Ontologies and OWL

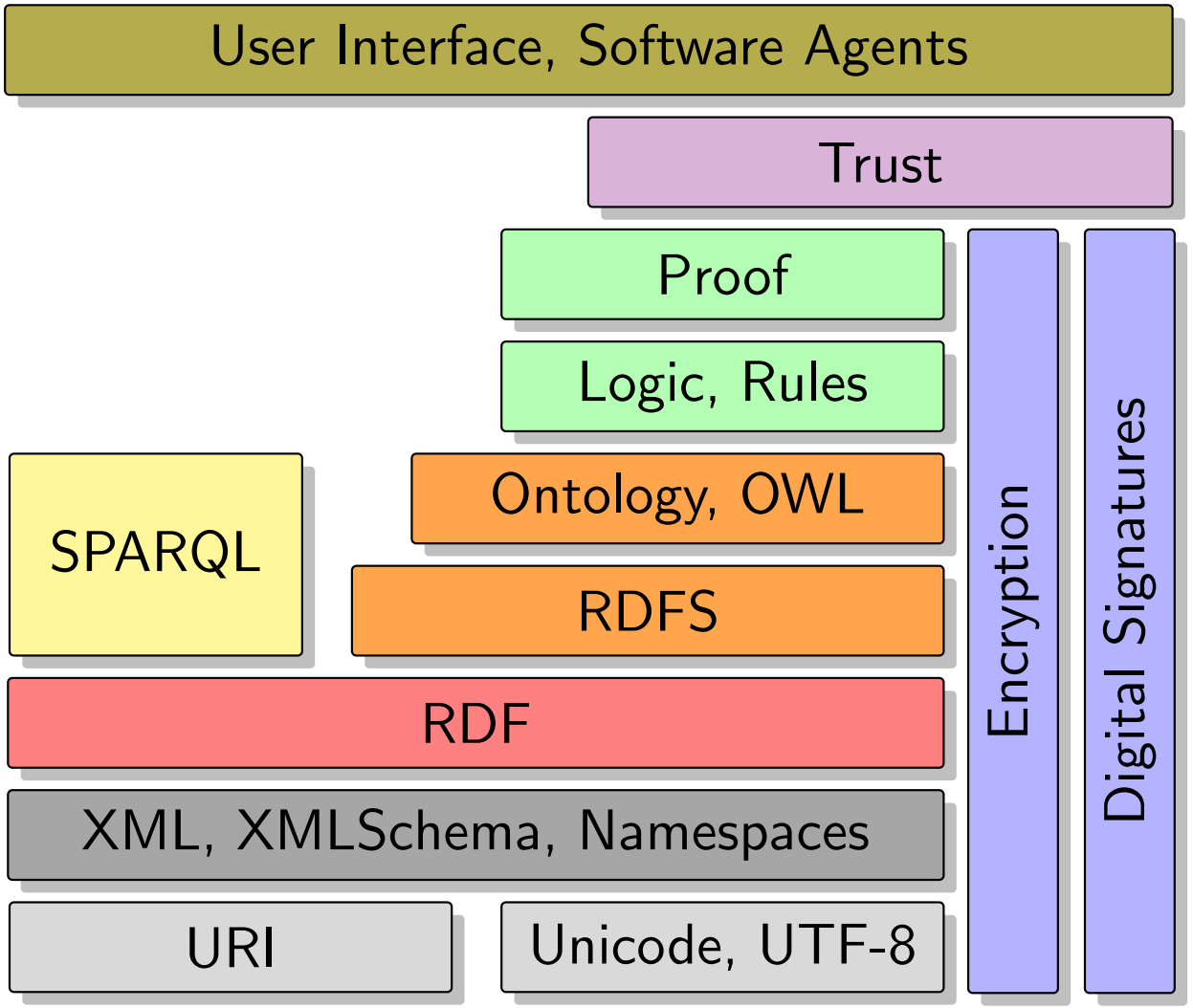
Camilo Thorne

Room 00.012
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
+49 (0) 711 685-81369
camilo.thorne@ims.uni-stuttgart.de

Semantic Web, SS 2017
(based on slides by W. Kessler)

Institut für Maschinelle Sprachverarbeitung

The Semantic Web Stack [W3C, Tim Berners-Lee]



Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

Quiz: Limitations of RDFS

Which of the statements can be modeled with RDF or RDFS?

A) Every cat is a dog.

B) A mouse is never a fruit [or: no mouse is a fruit].

C) If customer X ordered item Y,
then item Y is ordered by customer X.

D) Everything that is a pet is an animal.

E) If someone has internal medicine as specialization,
then she's a doctor.

F) Every box of organic fruit contains at least one fruit fly.

Institut für Maschinelle Sprachverarbeitung

Limitations of RDFS

- We cannot express that two classes are disjoint (e.g., mouse and fruit).
- We cannot say that a property is the inverse of another (e.g., if X ordered Y, Y is ordered by X).
- We cannot declare restrictions on property values for one class (e.g., every box must contain a fruit fly).
- We cannot build new classes by combining other classes (e.g., vegetarian is everything that is not meat or fish).
- We cannot define cardinality restrictions for properties (e.g., every person can have only one ID).

RDF and RDFS lay the foundation for a formal representation of domain knowledge in an ontology, but there is a need for more complex descriptions ▷ OWL.

Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

Ontology Definitions (1)

Definition: Ontology [Gruber, 1993]

An ontology is an explicit [formal] specification of a [shared] conceptualization.

Explicit Clear, explicit, unambiguous definitions of concepts (vocabulary) and constraints of the domain.

Formal Readable and interpretable by computers, with a formal semantics (i.e. Description Logics).

Shared Describes a common, shared standard, not the view of a single person.

Conceptualization An ontology describes an abstract, simplified model of (a part of) the real world, that is being used for representing knowledge.

Institut für Maschinelle Sprachverarbeitung

Ontology Definition (2)

Definition: Ontology [W3C, OWL Web Ontology Language Guide (2004)]

Ontology is a term borrowed from philosophy that refers to the science of **describing the kinds of entities in the world and how they are related**. An OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL **formal semantics** specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics.

Institut für Maschinelle Sprachverarbeitung

Ontology Definition (3)

Definition: Ontology [W3C, Linked Data Glossary (2013)]

A **formal model** that allows **knowledge to be represented** for a **specific domain**. An ontology describes the types of things that exist (classes), the relationships between them (properties) and the logical ways those classes and properties can be used together (axioms).

Institut für Maschinelle Sprachverarbeitung

Parts of an Ontology

Classes describe concepts from the domain to be modeled.

Subclasses describe more specific concepts.

Instances are concrete individuals, they are always instances of a specific class and can have relations with classes and other instances.

Data properties describe attributes (literals of a certain data type) of a class or instance.

Object properties describe relations of a class or instance with other classes or instances.

Institut für Maschinelle Sprachverarbeitung

Quiz: Fruit Ontology

apple, *orange*, *citrus fruit*, *Granny Smith*, *vitamins*, *ripe in summer*, *grows on tree*, *has stone*, *berry*, *red*, *Spanish*

We want to create an ontology about fruit. Given the above list of terms, decide for each term if it should be:

- a class on one of the top levels in the hierarchy,
- a class on the lowest level in the hierarchy,
- an instance,
- a data property, or
- an object property.

Institut für Maschinelle Sprachverarbeitung

OWL[1]: Web Ontology Language

- OWL is a W3C recommendation since February 2004,
  OWL 2 is a W3C recommendation since December 2012.
- Like RDFS, OWL lets us define the semantics of a domain
- OWL builds on RDFS, but offers more complex descriptions than RDFS
- Because OWL is based on logics, it has formal semantics which allows reasoning about the knowledge
- OWL namespace (usually abbreviated as `owl`):
  `http://www.w3.org/2002/07/owl#`

[1] Yes, the wrong sequence of letters is on purpose!

Institut für Maschinelle Sprachverarbeitung

OWL Varieties

OWL Full Includes full RDFS, powerful, but undecidable[2]

OWL DL Sublanguage of OWL Full, based on description logics, efficient reasoning support, decidable, no full compatibility with RDFS

OWL Lite Sublanguage of OWL DL, only simple constraints

OWL 2 Profiles Sublanguages of OWL DL (new in OWL 2): OWL 2 EL, OWL 2 QL, OWL 2 RL.

▷ We will use OWL DL in this course.

[2] A logical system is decidable if there is an effective method for determining whether arbitrary formulas are theorems of the logical system [Wikipedia]

Institut für Maschinelle Sprachverarbeitung

Writing OWL

- There are many ways to write OWL, we can use:

Turtle
```
ex:JohnSmith rdf:type ex:Person .
```

RDF/XML
```
<ex:Person rdf:about="#JohnSmith"/>
```

OWL/XML
```
<ClassAssertion>
    <Class IRI="Person" />
    <NamedIndividual IRI="JohnSmith" />
</ClassAssertion>
```

Functional
```
ClassAssertion( :Person :JohnSmith )
```

Manchester
```
Individual : JohnSmith
Types : Person
```

DLs $Person(\text{JohnSmith})$

- We will use Turtle syntax in this course, and to some extent Description Logic syntax

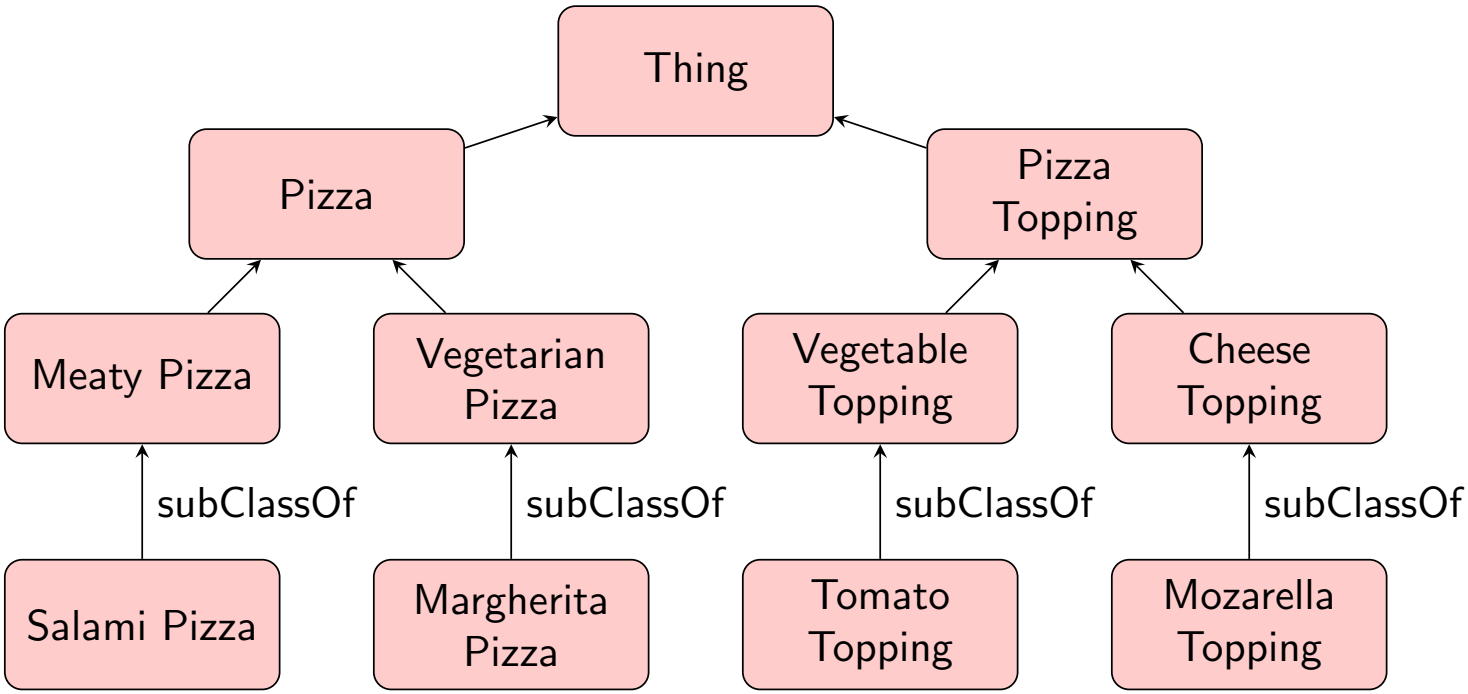
Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

The Pizza Ontology – Classes

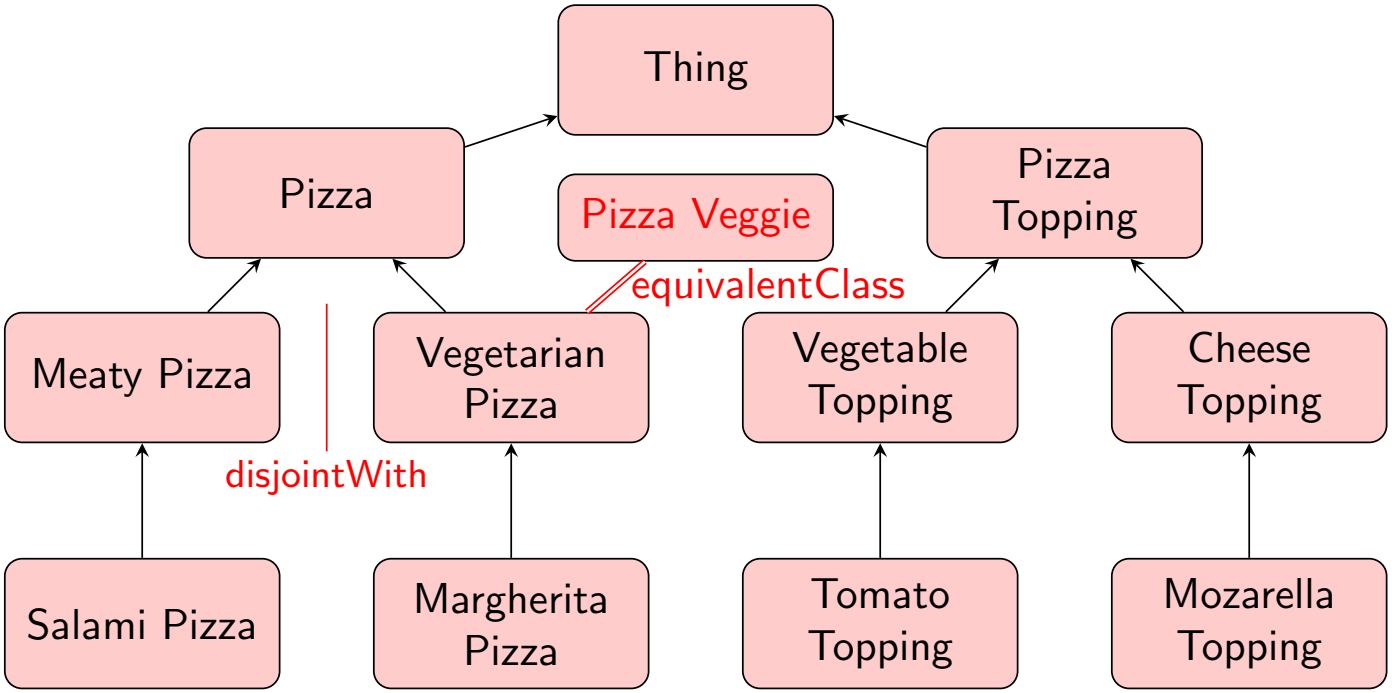
- An OWL ontology contains a hierarchy (taxonomy) of classes



Institut für Maschinelle Sprachverarbeitung

The Pizza Ontology – Classes

- An OWL ontology contains a hierarchy (taxonomy) of classes
- An OWL ontology can describe complex relations between classes



Institut für Maschinelle Sprachverarbeitung

OWL Classes ($A \equiv B,\ A \sqsubseteq \neg B$)

```
ex:Pizza rdf:type owl:Class .
ex:Pizza rdfs:subClassOf ex:Food .
ex:Pizza owl:disjointWith ex:IceCream .
ex:Pizza owl:equivalentClass dbpedia:Pizza .
```

- `owl:Class` is a *subclass* of `rdfs:Class`
- Class hierarchy is specified with `rdfs:subClassOf` ($A \sqsubseteq B$)
- A class can be equivalent to (`owl:equivalentClass`) another class, i.e., all instances of the first class are aso instances of the second class ($A \equiv B$)
- A class can be disjoint with (`owl:disjointWith`) another class, i.e., nothing can be an instance of both classes ($A \sqcap B \sqsubseteq \bot$)

Institut für Maschinelle Sprachverarbeitung

OWL Predefined Classes ($\top$, $\bot$)

- OWL contains two predefined classes
- `owl:Thing` is the class that contains everything
  (every element in the ontology is a thing, $\top \equiv A \sqcup \neg A$)
- `owl:Nothing` is the class that contains nothing
  (the empty class, $\bot \equiv A \sqcap \neg A$).
- Every class is a subclass of `owl:Thing`
- Every class is a superclass of `owl:Nothing`
- If a class is a subclass of `owl:Nothing`, it cannot have instances, this is usually not what you want

Institut für Maschinelle Sprachverarbeitung

Quiz: OWL Classes

```
ex:Apple rdfs:subClassOf ex:Fruit .
ex:Apple owl:disjointWith ex:Pear .
```

Which of these statements are correct?

A) Every `ex:Apple` is a `ex:Fruit`.

B) Every `ex:Fruit` is a `ex:Apple`.

C) `ex:Apple` and `ex:Fruit` contain exactly the same instances.

D) No `ex:Apple` is a `ex:Pear`.

E) Every `ex:Apple` is a `ex:Pear`.

F) Nothing can be a `ex:Pear` and a `ex:Fruit` at the same time.

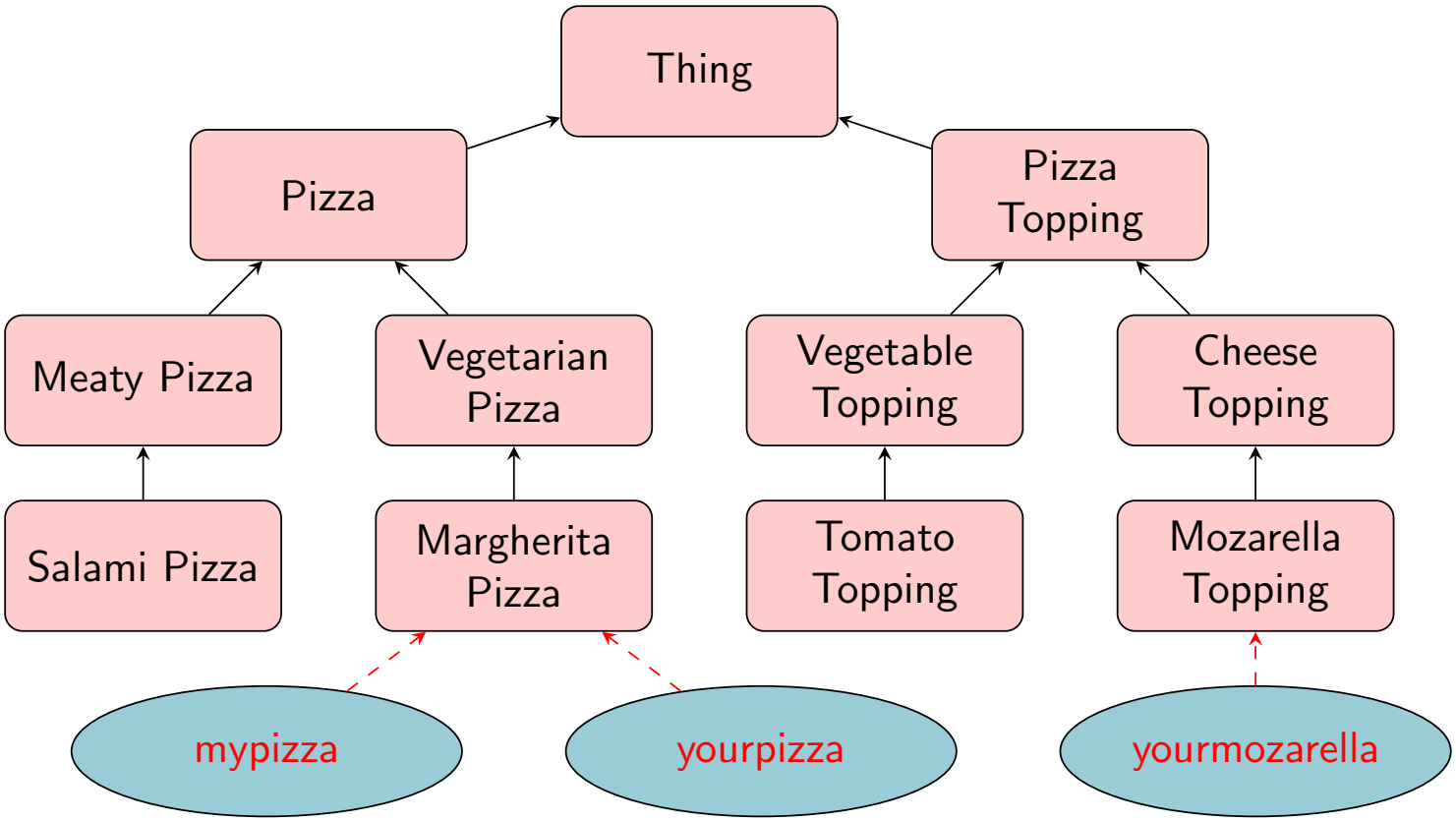
G) Every `ex:Pear` is a `ex:Fruit`.

H) Every `ex:Fruit` is a `ex:Pear`.

Institut für Maschinelle Sprachverarbeitung

The Pizza Ontology – Instances

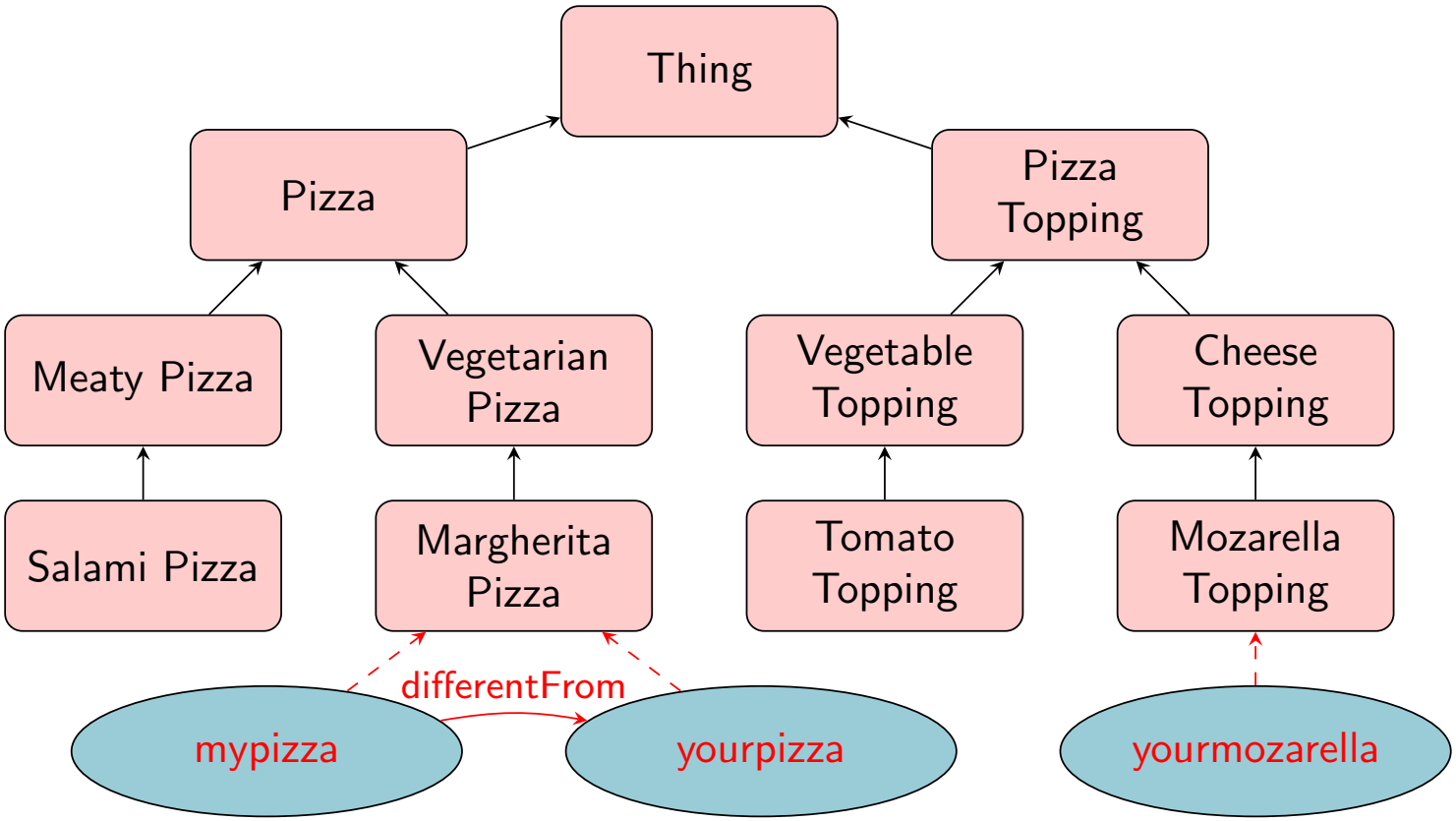
- An OWL ontology contains individuals that are instances of classes



Institut für Maschinelle Sprachverarbeitung

The Pizza Ontology – Instances

- An OWL ontology contains individuals that are instances of classes
- Individuals can have relations with each other



Institut für Maschinelle Sprachverarbeitung

OWL Instances ($\{a\} \equiv \{b\},\ \{a\} \sqsubseteq \neg\{b\}$)

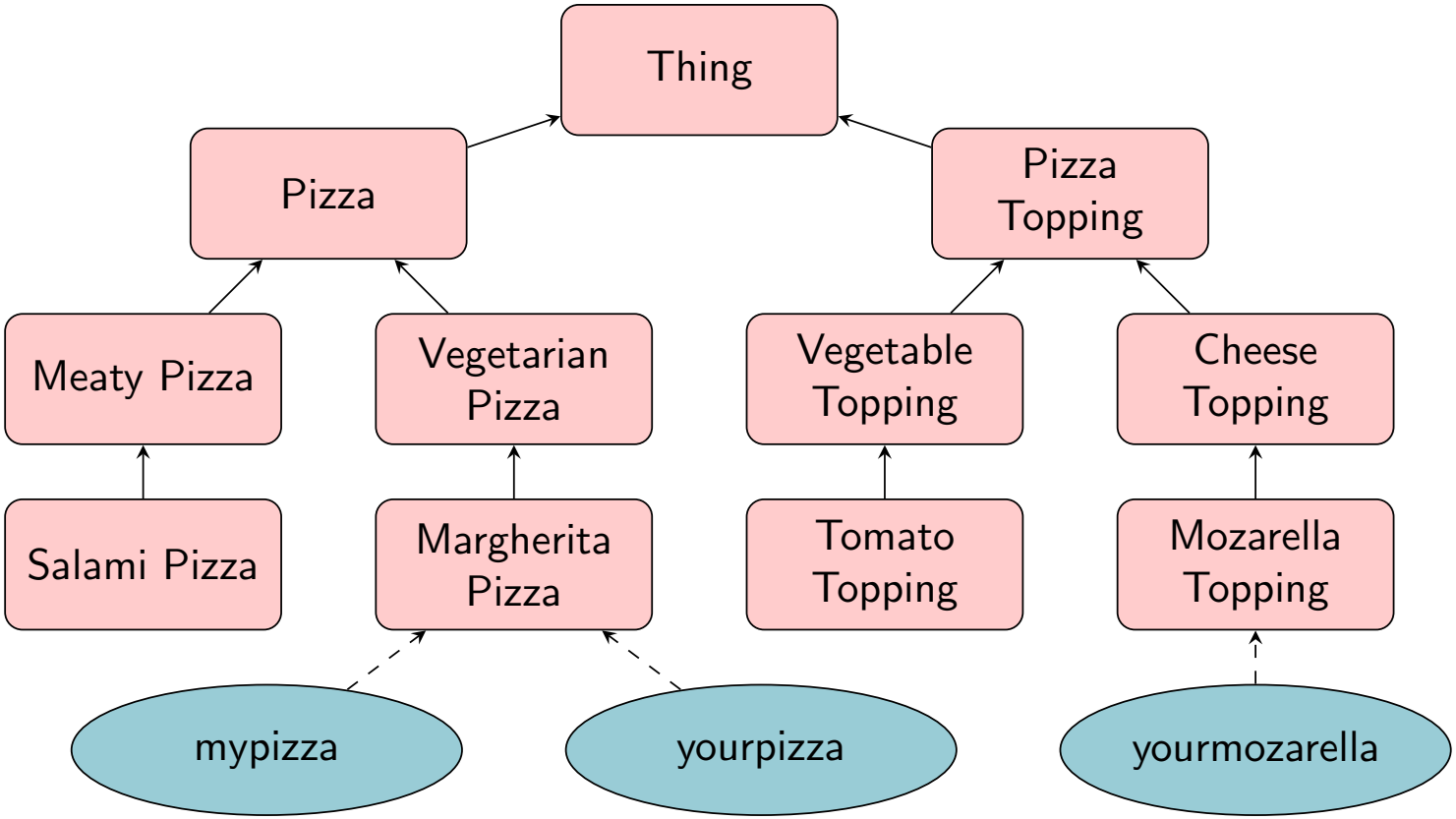
```
ex:mymargheritapizza owl:differentFrom ex:yourmargheritapizza .
ex:mymargheritapizza owl:sameAs ex:myTastymargheritapizza .
```

- Generally two different URIs may refer to the same individual (no unique name assumption).
- `owl:sameAs` explicitly specifies that two URIs denote the same real-world individual ($\{a\} \equiv \{b\}$).
- To specify that two individuals are certainly different, `owl:differentFrom` can be used ($\{a\} \sqsubseteq \neg\{b\}$).

Institut für Maschinelle Sprachverarbeitung

The Pizza Ontology – Properties

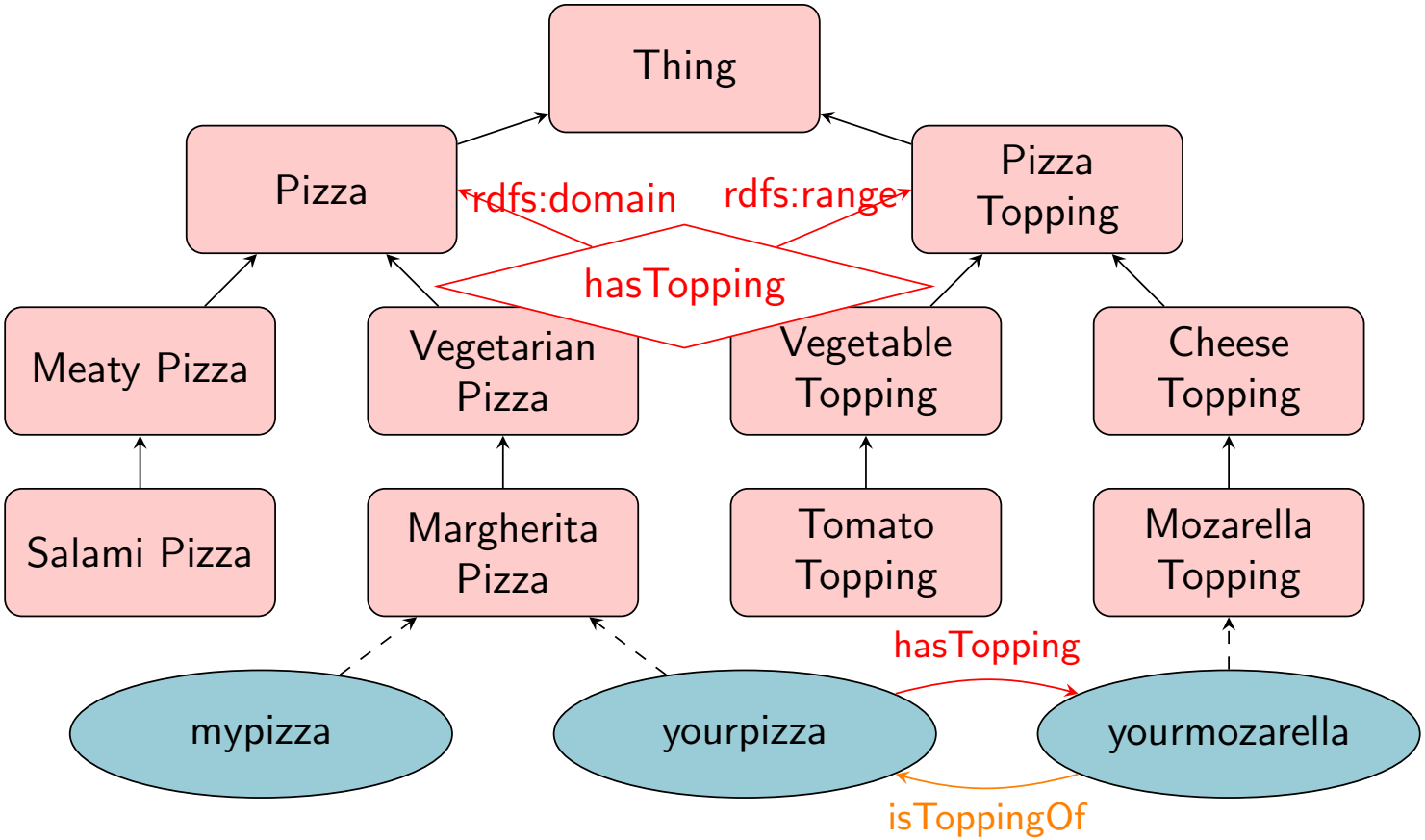
- An OWL ontology allows arbitrary properties



Institut für Maschinelle Sprachverarbeitung

The Pizza Ontology – Properties

- An OWL ontology allows arbitrary properties
- Many different characteristics of properties can be modeled



Institut für Maschinelle Sprachverarbeitung

OWL Properties - Data vs. Object

```
ex:hasPrice rdf:type owl:DatatypeProperty .
ex:hasPrice rdfs:domain ex:Pizza .
ex:hasPrice rdfs:range xs:float .

ex:hasTopping rdf:type owl:ObjectProperty .
ex:hasTopping rdfs:domain ex:Pizza .
ex:hasTopping rdfs:range ex:PizzaTopping .
```

- OWL distinguishes **data properties** from **object properties**:
  1. **data** properties have **literals** as objects of statements
  2. **object** properties have **resources** as objects of statements
- OWL uses XML Schema data types
- Domain and range are specified using RDFS

Institut für Maschinelle Sprachverarbeitung

Equivalent and Disjoint Properties ($r \equiv s,\ r \sqsubseteq \neg s$)

```
ex:hasTopping owl:equivalentProperty ex:hasPizzaTopping .
ex:hasTopping owl:propertyDisjointWith ex:hasPizzaBase .
ex:hasTopping rdfs:subPropertyOf ex:hasIngredient .
```

- Two properties can be defined as equivalent (connecting the same subject-object pairs) with `owl:equivalentProperty`
- Two properties can be defined as disjoint (a subject-object pair can only be connected by at most one, never both) with `owl:propertyDisjointWith`
- Properties can form hierarchies with `rdfs:subPropertyOf`

Institut für Maschinelle Sprachverarbeitung

Inverse Properties

```
ex:hasTopping rdfs:domain ex:Pizza .
ex:hasTopping rdfs:range ex:PizzaTopping .
ex:hasTopping owl:inverseOf ex:isToppingOf .
```

- Inverse properties can be declared using `owl:inverseOf`
- If `s` is the inverse property of `r`, for every OWL statement `a r b.` that is given, we can infer that `b s a.`
- **Example:** `pz:mymizza pz:hasTopping pz:mymozarella .`
  $\rightarrow$ `pz:mymozarella pz:isToppingOf pz:mypizza .`

Institut für Maschinelle Sprachverarbeitung

Quiz: OWL Properties

```
ex:isRipeInMonth rdf:type owl:ObjectProperty .
ex:isRipeInMonth rdfs:domain ex:Fruit .
ex:isRipeInMonth rdfs:range ex:Month .
ex:isRipeInMonth owl:inverseOf ex:isHarvestMonthOf .
ex:elstar owl:ripeInMonth ex:september .
```

Which of these statements are correct?

A) `ex:isHarvestMonthOf` has the domain `ex:Month`

B) `ex:isHarvestMonthOf` has the domain `ex:Fruit`

C) `ex:isHarvestMonthOf` has the range `ex:Month`

D) `ex:isHarvestMonthOf` has the range `ex:Fruit`

E) `ex:september ex:isHarvestMonthOf ex:elstar .`
is inferred from the above

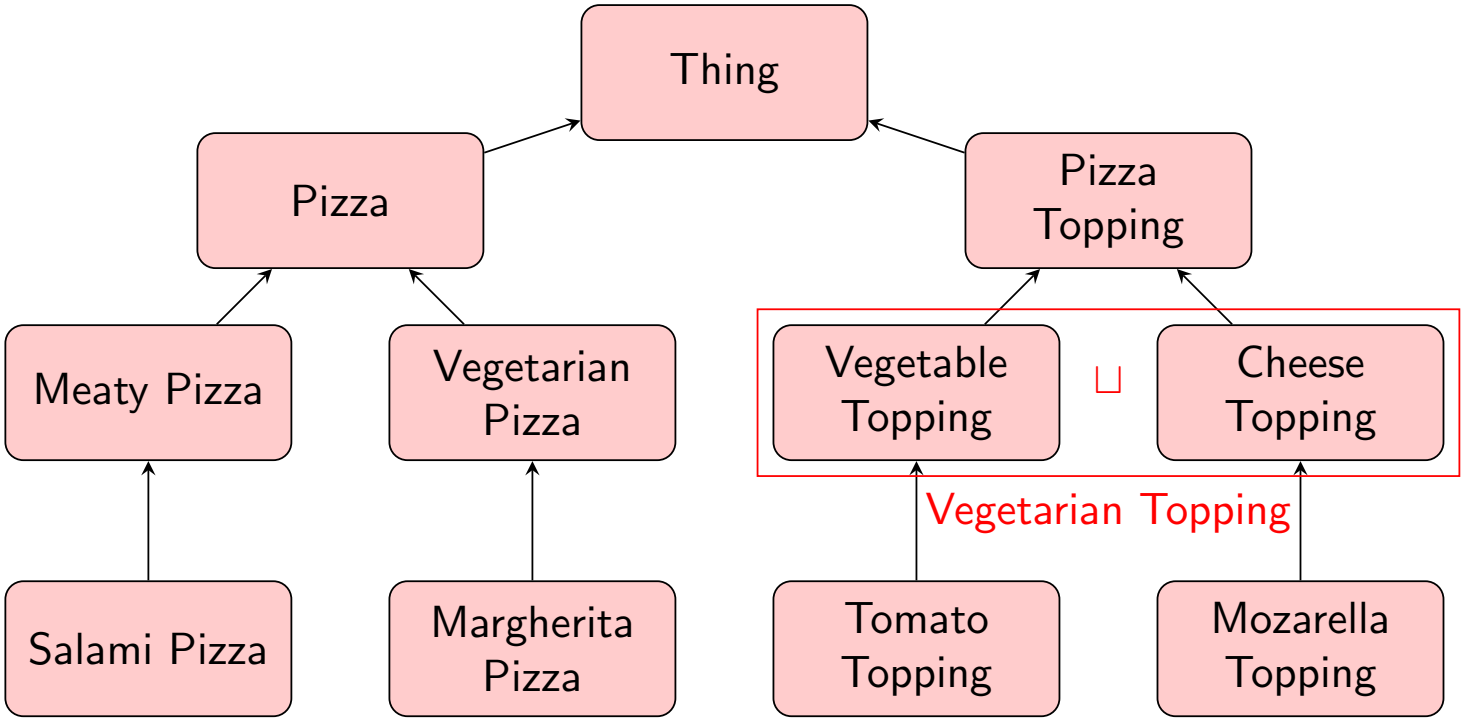
Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

The Pizza Ontology – New Classes with Logical Operators

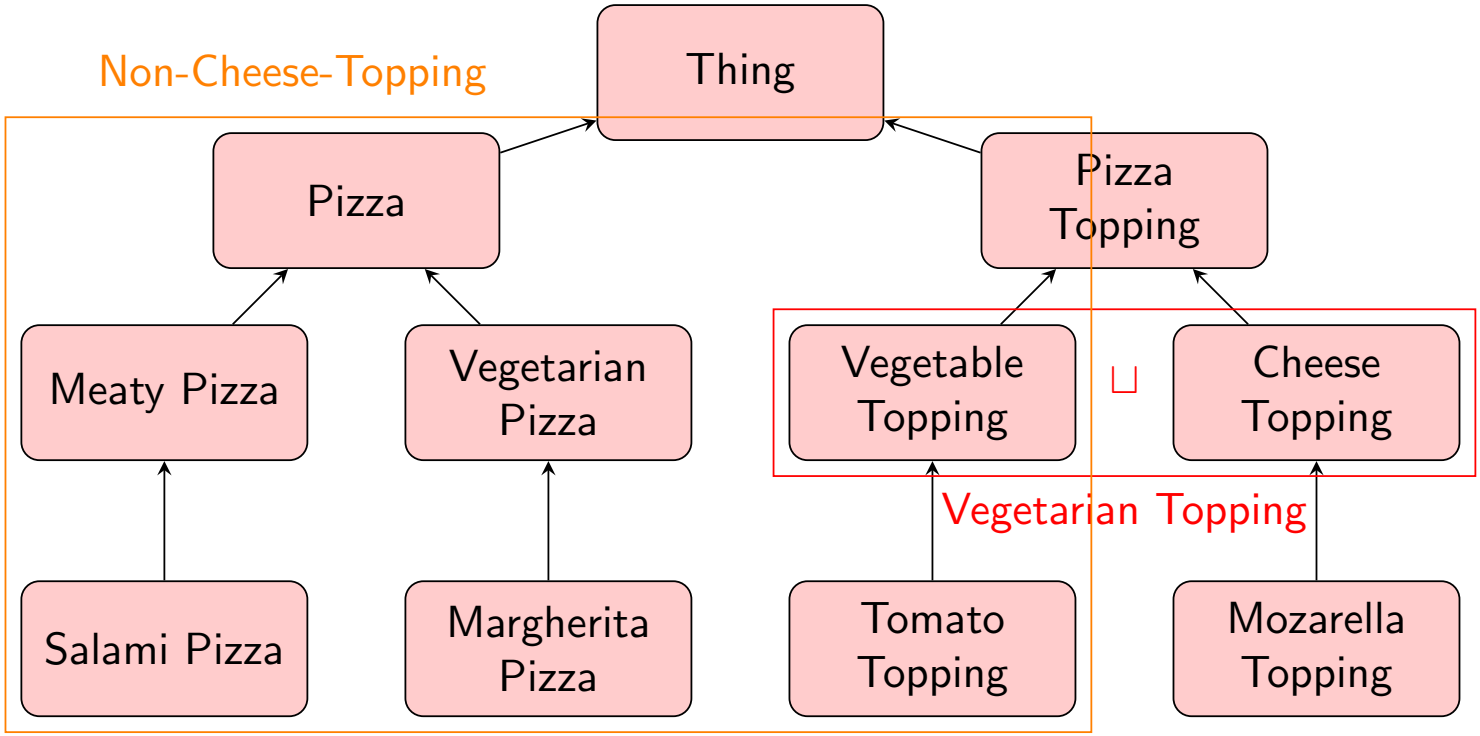
- An ontology allows the definition of new classes by combining existing classes with logical operators (*and*, *or* and *not*)



Institut für Maschinelle Sprachverarbeitung

The Pizza Ontology – New Classes with Logical Operators

- An ontology allows the definition of new classes by combining existing classes with logical operators (*and*, *or* and *not*)

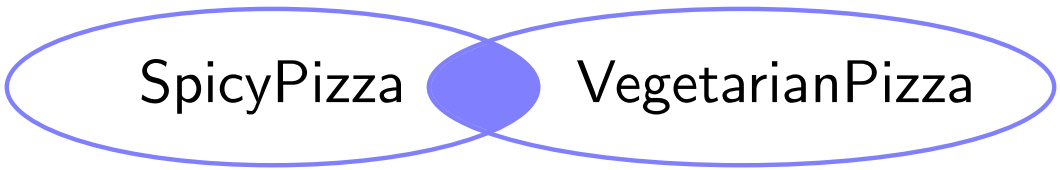


Institut für Maschinelle Sprachverarbeitung

New Classes with Logical Operators – Intersection ($A \cap B$)

```
_:X owl:intersectionOf (ex:SpicyPizza ex:VegetarianPizza) .
```

- `owl:intersectionOf` (logical *and*, $C_1 \sqcap C_2$) contains all individuals that are members of both intersected classes
- **Example:** A spicy vegetarian pizza is spicy *and* vegetarian

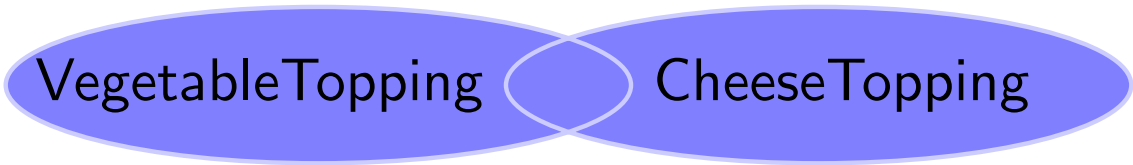


Institut für Maschinelle Sprachverarbeitung

New Classes with Logical Operators – Union ($A \cup B$)

```
_:X owl:unionOf (ex:VegetableTopping ex:CheeseTopping) .
```

- `owl:unionOf` (logical *or*, $C_1 \sqcup C_2$) contains all individuals that are members of one of the two intersected classes
- **Example:** A vegetarian topping may be a cheese topping *or* a vegetable topping (or both)

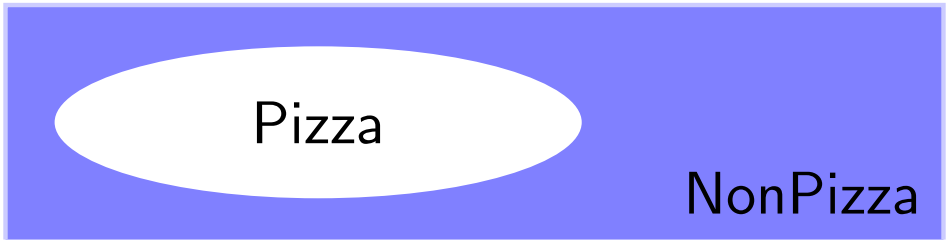


Institut für Maschinelle Sprachverarbeitung

New Classes with Logical Operators – Complement ($\neg A$)

```
_:X owl:complementOf ex:Pizza .
```

- `owl:complementOf` (logical *not*, negation, $\neg C$) describes the complement of a class C i.e., all individuals that are not in C
- **Example:** A non-pizza is everything that is not a pizza[3]



[3] This includes other food, objects, persons, animals, . . . , without further restrictions this class is probably not very useful.

Institut für Maschinelle Sprachverarbeitung

Using Classes with Logical Operators

- The combination of classes with logical operators describes an anonymous (unnamed) class
- Other classes can be a declared to be a subclass or equivalent to this anonymous class
- **Examples:**

```
ex:NonPizza rdfs:subClassOf _:X .
_:X rdf:type owl:Class .
_:X owl:complementOf ex:Pizza .

ex:VegetarianTopping owl:equivalentClass [
   rdf:type owl:Class ;
   owl:unionOf (ex:VegetableTopping ex:CheeseTopping)
] .
```

Institut für Maschinelle Sprachverarbeitung

Quiz: OWL Logical Operators

Given this OWL fragment:

```
ex:abc rdf:type owl:ObjectProperty .
ex:abc rdfs:range [
   rdf:type owl:Class ;
   owl:intersectionOf (ex:CitrusFruit ex:StoneFruit)
] .
```

Which of these statements are correct?

A) Subjects of `ex:abc` may be either stone fruits or citrus fruits

B) Objects of `ex:abc` may be either stone fruits or citrus fruits

C) Subjects of `ex:abc` must be both stone fruits and citrus fruits

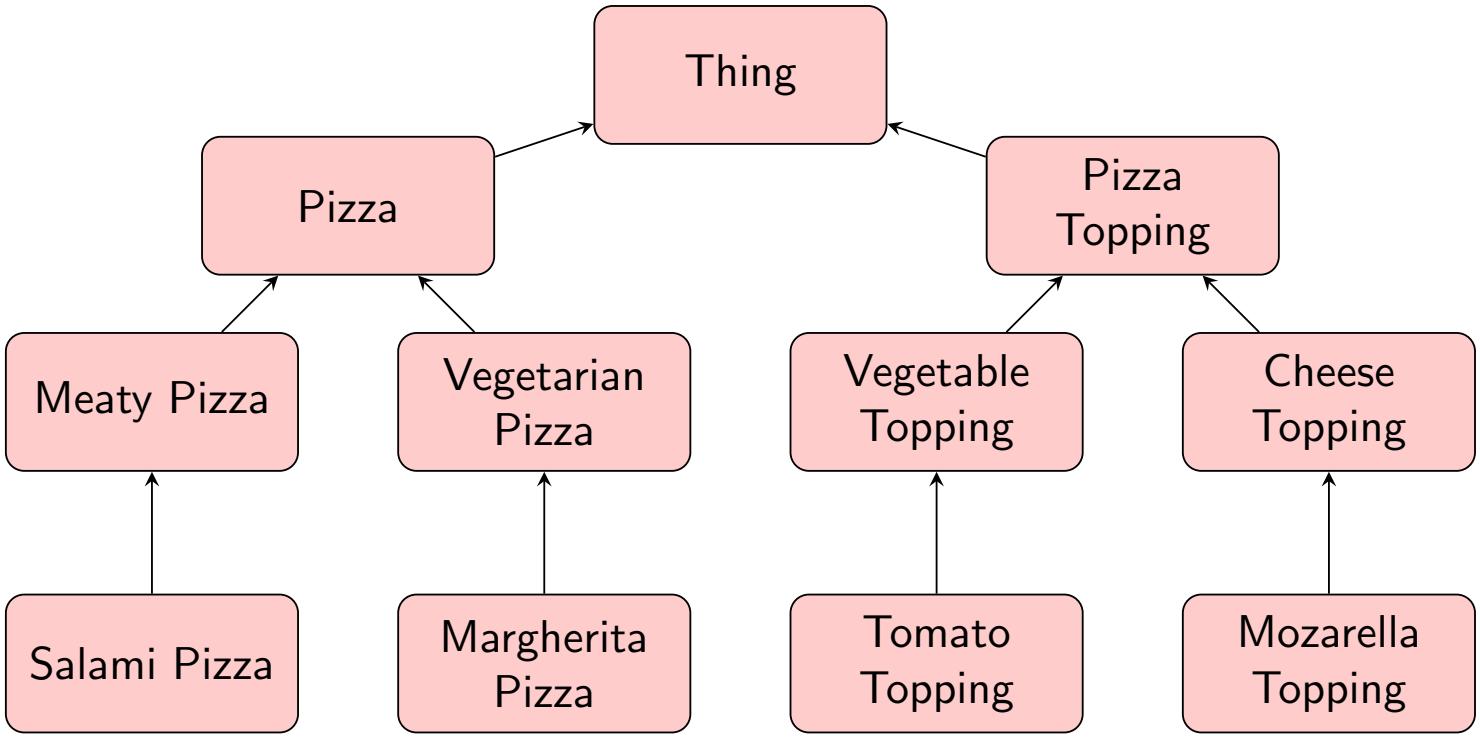
D) Objects of `ex:abc` must be both stone fruits and citrus fruits

E) We could have said the same with two `rdfs:range` statements

Institut für Maschinelle Sprachverarbeitung

The Pizza Ontology – New Classes with Restrictions

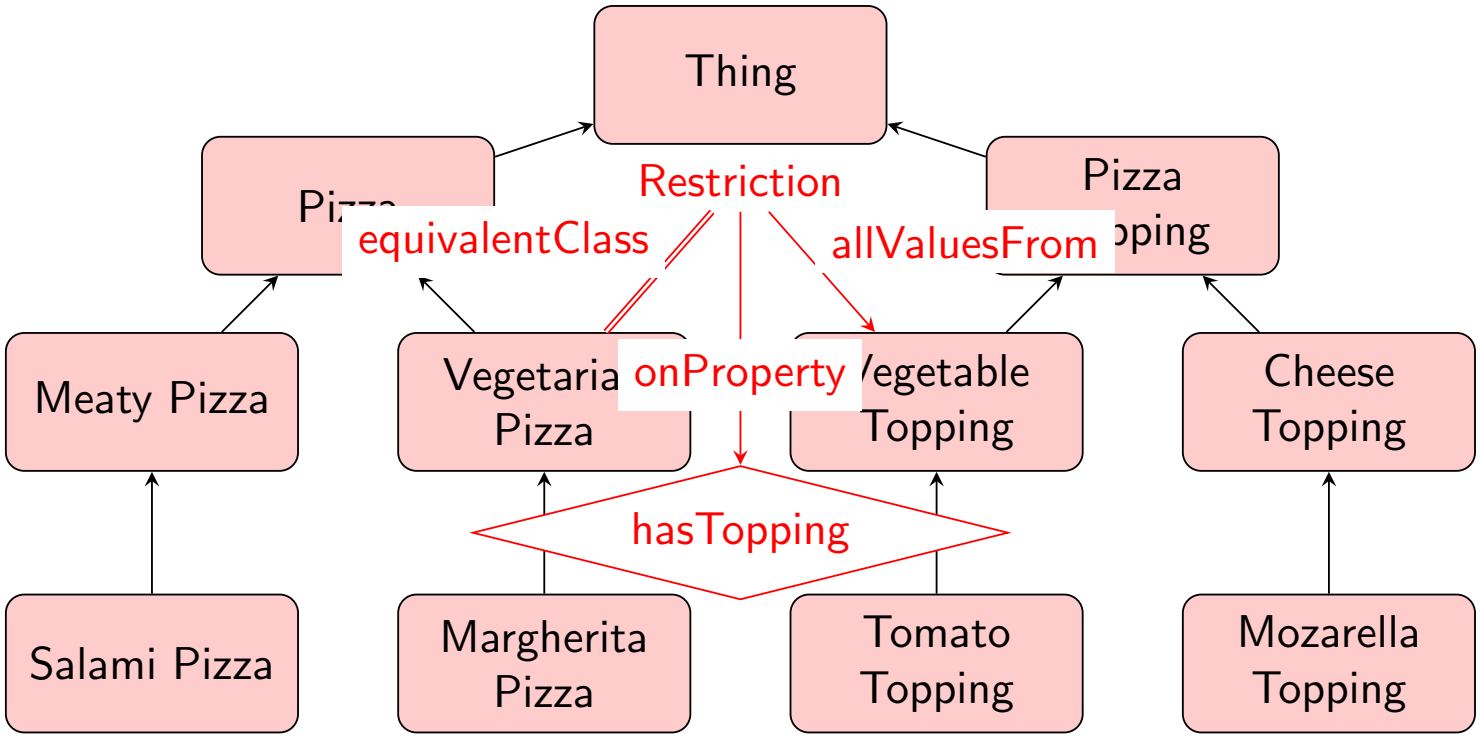
- An ontology allows the definition of **restrictions** **on the number and nature of objects connected by a given property**



Institut für Maschinelle Sprachverarbeitung

The Pizza Ontology – New Classes with Restrictions

- An ontology allows the definition of **restrictions** **on the number and nature of objects connected by a given property**



New Classes with Restrictions

- A restriction works on a specific property by constraining its values (its range)
- A restriction describes an anonymous class which contains all individuals that are connected to the restricted property
- The class we want to describe with the restriction is usually a subclass of the anonymous class

- **Examples:**
  - `InterestingPizza` is (among other things) a subclass of the things that have at least three toppings
  - `CheesePizza` is (among other things) a subclass of the things where all toppings are `CheeseTopping`
  - `ItalianPizza` is (among other things) a subclass of the things that originate in Italy
  - `FishPizza` is (among other things) a subclass of the things where some toppings are `FishTopping`

Institut für Maschinelle Sprachverarbeitung

Universal Restrictions ($\forall r.A$)

```
ex:CheesePizza rdf:type owl:Class .
ex:CheesePizza rdfs:subClassOf [
   rdf:type owl:Restriction ;
   owl:onProperty ex:hasTopping ;
   owl:allValuesFrom ex:CheeseTopping
] .
```

- `owl:allValuesFrom` is used to specify the class of possible values the property specified by `owl:onProperty` can take (equivalent to universal quantification $\forall$)
- **Example:** A cheese pizza has *only* cheese toppings, *all* toppings of a cheese pizza are cheese toppings

Institut für Maschinelle Sprachverarbeitung

Existential Restrictions ($\exists r.A$)

```
ex:FishPizza rdf:type owl:Class .
ex:FishPizza rdfs:subClassOf [
   rdf:type owl:Restriction ;
   owl:onProperty ex:hasTopping ;
   owl:someValuesFrom ex:FishTopping
] .
```

- `owl:someValuesFrom` specifies that there must be at least one statement where the value of the property is from this class (equivalent to existential quantification $\exists$)
- **Example:** A fish pizza has *at least one* fish topping, *some* toppings of a fish pizza are fish toppings

Institut für Maschinelle Sprachverarbeitung

Cardinality Restrictions ($\exists_{\leq k} r.A$, $\exists_{\geq k} r.A$, $\exists_{=k} r.A$)

```
ex:InterestingPizza rdf:type owl:Class .
ex:InterestingPizza rdfs:subClassOf [
   rdf:type owl:Restriction ;
   owl:onProperty ex:hasTopping ;
   owl:someValuesFrom ex:PizzaTopping ;
   owl:minCardinality "3"^^xs:nonNegativeInteger
] .
```

- We can restrict the minimum (`owl:minCardinality`), maximum (`owl:maxCardinality`) or set a specific (`owl:cardinality`) number of objects connected to an instance of the class with a given property
- **Example:** An interesting pizza has at least three toppings

Institut für Maschinelle Sprachverarbeitung

Value Restrictions ($r.\{a\}$)

```
ex:ItalianPizza rdf:type owl:Class .
ex:ItalianPizza rdfs:subClassOf [
   rdf:type owl:Restriction ;
   owl:onProperty ex:hasCountryOfOrigin ;
   owl:hasValue ex:italy
] .
```

- `owl:hasValue` restricts the allowed value for a property to exactly one individual or data value, there has to be at least one statement with this value
- **Example:** An Italian pizza has to originate from Italy (Italy is an instance, not a class)

Institut für Maschinelle Sprachverarbeitung

Quiz: OWL Restrictions 1

```
pz:mypizza pz:hasCountryOfOrigin pz:italy .
pz:mypizza pz:hasTopping pz:mymozarella1 .
pz:mypizza pz:hasTopping pz:mymozarella2 .
pz:mypizza pz:hasTopping pz:mysalmon1 .
```

Of which classes is `pz:mypizza` an instance (assuming the definitions given in the previous slides)?

A) `pz:InterestingPizza`

B) `pz:CheesePizza`

C) `pz:FishPizza`

D) `pz:ItalianPizza`

Institut für Maschinelle Sprachverarbeitung

Quiz: OWL Restrictions 2

```
_:A owl:intersectionOf (_:X _:Y) .

_:X rdf:type owl:Restriction .
_:X owl:onProperty ex:harvestedInMonth .
_:X owl:hasValue ex:September .

_:Y rdf:type owl:Restriction .
_:Y owl:onProperty ex:harvestedInMonth .
_:Y owl:cardinality "1"^^xs:nonNegativeInteger .
```

Fruit from this class is harvested . . .

A) . . . in the month September (and maybe in another month)

B) . . . in the month September (and at least one other month)

C) . . . only in the month September (and no other month)

D) . . . in any month but September

Institut für Maschinelle Sprachverarbeitung

Subclass and Equivalent Class

There is an important distinction between `owl:equivalentClass` and `rdfs:subClassOf`:

- `owl:equivalentClass` means that the classes are **exactly the same**, it gives a definition of the class. Every instance of one class is also an instance of the other
- `rdfs:subClassOf` means that one class is a **subclass** of the other; every instance of the subclass is also an instance of the superclass, but there are instances of the superclass that are not instances of the subclass

Confusing the two relations is a common modeling mistake.

Institut für Maschinelle Sprachverarbeitung

Quiz: Subclasses and Equivalent Classes

Should `<?????>` be replaced with `owl:equivalentClass` or `rdfs:subClassOf`?

A)
```
ex:VegetarianPizza rdf:type owl:Class .
ex:VegetarianPizza <?????> [
   rdf:type owl:Restriction ;
   owl:onProperty ex:hasTopping ;
   owl:allValuesFrom ex:VegetarianTopping
] .
```

B)
```
ex:VegetarianPizza rdf:type owl:Class .
ex:VegetarianPizza <?????> [
   rdf:type owl:Class ;
   owl:intersectionOf (ex:Pizza _:Y)
] .
_:Y rdf:type owl:Restriction .
_:Y owl:onProperty ex:hasTopping .
_:Y owl:allValuesFrom ex:VegetarianTopping .
```

Institut für Maschinelle Sprachverarbeitung

Concluding Remarks

- **No Unique Name Assumption**: If two individuals (or classes or properties) have different names, we may still derive by inference that they must be the same
  - **Example:** `ex:johnsmith` and `ex:pizzaeater1234` may refer to the same person

- **Open World Assumption**: We assume our knowledge-base is incomplete. The lack of a given assertion does not imply whether it is true or false, it is simply not known
  - **Example:** We know that "Mike is a philosopher" and "Rudolph is a professor". We cannot say anything about the membership of Mike in the class "professor"

Institut für Maschinelle Sprachverarbeitung

Expressiveness (Recap)

RDFS	OWL DL	DLs
`A rdfs:subClassOf B.`	=	$A \sqsubseteq B$
`r rdfs:subPropertyOf s.`	=	$r \sqsubseteq s$
`r rdfs:domain A.`	=	$\exists r.\top \sqsubseteq A$
`r rdfs:range B.`	=	$\exists r^{-}.\top \sqsubseteq B$
`a rdf:type A.`	=	$A(a)$
`a r b.`	=	$r(a, b)$
`A rdf:type rdfs:Class.`	`A rdf:type owl:Class.`	
`R rdf:type rdfs:Property.`	`A rdf:type owl:Property.`	
	`A owl:equivalentClass B.`	$A \equiv B$
	`A owl:disjointWith B.`	$A \sqsubseteq \neg B$
	`a owl:sameAs b.`	$\{a\} = \{b\}$
	`a owl:differentFrom b.`	$\{a\} \sqsubseteq \neg\{b\}$
	⋮	

See also https://ragrawal.wordpress.com/2007/02/20/difference-between-owl-lite-dl-and-full/ for an informal comparison

Institut für Maschinelle Sprachverarbeitung

Exercise: Protégé

- Load the pizza ontology with Protégé
- Find the classes we discussed and look at their definitions.
- Create a new class: `pz:PizzaDellaCasa`
- Restrict the newly created `pz:PizzaDellaCasa` so that it has all your favorite toppings
- Compare your pizza with your neighbour's

Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

Summary: Ontologies and OWL

- An **ontology** is a clear, unambiguous, formal model of some part of the real world that is shared by several people
- **OWL** can be used to describe ontologies
- OWL adds to RDFS the possibility to
  - clarify that two individuals are identical or different
  - express that two classes are equivalent or disjoint
  - declare a property to be the inverse of another property
  - construct new classes by taking the union, intersection or complement of other existing classes
  - define classes by specifying restrictions on the cardinality or possible values of properties
- OWL is based on **description logics**, which makes its semantics machine accessible

Institut für Maschinelle Sprachverarbeitung

Outline

Institut für Maschinelle Sprachverarbeitung

Suggested Reading

- Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph and York Sure. Semantic Web. Grundlagen. Springer textbook, 2008. (Chapter 5)
- Pascal Hitzler, Markus Krötzsch and Sebastian Rudolph. Foundations of Semantic Web Technologies. Chapman & Hall/CRC, 2009. (Chapter 4)
- Matthew Horridge. A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools. Edition 1.3. The University Of Manchester, March 24, 2011.

Institut für Maschinelle Sprachverarbeitung