COMP318

Ontologies and Semantic Web

OWL - Part 6

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

OWL semantics

- Checking for anomalies depends on being able to reason with the content of the ontologies

- Understanding OWL requires some understanding of how DL models the world.

- OWL2 ontologies are represented by axioms
  - But not all axioms correspond to only one triple

- Axioms are typically classified in T-box, R-Box, and A-Box

- T-Box: terminological knowledge
  - **independent of** any actual instance data
    - Class equivalence: $\mathtt{Offspring} \equiv \mathtt{Children}$
    - Class inclusion: $\mathtt{House} \sqsubseteq \mathtt{ResidentialUnit}$
    - OWL uses the `rdfs:subClassOf` for representing subsumption.
  - The set of property axioms also known as R-Box

- A-Box: assertional knowledge
  - facts about concrete instances
  - represented in RDF

OWL direct semantics

- The direct model-theoretic semantics of OWL 2 is strongly related to and extends the semantics of a particular description logic (SROIQ)
  - As the definition of SROIQ does not provide for datatypes and punning, the semantics of OWL 2 is defined directly on the constructs of the structural specification of OWL 2
- Since each OWL 1 DL ontology is an OWL 2 ontology, this is also a direct semantics for OWL 1 Lite and OWL 1 DL ontologies;
  - this semantics is equivalent to the direct model-theoretic semantics of OWL 1 Lite and OWL 1 DL

Interpretations for DLs

- The semantics of description logics are defined by providing an **interpretation** of
  - classes (concepts) as sets of individuals and
  - properties (roles) as sets of ordered pairs of individuals
    - these individuals are typically assumed from a given domain.

- Interpretations ($\mathcal{I}$) are thought of as possible "realities"
  - we can see them as a function from abstract representations to the actual elements in set theory (things that exist in the domain)
    - the function assigns values to elements and may be the model of a graph or an ontology

$$
\begin{aligned}
\text{dbo:City}^{\mathcal{I}} &= \{\text{dbp:amsterdam}^{\mathcal{I}}\} \\
\text{dbo:Country}^{\mathcal{I}} &= \{\text{dbp:netherlands}^{\mathcal{I}}\} \\
\text{dbo:PopulatedPlace}^{\mathcal{I}} &= \{\text{dbp:amsterdam}^{\mathcal{I}}, \text{dbp:Netherlands}^{\mathcal{I}}\} \\
\text{dbo:location}^{\mathcal{I}} &= \{\langle \text{dbp:amsterdam}^{\mathcal{I}}, \text{dbp:Netherlands}^{\mathcal{I}} \rangle\}
\end{aligned}
$$

Examples

Triples	DL syntax	Semantics
dbp:amsterdam dbo:location dbp:Netherlands .	location(amsterdam, netherlands)	$\langle \mathrm{dbp : amsterdam}^{\mathcal{I}}, \mathrm{dbp : Netherlands}^{\mathcal{I}} \rangle \in \mathrm{location}^{\mathcal{I}}$
dbp:amsterdam rdf:type dbo:City .	City(amsterdam)	$\mathrm{dbp : amsterdam}^{\mathcal{I}} \in \mathrm{dbo : City}^{\mathcal{I}}$
dbo:City rdfs:subClassOf dbo:Place .	$\mathrm{City} \sqsubseteq \mathrm{Place}$	$\mathrm{dbo : City}^{\mathcal{I}} \subseteq \mathrm{dbo : Place}^{\mathcal{I}}$
dbo:capitalOf rdfs:subPropertyOf dbo:location .	$\mathrm{capitalOf} \sqsubseteq \mathrm{location}$	$\mathrm{dbo : capitalOf}^{\mathcal{I}} \subseteq \mathrm{dbo : location}^{\mathcal{I}}$

Classes and individuals in OWL2

- owl:Class specialises rdfs:Class to represent classes
  - set of individuals that can be:
    - atomic (i.e. identified by a IRI, a more general form of URI): :Building rdf:type owl:Class
    - complex, i.e. built as boolean combination from other entities (classes and properties)
  - owl:NamedIndividual is the set of concrete individuals, used instead of rdfs:Resource
    - :BaronWayApartment rdf:type owl:NamedIndividual
    - :BaronWayApartment rdf:type :Building

Top and bottom classes

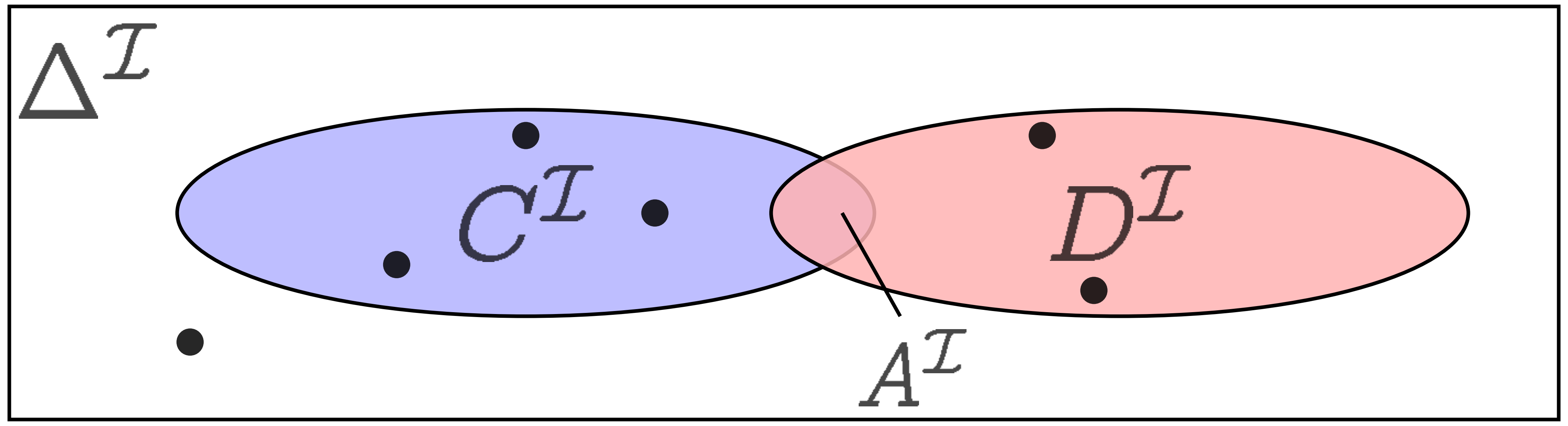
- `owl:Thing`
  - in DL syntax: $\top$
    - class containing **all individuals,** its interpretation is $\Delta^{\mathcal{I}}$
    - for every `owl:Class` C, C is a subclass of `owl:Thing`
- `owl:Nothing`
  - in DL syntax: $\bot$
    - empty class that contains **no individuals,** its interpretation is the empty set $\emptyset$
      - set of all individuals
    - for every `owl:Class` C, `owl:Nothing` is a subclass of C

T-Box axioms

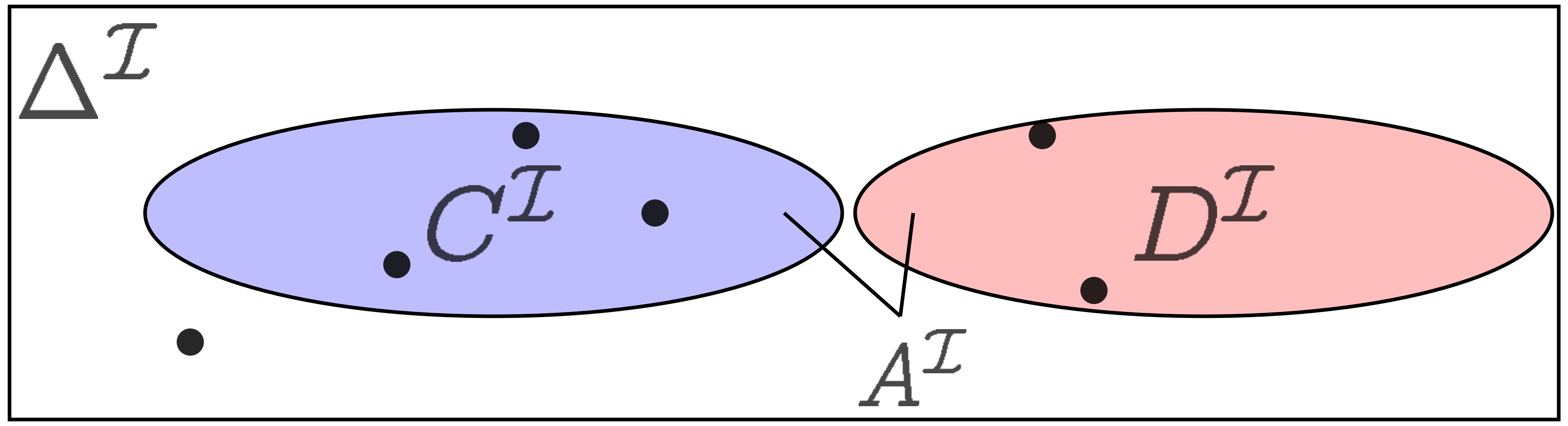
- The T-Box (classes only) is composed by:
  - Subsumption Axioms: $\mathtt{C} \sqsubseteq \mathtt{D}$ $(C^I \subseteq D^I)$
    - Declare Primitive classes, expressing necessary conditions for membership (but not sufficient)
      - Typically found near the top of the hierarchy
  - Equivalence Axioms: $\mathtt{C} \equiv \mathtt{D}$ $(C^I = D^I)$
    - Declare Defined classes, expressing necessary and sufficient conditions for membership
      - Similar to if and only if statements in predicate logic
      - Typically found as we move further down by specialising general concepts with various restrictions.
- $\mathtt{C}$ and $\mathtt{D}$ can be named concepts (e.g. **foaf:Agent**) or complex concepts obtained by using logical operators:
  - Negation: $\neg\, \mathbf{E}$
  - Intersection: $\mathbf{E} \sqcap \mathbf{F}$
  - Union: $\mathbf{E} \sqcup \mathbf{F}$
  - Property restrictions:
    - E.g. local scope for domain and ranges: $\exists \mathbf{R}.\mathbf{E}$, $\forall \mathbf{R}.\mathbf{E}$

Union and Intersection

- $\mathtt{A} \sqsubseteq \mathtt{C} \sqcap \mathtt{D}$
  - MargheritaTopping $\sqsubseteq$ CheeseTopping $\sqcap$ TomatoTopping



- $\mathtt{A} \sqsubseteq \mathtt{C} \sqcup \mathtt{D}$
  - VegetarianPizza $\sqsubseteq$ CheeseTopping $\sqcup$ VegetableTopping



Ontology representation in OWL

- Boolean Combination: Union (disjunction)
  - A vegetarian pizza is any pizza which, amongst other things, has only vegetable and/or cheese toppings

```
:VegetarianPizza rdf:type owl:Class ;
          owl:equivalentClass
            [ rdf:type owl:Class ;
              owl:intersectionOf ( :Pizza
                          [ rdf:type owl:Restriction ;
                            owl:onProperty :hasTopping ;
                            owl:allValuesFrom
                              [ rdf:type owl:Class ;
                                owl:unionOf ( :CheeseTopping
                                            :VegetableTopping
                                           )
                              ]
                          ]
                         )
            ]
```

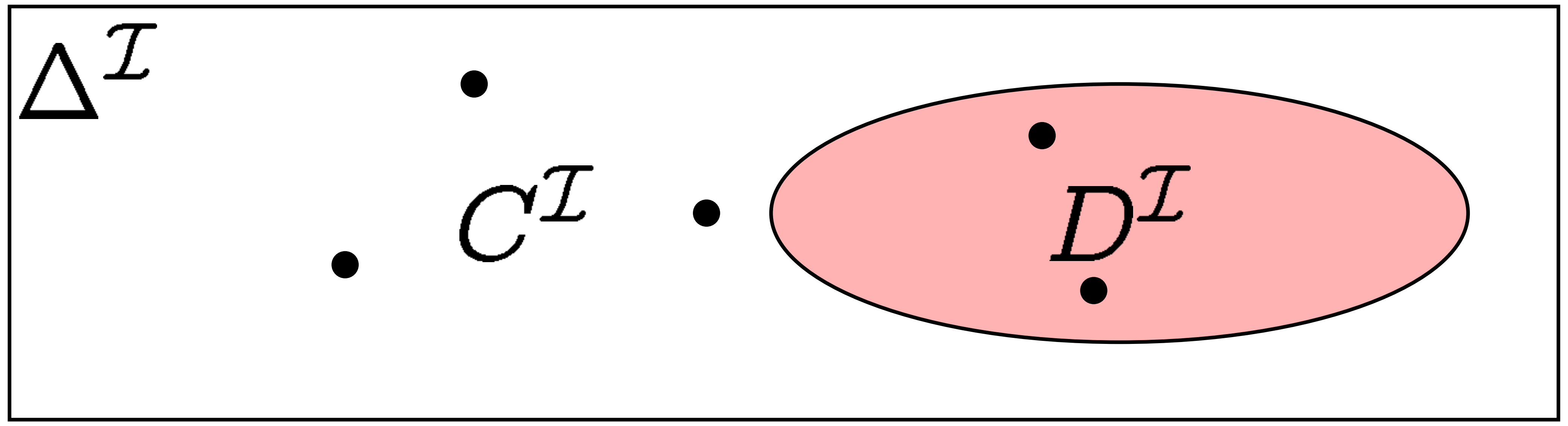
Ontology representation in OWL

- Boolean Combination: Intersection (conjunction)
  - A ProteinLover's is any pizza that has, amongst other things, has only toppings that are both meat and seafood

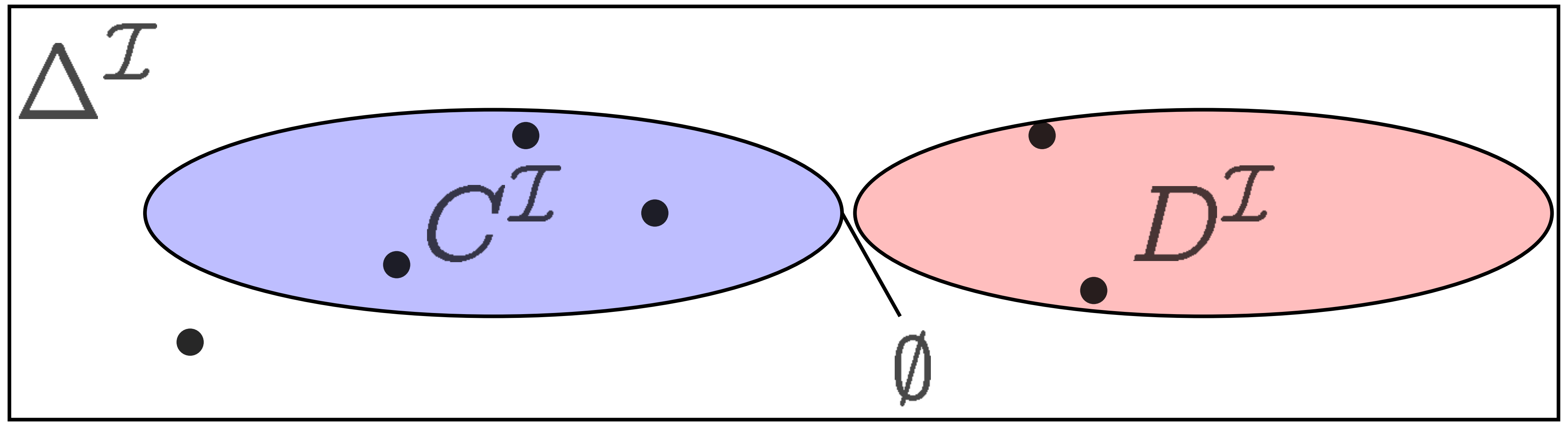
```
:ProteinLoverPizza rdf:type owl:Class ;
            owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:intersectionOf ( :Pizza
                           [ rdf:type owl:Restriction ;
                             owl:onProperty :hasTopping ;
                             owl:allValuesFrom
                               [ rdf:type owl:Class ;
                                 owl:intersectionOf
                                   ( :FishTopping
                                     :MeatTopping
                                   )
                               ]
```

Negation and Disjointness

- $\mathtt{C} \sqsubseteq \neg \mathtt{D}$
  - A $\mathtt{C}$ is not a $\mathtt{D}$
  - VegetarianTopping $\sqsubseteq \neg$ MeatTopping



- $\mathtt{C} \sqcap \mathtt{D} \sqsubseteq \bot$
  - Nothing is both a $\mathtt{C}$ and a $\mathtt{D}$
  - Equivalent to $\mathtt{C} \sqsubseteq \neg \mathtt{D}$ (and $\mathtt{D} \sqsubseteq \neg \mathtt{C}$)



Ontology representation in OWL

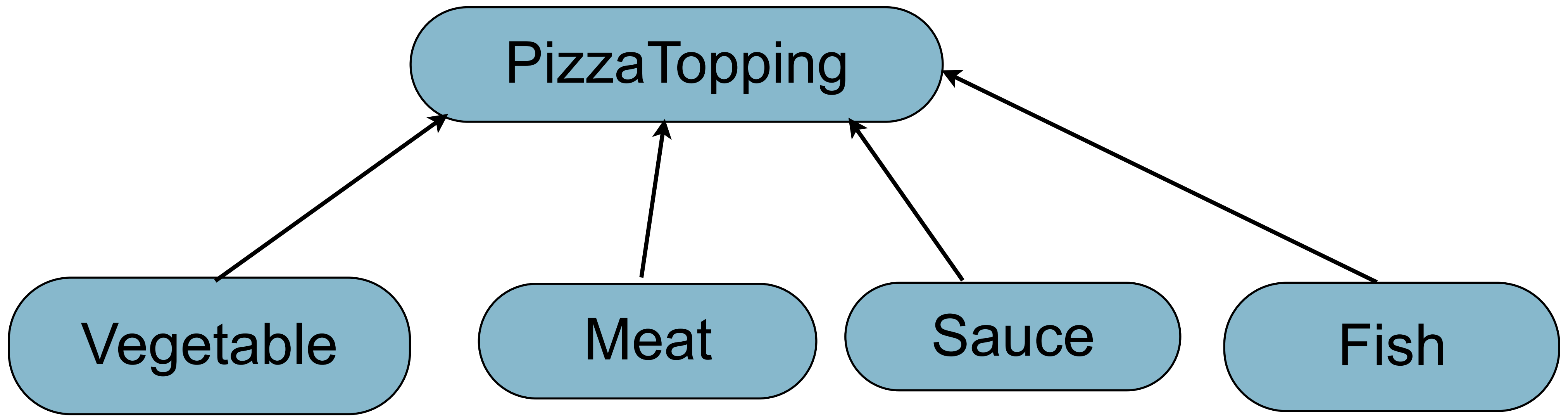
- Boolean Combination: Complement (conjunction)
  - A non vegetarian pizza is any pizza that is not a vegetarian one

```
:NonVegetarianPizza rdf:type owl:Class ;
          owl:equivalentClass
            [ rdf:type owl:Class ;
              owl:intersectionOf ( :Pizza
                          [ rdf:type owl:Class ;
                            owl:complementOf :VegetarianPizza
                          ]
                        )
            ] ;

          owl:disjointWith :VegetarianPizza ;
```

Ontology representation in OWL

- Disjointness
  - States that all disjoint classes belong to different branches in the ontology tree
  - The default is for classes to overlap
    - Disjointness needs to be stated explicitly




```
:MeatTopping rdf:type owl:Class ;
        rdfs:subClassOf :PizzaTopping ;
        owl:disjointWith :FishTopping ,
                         :SauceTopping ,
                         :VegetableTopping .
```

Check for anomalies

- An important advantage of the use of OWL over RDF Schema is the possibility to detect inconsistencies
  - In ontology
    - incoherent ontology: at least an unsatisfiable class, class that cannot have any instance
  - In ontology+instances
    - inconsistent ontology: every class is interpreted as the empty set
- Examples of common inconsistencies
  - incompatible domain and range definitions for transitive, symmetric, or inverse properties
  - cardinality properties
  - requirements on property values can conflict with domain and range restriction
- Examples from the Pizza tutorial for Protege
  - http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/

COMP318

Ontologies and Semantic Web

nd OWL - Part 6

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