Semantic Web

Lecture 9: OWL & Protégé

Knarig Arabshian
knarig@cs.columbia.edu

Role Restrictions

- Another type of logic-based constructors for complex classes
- Role restrictions are constructors involving roles
- Universal Quantifier
  - First role restriction is derived from universal quantifier in predicate logic
  - Defines a class as the set of all object for which the given role only attains values from given class
  - owl:allValuesFrom role restriction

Role Restrictions

```
<owl:Class rdf:about=”Exam”>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource=”hasExaminer”/>
            <owl:allValuesFrom rdf:resource=”Professor”/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

- Examiners must always be professors
- All examiners of an exam must be professors

Role Restrictions

- Existential Quantifier
  - Declare that any exam must have at least one examiner
  - Role restriction owl:someValuesFrom is used

```
<owl:Class rdf:about=”Exam”>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource=”hasExaminer”/>
            <owl:someValuesFrom rdf:resource=”Person”/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Role Restrictions

- owl:allValuesFrom
  - We can say something about all of the examiners
- owl:someValuesFrom
  - We can say something about at least one of the examiners

Example

```
<Person rdf:about="anton">
  <likesToWorkWith rdf:resource="doris"/>
  <likesToWorkWith rdf:resource="dagmar"/>
</Person>
<Person rdf:about="doris">
  <likesToWorkWith rdf:resource="dagmar"/>
  <likesToWorkWith rdf:resource="bernd"/>
</Person>
<Person rdf:about="gustav">
  <likesToWorkWith rdf:resource="bernd"/>
  <likesToWorkWith rdf:resource="doris"/>
  <likesToWorkWith rdf:resource="desiree"/>
</Person>
<Person rdf:about="charles"/>
<owl:Class rdf:about="FemaleColleagues">
  <owl:oneOf rdf:parseType="Collection">
        <Person rdf:about="dagmar"/>
        <Person rdf:about="doris"/>
        <Person rdf:about="desiree"/>
  </owl:oneOf>
</owl:Class>
```

```
<owl:AllDifferent>
   <owl:distinctMembers rdf:parseType="Collection">
        <Person rdf:about="anton"/>
        <Person rdf:about="bernd"/>
        <Person rdf:about="charles"/>
        <Person rdf:about="dagmar"/>
        <Person rdf:about="doris"/>
        <Person rdf:about="desiree"/>
   </owl:distinctMembers>
 </owl:AllDifferent>

<owl:Class rdf:about="Class1">
   <owl:equivalentClass>
        <owl:Restriction>
         <owl:onProperty rdf:resource="likeToWorkWith"/>
         <owl:someValuesFrom
 rdf:resource="FemaleColleagues"/>
        </owl:Restriction>
   </owl:equivalentClass>
 </owl:Class>
```

Open World Assumption

- Infer that anton, doris and gustav are in Class1 since they all like to work with one of the individuals that are in the FemaleColleagues class
- Can not infer that charles is in Class1 because the individual has not indicated what the value of the likesToWorkWith property is
- Reason for this is the Open World Assumption
  - Implicitly assumed that a knowledge base may always be incomplete
  - Charles could be in the relation likesToWorkWith an instance of the Female Colleagues class but this is not known yet

Open World Assumption

- Easily lead to mistakes in the modeling of knowledge
- Other paradigms, like databases, have a Closed World Assumption
  - Knowledge base is considered to be complete
  - With a CWA, inference is that charles is NOT in Class1 since it has not bee asserted
- OWA is reasonable for Semantic Web because the WWW is expanding and new knowledge is added all the time

Open World Assumption

- Impacts on other situations

<Professor rdf:about=”rudiStuder”/>

<Philosopher rdf:about=”mikeStange”/>

- Can not infer anything about the membership or non-membership of mikeStange in the class Professor because further knowledge may not yet be known to us
  - mikeStange may or may not be a professor

Open World Assumption

```
<owl:Class rdf:about="Class2">
        <owl:equivalentClass>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="likesToWorkWith"/>
                        <owl:allValuesFrom rdf:resource="FemaleColleagues"/>
                </owl:Restriction>
        </owl:equivalentClass>
</owl:Class>
```

- Infer that doris and gustav do NOT belong to Class2
- Because of OWA, can NOT say anything about the membership of anton or charles in Class2

Open World Assumption

```
<owl:Class rdf:about="Class3">
        <owl:equivalentClass>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="likesToWorkWith"/>
                        <owl:hasValue rdf:resource="doris"/>
                </owl:Restriction>
        </owl:equivalentClass>
</owl:Class>
```

- Here, you can infer that anton and gustav belong to Class3 because both like to work with doris

Open World Assumption

```
<owl:Class rdf:about="Class4">
        <owl:equivalentClass>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="likesToWorkWith"/>
                        <owl:minCardinality
rdf:datatype="&xsd;nonNegativeInteger">
                                3
                        </owl:minCardinality>
                </owl:Restriction>
        </owl:equivalentClass>
</owl:Class>
```

- What can we infer with this?

Open World Assumption

```
<owl:Class rdf:about="Class5">
        <owl:equivalentClass>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="likesToWorkWith"/>
                        <owl:maxCardinality
rdf:datatype="&xsd;nonNegativeInteger">
                                0
                        </owl:minCardinality>
                </owl:Restriction>
        </owl:equivalentClass>
</owl:Class>
```

- Due to OWA we cannot infer that charles is in Class5

Open World Assumption

```
<owl:Class rdf:about="Class5">
        <owl:equivalentClass>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="likesToWorkWith"/>
                        <owl:allValuesFrom rdf:resource="&owl;Nothing">
                </owl:Restriction>
        </owl:equivalentClass>
</owl:Class>
```

- Alternate definition of Class5

Role Relationships

```
<owl:ObjectProperty rdf:about="hasExaminer">
        <rdfs:subPropertyOf rdf:resource="hasParticipant"/>
</owl:ObjectProperty>
```

- Examiners of an event are also present at the event
- Can also state that tw roles are equivalent to one another with owl:equivalentProperty

Role Relationships

```
<Exam rdf:about="semanticWebExam">
        <hasExaminer rdf:resource="rudiStuder"/>
</Exam>
<owl:ObjectProperty rdf:about="hasExaminer">
        <rdfs:subPropertyOf rdf:resource="hasParticipant"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="hasParticipant">
        <rdfs:equivalentPropertyOf rdf:resource="hasAttendee"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="hasAttendee">
        <rdfs:inversePropertyOf rdf:resource="participatesIn"/>
</owl:ObjectProperty>
```

Role Relationships

```
<Exam rdf:about="semanticWebExam">
        <hasExaminer rdf:resource="rudiStuder"/>
</Exam>

<owl:ObjectProperty rdf:about="hasExaminer">
        <rdfs:subPropertyOf rdf:resource="hasParticipant"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="hasParticipant">
        <owl:equivalentProperty rdf:resource="hasAttendee"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="hasAttendee">
        <owl:inverseOf rdf:resource="participatesIn"/>
</owl:ObjectProperty>
```

- semanticWebExam and rudiStuder are in the relation has Participant and also in the equivalent relation hasAttendee
- Infer that rudiStuder and semanticWebExam are in the relation participatesIn

Examples

- Class Vegetable is subclass of PizzaTopping
- Class PizzaToping does not share any elements with the class Pizza
- Individual aubergine is an element of the class Vegetable
- Abstract role hasTopping is only used for relationships between elements of the classes Pizza and PizzaTopping
- The class VegPizza consists of those elements which are in the class NoMeatPizza and in the class NoFishPizza
- The role hasTopping is a subrole of hasIngredient

Examples

- Every pizza has at least two toppings
- Every pizza has tomato as a topping