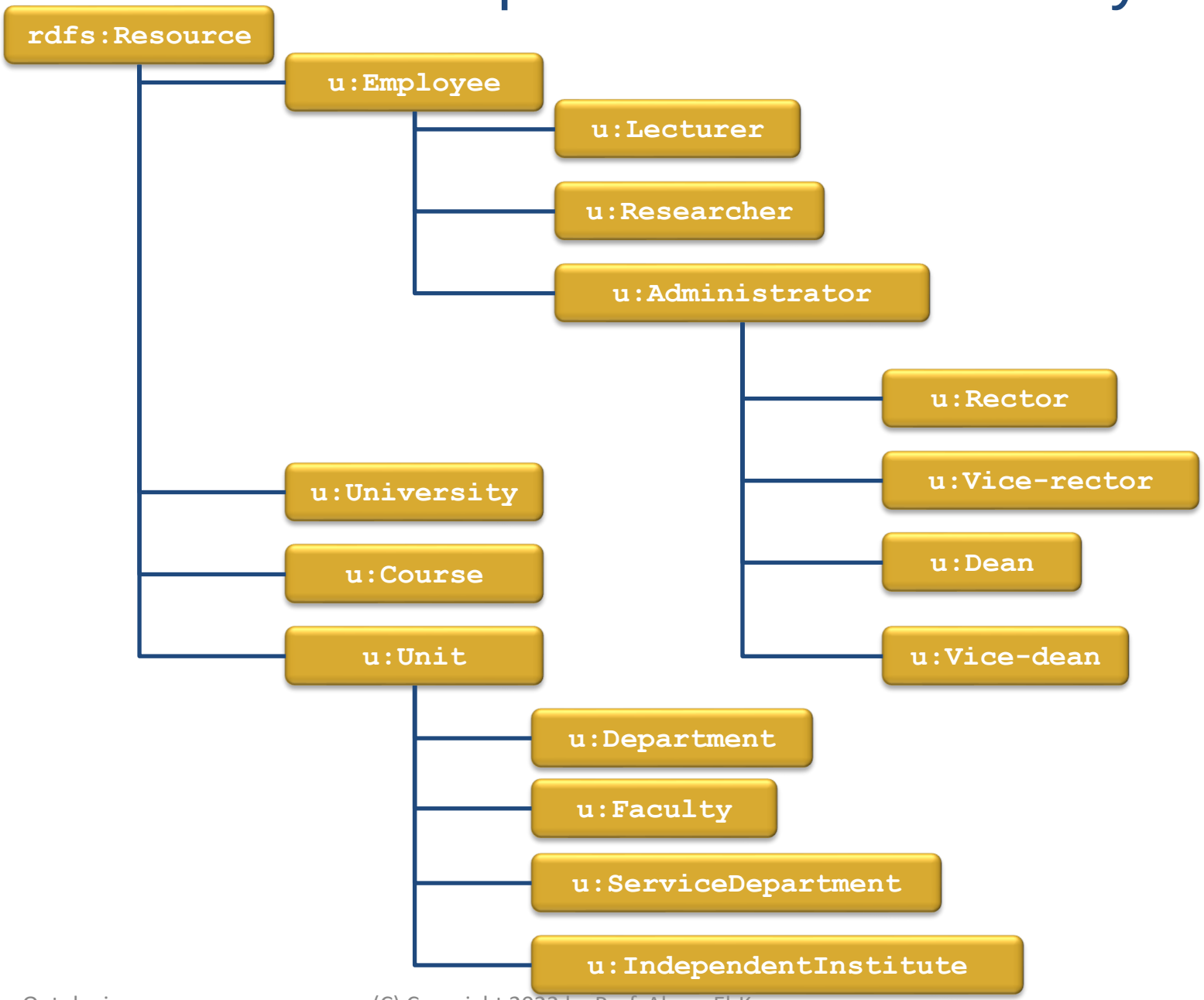
Ontology Development

- Lesson recap : Steps to develop An ontology
- Class and class relations
- Property and property restrictions

Class hierarchy: Modes of Development

- top-down – define the most general concepts first and then specialize them
- bottom-up – define the most specific concepts and then organize them in more general classes
- combination – define the more salient concepts first and then generalize and specialize them

Example of Class Hierarchy



Classes types

- **Primitive concept**:

Usually found near the top of a generalization hierarchy

- **Defined concepts:**

Appear as we move further down by specializing general concepts with various restrictions.

Defining Classes and a Class Hierarchy (cont.)

- Things to remember:
  - There is no single correct class hierarchy
  - But there are some guidelines
- The question to ask:
  - "Is each instance of the subclass an instance of its superclass?"

Class Inheritance

- Classes usually constitute a taxonomic hierarchy (a subclass-superclass hierarchy)
- A class hierarchy is usually an IS-A hierarchy:
  - *An instance of a subclass is an instance of a superclass.*
- If you think of a class as a set of elements, a subclass is a subset
  - Apple is a subclass of Fruit

    Every apple is a fruit

Golden Rules

- *There is no one correct way to model a domain*
  - There are always viable alternatives.
  - Best solution depends on
    - Application
    - Ability to cover extensions
- *Ontology development is an iterative process.*

Golden Rules

- *Concepts in the ontology should be close to objects (physical or logical) and relationships*
- *Relations between Concepts (inheritance, disjoints, equivalents)*

- Very likely: in sentences that describe your domain:
  - Nouns are
  - Verbs and prepositions are

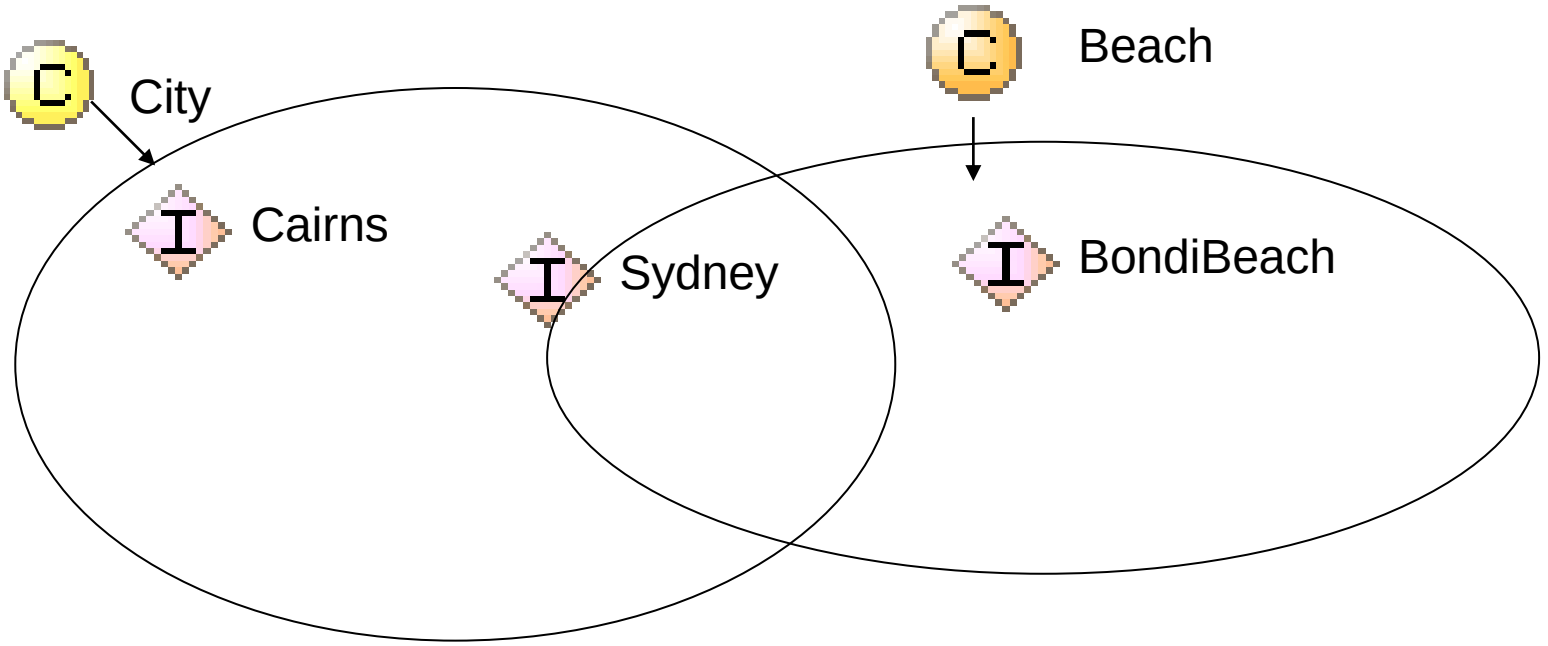
Golden Rules

- *Concepts in the ontology should be close to objects (physical or logical) and relationships*
- *Relations between Concepts (inheritance, disjoints, equivalents)*

- Very likely: in sentences that describe your domain:
  - Nouns are **objects (concepts)**
  - Verbs and prepositions are **relationships**

Class Relationships

- Classes can be organized in a hierarchy
- Direct instances of subclass are also (indirect) instances of superclasses
- Classes can overlap arbitrarily



Disjoint Classes

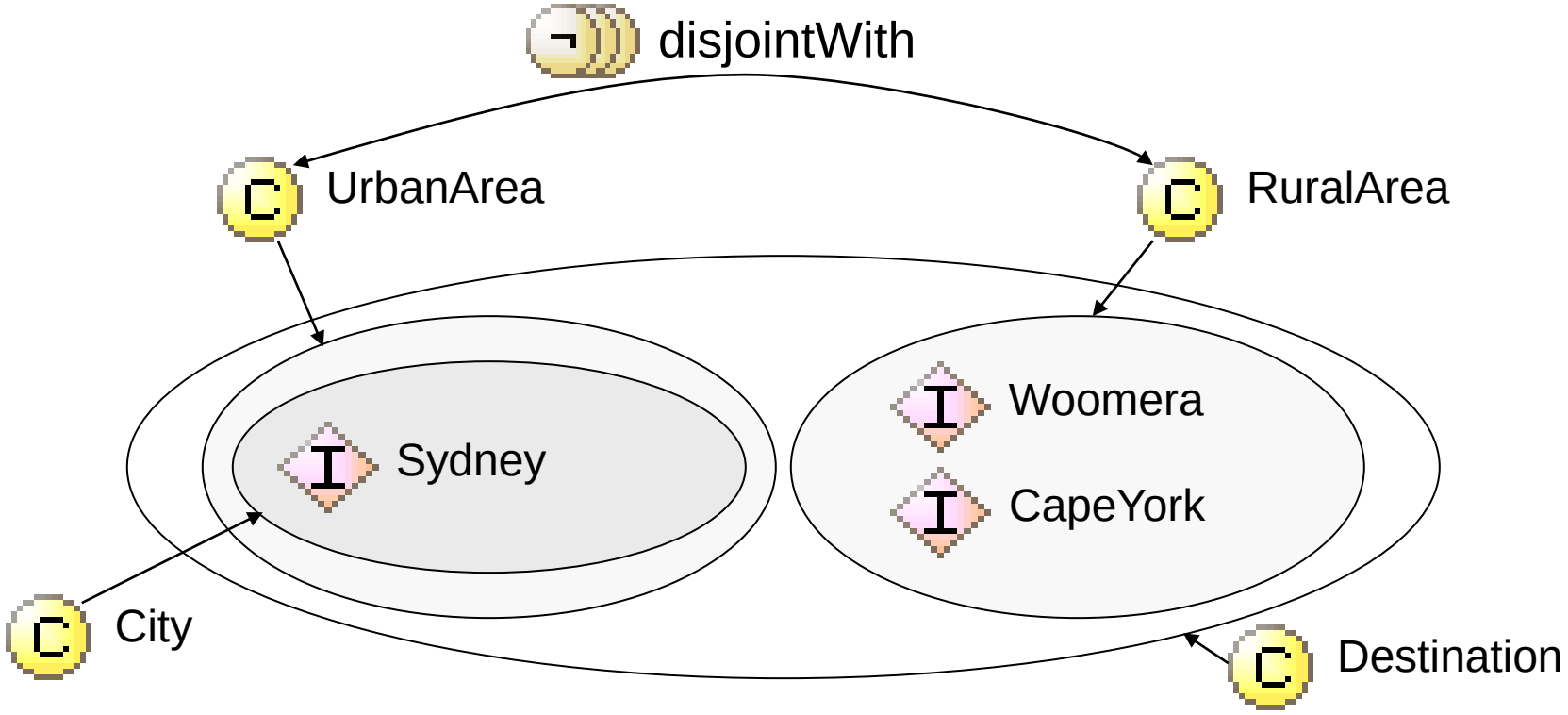
- They are classes that members of the selected class cannot also belong to.
- Classes are disjoint if they cannot have common instances.
- Example: There can be no animal that can be both an Elephant and a Dog.

Classes distinction

- To keep primitives in disjoint
  - need to distinguish the roles things play in different situations from what they are for example:
  - “pet”, “farm animal”, “draft animal”,
  - “doctor”, “nurse”, “patient”
  - “professor”, “student”, …
- Often need to distinguish qualifications from roles
  - A person may be qualified as a doctor but playing the role of a patient

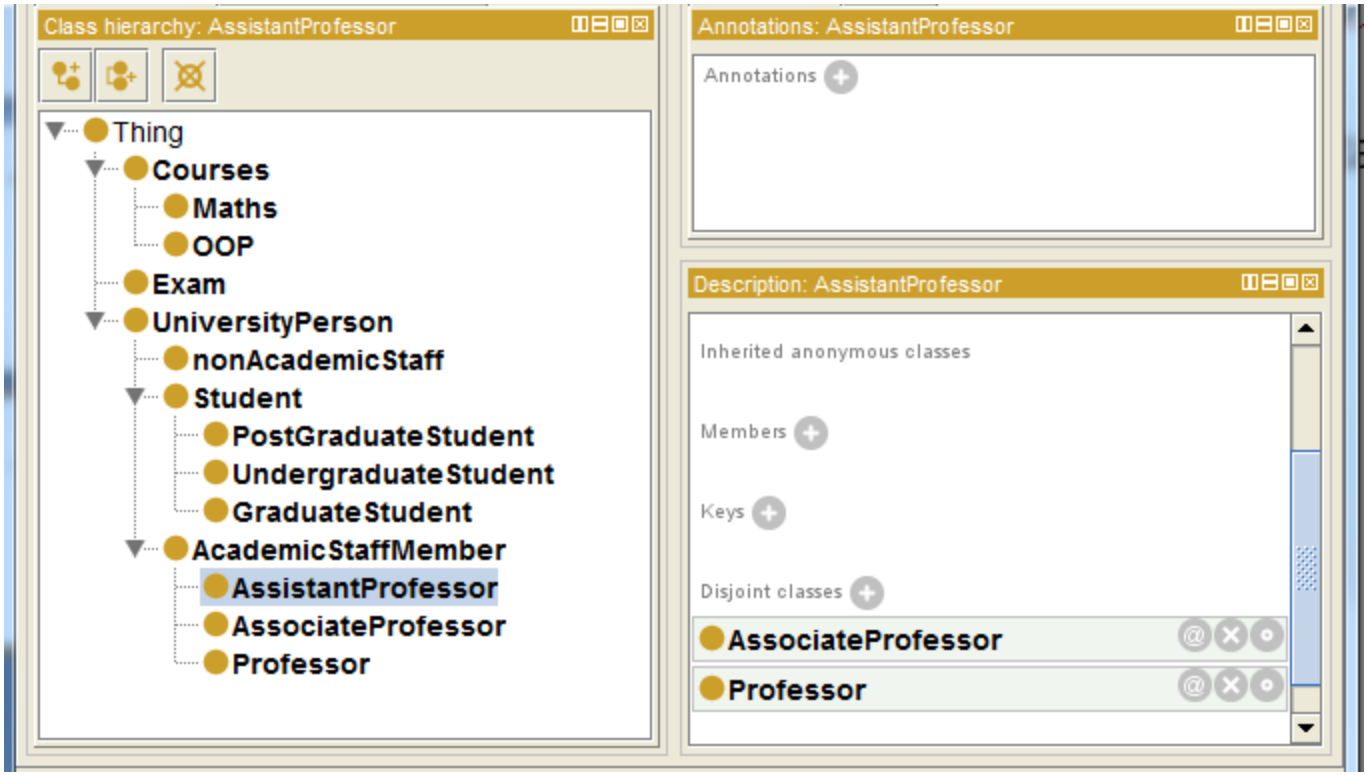
Class Disjointness

- All classes **could potentially** overlap
- In many cases we want to make sure they **don't share instances**



DisjointWith

- Professor and Assistant professor are disjoint

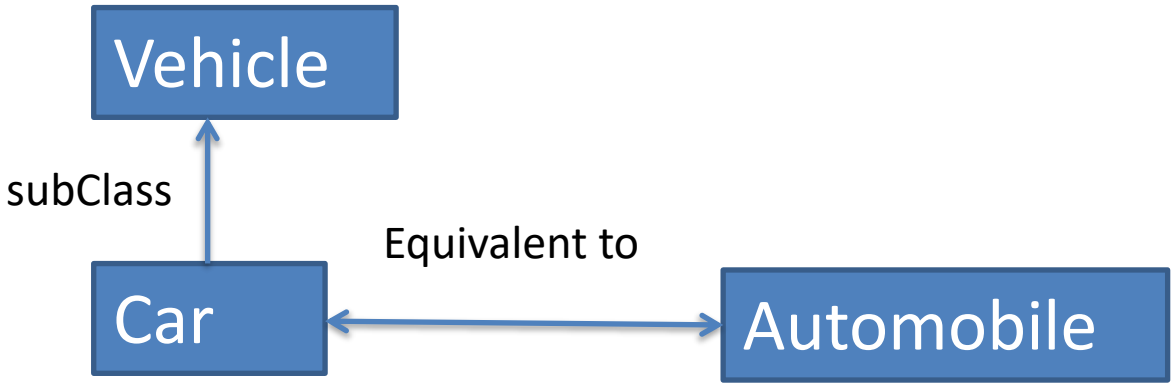


Equivalent classes

- Equivalent classes: other classes or groups that are equivalent to the selected class.

- In the case of **Parent**: it is "any **Person** who has a child".
- Example:
  - USPresident
  - PrincipalResidentOfWhiteHouse
  - <owl:Class rdf:about="#US_President">
    <owl:equivalentClass
    rdf:resource="#PrincipalResidentOfWhiteHouse"/>
    </owl:Class>

Equivalent classes: Example



Define Properties of Classes (Slots)

- Slots in a class definition describe Part of the meaning of a class (and thus participate in classification).
- Derived property to be inferred once class membership is known.

E.g.: Each plant will have name, stem, root, leaves, etc.

Property and Class Inheritance

- A subclass inherits all the slots from the superclass.
  - *If a student has a name and an id, an undergraduate student also has a name and an id.*

- If a class has multiple superclasses, it inherits slots from all of them (Yes, there is multiple inheritance).

Specify Values for each:
Two methods

- Value partitions
  - Classes that partition a Quality
    - The disjunction of the partition classes equals the quality class

- Symbolic values (value set)
  - Individuals that enumerate all states of a Quality
    - The enumeration of the values equals the quality class

Note any hierarchies of values

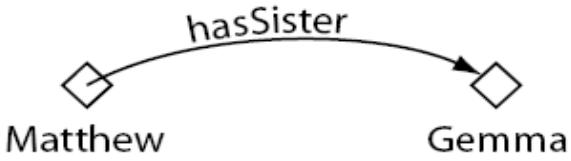
- Modifiers
  - **Domestication**
    - **Domestic**
    - **Wild**
    - **Feral**
  - **Risk**
    - **Dangerous**
    - **Risky**
    - **Safe**
  - **Gender**
    - **Male**
    - **Female**
  - **Age**
    - **Child**
      - **Infant**
      - **Toddler**
    - **Adult**
      - **Elderly**

- Identify modifiers that have mutually exclusive values
  - Domestication
  - Risk
  - Gender
  - Age
- Make meaning precise
  - Age → Age_group

- NB: Some Uses are not mutually exclusive
  - Can be(cattle are both a draft and a food animal)

Properties (Slots)

- Types of properties
  - **Fundamental properties (data properties) :**
    - "intrinsic" properties: flavor and toping of Pizza , **name** and **address** of student
    - "extrinsic" properties: name and price of Pizza/ dish
    - Parts of : **engine** in a car; **ingredients** in a dish
  - **Relations to other object (object properties):**
    - **publisher** of book or producer of an ingredient
- Simple and complex properties
  - **simple properties (data properties**): contain primitive values (string numbers).
  - **complex properties (object properties):** contain (or point to) other objects (e.g., producer of an ingredient, a publisher of a book).

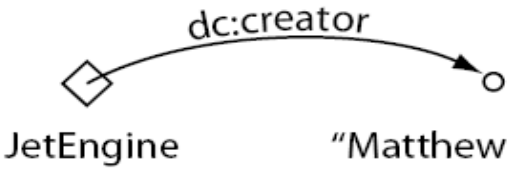


Object Property

An object property linking the individual Matthew to the individual Gemma

Data Property

A datatype property linking the individual Matthew to the data literal '25', which has a type of an xml:integer.

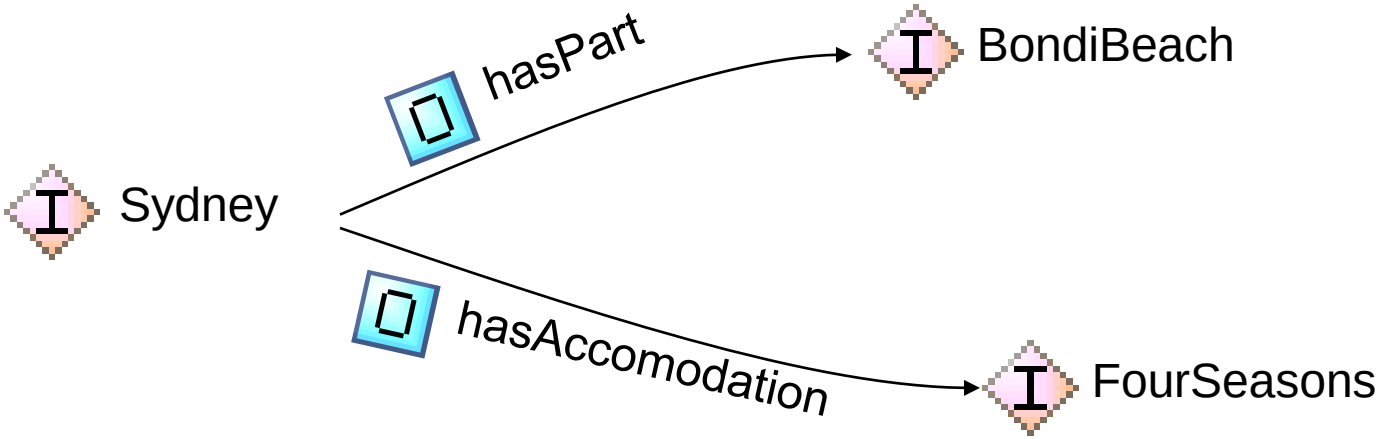


Data Property

An annotation property, linking the class 'JetEngine' to the data literal (string) "Matthew Horridge".

ObjectProperties

- Link two individuals together
- Relationships (0..n, n..m)



Domain and Range of Property

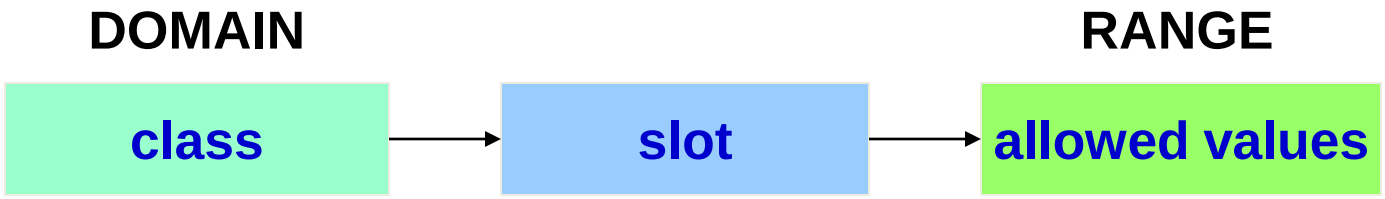
- Domain of a slot – the class (or classes) that have the slot
  - More precisely: class (or classes) instances of which can have the slot
- Range of a slot – the class (or classes) to which **slot values belong**

  - Example: TV show is 'produced by' TV
    - **Domain: TV show**
    - **Range: TV**

Domain and Ranges of Property(cont.)

- *When defining a domain or a range for a slot, find the most general classes or class that can be respectively the domain or the range for the slots .*
- *All the classes in the domain of a slot should be described by the slot*
- *Instances of all the classes in the range of a slot should be potential fillers for the slot.*
- *Do not choose an overly general class for range (i.e., one would not want to make the range THING)*

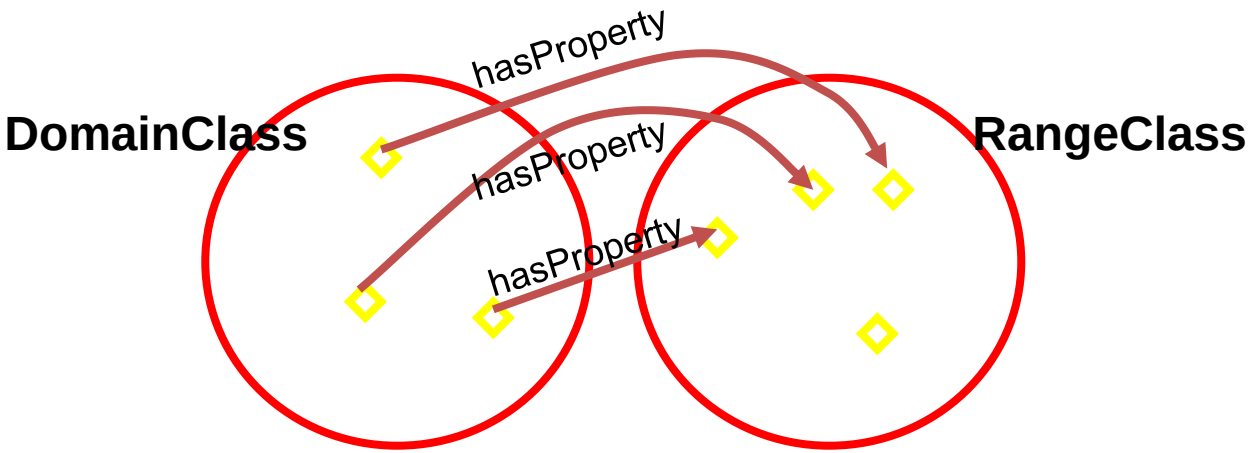
Domain and Ranges of Property (cont.)



- When defining a domain or range for a slot, find **the most general class** or classes
- Consider the **publisher** slot for a **book**:
  - Domain: Book
  - Range: Publisher Instance [Amazon, Elsevier, Springer,…]

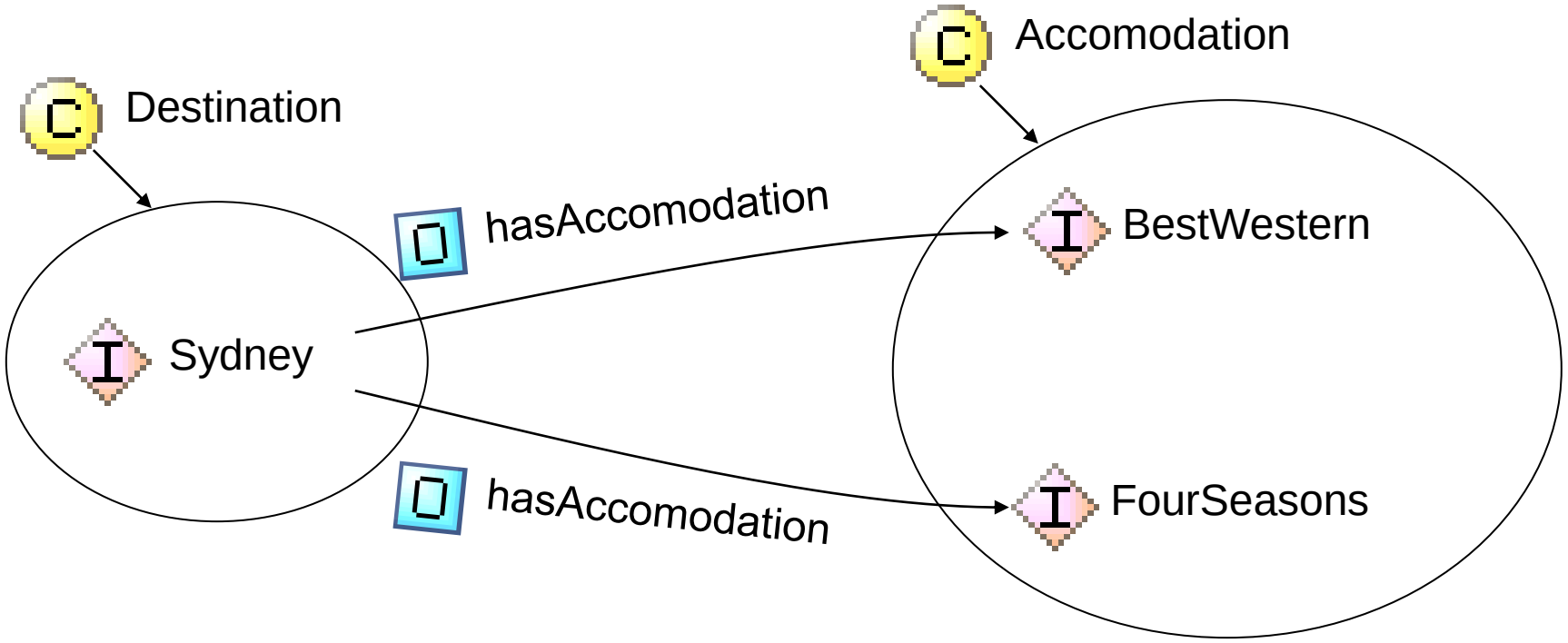
Property Domain & Range

- If a relation is:
  subject_individual → hasProperty → object_individual
- The domain is the class of the subject individual
- The range is the class of the object individual (or a datatype if hasProperty is a Datatype Property)

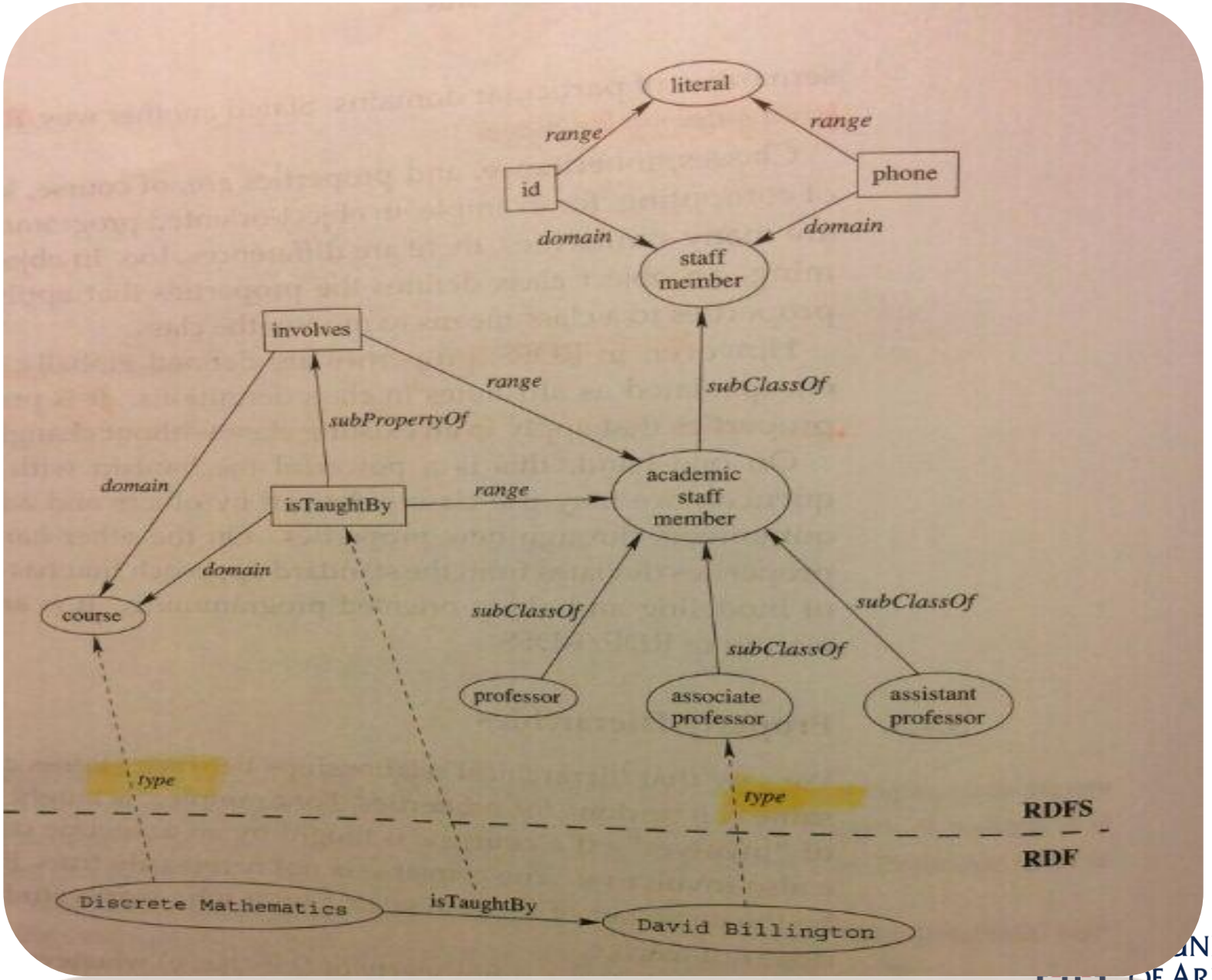


Properties, Range and Domain

- Property characteristics
  - Domain: “left side of relation” (Destination)
  - Range: “right side” (Accomodation)



Example


literal
range
range
id
phone
domain
domain
staff member
involves
range
subClassOf
subPropertyOf
domain
isTaughtBy
range
academic staff member
domain
course
subClassOf
subClassOf
subClassOf
professor
associate professor
assistant professor
type
type
RDFS
RDF
Discrete Mathematics
isTaughtBy
David Billington

Golden Rules

- A subclass inherits all the slots from the superclass
- A subclass can override the restrictions to "narrow" the list of allowed values
  - Make the cardinality range smaller
  - Replace a class in the range with a subclass

Property Restriction

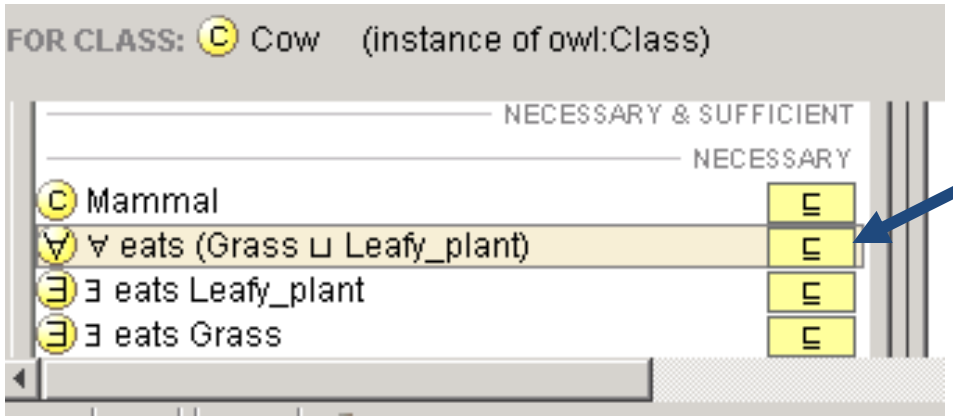
- Whenever required to describe the leave of a class, it is required to *close* down descriptions of entities

- Person owns/eat LivingThing except Person
  - domain: Person
  - range: LivingThing and not Person

- *"A 'Herbivore' is an animal that only eats plants"*
  (NB All animals eat some living thing)
  - Herbivore=
  - *eats* domain: *Herbivore*
    range: Living_thing & not Plants
  - Animal and *eats* only Plant
- "An 'omnivore' is an animal that eats both plants and animals"
  - Omnivore=
    Animal and *eats* some Animal and *eats* some Plant

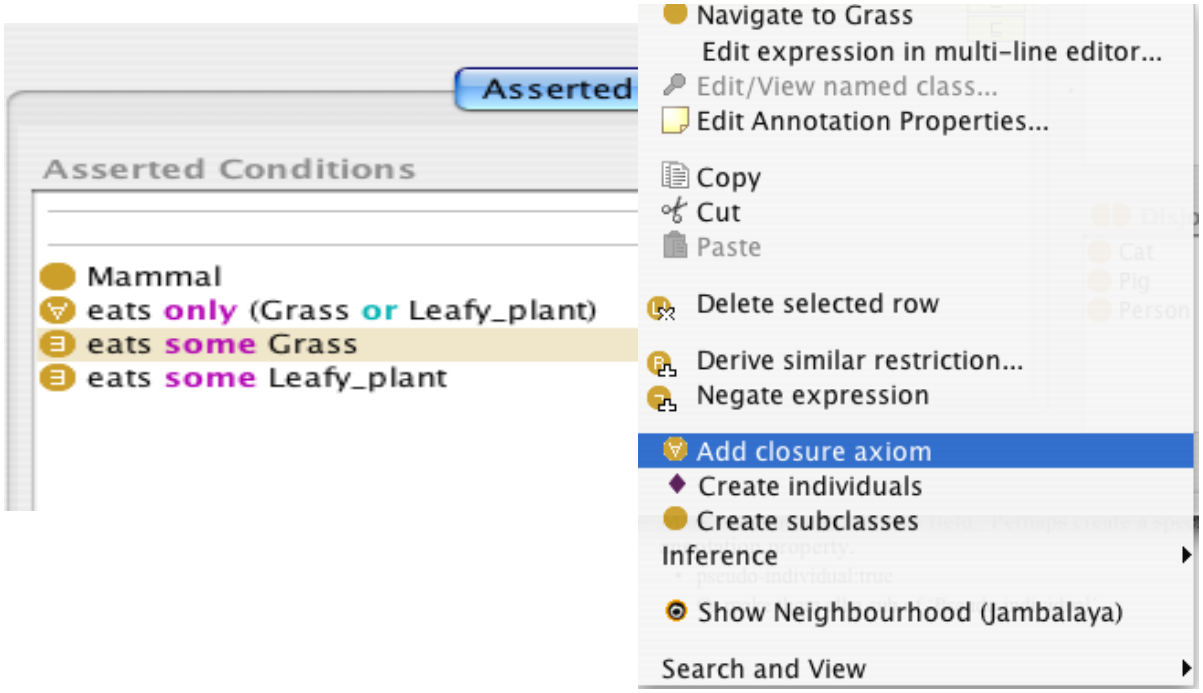
Which properties can be filled in at the class level

- What can we say about all members of a class:
- For example: Eats
  - **Cows** are **animals** and eat *only* **plants**
  - all Cats eat some Animals
  - all Omnivores eat some Animals and eat some Plants



In the tool

- Right mouse button short cut for closure axioms
  - for any existential restriction
  - adds closure axiom



Property Constraints/ Cardinality restrictions

- Property constraints describe or limit the set of possible values for a slot
  - *The name of a student is a string*
  - *The book publisher is an instance of class publisher*
  - *A lecture has exactly one location*
  - *A lecture has exactly one time slot*
  - *a person has exactly two parents*
  - *a course is taught by at least one lecturer*

Define the Values of the data property

- **Slot cardinality**
  - defines how many values a slot can have.
  - Minimum and maximum value – a range of values for a numeric slot.
  - Default value – the value a slot has unless explicitly specified otherwise.

- **Slot-value type**
  - what types of values can fill in the slot.
  - common value types:
    - **String**
    - **Number**
    - **Boolean**
    - **Enumerated**

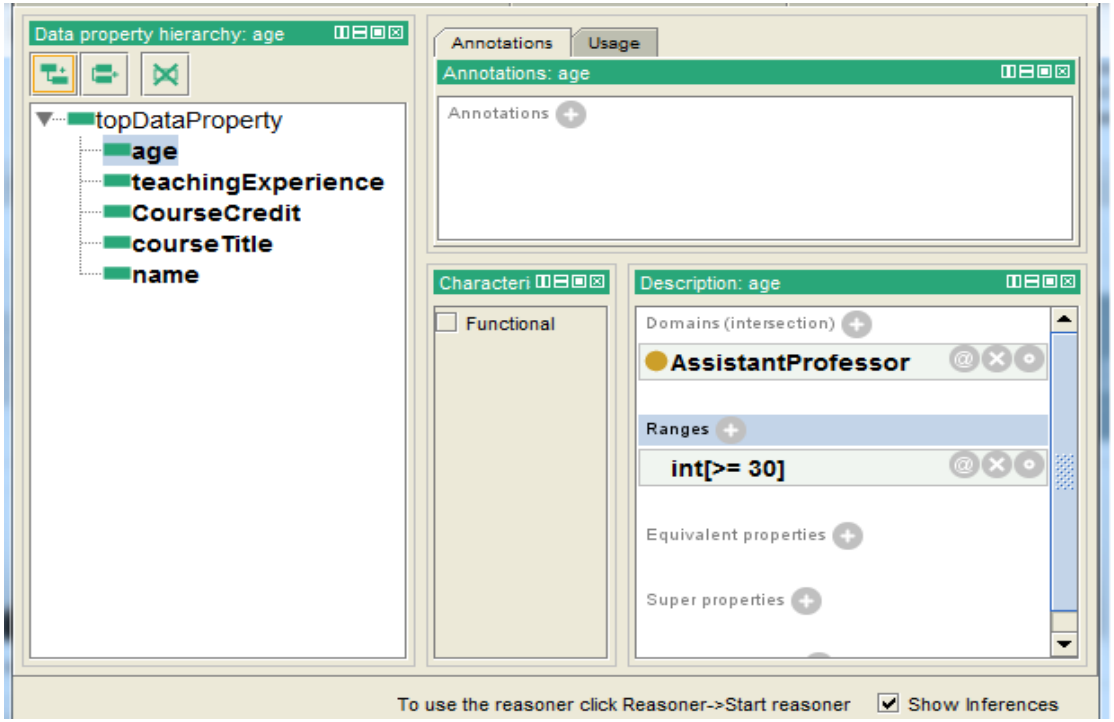
1-Common Facets: Value Type

- String: a string of characters (“Château Lafite”)
- Number: an integer or a float (15, 4.5)
- Boolean: a true/false flag
- Enumerated type: a list of allowed values (high, medium, low).
- Complex type: an instance of another class
  - Specify the class to which the instances belong

The Authors is the value type for the slot “Authored by” at the Book.

Data Type property

- Data type properties will have ranges assigned to literal types
  - name, courseTitle, …



2- Slot Cardinality

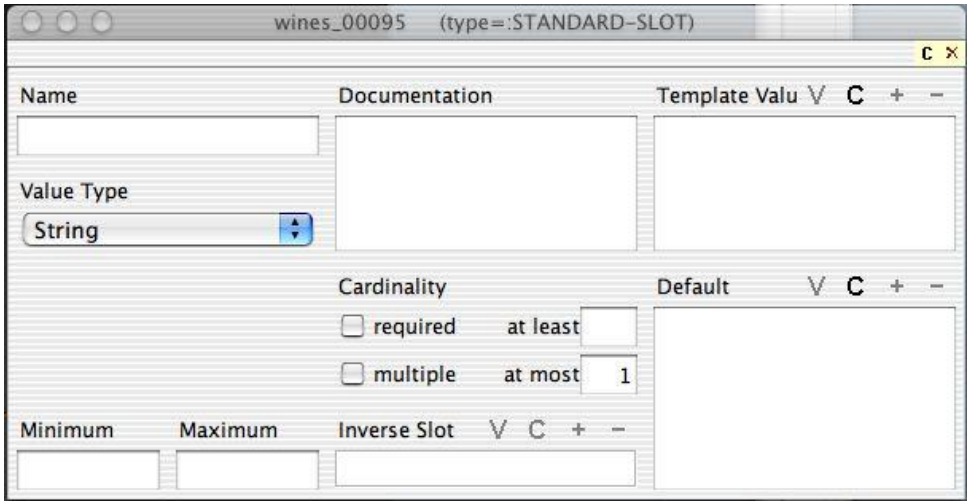
- Cardinality
  - Cardinality N means that the slot must have N values
- Minimum cardinality
  - Minimum cardinality 1 means that the slot must have a value **(required)**
  - Minimum cardinality 0 means that the slot value is **(optional)**
- Maximum cardinality
  - Maximum cardinality 1 means that the slot can have at most one value **(single-valued slot)**
  - Maximum cardinality greater than 1 means that the slot can have more than one value (**multiple-valued slot)**

3-Default Values

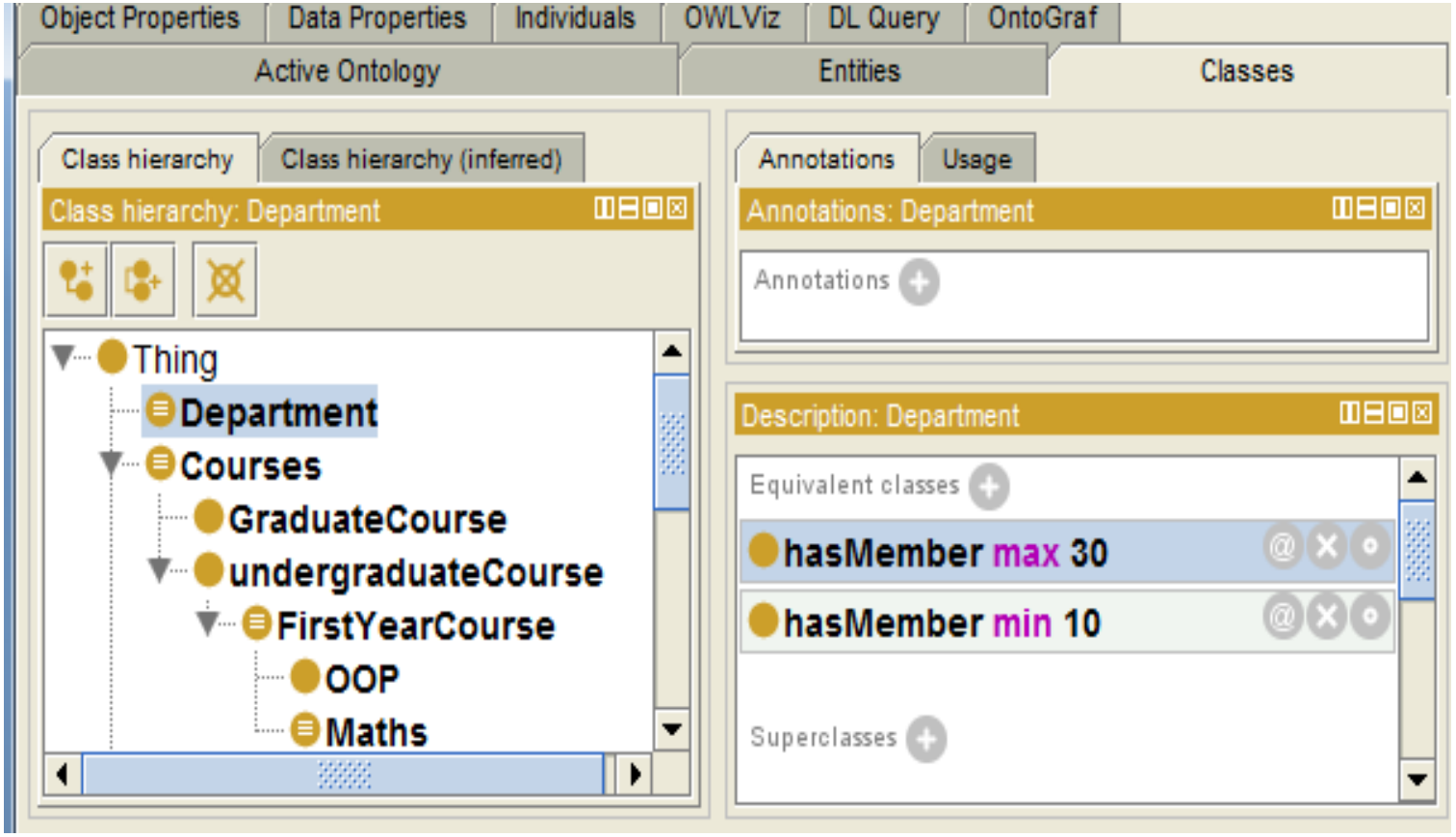
- Default value – a value the slot gets when an instance is created.
- A default value can be changed.
- The default value is a common value for the slot, but is not a required value.
- For example, the default value for quadratic shape can be Parallelogram.

Slot cardinality summary

- Slot cardinality – the number of values a slot has
- Slot value type – the type of values a slot has
- Minimum and maximum value – a range of values for a numeric slot
- Default value – the initial value for a slot when the instance is created



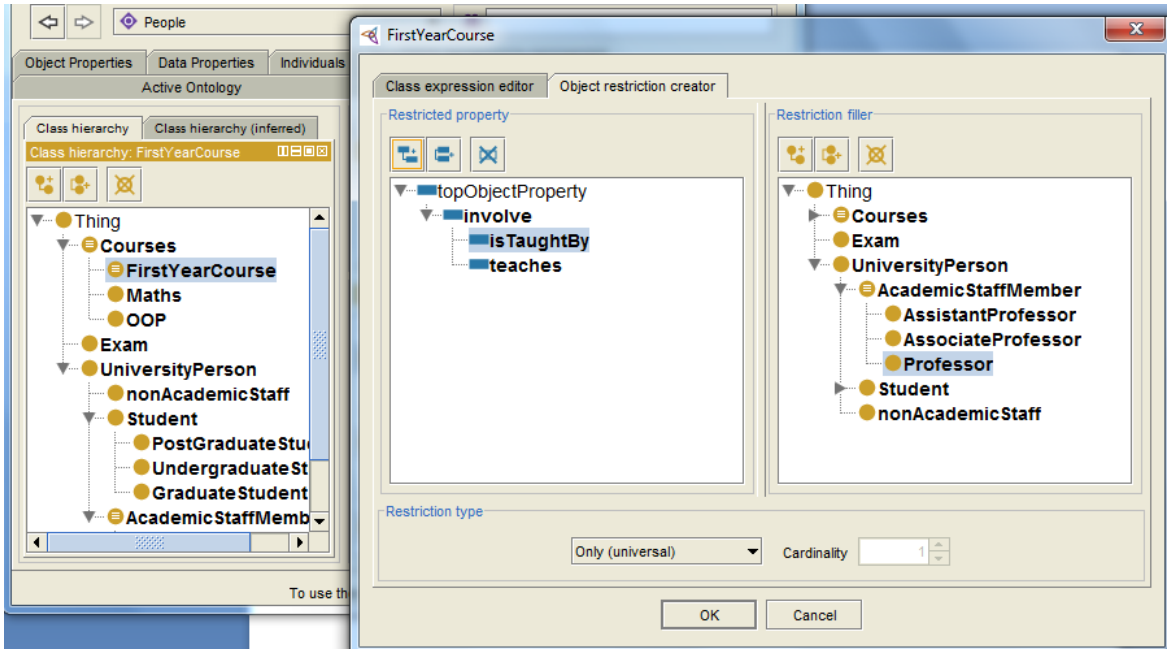
Cardinality Restriction



- Department has min 10 members
- Department has max 30 members

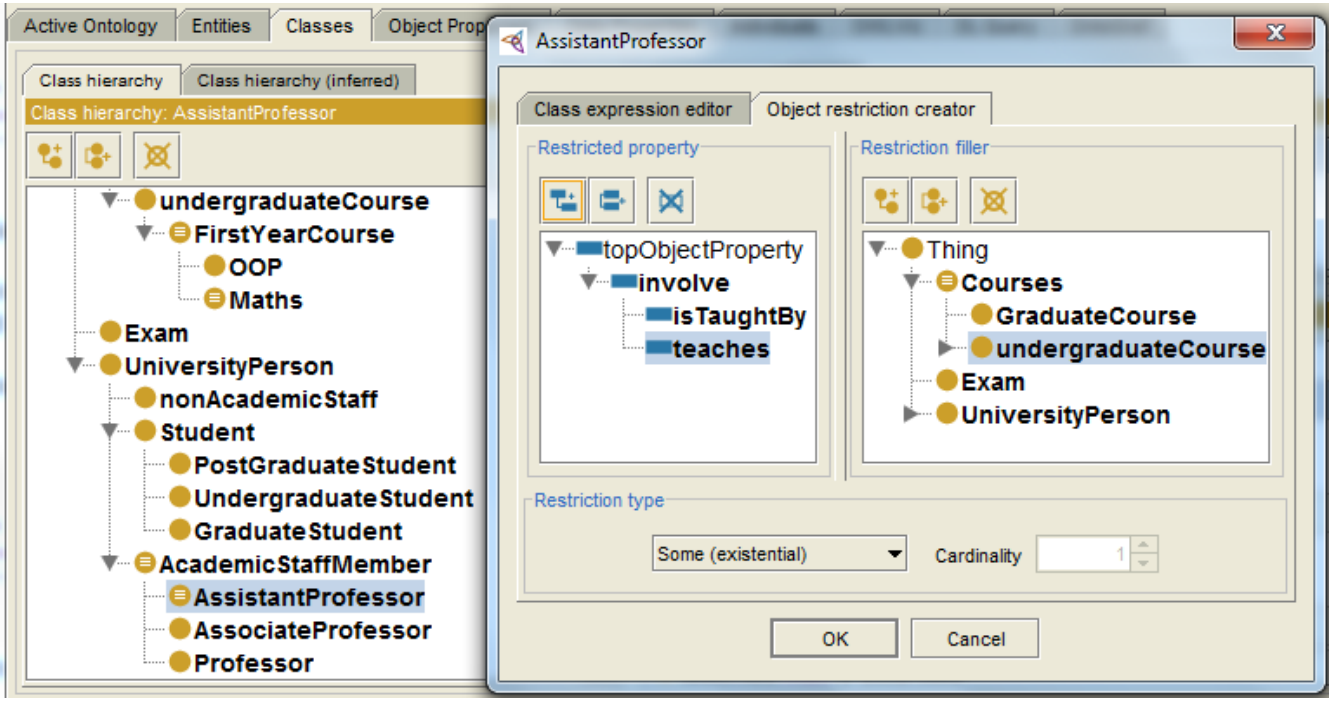
Property restriction

- allValuesFrom
- This constraints is analogous to the universal (for-all) quantifier of Predicate logic



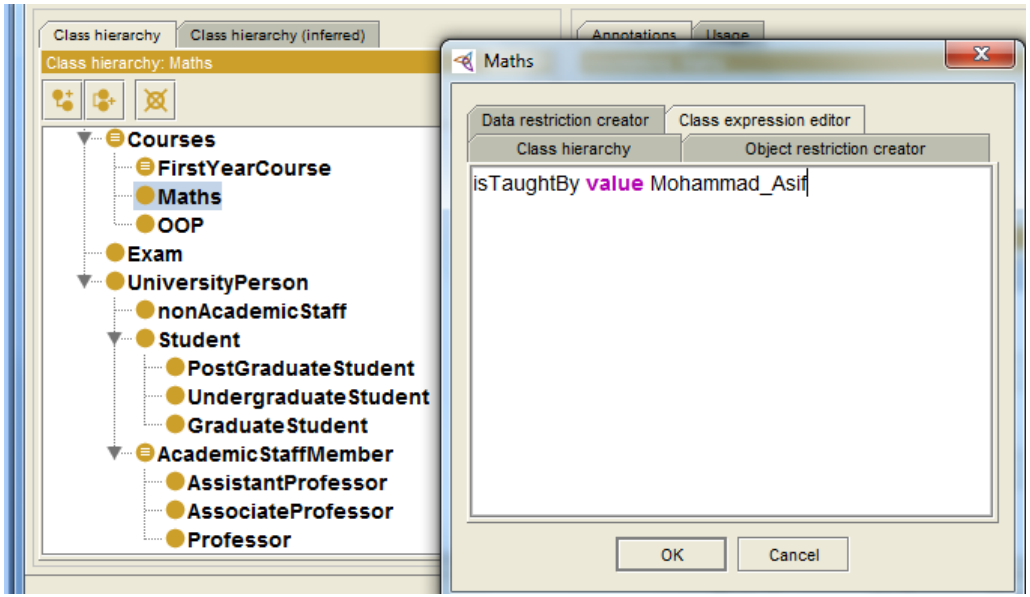
Property restriction

- SomeValuesFrom
- constraint is analogous to the existential quantifier of Predicate logic



Property Restriction

- hasValue

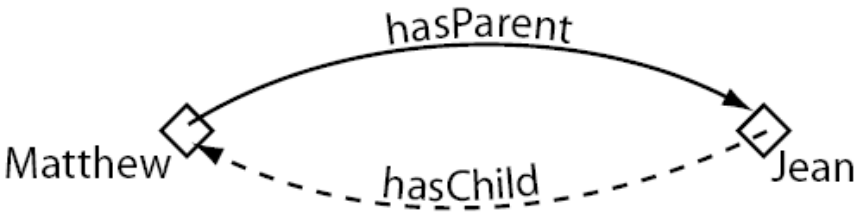


It is used to refer to an individual or a data value.

A restriction containing a hasValue constraint describes a class of all individuals for which the property concerned has at least one value *semantically* equal to V

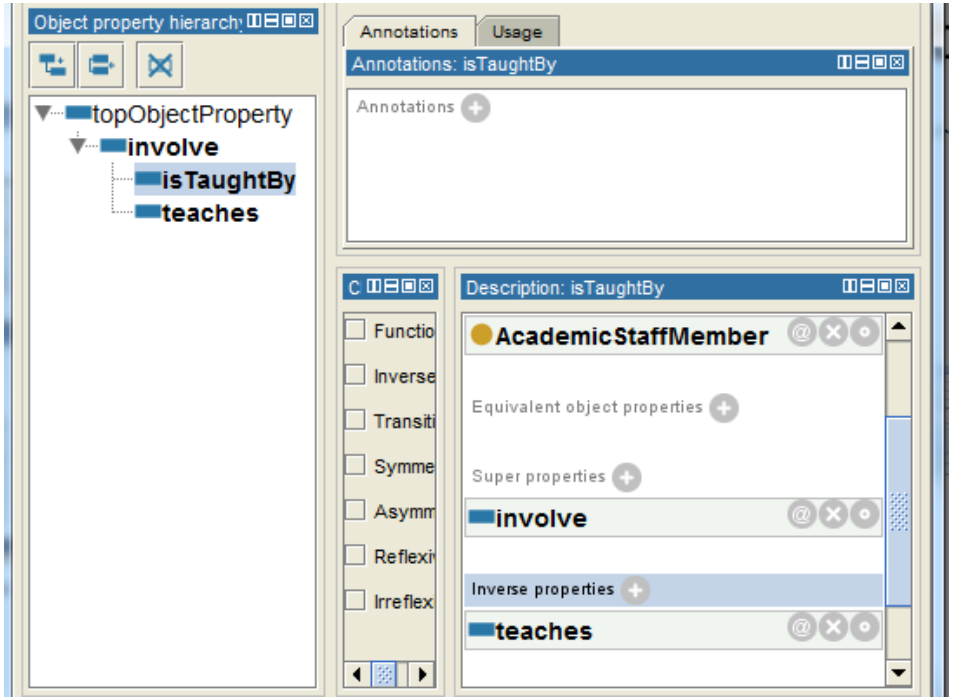
Inverse Slot

- Each object property may have a corresponding inverse property.
- If some property links individual **A** to individual **B**, then its inverse property will link individual b to individual a.
- Allow acquisition/presentation of the information in either direction
- Enable additional verification :Teach/is-taught-by



Inverse object properties

- teaches is inverse of isTaughtBy



Ontology Engineering
Steps to develop An ontology

1. determine domain and scope
   - what is the domain that the ontology will cover?
   - what we are going to use the ontology for?
   - what types of questions the information in the ontology should provide answers for (competence questions)?

2 informal/semiformal knowledge acquisition

- Collect the terms
- Oganize them informally
- Clarify terms to produce informal concept definitions
- Diagram informally

3 refine requirements and tests

Steps to develop An ontology(cont.)

4. implementation
   - implement prototype recording the intension as a summary
   - scale up a bit and check performance

5 evaluation and quality assurance

- against goals (ontology design is subjective!)
- include tests for evolution and change management
- design regression tests and ‘probes’

6 maintenance: usage monitoring and evolution

- – compatibility between different versions of the same ontology and between versions of an ontology and instance data

Open vs Closed World reasoning

- **Open world reasoning**
  - Negation as contradiction
    - Anything might be true unless it can be proven false
      - Reasoning about *any world consistent with this one*
- **Closed world reasoning**
  - Negation as failure
    - Anything that cannot be found is false
      - Reasoning about *this world*
- ***Ontologies are not databases***

Database -v- Ontology

Database:

- Closed world assumption (**CWA**)
  - Missing information treated as false.
- Unique name assumption (**UNA**)
  - Each individual has a single, unique name.
- Schema behaves as **constraints** on structure of data
  - Define legal database states

Ontology:

- Open world assumption (**OWA**)
  - Missing information treated as unknown
- **No UNA**
  - Individuals may have more than one name
- Ontology axioms behave like **implications** (inference rules)
  - Entail implicit information

Exercise

Convert the following Statements into ontology component

George is an employee. → An object is an instance of the Employee class.

George works for Sony. → An object in the Employee class is linked with an object in the Company class via the works_for relationship.

George reports to Adam. → An object in the Employee class is linked with another object in the same class via the reports_to relationship.

Fred works for a company.

Fred reports to two other employees.

homework

- Use the Proteg´e editor to define a normalised ontology for use by a travel
- agency covering the following:
- Hotel, restaurant, sports, luxury hotel, bed and breakfast, safari, activity, hiking, spa treatment, sunbathing, sightseeing, accommodation rating (three stars, etc.), campground, surfing.

Build a class hierarchy and indicate which classes in it are primitive and which are definable.

Define the required relations, their properties, domains and ranges as well as individuals.

- Define the following classes:
- 1. A two star hotel.
- 2. A spa resort (i.e., a destination offering a spa treatment).
- 3. A destination with sport activities but without safari.
- 4. A destination where all hotels have three star rating.
- 5. A destinations with at least three restaurants and at least four hotels.