COMP318

Ontologies and Semantic Web

OWL - Part 5

Dr Valentina Tamma

V.Tamma@liverpool.ac.uk

Where were we

- OWL preliminaries
- OWL class constructors

- `https://www.w3.org/TR/owl2-primer/`

OWL properties

- As in RDFS, there are two types of properties in OWL datatypes and object properties.
  - datatype properties relate objects to datatype values:
    - name, phoneNumber, age...
  - OWL does not have any predefined datatypes
    - but it allows users to use XML Schema data types
    - &xsd;nonNegativeInteger is an abbreviation for “http://www.w3.org/2001/XMLSchema#nonNegativeInteger”

```
<owl:DatatypeProperty rdf:ID=”age”>
  <rdfs:range rdf:resource=”
      &xsd;nonNegativeInteger”/>
</owl:DatatypeProperty>
```

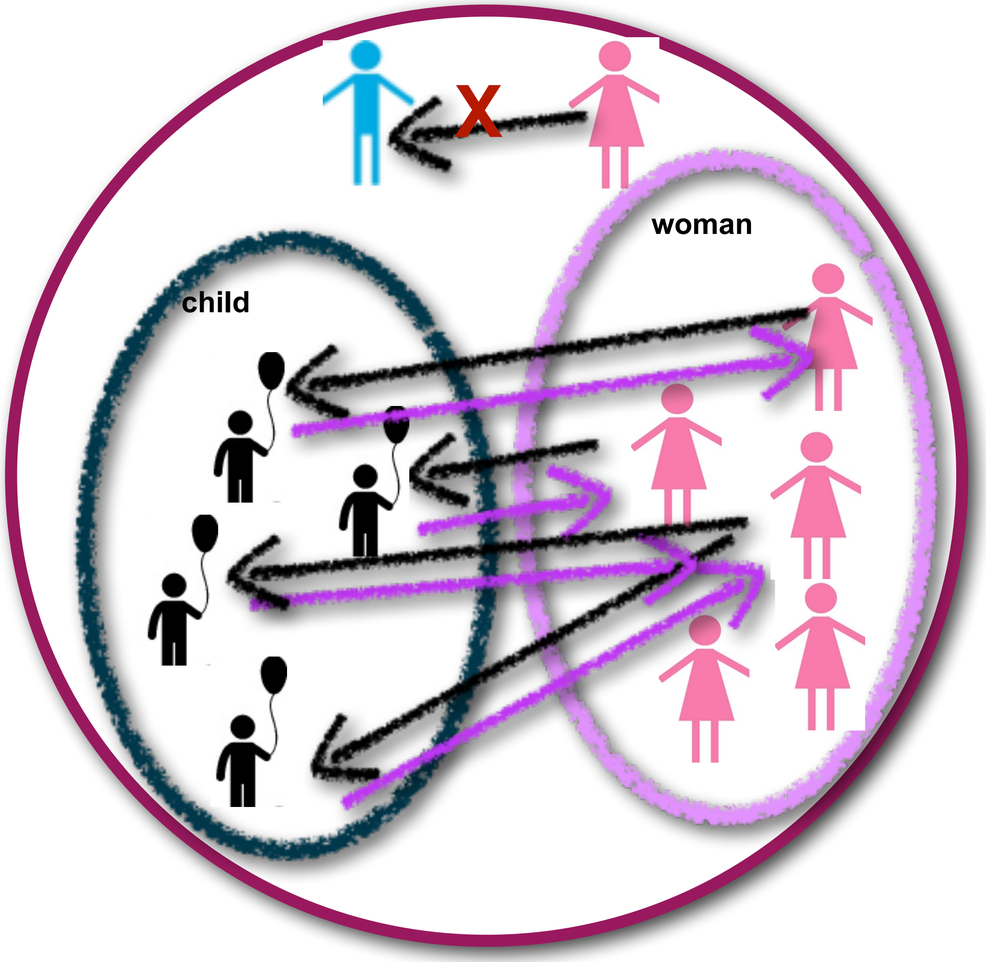
OWL properties

- ***Object*** properties.
  - relate objects to other objects
    - *marriedTo, father, spouse...*

```
<owl:ObjectProperty rdf:about=”#motherOf”>
  <rdfs:domain rdf:resource=”#woman”/>
  <rdfs:range rdf:resource=”#person”/>
  <rdfs:subPropertyOf rdf:resource=”#parentOf”/>
</owl:ObjectProperty>
```

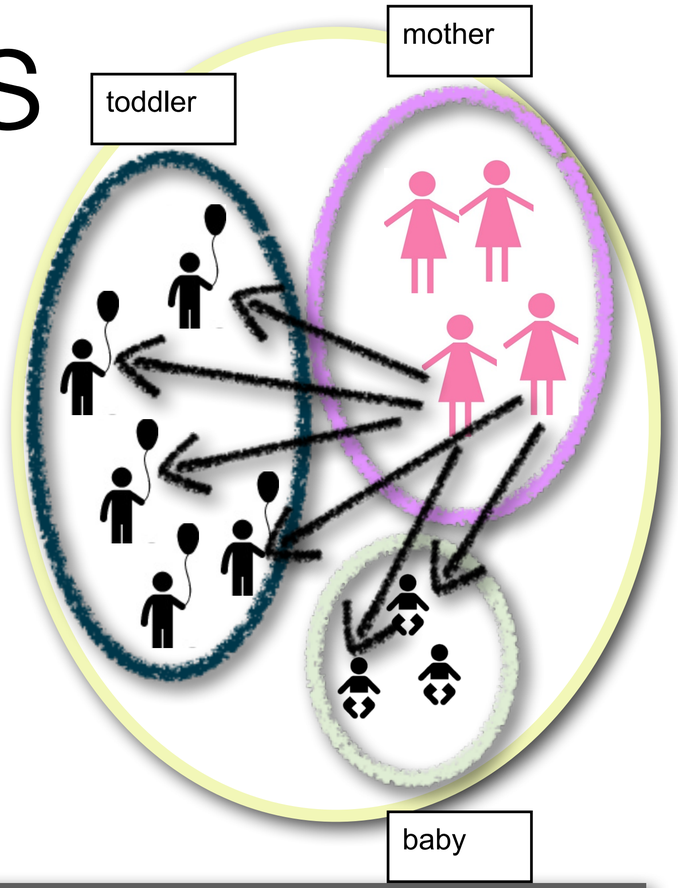
```
<owl:ObjectProperty rdf:about=”#childOf”>
  <rdfs:domain rdf:resource=”#person”/>
  <rdfs:range rdf:resource=”#woman”/>
  <owl:inverseOf rdf:resource=”#motherOf”/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about=”#offspringOf”>
  <owl:equivalentProperty rdf:resource=”#childOf”/>
</owl:ObjectProperty>
```



Property restrictions

- Restrictions allow us to build new classes from class, property and individual names
  - existential quantification:
    - define a class that consists of all objects for which there exist at least **one** toddler among the values of motherOf
    - the restriction defines an **anonymous** class with no ID and only local scope - it can only be used in the place the restriction appears



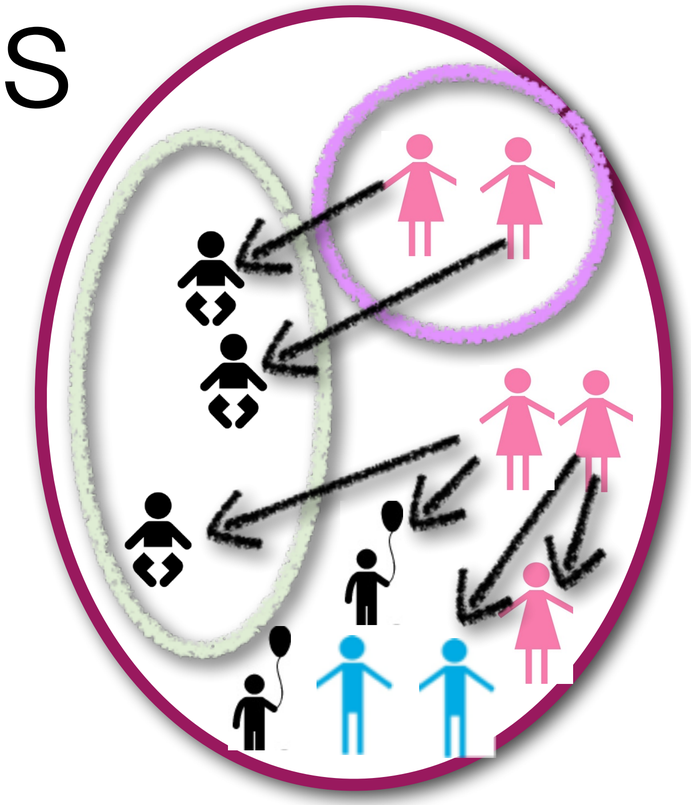

```
<owl:Class rdf:about=”#motherOfToddler”>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:Property rdf:resource=”#motherOf”/>
      <owl:someValuesFrom rdf:resource=”#toddler”/>
    </owl:Restriction>
  </<rdfs:subClassOf>
</owl:Class>
```

```
:motherOfToddler  rdf:type    owl:Class ;
    rdfs:subClassOf  [
      rdf:type          owl:Class ;
      rdf:type          owl:Restriction ;
      owl:onProperty      :motherOf ;
      owl:someValuesFrom   :toddler .]
```

```
Class: motherOfToddler SubClassOf: motherOf some toddler
```

Property restrictions

- Restrictions allow us to build new classes from class, property and individual names
  - universal quantification:
    - define a class that consists of all objects for which **all** values of motherOf are babies



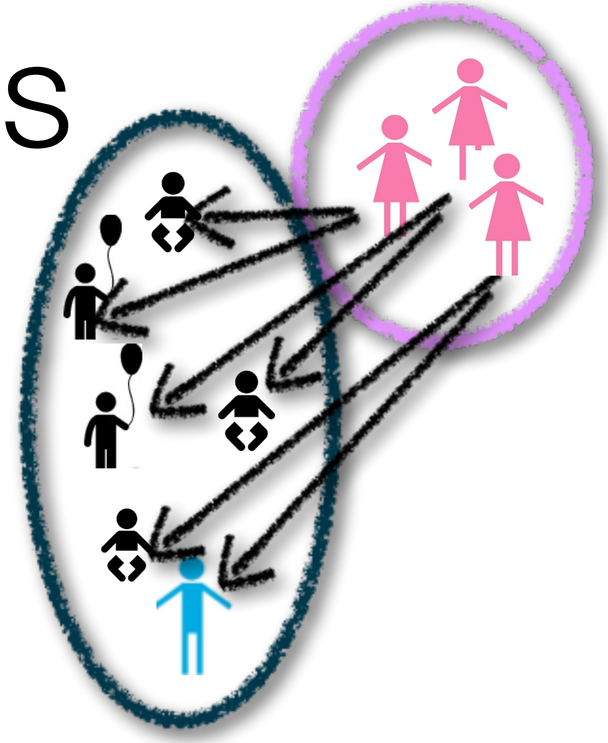
```
<owl:Class rdf:about="#motherOfBaby">
  <owl:subClassOf>
    <owl:Restriction>
      <owl:Property rdf:resource="#motherOf"/>
      <owl:allValuesFrom rdf:resource="#baby"/>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>
```

```
:motherOfBaby  rdf:type    owl:Class ;
    rdfs:subClassOf  [
      rdf:type          owl:Class ;
      rdf:type          owl:Restriction ;
      owl:onProperty      :motherOf ;
      owl:allValuesFrom   :Baby . ]
```

Class: motherOfBaby SubClassOf: motherOf all baby

Cardinality restrictions

- Restrictions allow us to build new classes from class, property and individual names
  - *min, max* and *exactly* cardinality restrictions

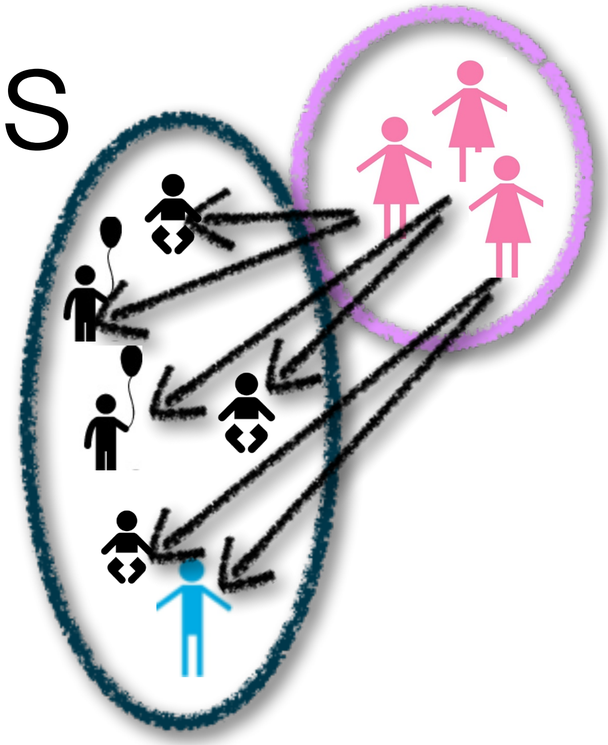


```
<owl:Class rdf:about=”#motherOfChildren”>
  <owl:subClassOf>
    <owl:Restriction>
      <owl:Property rdf:resource=”#motherOf”/>
      <owl:minQualifiedCardinality
          rdf:datatype=”&xsd;nonNegativeInteger”/> 2
      </owl:minQualifiedCardinality >
      <owl:Class rdf:resource=”#offspring”/>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>
```

```
Class: motherOfChildren SubClassOf: motherOf min 2
Offspring
```

```
:motherOfChildren  rdf:type    owl:Class ;
    rdfs:subClassOf  [
      rdf:type          owl:Class ;
      rdf:type          owl:Restriction ;
      owl:minQualifiedCardinality “2"^^&xsd:nonNegativeInteger ;
      owl:onProperty      :motherOf ;
      owl:onClass   :Offspring . ]
```

Cardinality restrictions



- Restrictions allow us to build new classes from class, property and individual names
  - *min*, *max* and *exactly* cardinality restrictions

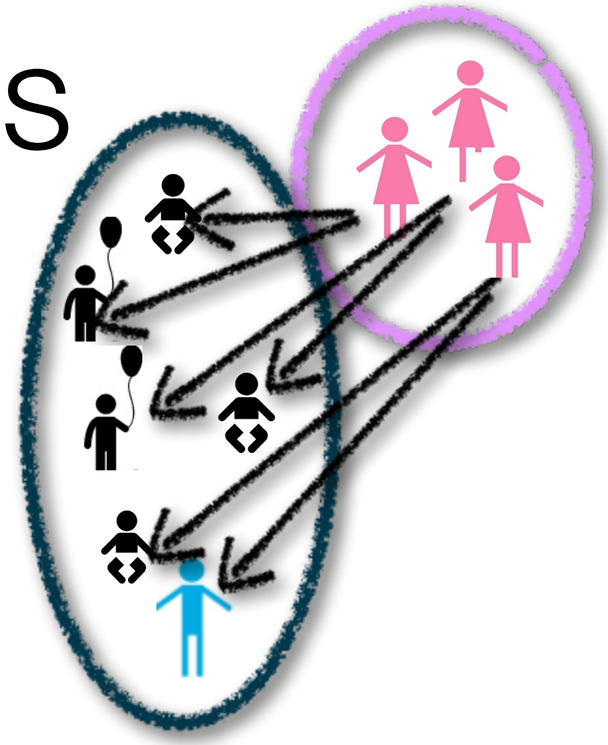
```
<owl:Class rdf:about="#motherOfChildren">
  <owl:subClassOf>
    <owl:Restriction>
      <owl:Property rdf:resource="#motherOf"/>
      <owl:maxQualifiedCardinality
          rdf:datatype="&xsd;nonNegativeInteger"/> 4
      </owl:maxQualifiedCardinality >
      <owl:Class rdf:resource="#offspring"/>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>
```

```
Class: motherOfChildren SubClassOf: motherOf max 4 Offspring
```

```
:motherOfChildren  rdf:type    owl:Class ;
    rdfs:subClassOf  [
      rdf:type          owl:Class ;
        rdf:type  owl:Restriction ;
        owl:maxQualifiedCardinality "4"^^&xsd:nonNegativeInteger;
        owl:onProperty      :motherOf ;
        owl:onClass   :Offspring . ]
```

Cardinality restrictions

- Restrictions allow us to build new classes from class, property and individual names
  - *min*, *max* and *exactly* cardinality restrictions



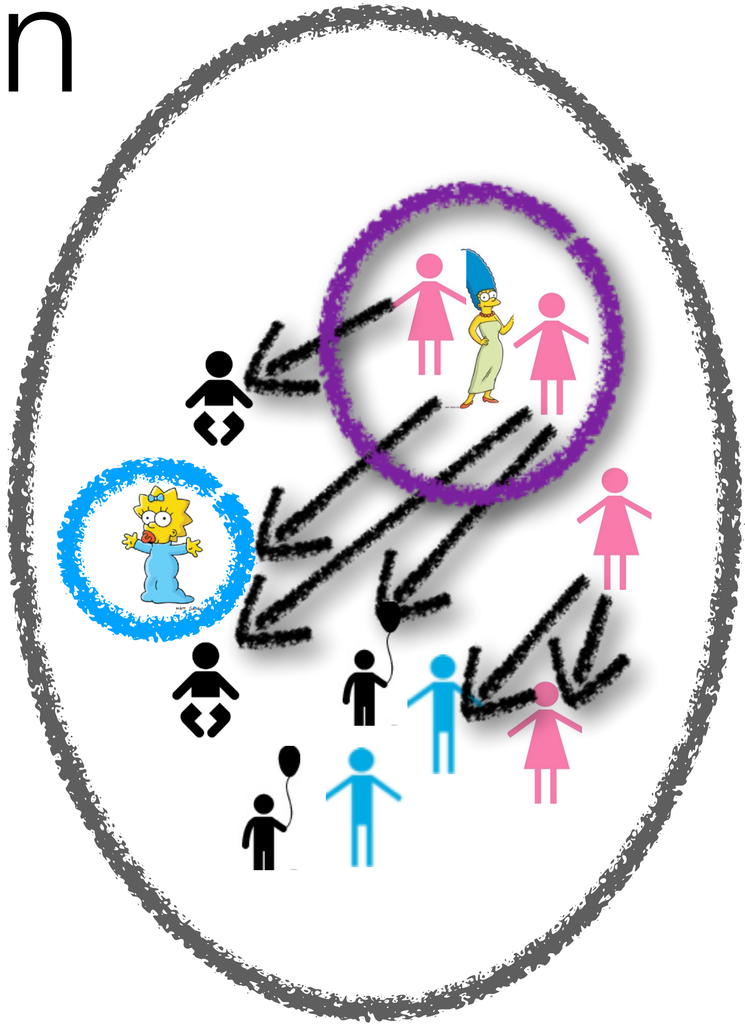
```
<owl:Class rdf:about="#motherOfChildren">
  <owl:subClassOf>
    <owl:Restriction>
      <owl:Property rdf:resource="#motherOf"/>
      <owl:qualifiedCardinality
          rdf:datatype="&xsd;nonNegativeInteger"/> 2
      </owl:qualifiedCardinality >
      <owl:Class rdf:resource="#offspring"/>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>
```

```
Class: motherOfChildren SubClassOf: motherOf exactly 2
Offspring
```

```
:motherOfChildren  rdf:type    owl:Class ;
    rdfs:subClassOf  [
     rdf:type          owl:Class ;
       rdf:type  owl:Restriction ;
       owl:cardinality "2"^^&xsd:nonNegativeInteger;
       owl:onProperty      :motherOf ;
       owl:onClass   :Offspring . ]
```

hasValue restriction

- Restrictions allow us to build new classes from class, property and individual names
  - Simpsons children are children of Marge



```
<owl:Class rdf:about="#motherOfSimpsons">
  <owl:subClassOf>
    <owl:Restriction>
      <owl:Property rdf:resource="#offspringOf"/>
      <owl:hasValue rdf:resource="#marge"/>
    </owl:Restriction>
  </owl:subClassOf>
</owl:Class>
```

```
:motherOfSimpsons  rdf:type    owl:Class ;
    owl:EquivalentClass  [
      rdf:type       owl:Restriction ;
                owl:onProperty  :offspringOf ;
                owl:hasValue    :Marge
```

```
Class: motherOfSimpsons SubClassOf: offspringOf value Marge
```

OWL property characteristics

- We can explicitly states the characteristics of object properties, and use these characteristics to refine reasoning:
  - owl:TransitiveProperty;
  - owl:SymmetricProperty;
  - owl:InverseOf;
  - owl:FunctionalProperty and owl:InverseFunctionalProperty.

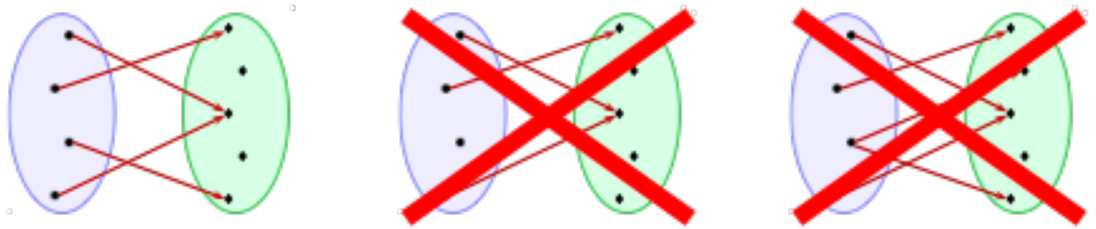
```
<owl:TransitiveProperty rdf:about=“hasAncestor"/>

<owl:SymmetricProperty rdf:about="hasSpouse"/>

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

 <owl:FunctionalProperty rdf:about="hasSpouse"/>
 <owl:InverseFunctionalProperty rdf:about=“hasSpouse"/>
```

OWL property characteristics

- OWL:TransitiveProperty
  - for all x, y, z if R(x, y) and R(y, z) then R(x, z)
    - isTallerThan, hasSameGradeAs, isSiblingOf, ...
- OWL:SymmetricProperty
  - for all x, y if R(x, y) then R(y, x)
    - isSiblingOf, hasSameGradeAs, isFriendOf ...
- OWL:InverseProperty
  - for all x, y if R(x, y) then R(y, x) ≡ R-(x, y)
  - hasParent
    - isParentOf

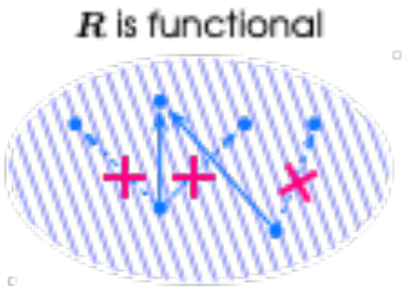
Functions

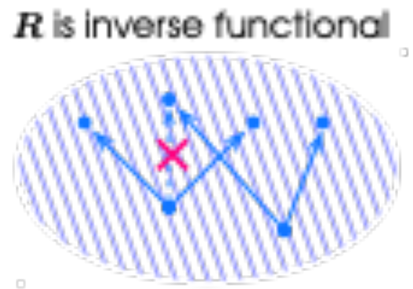
- A function from a set A to a set B is a binary relation $R \subseteq A \times B$ in which every element of A is R-related to a unique element of B
  - in other words for each $a \in A$, there is precisely one pair (a, b) in R



OWL property characteristics

- OWL:FunctionalProperty
  - for every x there is at most one y with R(x, y)
    - at most one value is associated to each object
      - directSupervisor





- OWL:inverseFunctionalProperty
  - for every y there is at most one x with R(x, y)
  - two different objects cannot have the same value associated to them
    - hasStudentNumber
      - for each StudentNumber, there can only be one student associated to that number.

OWL Property chains

- In OWL an **object property** is a binary predicate used to state facts in the form of a triple
  - where the subject and the object are entities identified by a IRI

- OWL 2 allows developers to compose two or more object property statements together in a chain,
  - the entity in the object position of one statement (other than the last one) is also the subject of the following fact.
  - Similar to an equijoin in databases

```
<rdf:Description rdf:about="hasGrandparent">
    <owl:propertyChainAxiom
rdf:parseType="Collection">
        <owl:ObjectProperty rdf:about="hasParent"/>
        <owl:ObjectProperty rdf:about="hasParent"/>
    </owl:propertyChainAxiom>
 </rdf:Description>
```

```
:hasGrandparent  owl:propertyChainAxiom
( :hasParent  :hasParent ) .
```

```
ObjectProperty: hasGrandparent
  SubPropertyChain: hasParent o hasParent
```

Recap

- OWL property constructors
- OWL property characteristics

- `https://www.w3.org/TR/owl2-primer/`

COMP318

Ontologies and Semantic Web

End of OWL - Part 5

Dr Valentina Tamma

`V.Tamma@liverpool.ac.uk`