>
     <owl:Class>
       <owl:unionOf rdf:parseType="Collection">
         <owl:Class rdf:about="Mother"/>
         <owl:Class rdf:about="Father"/>
       </owl:unionOf>
     </owl:Class>
   </owl:equivalentClass>
 </owl:Class>
```

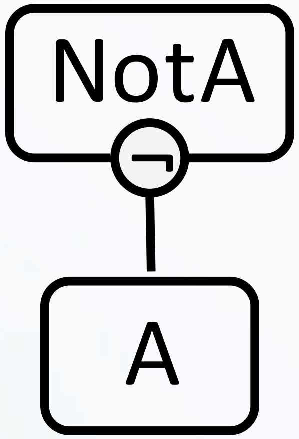


Complement of Class Expressions

owl:complementOf

The complement of a class corresponds to logical negation: it consists of exactly those objects which are not members of the class itself. The following definition of childless persons uses the class complement and also demonstrates that class constructors can be nested:

```
<owl:Class rdf:about="ChildlessPerson">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="Person"/>
        <owl:Class>
          <owl:complementOf rdf:resource="Parent"/>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```



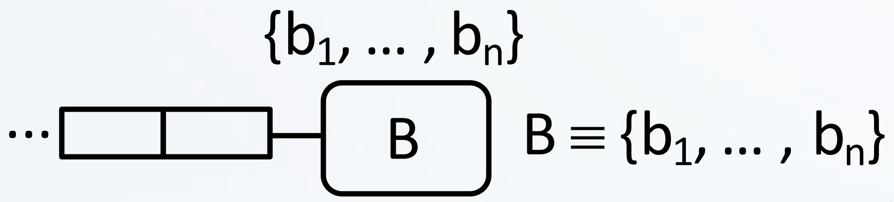
Proposed Notation

Enumeration of Individuals

owl:one of

- A way to describe a class is just to enumerate all its instances.
- These instances are the only members of PalestinianCities. Classes defined this way are sometimes referred to as closed classes or enumerated sets. We cannot have more, or less instances.

```
<owl:Class rdf:about=”PalestinianCities">
 <owl:equivalentClass>
  <owl:Class>
   <owl:oneOf rdf:parseType="Collection">
    <rdf:Description rdf:about="Jerusalem "/>
    <rdf:Description rdf:about="Aka"/>
    <rdf:Description rdf:about=”Ramallah"/>
     …
   </owl:oneOf>
  </owl:Class>
 </owl:equivalentClass>
</owl:Class>
```



OWL

Web Ontology Language

In this lecture:

- ❑ Part 1: **Introduction to OWL**
- ❑ Part 2: **OWL Basics**
- ❑ Part 3: **Class Expression Axioms**
- ❑ Part 4: **Property Axioms**
- ❑ Part 5: **Assertions**
- ❑ Part 6: **Class Expressions Propositional Connectives and Enumeration**
- ❑ Part 7: **Class Expressions -Property Restrictions**
- ❑ Part 8: **Class Expressions -Cardinality Restrictions**

Class Expressions -Property Restrictions

- ❑ Existential Quantification
- ❑ Universal Quantification
- ❑ Individual Value Restriction
- ❑ Self-Restriction

Object Property Restrictions- Existential Quantification

owl:someValuesFrom

- Defines a class as **the set of all individuals that are connected via a particular property** to another individual which is an instance of a certain class.
- Each Account must have at least one Owner that is a Person.

```
<owl:Class rdf:about=“Account">
<owl:equivalentClass>
   <owl:Restriction>
        <owl:onProperty rdf:resource="#HasOwner" />
        <owl:someValuesFrom rdf:resource="#Person" />
    </owl:Restriction>
</owl:equivalentClass>
</owl:Class>
```

Object Property Restrictions- Universal Quantification

owl:allValuesFrom

- To describe a class of individuals for which all related individuals must be instances of a given class
- For every Account, if they have owners, all the owners must be Persons.

```
<owl:Class rdf:about="Account">
 <owl:equivalentClass>
    <owl:Restriction>
         <owl:onProperty rdf:resource="#HasOwner" />
         <owl:allValuesFrom rdf:resource="#Person" />
     </owl:Restriction>
 </owl:equivalentClass>
</owl:Class>
```

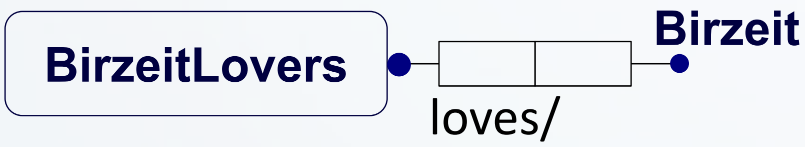
Object Property Restrictions -Individual Value Restriction

owl:hasValue

ObjectHasValue(OP a): consists of an object property OP and an individual a, and it contains all those individuals that are connected by OP to a. For example, we can create a class “BirzeitLovers” to include all those who love Birzeit.

```
<owl:Class rdf:about=“BirzeitLovers">
  <owl:equivalentClass>
      <owl:Restriction>
          <owl:onProperty rdf:resource=”#Loves"/>
          <owl:hasValue rdf:resource="#Birzeit"/>
     </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

individual



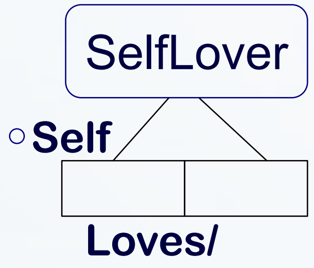
(Proposed Notation)

Object Property Restrictions- Self Restriction

owl:hasSelf

- ObjectHasSelf (OP): a class expression that contains all individuals that are connected to themselves via OPE
- ObjectHasSelf (*a:loves*) contains those individuals that love themselves.

```
<owl:Class rdf:about="SelfLover">
   <owl:equivalentClass>
        <owl:Restriction>
             <owl:onProperty rdf:resource="#loves"/>
                 <owl:hasSelf rdf:datatype="&xsd;boolean">true</owl:hasSelf>
        </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```



(Proposed Notation)

OWL
Web Ontology Language

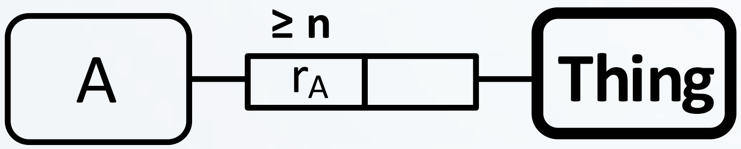
In this lecture:

Class Expressions – Min Cardinality

Owl:minCardinality

- OWL allows us to place some restrictions on properties, such as owl:minCardinality.
- Each student must be EnrolledIn in 3 things, **at least**

```
<owl:Class rdf:about="Student">
   <rdfs:subClassOf>
        <owl:Restriction>
               <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger"> 3
               </owl:minCardinality>
               <owl:onProperty rdf:resource="#EnrolledIn"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

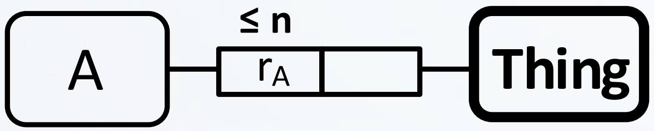


Class Expressions - Max Cardinality

Owl:maxCardinality

- OWL allows us to place some restrictions on properties, such as owl:maxCardinality.
- Each student must be EnrolledIn in 6 things, **at most**

```
<owl:Class rdf:about="Student">
   <rdfs:subClassOf>
        <owl:Restriction>
               <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger"> 6
               </owl:MaxCardinality>
               <owl:onProperty rdf:resource="EnrolledIn"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

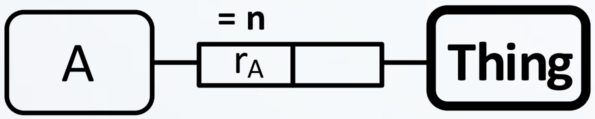


Class Expressions – Exact Cardinality

Owl:Cardinality

- OWL allows us to place some restrictions on properties, such as owl:Cardinality.
- Each student must be EnrolledIn in 5 things, **exactly**

```
<owl:Class rdf:about="Student">
   <rdfs:subClassOf>
       <owl:Restriction>
             <owl:Cardinality rdf:datatype="&xsd;nonNegativeInteger"> 5
             </owl:Cardinality>
             <owl:onProperty rdf:resource="EnrolledIn"/>
       </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

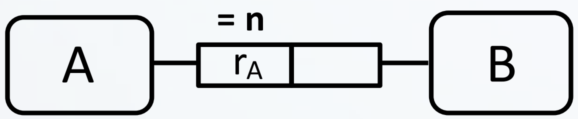


Class Expressions - Qualified Cardinality

owl:qualifiedCardinality

- Same as owl:Cardinality, but we restrict the property with its range.
- Each student must be EnrolledIn in 5 courses, **exactly**

```
<owl:Class rdf:about="“Student">
   <rdfs:subClassOf>
       <owl:Restriction>
           <owl:qualifiedCardinality rdf:datatype="&xsd;nonNegativeInteger“> 5
           </owl:qualifiedCardinality>
           <owl:onProperty rdf:resource=”EnrolledIn"/>
           <owl:onClass rdf:resource=”Course"/>
       </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```



Similarly, there is **minQualifiedCardinality** and **maxQualifiedCardinality**

XML Datatypes

xsd:string
xsd:normalizedString
xsd:boolean
xsd:decimal
xsd:float
xsd:double
xsd:integer
xsd:nonNegativeInteger
xsd:positiveInteger
xsd:nonPositiveInteger
xsd:negativeInteger
xsd:long

xsd:int
xsd:short
xsd:byte
xsd:unsignedLong
xsd:unsignedInt
xsd:unsignedShort
xsd:unsignedByte
xsd:hexBinary
xsd:base64Binary
xsd:dateTime
xsd:time
xsd:date

xsd:gYearMonth
xsd:gYear
xsd:gMonthDay
xsd:gDay
xsd:gMonth
xsd:anyURI
xsd:token
xsd:language
xsd:NMTOKEN
xsd:Name
xsd:NCName

The above datatypes, plus rdfs:Literal, form the built-in OWL datatypes.

References

[1] Mustafa Jarrar: Lecture Notes on RDF Data Model, Birzeit University, 2018

[2] Mustafa Jarrar: Lecture Notes on RDF Schema, Birzeit University, 2018

[3] Mustafa Jarrar: Lecture Notes on Ontology Web Language (OWL) Birzeit University, 2018

[4] Mustafa Jarrar: Lecture Notes on Data Web and Linked Data, Birzeit University, 2018

[5] Mustafa Jarrar, Anton Deik: The Graph Signature: A Scalable Query Optimization Index For RDF Graph Databases Using Bisimulation And Trace Equivalence Summarization. International Journal on Semantic Web and Information Systems, 11(2), 36-65, 2015

[6] Mustafa Jarrar, Anton Deik, Bilal Faraj: Ontology-Based Data And Process Governance Framework -The Case Of E-Government Interoperability In Palestine . In pre-proceedings of the IFIP International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA'11). Pages(83-98). 2011.

[7] Anton Deik, Bilal Faraj, Ala Hawash, Mustafa Jarrar: Towards Query Optimization For The Data Web - Two Disk-Based Algorithms: Trace Equivalence And Bisimilarity . In proceedings of the International Conference on Intelligent Semantic Web – Applications and Services. Pages 131-137. ACM. 2010.

[8] Rami Hodrob, Mustafa Jarrar: ORM To OWL 2 DL Mapping.. In proceedings of the International Conference on Intelligent Semantic Web – Applications and Services. Pages 131-137. ACM, 2010.

[9] Mustafa Jarrar: Towards Automated Reasoning On ORM Schemes. -Mapping ORM Into The DLR_idf Description Logic. In proceedings of the 26th International Conference on Conceptual Modeling (ER 2007). Pages (181-197). LNCS 4801, Springer. Auckland, New Zealand. 2007

[10] Mustafa Jarrar: Towards Methodological Principles For Ontology Engineering . PhD Thesis. Vrije Universiteit Brussel. (May 2005)