
# COMP318

## Ontologies and Semantic Web

# OWL - Part 7

**Dr Valentina Tamma**
`V.Tamma@liverpool.ac.uk`

# OWL 2 properties

- owl:DatatypeProperty
  - the range is a set of data values, e.g. foaf:birthday rdf:type owl:DatatypeProperty
  - Universal data property: $\mathscr{D}^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Lambda$ ($\Lambda$ set of all literal values)
- owl:ObjectProperty
  - the range is a set of individuals, e.g.:rents rdf:type owl:ObjecteProperty
  - Universal data property: $\mathscr{U}^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
- owl:AnnotationProperty
  - Does not yield any logical implication, e.g. rdfs:label rdf:type owl:AnnotationProperty

The set of classes, named individuals and properties (datatype, object and annotation) are all ***mutually disjoint***

# Ontology representation in OWL

- Class definition: SubClassOf vs EquivalentClasses
  - *All PizzaMargherita have, amongst other things, some mozzarella topping and some tomato topping*
    - all examples are in Turtle syntax

```
:Margherita rdf:type owl:Class ;
        rdfs:subClassOf :NamedPizza ,
                [ rdf:type owl:Restriction ;
                  owl:onProperty :hasTopping ;
                  owl:someValuesFrom :TomatoTopping
                ] ,

                [ rdf:type owl:Restriction ;
                  owl:onProperty :hasTopping ;
                  owl:someValuesFrom :MozzarellaTopping
                ] ;
```

# Understanding restrictions in OWL

- OWL uses the `rdfs:subClassOf` for representing subsumption
  - Suppose we want to state the cheesy toppings have some ingredient "cheese" as their "main ingredient"
    - *Cheesy topping is a subclass of all things that have as main ingredient some cheese*

```
:CheeseTopping
  a owl:Class ;
  rdfs:subClassOf
    [ a owl:Restriction ;
      owl:onProperty :mainIngredient ;
      owl:someValueFrom Cheese ].
```

# Ontology representation in OWL

- Class definition: SubClassOf vs EquivalentClasses
  - *A MeatyPizza is any pizza that has, amongst other things, at least one meat topping*

```
:MeatyPizza rdf:type owl:Class ;
        owl:equivalentClass [ rdf:type owl:Class ;
                              owl:intersectionOf (
                                :Pizza
                                [ rdf:type owl:Restriction ;
                                  owl:onProperty :hasTopping ;
                                  owl:someValuesFrom :MeatTopping
                                ]
                              )
                            ] ;
```

# Understanding restrictions in OWL

- OWL uses the `owl:EquivalentClass` for representing equivalence
  - The definition states that the set of CheeseTopping things is exactly the same as the class of things that have as main ingredient at least one thing that is cheese.
  - 

```
:CheeseTopping
  a owl:Class ;
  owl:EquivalentTo
    [ a owl:Restriction ;
      owl:onProperty :mainIngredient ;
      owl:someValueFrom Cheese ].
```

# Understanding restrictions

- All cheesy toppings form a subset of all things that have as main ingredient some cheese.
  - If you omit the subClassOf construct you can read it as a UML class with an attribute mainIngredient and a value type constraint for the attribute.

- The subclass relation is essential for understanding the semantics of the restriction:
  - it is a necessary but not sufficient condition for the class.
    - There might be things that have cheese as main ingredient but are not a cheesy topping, hence the subset/subclass definition.

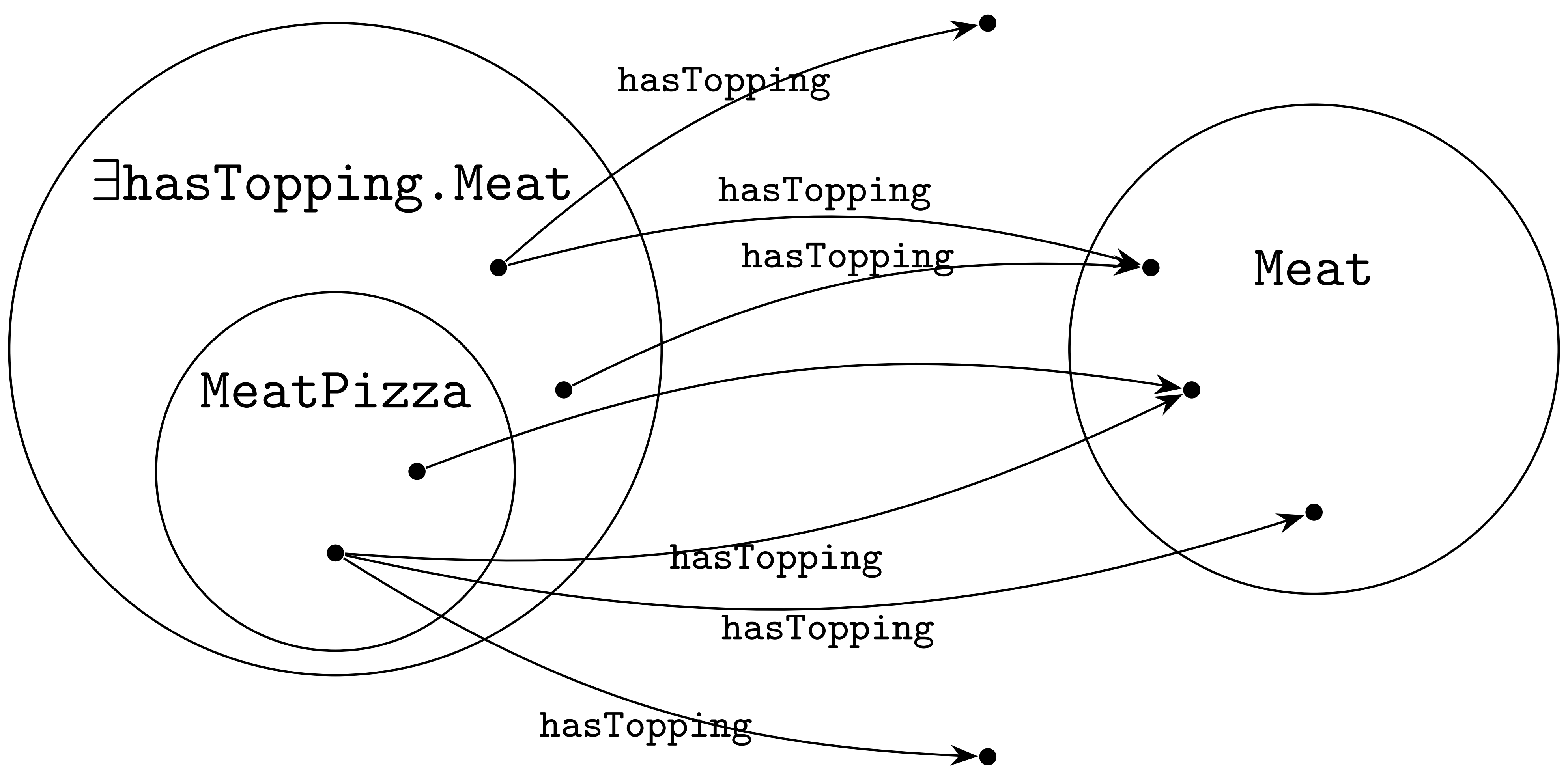
# Existential restrictions

- someValuesFrom
  - Can be used to declare primitive or defined classes
  - *Indicates some or at least one*
    - *A MeatyPizza is any pizza that has some (at least one) meet topping*

```
:MeatyPizza rdf:type owl:Class ;
        owl:equivalentClass
          [ rdf:type owl:Class ;
            owl:intersectionOf ( :Pizza
                                 [ rdf:type owl:Restriction ;
                                   owl:onProperty :hasTopping ;
                                   owl:someValuesFrom :Meat
                                 ]
                               )
          ] ;
```

# Existential Restrictions

- $\mathtt{A} \sqsubseteq \exists \mathtt{R}.\mathtt{C}$
  - $\mathtt{A}$ is $\mathtt{R}$-related to at least one $\mathtt{C}$
    - $(\exists \mathtt{R}.\mathtt{C})^{\mathcal{I}}$ = { $\mathtt{a} \in \Delta^{\mathcal{I}}$ | there is a $\mathtt{b}$ where $\langle \mathtt{a}, \mathtt{b} \rangle \in \mathtt{R}^{\mathcal{I}}$ and $\mathtt{b} \in \mathtt{C}^{\mathcal{I}}$ }
    - $\mathtt{MeatPizza} \sqsubseteq \exists \mathtt{hasTopping}.\mathtt{Meat}$



# Existential Restrictions and `rdfs:domain`

- Domain (global scope for R)
  - R rdfs:domain C means anything to which R applies is in C
    - :hasTopping rdfs:domain :Pizza
  - The domain can also be expressed as $\exists R.\top \sqsubseteq C$
    - $\exists hasTopping.\top \sqsubseteq Pizza$
  - Local scope for R:
    - $Pizza \sqsubseteq \exists hasTopping.Topping$

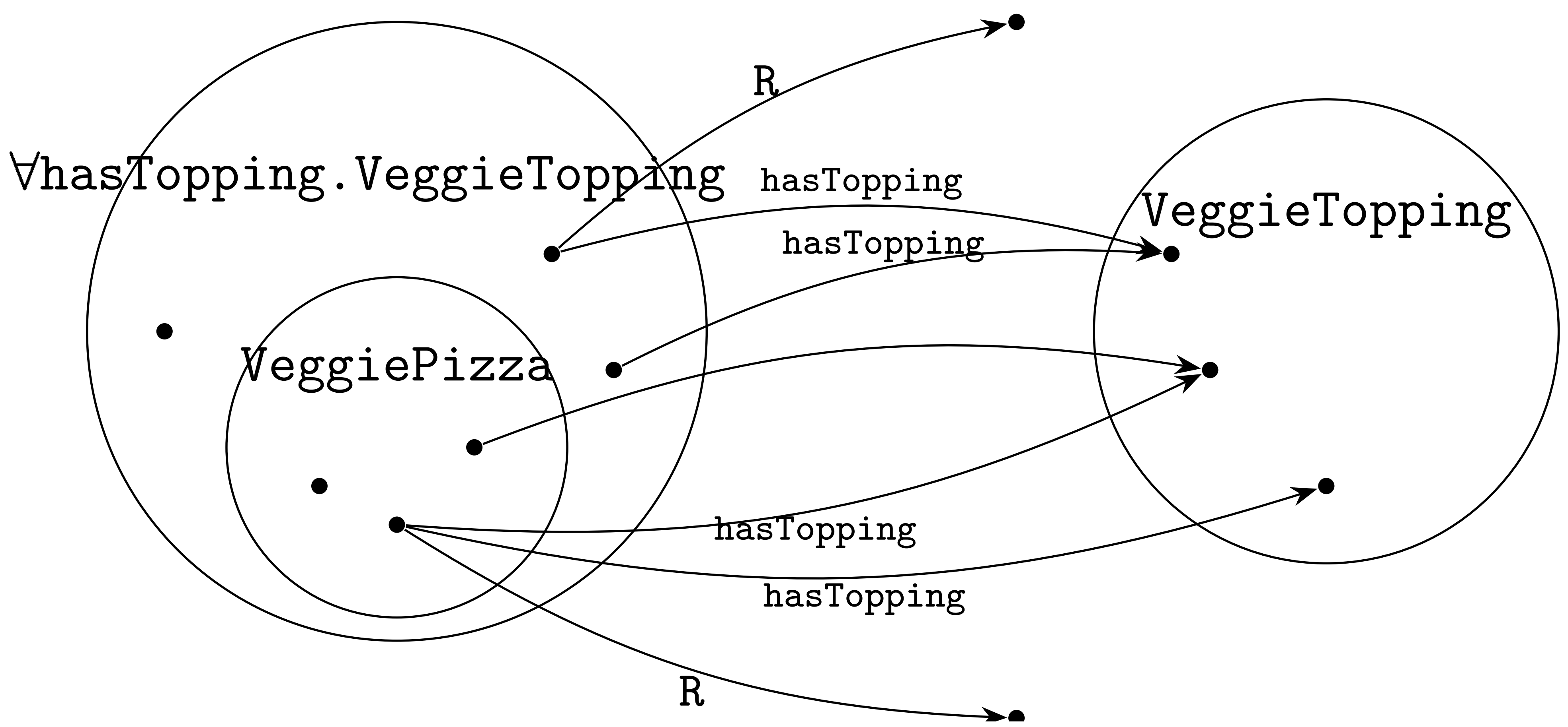
# Universal restrictions

- allValuesFrom
  - Can be used to declare primitive or defined classes
  - *Indicates only or no values except*
    - *A VegetarianPizza is any pizza that has only a VegetarianTopping*

```
:VeggiePizza rdf:type owl:Class ;
             owl:equivalentClass
               [ rdf:type owl:Class ;
                 owl:intersectionOf (
                   :Pizza [ rdf:type owl:Restriction ;
                            owl:onProperty :hasTopping ;
                            owl:allValuesFrom :VeggieTopping ]
                          )
               ] .
```

# Universal Restrictions

- $\mathtt{A} \sqsubseteq \forall \mathtt{R}.\mathtt{C}$
  - $\mathtt{A}$ is $\mathtt{R}$-related to only elements of $\mathtt{C}$
    - $(\forall \mathtt{R}.\mathtt{C})^{\mathcal{I}}$ = { $\mathtt{a} \in \Delta^{\mathcal{I}}$ | for all $\mathtt{b}$, if $\langle \mathtt{a}, \mathtt{b} \rangle \in \mathtt{R}^{\mathcal{I}}$ then $\mathtt{b} \in \mathtt{C}^{\mathcal{I}}$ }
    - $\mathtt{VeggiePizza} \sqsubseteq \forall \mathtt{hasTopping}.\mathtt{VeggieTopping}$



# Universal Restrictions and `rdfs:range`

- Range `R` `rdfs:range` `C` means anything one can reach by `R` is in `C`
    - :hasTopping rdfs:range :VeggieTopping
- The range can also be expressed as $\top \sqsubseteq \forall \mathtt{R}.\mathtt{C}$
    - $\top \sqsubseteq \forall \mathtt{hasTopping}.\mathtt{VeggieTopping}$
- Local scope for `R`:
    - $\mathtt{VeggiePizza} \sqsubseteq \forall \mathtt{hasTopping}.\mathtt{VeggieTopping}$

# Cardinality restrictions

- minQualifiedCardinality, maxQualifiedCardinality, exactCardinality
  - Can be used to declare primitive or defined classes
  - *Restricts the number of relations an object can have (min, max, exactly)*
    - *A Four Cheeses Pizza is any pizza that has minimum 4 toppings of type Cheese*

```
:FourCheeses  rdf:type rdf:type owl:Class ;
            owl:equivalentClass
              [ rdf:type owl:Class ;
               owl:intersectionOf (
                :Pizza [rdf:type   owl:Restriction ;
                        owl:minQualifiedCardinality
                        "4"^^xsd:nonNegativeInteger ;
                        owl:onProperty   :hasTopping ;
                        owl:onClass  :Cheese ] )
].
```

# A-Box: assertional axioms

- Contains facts about concrete individuals a, b, c, ....
  - A set of concept assertions, in RDF: :john_simmons rdf:type :Tennant
  - A set of role assertions, in RDF: :john :rents :baronWayApartment
  - Equality and inequality between individuals:
    - :john_simmons owl:sameAs :js
    - :john_simmons owl:differentFrom :mary_simmons
  - because the UNA does not hold in OWL


# COMP318

## Ontologies and Semantic Web

# End OWL - Part 7

**Dr Valentina Tamma**
V.Tamma@liverpool.ac.uk