COMP318: Introduction to OWL

www.csc.liv.ac.uk/~valli/Comp318

Dr Valentina Tamma

Room: Ashton 2.12

Dept of computer science

University of Liverpool

V.Tamma@liverpool.ac.uk

Where were we

- OWL preliminaries
- OWL class constructors

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

OWL class constructors

- Classes can also be defined through enumeration using **`owl:oneOf`**
  - allows a class to be defined extensionally,
    - with exactly the enumerated individual

```
<owl:Class rdf:about="#simpsonFamily">
   <owl:oneOf rdf:parseType="Collection">
      <owl:Thing rdf:about="#marge"/>
      <owl:Thing rdf:about="#homer"/>
      <owl:Thing rdf:about="#lisa"/>
      <owl:Thing rdf:about="#maggie"/>
      <owl:Thing rdf:about="#bart"/>
   </owl:oneOf>
</owl:Class>
```

```
:simpsonFamily  owl:equivalentClass  [
   rdf:type   owl:Class ;
   owl:oneOf  ( :marge, :homer, :lisa, :maggie, ;bart)
 ] .
```

```
Class: simpsonFamily
   EquivalentTo: { marge, homer, lisa, maggie, bart }
```

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>
```

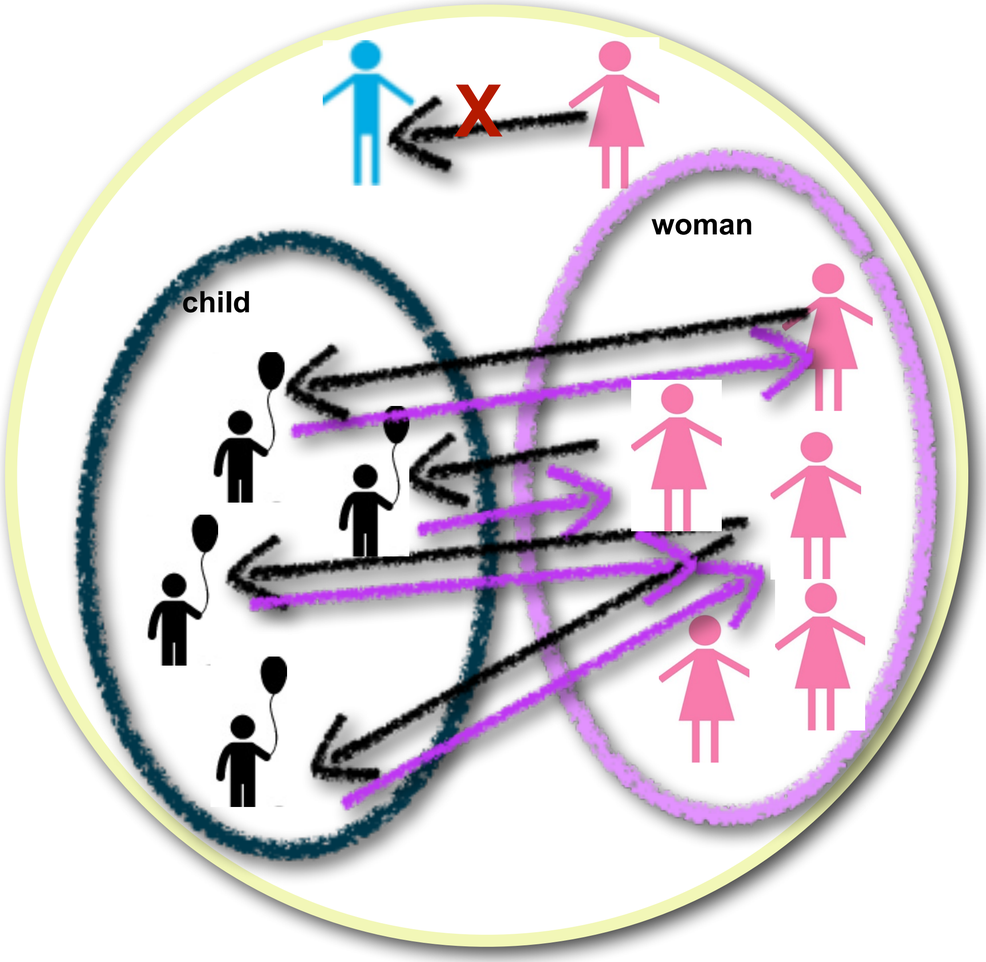
OWL properties

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

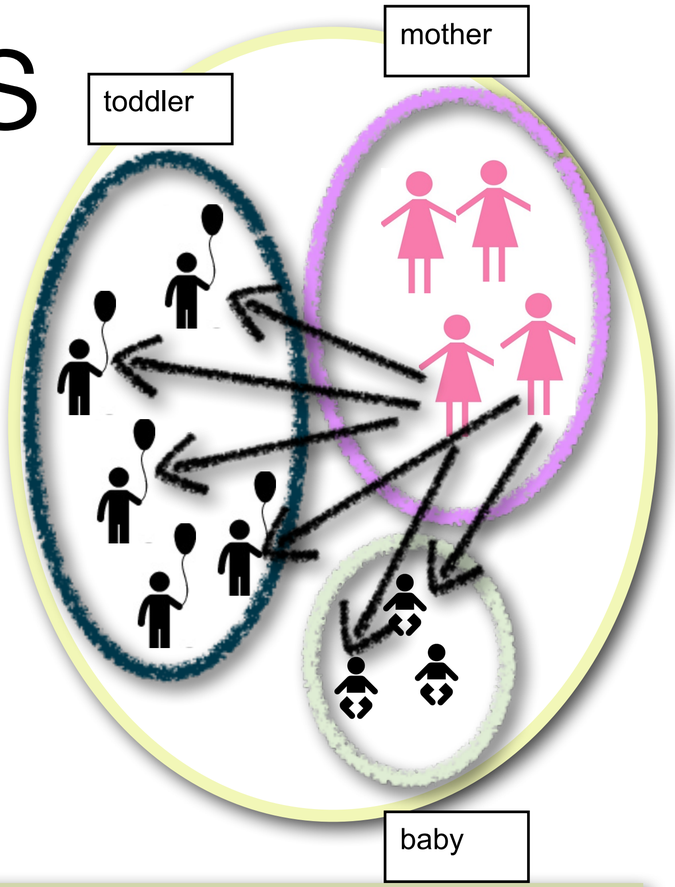
```
<owl:ObjectProperty rdf:about="#motherOf">
   <rdfs:domain rdf:resource="#woman"/>
   <rdfs:range rdf:resource="#person"/>
   <rdfs:subPropertyOf rdf:resource="#parentOf"/>
</owl:ObjectProperty>
```

```
<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

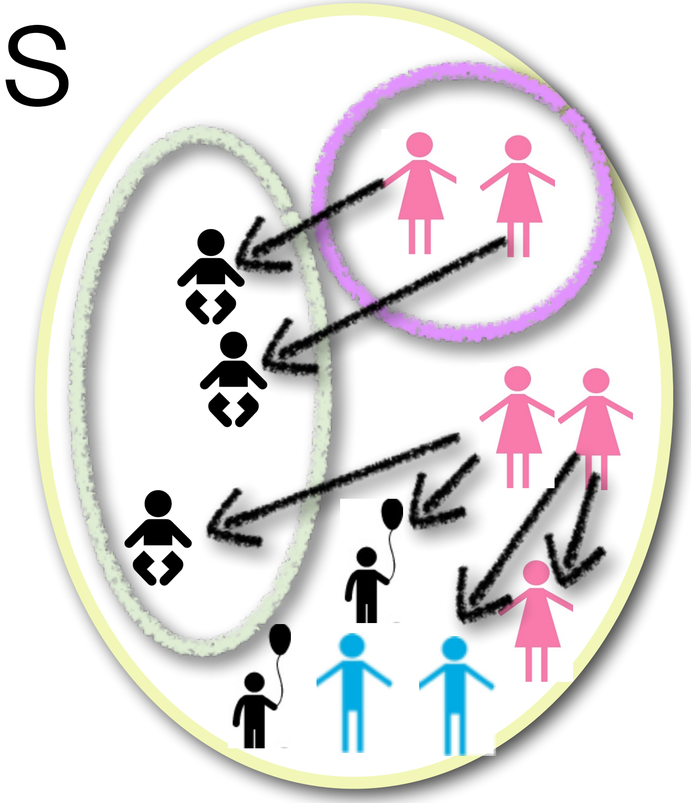
```
<owl:Class rdf:about="#motherOfToddler">
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:Property rdf:resource="#motherOf"/>
         <owl:someValuesFrom rdf:resource="#toddler"/>
      </owl:Restriction>
   </<rdfs:subClassOf>
</owl:Class>
```

```
: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

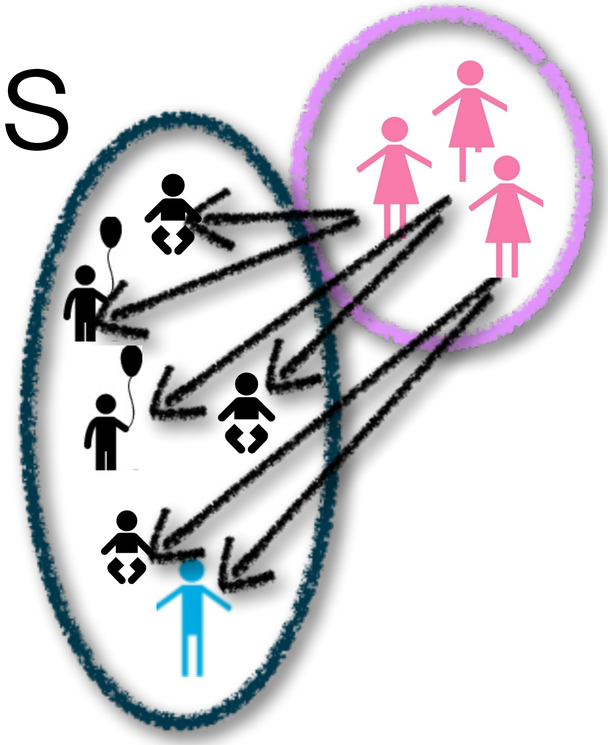


```
<owl:Class rdf:about="#motherOfBaby">
   <owl:subClassOf>
      <owl:Restriction>
         <owl:Property rdf:resource="#motherOf"/>
         <owl:allValuesFrom rdf:resource="#baby"/>
      </owl:Restriction>
   </owl:subClassOf>
</owl:Class>
```

```
: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



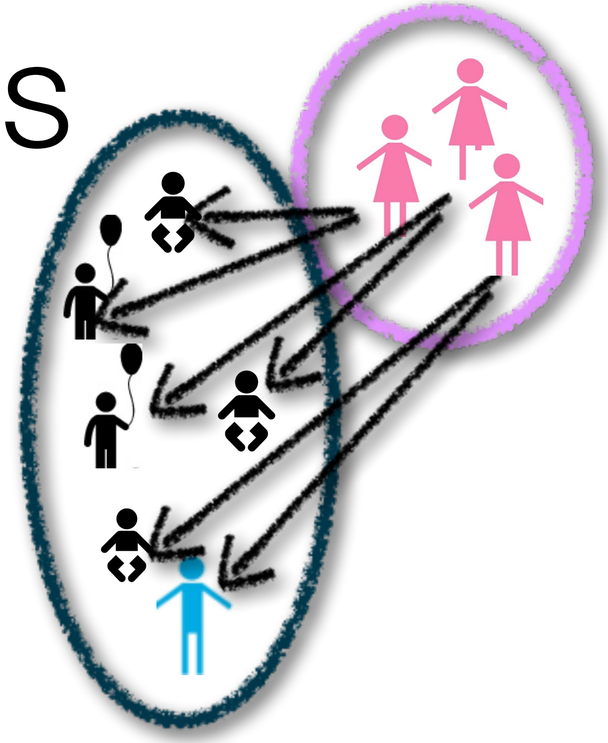
- Restrictions allow us to build new classes from class, property and individual names
  - *min*, *max* and *exactly* cardinality restrictions

```
<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



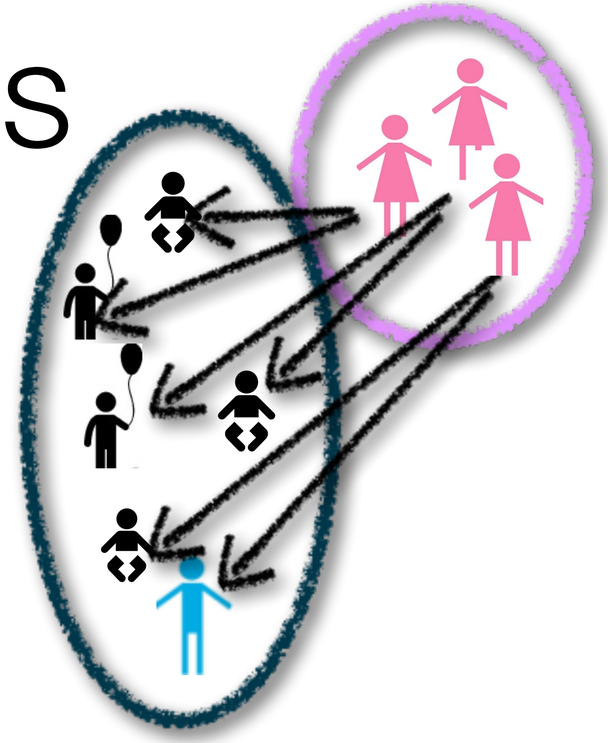
- Restrictions allow us to build new classes from class, property and individual names
  - *min*, *max* and *exactly* cardinality restrictions

```
<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

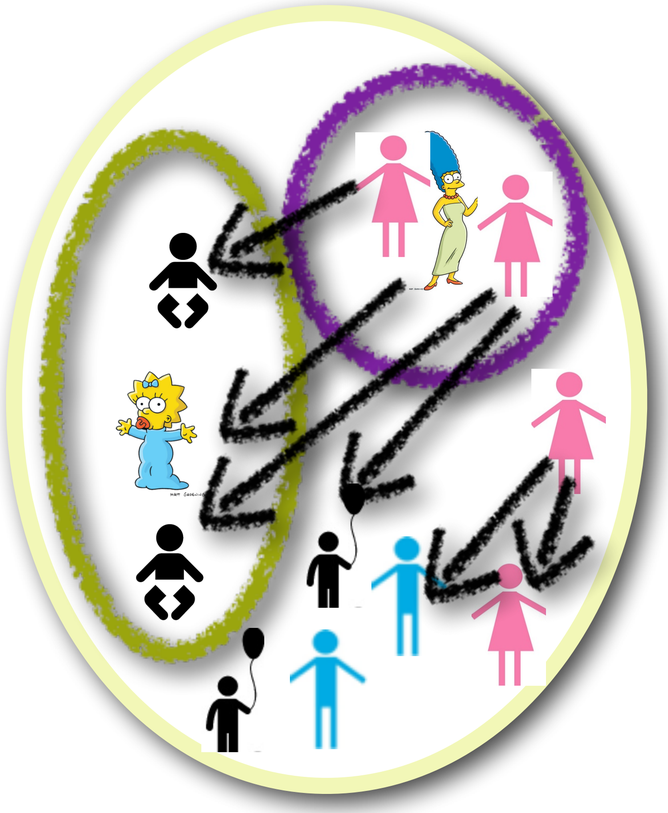
```
<owl:Class rdf:about="#motherOfTwo">
   <owl:subClassOf>
      <owl:Restriction>
         <owl:Property rdf:resource="#motherOf"/>
         <owl:qualifiedCardinality
              rdf:datatype="&xsd;nonNegativeInteger"/> 2
         </owl:qualifiedCardinality >
         <owl:Class rdf:resource="#offspring"/>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

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

```
:motherOfChildren  rdf:type    owl:Class ;
     rdfs:subClassOf  [
       rdf:type             owl:Class ;
       owl:intersectionOf
         ( [ rdf:type  owl:Restriction ;
             owl:Cardinality "2"^^&xsd:nonNegativeInteger ;
             owl:onProperty      :motherOf .])
```

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 1.0: characterising properties

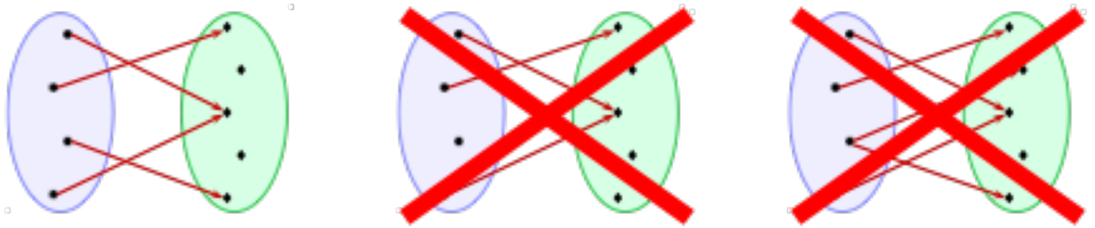
- We can explicitly states the characteristics of object properties, and use these characteristics to refine reasoning:
  - owl:TransitiveProperty;
  - owl:SymmetricProperty;
  - owl:InverseOf;
  - owl:FunctionalProperty and owl:InverseFunctionalProperty.

OWL 1.0: characterising properties

- 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

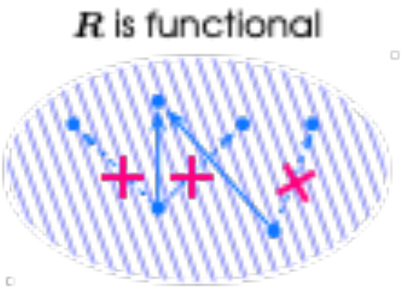
Functions

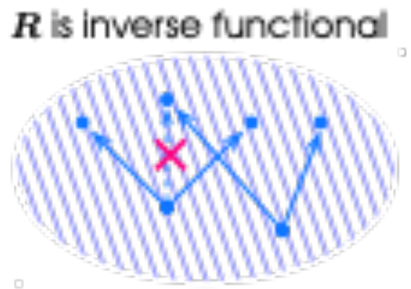
- A function from a set A to a set B is a binary relation $R \subseteq A \times B$ in which every element of A is R-related to a unique element of B
  - in other words for each $a \in A$, there is precisely one pair (a, b) in R



OWL 1.0: characterising properties

- 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.

Example

- Translate in turtle syntax the following statements, and add any axiom you think appropriate:
  - john is a lecturer
  - mary is an academic staff member
  - mary is 39 years old
  - COMP1111 is a course
  - each course is taught by at most one staff member
  - john teaches COMP1111
  - mary teaches COMP1111

Example (ctd)

- Is the model we obtain correct, or does it contain contradictory information?
  - If so, what are the statements that cause a contradiction?
  - how would you solve it?

Example

- Translate in turtle syntax the following statements:
  - first year courses are courses taught only by professors

Recap

- OWL class constructors
- OWL properties
- Restrictions

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