

Week 11 Suggested Tutorial Solution: OWL

Semester 1, 2006

Consider the rental accommodation exchange from the week 2 tutorial and the representation in the solution to the week 4 tutorial and following, as represented in RDFS in the last tutorial.

Elaborate a significant portion of the ontology in OWL using the OWL facilities from the lecture.

1. Simple Properties

- owl:ObjectProperty – relations between instances of two classes. For example, property *implements* is the relations between instances of class *Estate Agent* and *Lease* shown on figure 1.

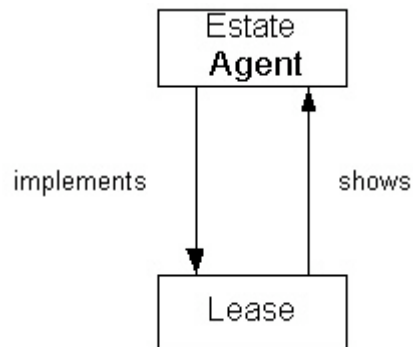


Figure 1

```
<owl:ObjectProperty rdf:ID="implements">
  <rdfs:domain rdf:resource="#Estate Agent"/>
  <rdfs:range rdf:resource="#Lease"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="shows">
  <rdfs:domain rdf:resource="#Lease"/>
  <rdfs:range rdf:resource="#Estate Agent"/>
</owl:ObjectProperty>
```

- owl:DatatypeProperty – relations between instances of classes and RDF literal and XML Schema datatypes. For example, property *hasLeaseNumber* is a relation between an instance of class *Lease* and a literal – string shown on figure 2.

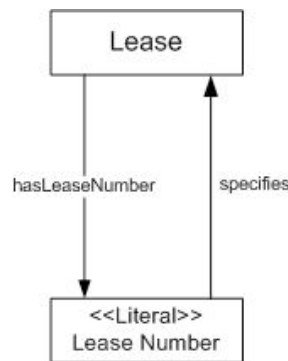


Figure 2

```

<owl:Class rdf:ID="Lease" />
<owl:DatatypeProperty rdf:ID="hasLeaseNumber">
  <rdfs:domain rdf:resource="#Lease" />
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

```

2. Property Characteristics

- owl:FunctionalProperty – on figure 1, property *shows* is functional because a particular instance of *Lease* has the name of a particular instance of *Estate Agent* printed on that lease, e.g. Ray White. Meanwhile, there could be at least one instance of *Lease* pointing to a particular instance of *Estate Agent*. Thus, the property like *shows* holding many (instances on its domain) to one (instance on its range) relationship is regarded as a functional property.

```

<owl:Class rdf:ID="Lease"/>
<owl:ObjectProperty rdf:ID="shows">
  <rdf:type rdf:resource="&owl:FunctionalProperty" />
  <rdfs:domain rdf:resource="#Lease" />
  <rdfs:range rdf:resource="#Estate Agent" />
</owl:ObjectProperty>

```

- owl:InverseFunctionalProperty – for example, on figure 1, obviously, property *implements* has a opposite direction of *shows*. It ends up that there is a one-to-many relationship between the domain and range of *implements*. Thus, *implements* is considered as a inverse functional property.

```

<owl:ObjectProperty rdf:ID="shows" />
<owl:ObjectProperty rdf:ID="implements">
  <rdf:type rdf:resource="&owl:InverseFunctionalProperty"/>
  <owl:inverseOf rdf:resource="#shows" />
</owl:ObjectProperty>

```

- owl:InverseOf – for example, on figure 2, property *specifies* is inverse of *hasLeaseNumber* because each instance of domain of the former – *Lease Number*, i.e. the range of the latter, has a particular instance of range of the former – *Lease*, i.e. the domain of the latter. Furthermore, they are

functional property and have one-to-one relationships between their domains and ranges.

```
<owl:ObjectProperty rdf:ID="hasLeaseNumber">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="specifies">
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="specifies">
  <owl:inverseOf rdf:resource="#hasLeaseNumber" />
</owl:ObjectProperty>
```

- owl:SymmetricProperty - for example, Mary and John, both of whom are lessees, are sharing a house. We can say either Mary is the housemate of John, or John is the housemate of Mary.

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

<Lessee rdf:ID="Mary">
  <housemate rdf:resource="#John" />
</Lessee>
```

- owl:TransitiveProperty - for example, a particular clause such as *rent consists of* a number of clauses such as *rent in advance*, in turn; the *rent in advance* also *consists of* several clauses such as *for a period agreement – 2 weeks rent*. Hence, the property *consists of* is transitive.

```
<owl:ObjectProperty rdf:ID="consistsOf">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdfs:domain rdf:resource="#Lease"/>
  <rdfs:range rdf:resource="#Clause" />
</owl:ObjectProperty>

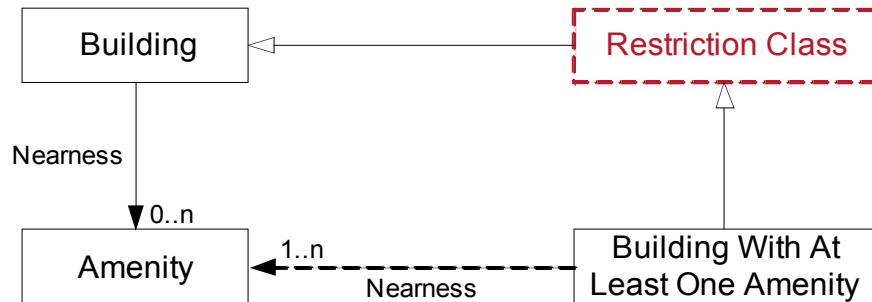
<Clause rdf:ID="rent">
  <consistsOf rdf:resource="#rent_in_advance" />
</Clause>

<Clause rdf:ID="rent_in_advance">
  <consistsOf rdf:resource
    ="#for_a_period_agreement-2-weeks_rent" />
</Clause >
```

In this case, clause *for_a_period_agreement—2-week rent* is also of a clause on a *Lease*. A transitive property must have its domain and range intersecting. Simplest, if domain and range are the same, both are “clause”.

3. Property Restrictions – In addition to designating property characteristics, it is possible to further constrain the range of a property in specific contexts in a variety of ways.

- owl:minCardinality, for example, on figure 3, there could be some of instances of *nearness* domain – *Building*, which have at least one instance of *Amenity*.



Legend

Subsumption	◁
Property	◄
Subproperty	◄---

Figure 3

```

<owl:ObjectProperty rdf:ID="nearness">
  <rdfs:domain rdf:resource="#Building"/>
  <rdfs:range rdf:resource="#Amenity"/>
</owl:ObjectProperty>

<owl:Class rdf:ID =
  "Restriction_Class">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#nearness"/>
      <owl:minCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:minCardinality>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:ID=
  "Restriction_Class">
  <rdfs:subClassOf rdf:resource="#Building"/>
</owl:Class>

```

- owl:allValuesFrom – there could be a subset of instances of *Building*, which are close to bus stops alone on figure 4.

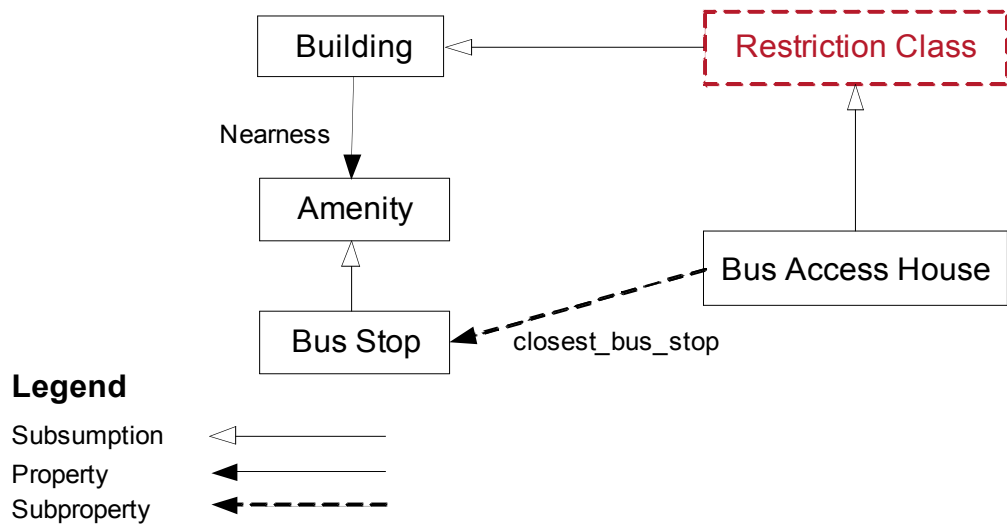


Figure 4

```

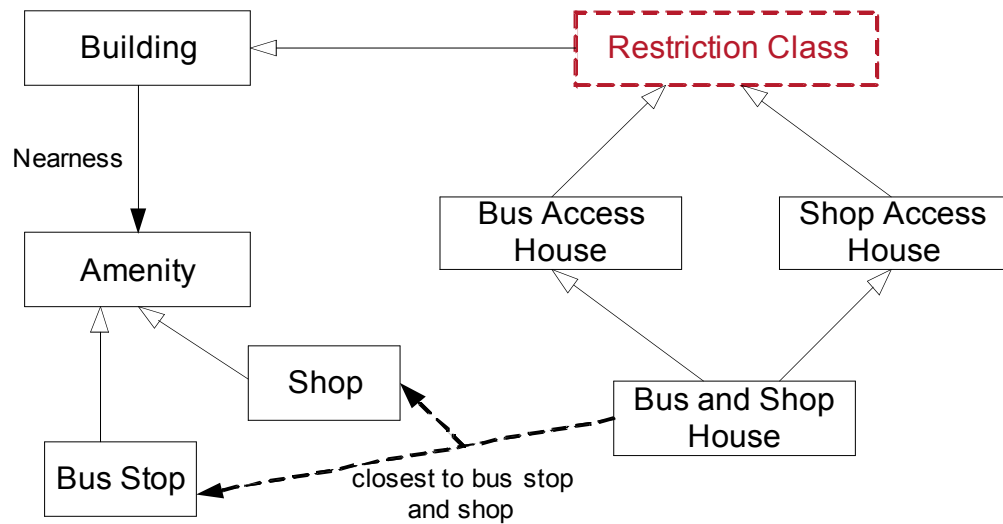
<owl:ObjectProperty rdf:ID="closest_bus_stop">
  <rdfs:subPropertyOf rdf:resource="#nearness" />
  <rdfs:range rdf:resource="#Bus_Stop" />
</owl:ObjectProperty>

<owl:Class rdf:ID=
  "Restriction_Class">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#closest_bus_stop" />
      <owl:allValuesFrom rdf:resource="#Bus_Stop" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:ID=
  " Restriction_Class ">
  <rdfs:subClassOf rdf:resource="#Building"/>
</owl:Class>

```

- owl:someValuesFrom, there are instances of building, which are close to both bus stops and shops simultaneously on figure 5.



Legend

Subsumption	◁
Property	→
Subproperty	- - - →

Figure 5

```

<owl:Class rdf:ID="Bus_Stop_and_Shop">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Bus_Stop" />
    <owl:Class rdf:about="#Shop" />
  </owl:unionOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="closest_bus_stop_and_shop">
  <rdfs:subPropertyOf rdf:resource="#nearness" />
  <rdfs:range rdf:resource="#Bus_Stop_and_Shop" />
</owl:ObjectProperty>

<owl:Class rdf:ID="Restriction_Class">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#closest_bus_stop_and_shop"/>
      <owl:someValuesFrom rdf:resource="#Bus_Stop" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:ID="Restriction_Class">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#closest_bus_stop_and_shop"/>
      <owl:someValuesFrom rdf:resource="#Shop" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
  
```

```

    </owl:equivalentClass>
  </owl:Class>

  <owl:Class rdf:ID=" Restriction_Class">
    <rdfs:subClassOf rdf:resource="#Building"/>
  </owl:Class>

  <owl:Class rdf:ID= " Restriction_Class ">
    <rdfs:subClassOf rdf:resource="#Building"/>
  </owl:Class>

```

- owl:hasValue, for example, there are instances of building which are only close to a particular bus stop such as bus stop number 18.

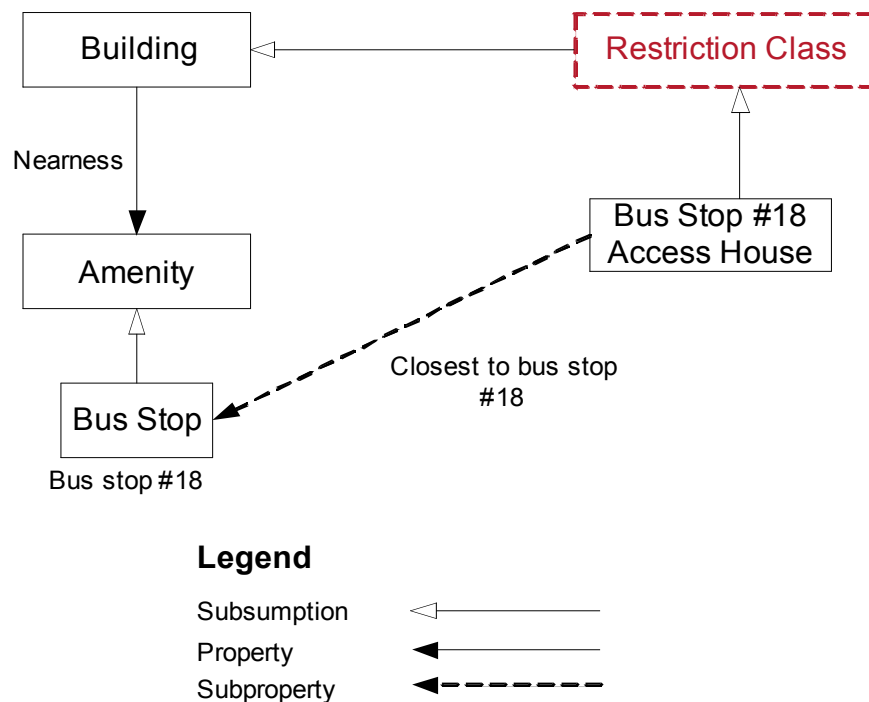


Figure 6

```

<owl:ObjectProperty rdf:ID="closest_bus_stop_18">
  <rdfs:subPropertyOf rdf:resource="#nearness" />
  <rdfs:range rdf:resource="#Bus_Stop" />
</owl:ObjectProperty>

<owl:Class rdf:ID="Restriction_Class">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#closest_bus_stop_18" />
      <owl:hasValue rdf:resource="#bus_stop#18" />
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>

<owl:Class rdf:ID="Restriction_Class">
  <rdfs:subClassOf rdf:resource="#Building"/>
</owl:Class>

```

4. Identity between individuals

- owl:sameAs, - two individuals may be stated to be the same in OWL. For example, the name, e.g. Ray White, and ABN such as 490589675432, of an estate agent refer to the same object.

```
<Estate_Agent rdf:ID="Ray_White">
  <owl:sameAs rdf:resource="#ABN_490589675432" />
</Estate_Agent>
```

5. Different Individuals

- owl:differentFrom - for example, Mary, a lessee is different from John who is a lessee as well.

```
<Lessee rdf:ID="Mary">
  <owl:differentFrom rdf:resource="#John"/>
</Lessee>
```

- owl:allDifferent - for example, Smith, Jane and Bill, all of whom are lessor, are different.

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
    <rental:Lessor rdf:about="#Smith" />
    <rental:Lessor rdf:about="#Jane" />
    <rental:Lessor rdf:about="#Bill" />
  </owl:distinctMembers>
</owl:AllDifferent>
```

6. Enumerated Class

- Regarding to oneOf, for example, there are eight states in Australia: they are QLD, NSW, ACT, TAS, VIC, SA, WA and NT.

```
<owl:Class rdf:ID="States_in_Australia">
  <owl:oneOf rdf:parseType="Collection">
    <owl:States_in_Australia rdf:about="#QLD"/>
    <owl:States_in_Australia rdf:about="#NSW"/>
    <owl:States_in_Australia rdf:about="#ACT"/>
    <owl:States_in_Australia rdf:about="#TAS"/>
    <owl:States_in_Australia rdf:about="#VIC"/>
    <owl:States_in_Australia rdf:about="#SA"/>
    <owl:States_in_Australia rdf:about="#WA"/>
    <owl:States_in_Australia rdf:about="#NT"/>
  </owl:oneOf>
</owl:Class>
```

7. Set Operator - OWL DL and OWL Full allow arbitrary Boolean combinations of classes and restrictions

- owl:IntersectionOf and owl:ComplementOf, for example, there are units except 2-bedroom ones.

```
<owl:Class rdf:ID="Non_2-Bedroom_Unit">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Unit"/>
    <owl:Class>
      <owl:complementOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#has" />
          <owl:hasValue rdf:resource="#2_bedroom" />
        </owl:Restriction>
      </owl:complementOf>
    </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

- owl:UnionOf, for example, find the leases that are implemented by Ray White and PRD respectively.

```
<owl:Class rdf:ID="Wanted_Lease">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class
      rdf:about="#Lease_implemented_by_Ray_White"/>
    <owl:Class
      rdf:about="#Lease_implemented_by_PRD " />
  </owl:unionOf>
</owl:Class>
```

8. Disjoint Class

- Regarding to owl:disjointWith, for units, there are studios, one-bedroom units and 2-bedroom units, all of which are disjoint.

```
rental:One-Bedroom_Unit    rdfs:subClassOf  rental:Unit
rental:Two-Bedroom_Unit   rdfs:subClassOf  rental:Unit
<owl:Class rdf:ID="One-Bedroom_Unit">
  <rdfs:subClassOf rdf:resource="#Unit"/>
</owl:Class>

<owl:Class rdf:ID="Two-Bedroom_Unit">
  <rdfs:subClassOf rdf:resource="#Unit"/>
</owl:Class>

<owl:Class rdf:ID="One-Bedroom_Unit">
  <owl:disjointWith
    rdf:resource="#Two-Bedroom_Unit"/>
</owl:Class>
```

Describe a concept you cannot represent in OWL (make a plausible extension if necessary).

OWL cannot represent the following concepts:-

- Causal relationship – the relationship of cause and effect. There are no syntaxes in OWL for representing that relationship. For example, a lessee violates the clauses on a lease such as owing rent for a long period, the estate agent implementing the lease complains RTA such violation. Hence, RTA puts the name of the lessee into a bad list. In this case, a single cause and effect relationship can be represented as follows

rental:violate	rdfs:domain	rental:Lessee
rental:violate	rdfs:range	rental:Lease

However, the further consequences as parts of the same causal chain cannot be done in OWL.

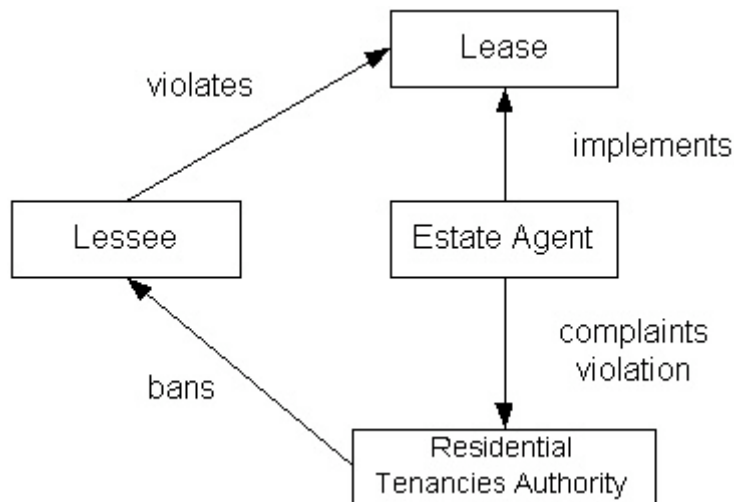


Figure 7

- Derived property cannot be represented as well since OWL does not support arithmetic. For example, the amount of a bond is set to be 4-week rent. However, there are no supports in OWL that the amount of a bond can be calculated by the amount of a weekly rent.

How would you use the facilities of OWL Full in this ontology (make a plausible extension if necessary)?

OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. There are three aspects discussed as follows: -

- **Class having property**
For example, *quantityOfLeasesImplemented* can be created for class *Lease* for indicating how many instances of *Lease* are currently managed.
- **Class having classes as instances.**
For example of *Bus*, obviously, the instances of the *Bus* are the route numbers of *Bus* such as route 411, 412 and 407. Actually, each route number can be regarded as a class, for instance of route 411, departure times in a particular stop can be regarded as instances of the *route 411*. In this case, *route 411* can be as an instance or class under different contexts.
- **Classes of properties**
For example, for committed lessees, there are a list of things to be done before those lessees move into units such as paying the bonds, signing the leases and taking keys. It is quite nature to have two subclasses: one indicates the lessees completing the whole list of things and other does not. In this case, allowing *classes of properties* would make it easy to fulfill such requirement.