Graduate Program in Production Engineering and Systems

(PPGEPS)

Important Terms in an Ontology (Part 3/6)

Dr. Yongxin Liao

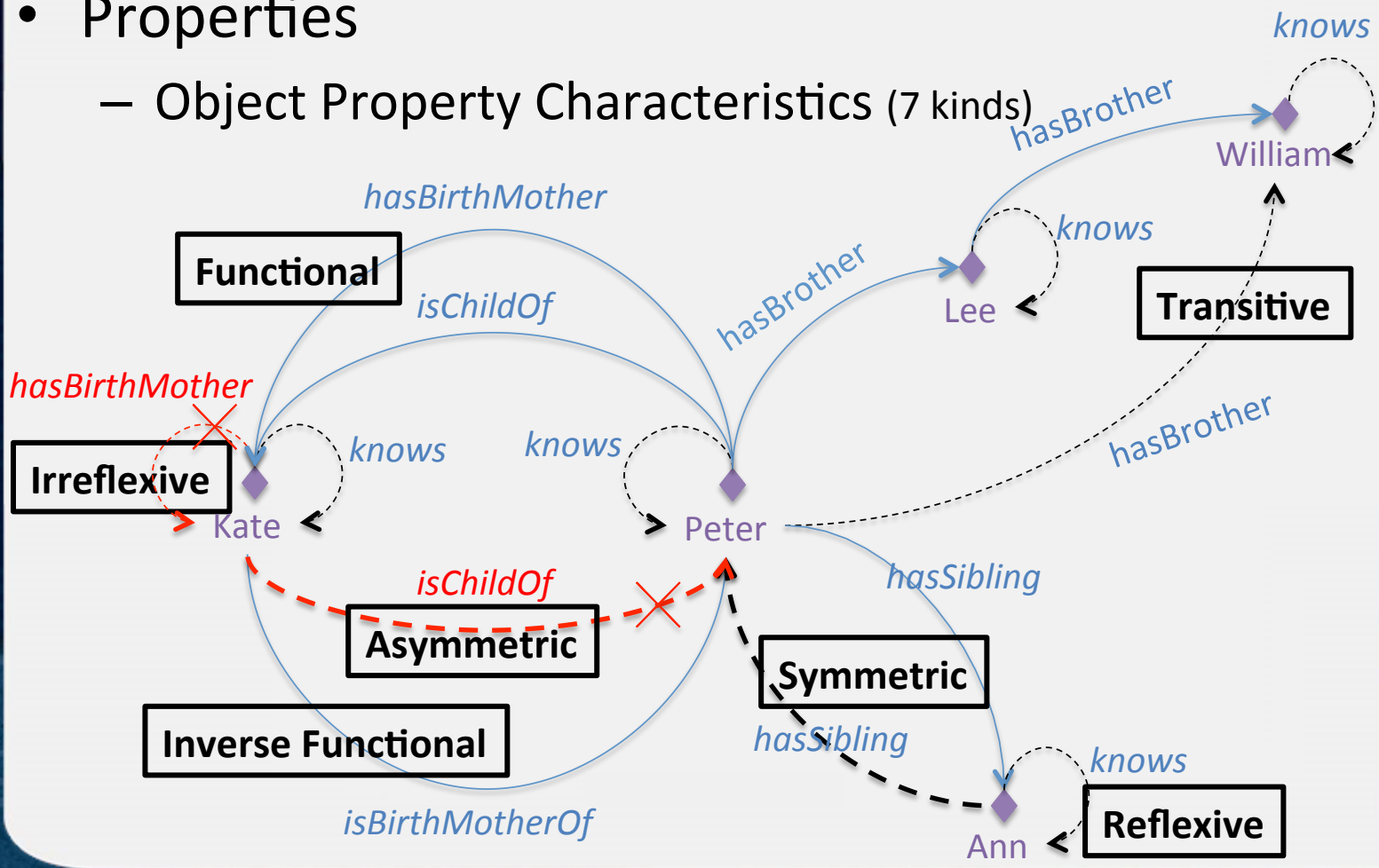
Course Outlines

- Previously on IOE
- Important Terms in an Ontology (3/6)
  - Complex Class Expressions (1/2)
    - Enumeration of Individuals
    - Propositional Connectives
    - Object Property Restrictions
    - Necessary and Sufficient Conditions
- Protégé Practices

PUCPR

Previously on the IOE

- Properties
  - Object Property Characteristics (7 kinds)

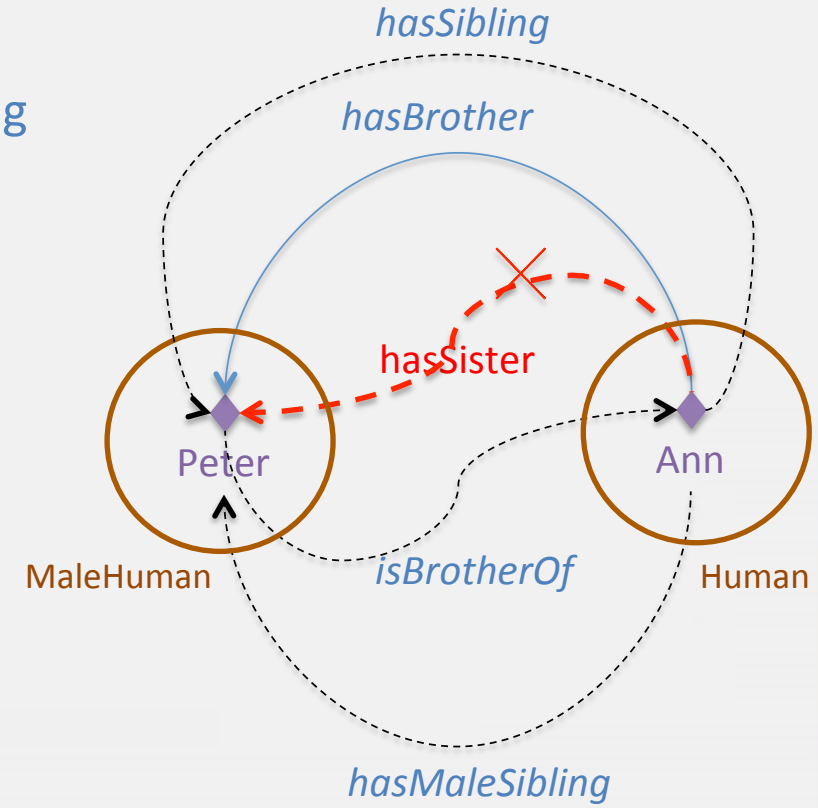


Previously on the IOE

- Properties
  - Object Property Descriptions (6 kinds)

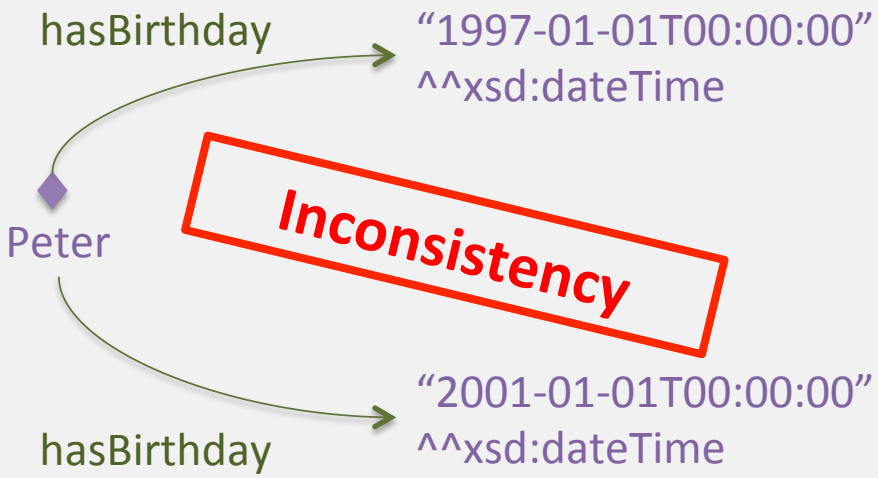
hasBrother

- is equivalent to hasMaleSibling
- is disjoint with hasSister
- is inverse of isBrotherOf
- has Domain Human
- has Range MaleHuman
- is sub property of hasSibling



Previously on the IOE

- Properties
  - Data Property Characteristics (1 kind)
    - Functional Data Property Axioms



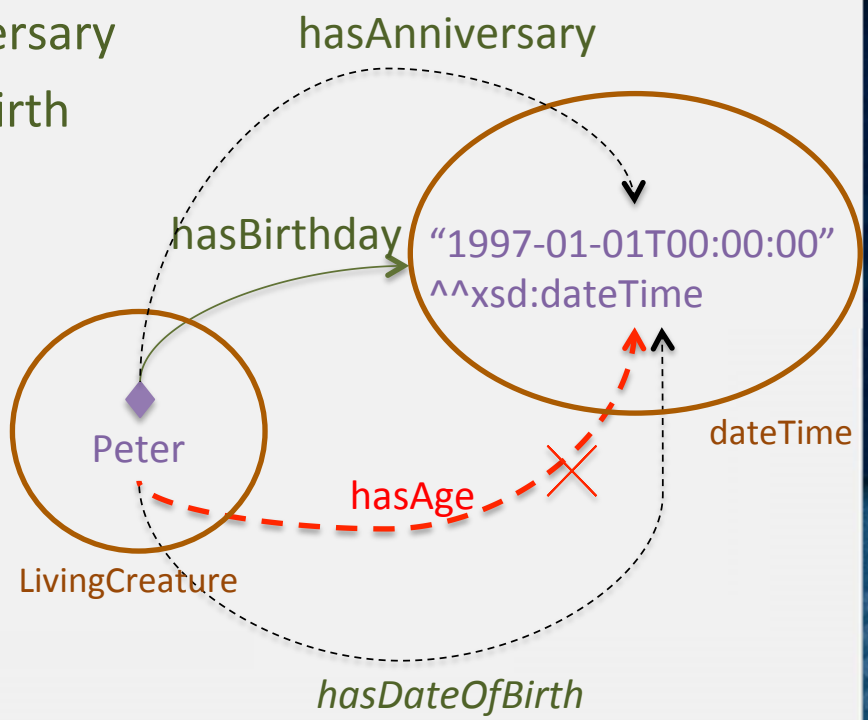
PUCPR

Previously on the IOE

- Properties
  - Data Property Descriptions (5 kinds)

hasBirthday

- is sub property of hasAnniversary
- is equivalent to hasDateOfBirth
- is disjoint with hasAge
- has Domain LivingCreature
- has Range dateTime



Important Terms in an Ontology

- Several Important Terms
  - Axioms
  - Concepts (Individuals and Classes)
  - Relationships (Class Assertions, Subclasses Disjoint/Equivalent Classes, Individual Equality/Inequality Properties, Property Assertions, Property Characteristics, and Property Descriptions)
  - **Complex Class Expressions (Enumeration of Individuals, Propositional Connectives, Object Property Restrictions, Necessary and Sufficient Conditions,** Data Property Restrictions)
  - Data Ranges (Data Types and Data Type Restrictions)
  - Reasoning Rules
  - Knowledge Base (T-box and A-box)
  - SPARQL Query

Part 1/6

Part 2/6

Part 3/6

Part 4/6

Part 5/6

Part 6/6

PUCPR

Class Expressions

The simplest form of class expressions

- Classes (Named Classes)
  - A named class is a class that identified using an IRI
    - It expresses semantics through giving a name to this class
      - E.g. The Class *Pizza*

      (IRI: http://www.semanticweb.org/yongxinliao/ontologies/2015/6/Pizzas#Pizza)

      The Class *SeafoodTopping*

      The Class *MeatTopping*

can not exist independently

- Complex Class Expressions (Anonymous Classes)
  - An anonymous Class is a class that has not IRI
    - It expresses semantics through placing constrains on the existing named classes and properties
      - E.g.

(a named class)
VegetarianPizza is equivalent to

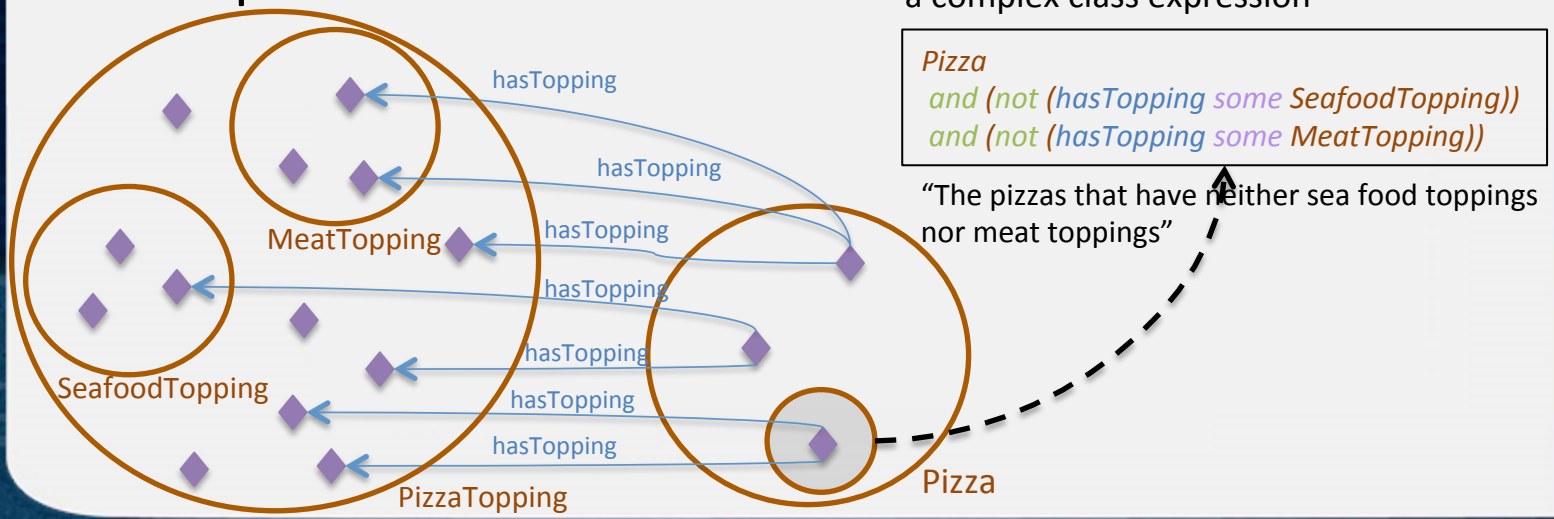
(a complex class expression)

Pizza
and (not (hasTopping some SeafoodTopping))
and (not (hasTopping some MeatTopping))

PUCPR

Class Expressions

- Complex Class Expressions
  - Complex class expressions specify a number of conditions that all its individuals should satisfied.
  - The individuals that satisfy these conditions are said to be members of the respective complex class expressions.



a complex class expression

Pizza
and (not (hasTopping some SeafoodTopping))
and (not (hasTopping some MeatTopping))

"The pizzas that have neither sea food toppings nor meat toppings"

Class Expressions

- Enumeration of Individuals ("{}")
- Propositional Connectives
  - Intersection of Class Expressions ("and")
  - Union of Class Expressions ("or")
  - Complement of Class Expressions ("not")
- Property Restrictions (for Object Properties)
  - Quantifier Restrictions ("some" or "only")
  - Value Restrictions ("value")
  - Cardinality Restrictions ("min", "max", or "exactly")
- Necessary and Sufficient Conditions

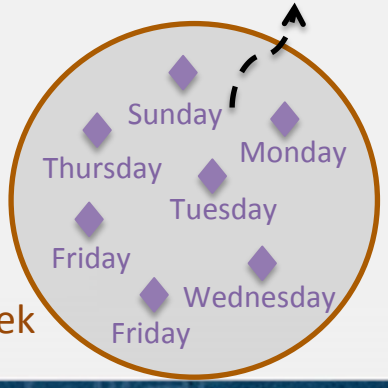
PUCPR

Enumeration of Individuals

- An enumeration of individuals defines a class that contains **exactly** the individuals in the enumeration.
- In protégé, it is described by
  - Precisely listing all individuals
  - Separating them by ","
  - Inside curly brackets ("{" and "}")

DaysOfTheWeek is equivalent to
{Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}

Enumeration of individuals

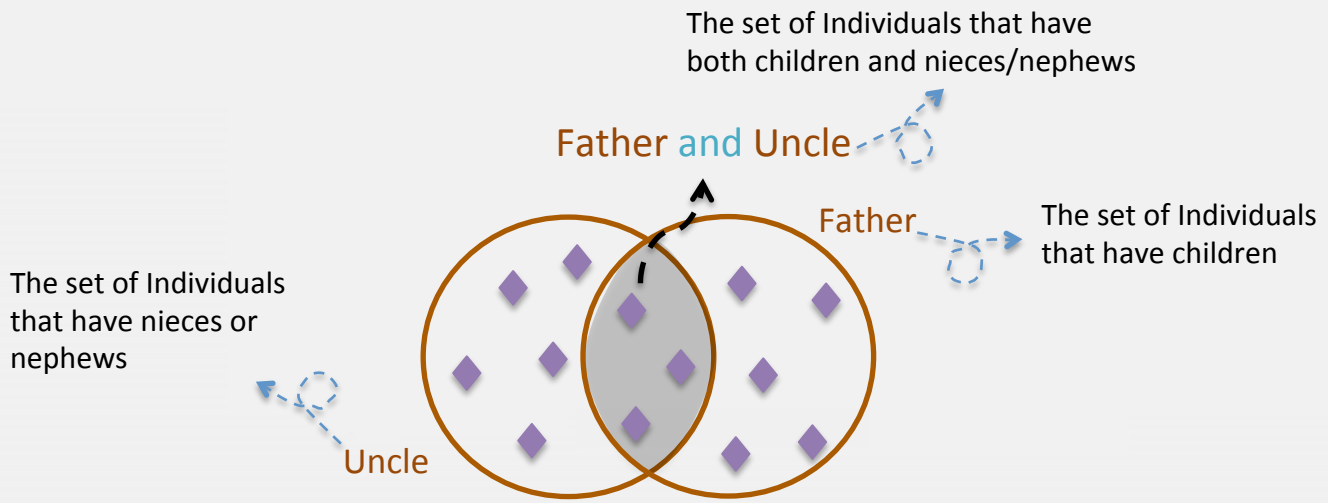


DaysOfTheWeek

Propositional Connectives

- Intersection of Class Expressions
  - It contains all individuals that are shared (co-owned) by all the class expressions in this intersection.
  - In protégé, it is described by combining two or more class expressions using the "and" operator.

Those class expressions can be
- Classes (Named Classes)
- Complex class expressions

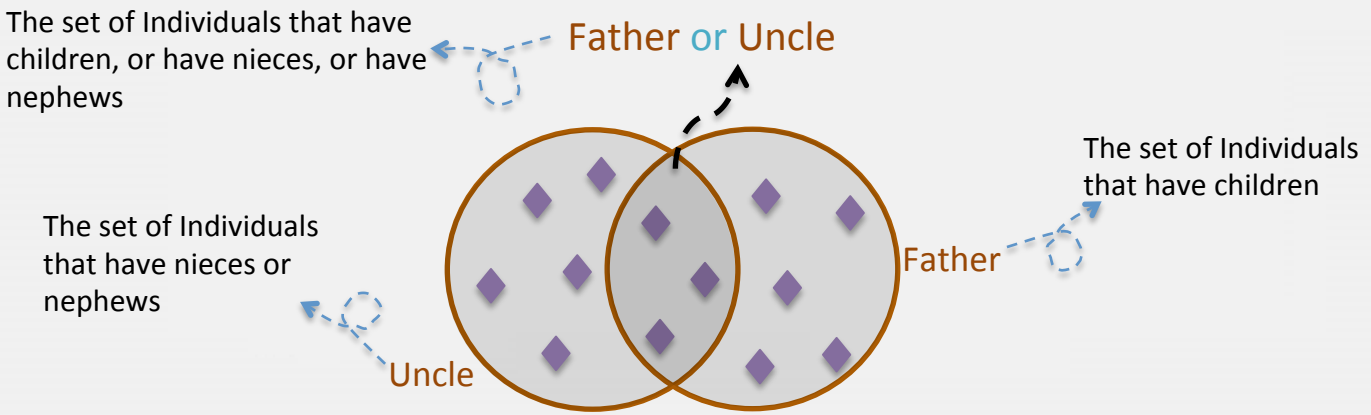


PUCPR

Propositional Connectives

- Union of Class Expressions
  - It contains all individuals that are members of at least one of the class expressions in this union
  - In protégé, it is described by combining two or more class expressions using the "or" operator.

Those class expressions can be
- Classes (Named Classes)
- Complex class expressions

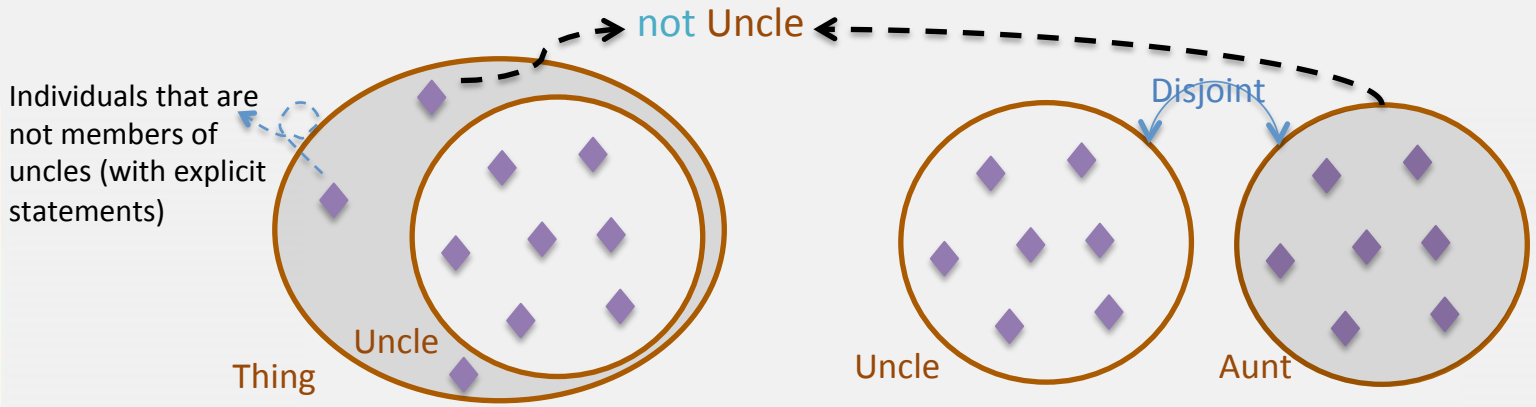


PUCPR

Propositional Connectives

This class expression can be
- A Class (Named Class)
- A complex class Expression

- Complement of Class Expressions
  - It contains of all the individuals that are **not** members of that class expression.
  - In protégé, it is described by placing a "not" operator **at the beginning of** that class expression.

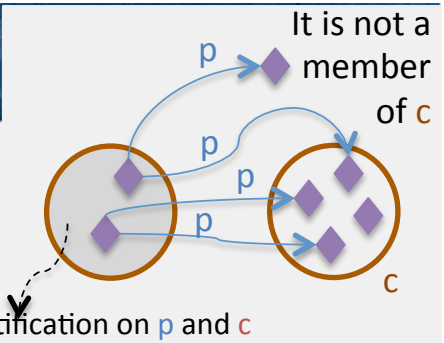


PUCPR

Object Property Restrictions

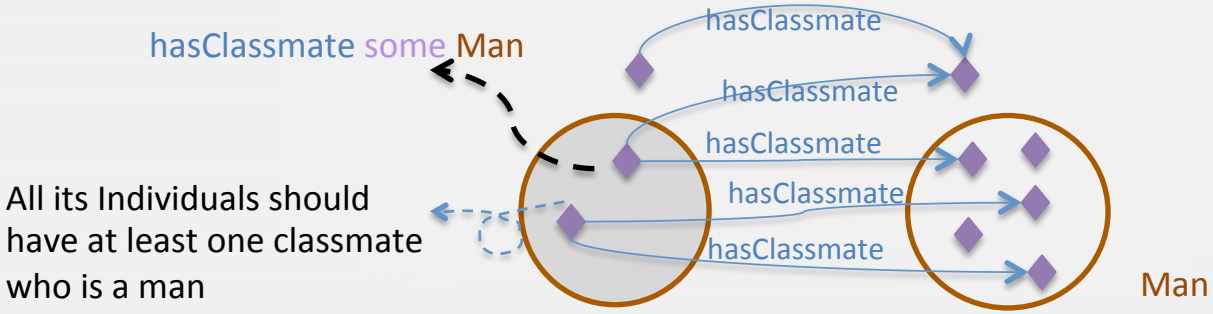
- A restriction describes a class of individuals based on the relationships that members of this class participate in.
- Three kinds of restrictions
  - Quantifier Restrictions ("some" or "only")
  - Value Restrictions ("value")
  - Cardinality Restrictions ("min", "max", or "exactly")

Quantifier Restrictions



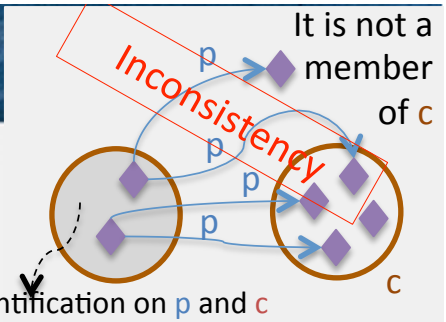
- Object Property Existential Quantification
  - An object property existential restriction expression consists of an object property p and a class expression c
  - It contains all those individuals that are connected, at least once, by p to the individuals that are members of c.
  - In Protégé, it is described as

an object property + "some" + a class expression



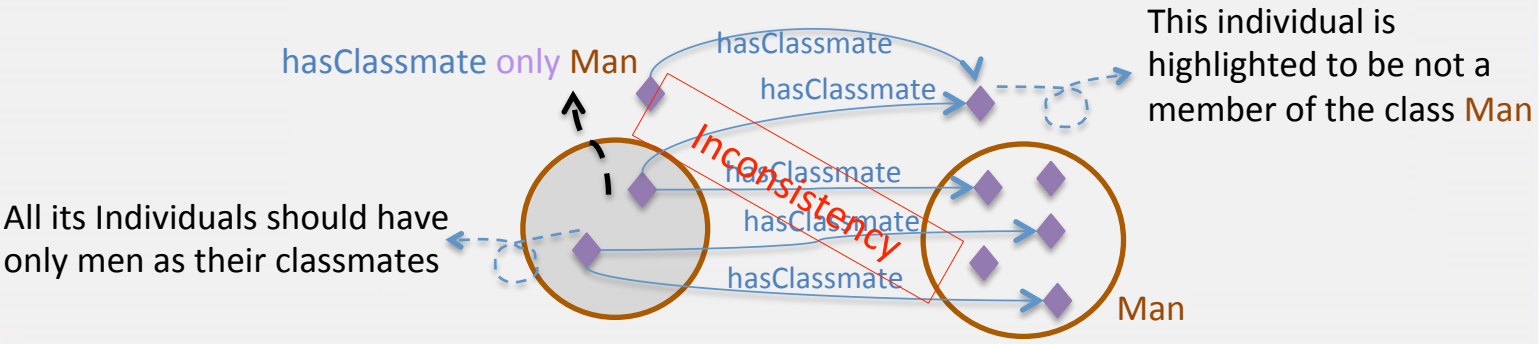
PUCPR

Quantifier Restrictions



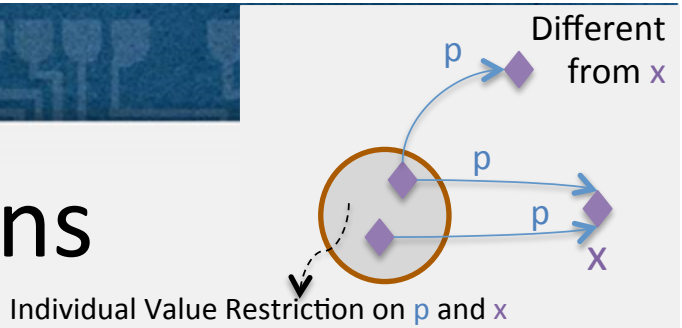
- Object Property Universal Quantification
  - An object property universal restriction expression consists of an object property p and a class expression c
  - it contains all those individuals that are only connected by p to the individuals that are members of c.
  - In Protégé, it is described as

an object property + "only" + a class expression



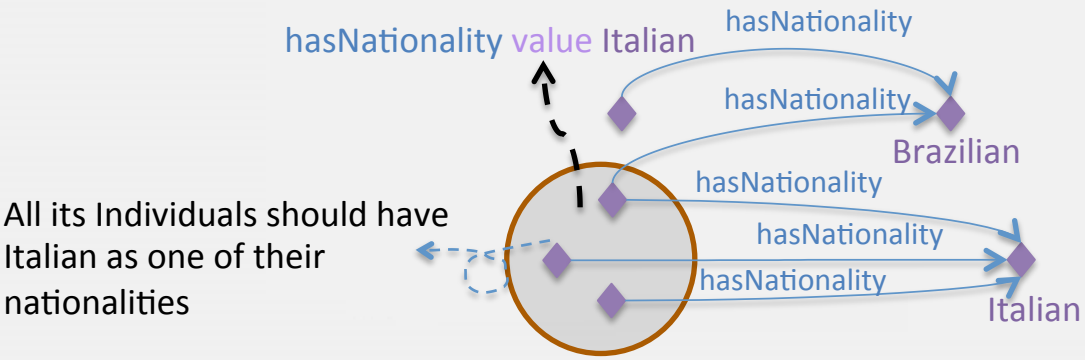
PUCPR

Value Restrictions



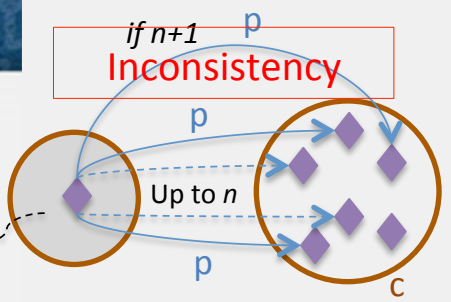
- Individual Value Restriction
  - An individual value restriction expression consists of an object property p and an individual x
  - It contains all those individuals that are connected by p to x.
  - In Protégé, it is described as

an object property + "value" + an individual



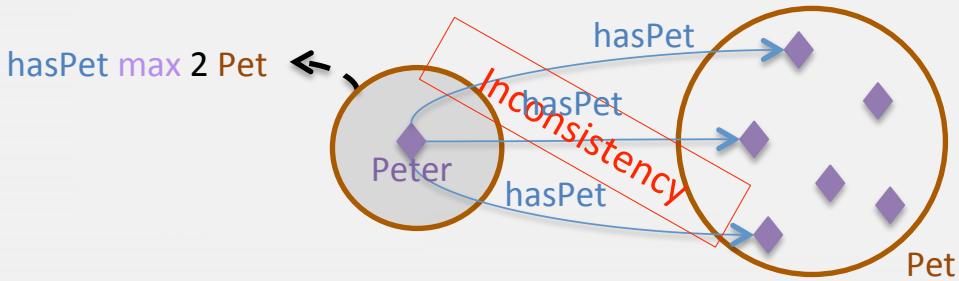
PUCPR

Cardinality Restrictions



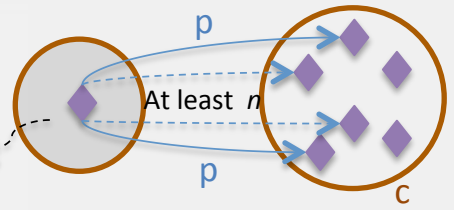
- Object Property Maximum Cardinality
  - An object property maximum cardinality expression consists of an object property p, a nonnegative integer *n*, and a class expression c
  - It contains all those individuals that are connected by p to *at most n different individuals* that are members of c.
  - In protégé, it is described as

an object property + "max" + a nonnegative integer + a class expression



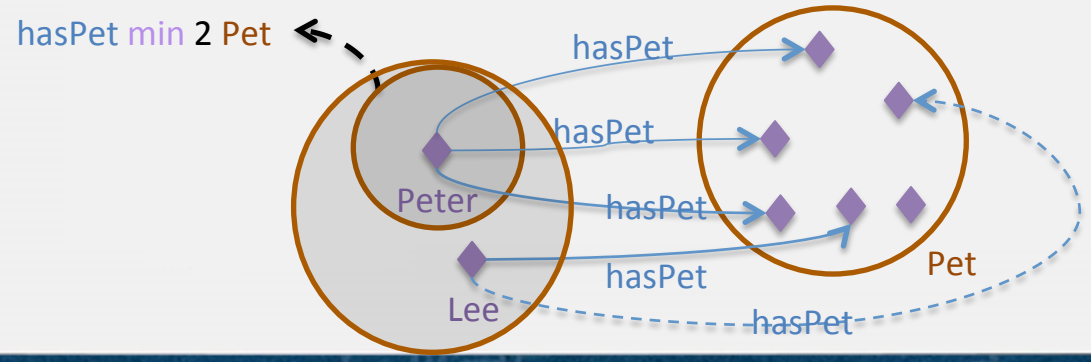
PUCPR

Cardinality Restrictions

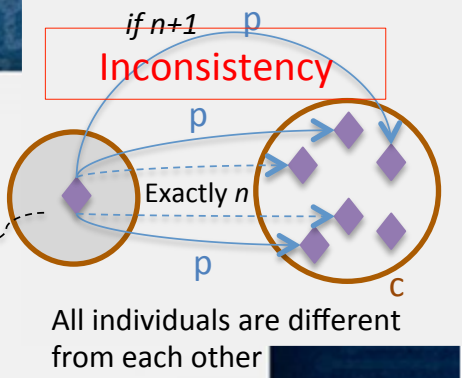


- Object Property Minimum Cardinality
  - An object property minimum cardinality restriction is consist of an object property p, a nonnegative integer *n*, and a class expression c
  - It contains all those individuals that are connected by p to *at least n different individuals* that are members of c.
  - In protégé, it is described as

an object property + "min" + a nonnegative integer+ a class expression



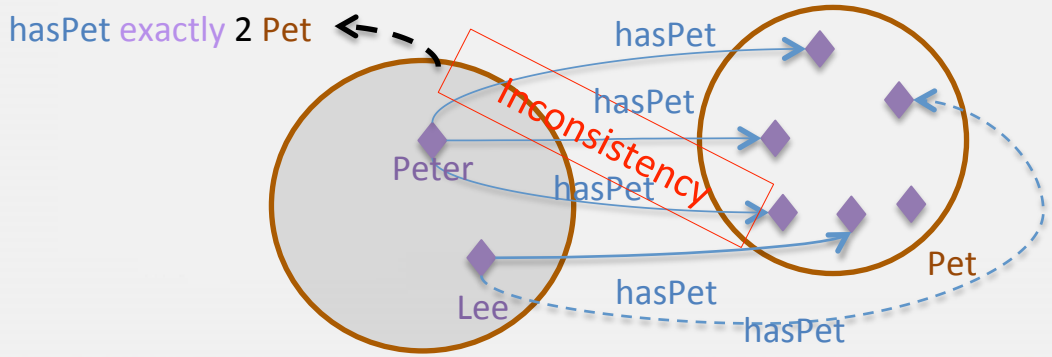
Cardinality Restrictions



Exact cardinality restriction on p, *n* and c

- Object Property Exact Cardinality
  - An object property exact cardinality restriction is consist of an object property p, a nonnegative integer *n*, and a class expression c
  - It contains all those individuals that are connected by p to *exactly n different individuals* that are members of c.
  - In protégé, it is described as

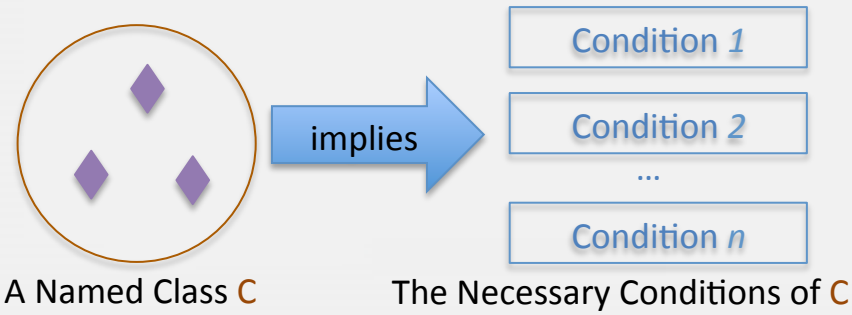
an object property + "exactly" + a nonnegative integer + a class expression



PUCPR

Necessary and Sufficient Conditions

- Necessary Conditions
  - The necessary conditions for a class C states that
    - if something is a member (or a sub class) of C **then it is necessary to fulfill** those conditions.
    - The class C is a sub class of the class that described by those conditions (complex class expressions).
  - In protégé, the necessary condition is simply called "SubClass of" in the "Class Description View".

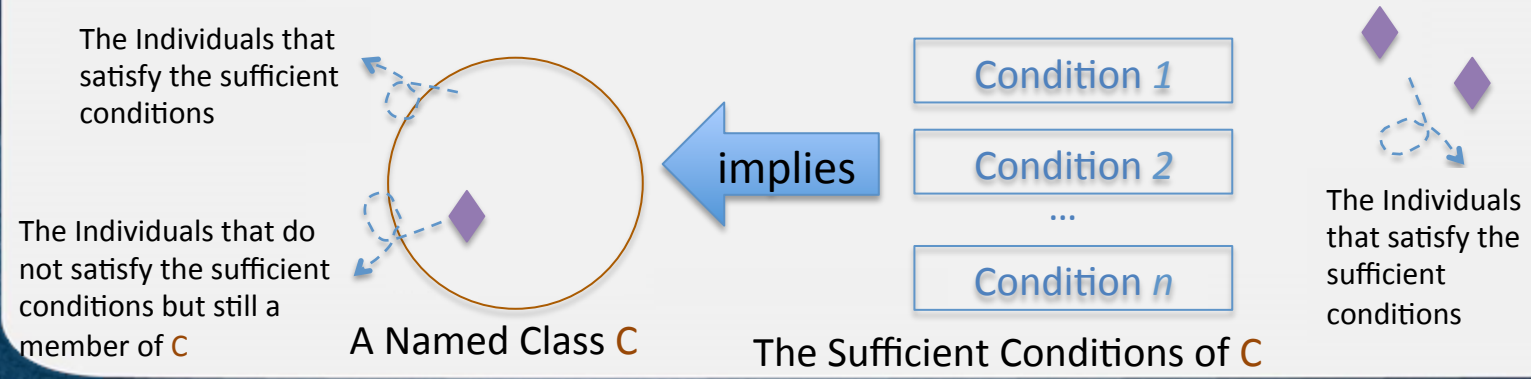



The Individuals that satisfy the necessary conditions but not a member of C

PUCPR

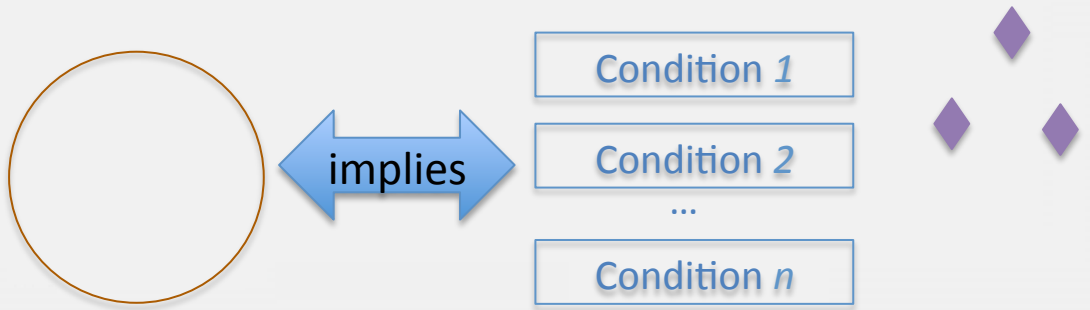
Necessary and Sufficient Conditions

- Sufficient Conditions
  - The sufficient conditions for a class C states that
    - if something satisfies those conditions then **it is sufficient to be** a member (or a sub class) of C.
    - The class C is a super class of the the class that described by those conditions (complex class expressions).
  - In Protégé it can be realized through the General Class Axioms in the "Active Ontology" Tab.



Necessary and Sufficient Conditions

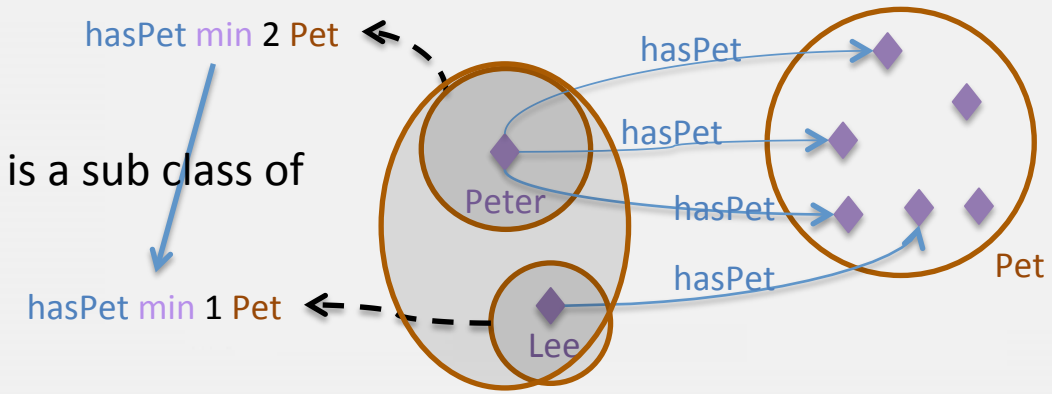
- Necessary & Sufficient Conditions
  - The necessary & sufficient conditions for a class C states that
    - If something satisfies those conditions then **it is sufficient to be** a member (or a sub class) of C.
    - Meanwhile, If something is a member (or a sub class) of C then **it is necessary to fulfill** those conditions.
    - The class C is equivalent to the the class that described by those conditions (complex class expressions).
  - In protégé, the necessary condition is simply called "Equivalent to" in the "Class Description View".



PUCPR

Reasoners

- Classification of Classes
  - Based on the descriptions (conditions) of existing classes to check whether or not a class is a subclass of another class(es).
  - Automatically compute the class hierarchy.



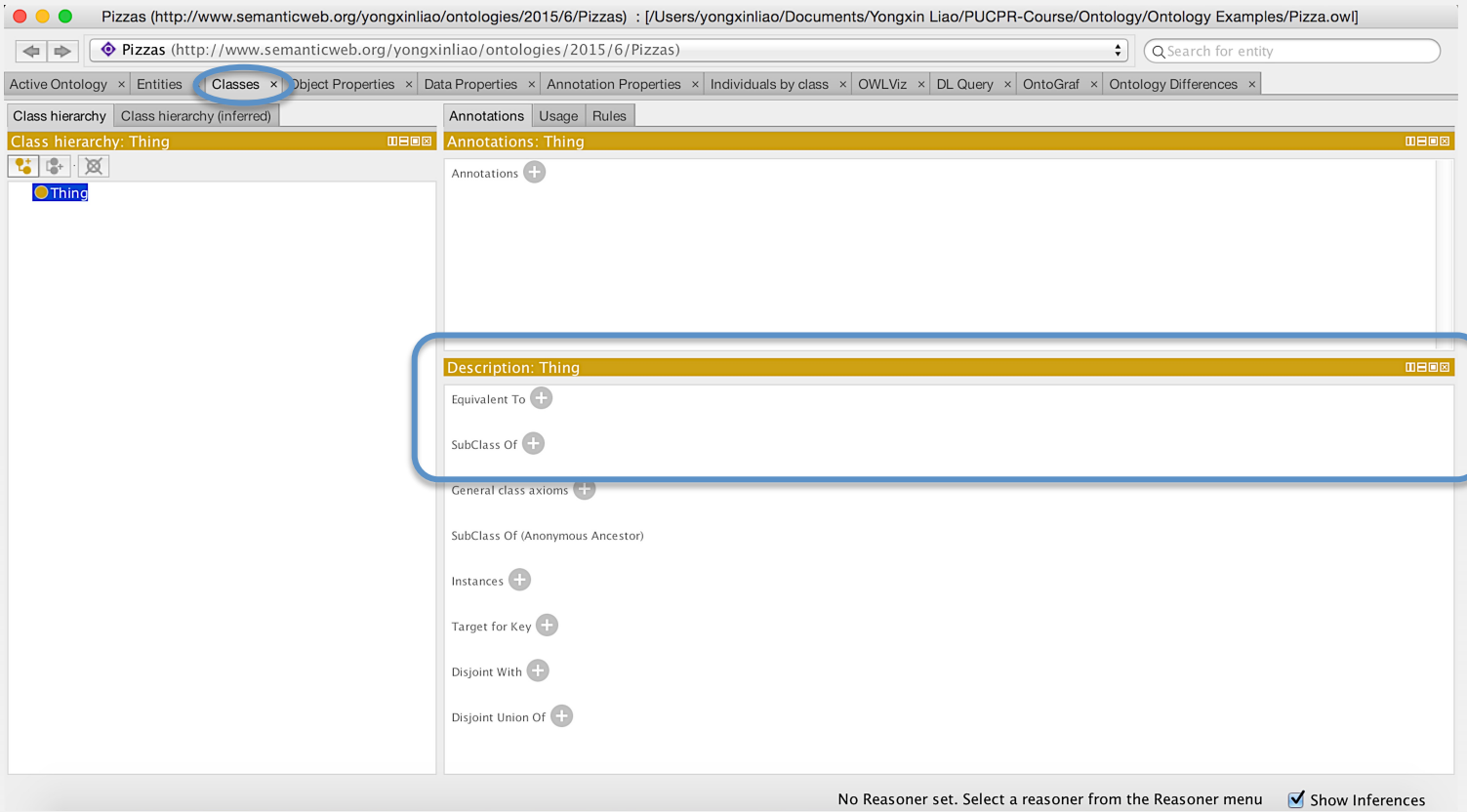
Protégé Practices

- The “Classes” Tab
  - Complex Class Expressions (via Object Properties)
    - Equivalent Classes Axioms
    - Subclass Axioms

PUCPR

Protégé Practices

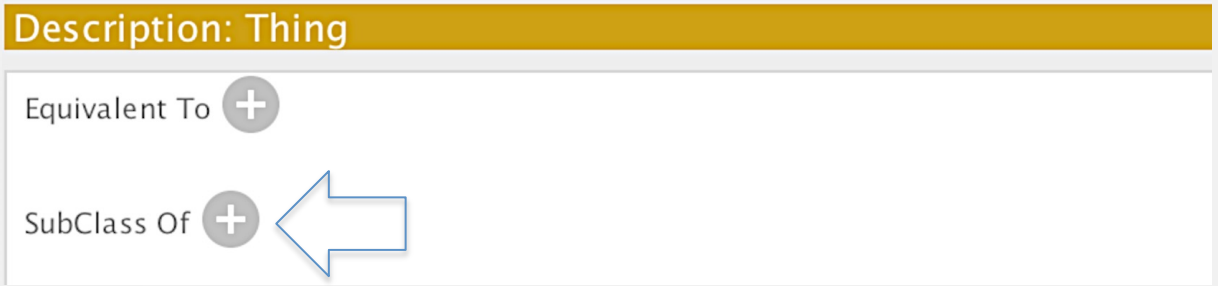
- The “Classes” Tab



PUCPR

Protégé Practices

- The “SubClass Of”
  - “SubClass Of” => necessary conditions
    - Let class C be a subclass of some necessary conditions
    - All the individuals and subclasses of C **are necessary to fulfill** those conditions.

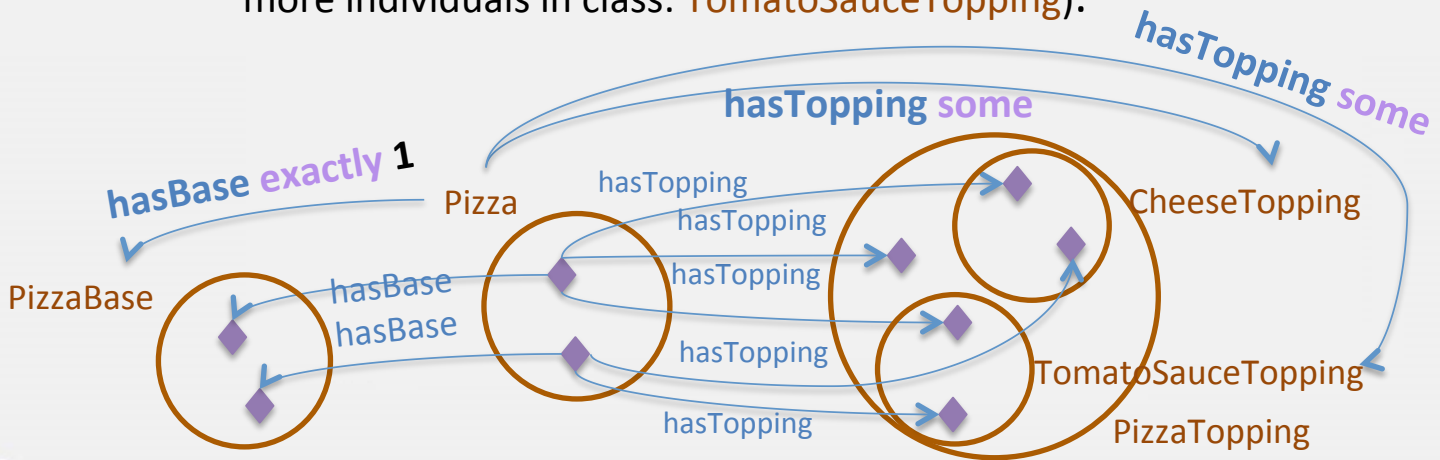


PUCPR

Protégé Practices

"SubClass Of"

- Pizzas
  - All pizza (all the individuals and subclasses of class: Pizza) should fulfill the following three necessary conditions:
    a) Each pizza contains exactly one pizza base (one individual in class: PizzaBase);
    b) Each pizza contains some cheese toppings (one or more individuals in class: CheeseTopping);
    c) Each pizza contains some tomato sauce toppings (one or more individuals in class: TomatoSauceTopping).



PUCPR

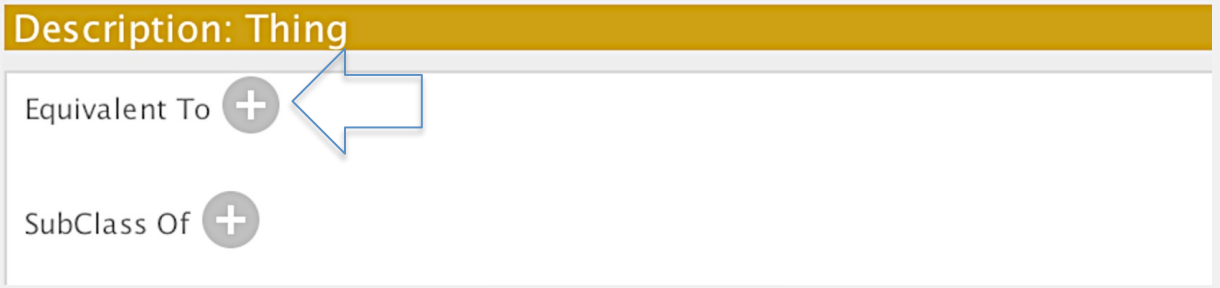
Protégé Practices

"SubClass Of"

- Four Kinds of Pizzas
  - **Margherita Pizza** has Mozzarella and Tomato Sauce Toppings.
  - **American Pizza** has Mozzarella, Pepperoni and Tomato Sauce Toppings.
  - **American Hot Pizza** has Jalapeno Pepper, Mozzarella, Pepperoni, and Tomato Sauce Toppings.
  - **Soho Pizza** has Garlic, Olive, Parmesan, and Tomato Sauce Toppings.

Protégé Practices

- The “Equivalent To”
  - “Equivalent To” => necessary & sufficient conditions
    - Let class C be an equivalent class of some necessary &sufficient conditions
    - All the individuals and subclasses of C **are necessary to fulfill** those conditions.
    - Meanwhile, an individual (a class) that satisfy those conditions **is sufficient to be** a member (a subclass) of C.



PUCPR

Protégé Practices

- Existential Restrictions ("some")
- Cheesy Mozzarella Pizzas
  - All Cheesy Mozzarella Pizzas (all the individuals and subclasses of class: CheesyMozzarellaPizza) should contains at least one mozzarella topping (one individual in Class: MozzarellaTopping).
  - A pizza (a collection of pizzas) that contains at least one mozzarella topping (one individual in Class: MozzarellaTopping) is sufficient to be a member (a subset) of Cheesy Mozzarella Pizzas

PUCPR

Protégé Practices

- Universal Restrictions ("only")
- Vegetarian Pizzas
  - All Vegetarian Pizzas (all the individuals and subclasses of class: VegetarianPizza) should contains only some cheese or vegetable toppings (some individuals from the classes: CheeseTopping or VegetableTopping).
  - A pizza (a collection of pizzas) that contains only some cheese or vegetable toppings (some individuals from the classes: CheeseTopping or VegetableTopping) is sufficient to be a member (a subset) of Vegetarian Pizzas

PUCPR

Protégé Practices

- Cardinality Restrictions ("min", "max", or "exactly")
- Interesting Pizzas
  - All Interesting Pizzas (all the individuals and subclasses of class: InterestingPizza) should contains three or more toppings (at least three individuals from the class: PizzaTopping).
  - A pizza (a collection of pizzas) that contains three or more toppings (at least three individuals from the class: PizzaTopping) is sufficient to be a member (a subset) of Interesting Pizzas

PUCPR

Protégé Practices

- Value Restrictions ("value")
- The Country of Creation
  - An object property named hasCountryOfOrigin
  - Specify the country of origin for different pizzas
    - **Margherita Pizza** was originally created in Italy.
    - **American Pizza** was originally created in America.
    - **American Hot Pizza** was originally created in America.
    - **Soho Pizza** was originally created in England.
  - A Class named ItalianPizza to classify all pizzas that has Italy as their country of origin.

Thank you
for your attention!

Any Questions?

PUCPR