Protégé-OWL Tutorial

Adriano Melo
André Chagas
Fred Freitas

Sistemas Inteligentes
http://www.cin.ufpe.br/~if684

Centro de Informática
U·F·P·E

Instalação

Download do Protégé

- public de astm
- stanford.edu (site oficial)

Protégé 3.4.4

- OWL 1.0
- Precisa ser instalado
- Muitos plugins

- **Protégé 4.1 alpha**
  - OWL 2.0
  - Não precisa ser instalado (apenas descompactado)
  - Poucos plugins (ainda)

Ontologia OWL

Modelar de forma declarativa um **domínio**.

- **Hierarquia** de conceitos e suas **relações**, **restrições**, **axiomas** e **terminologia** associada.

OWL: Linguagem para **representação de conhecimento** criada pela w3c.

- Basicamente: classes, propriedades e indivíduos.
- Hipótese de **mundo aberto**.

Dialetos da OWL

OWL – Lite

- **Hierarquia** entre classes e **restrições** simples;

OWL – DL

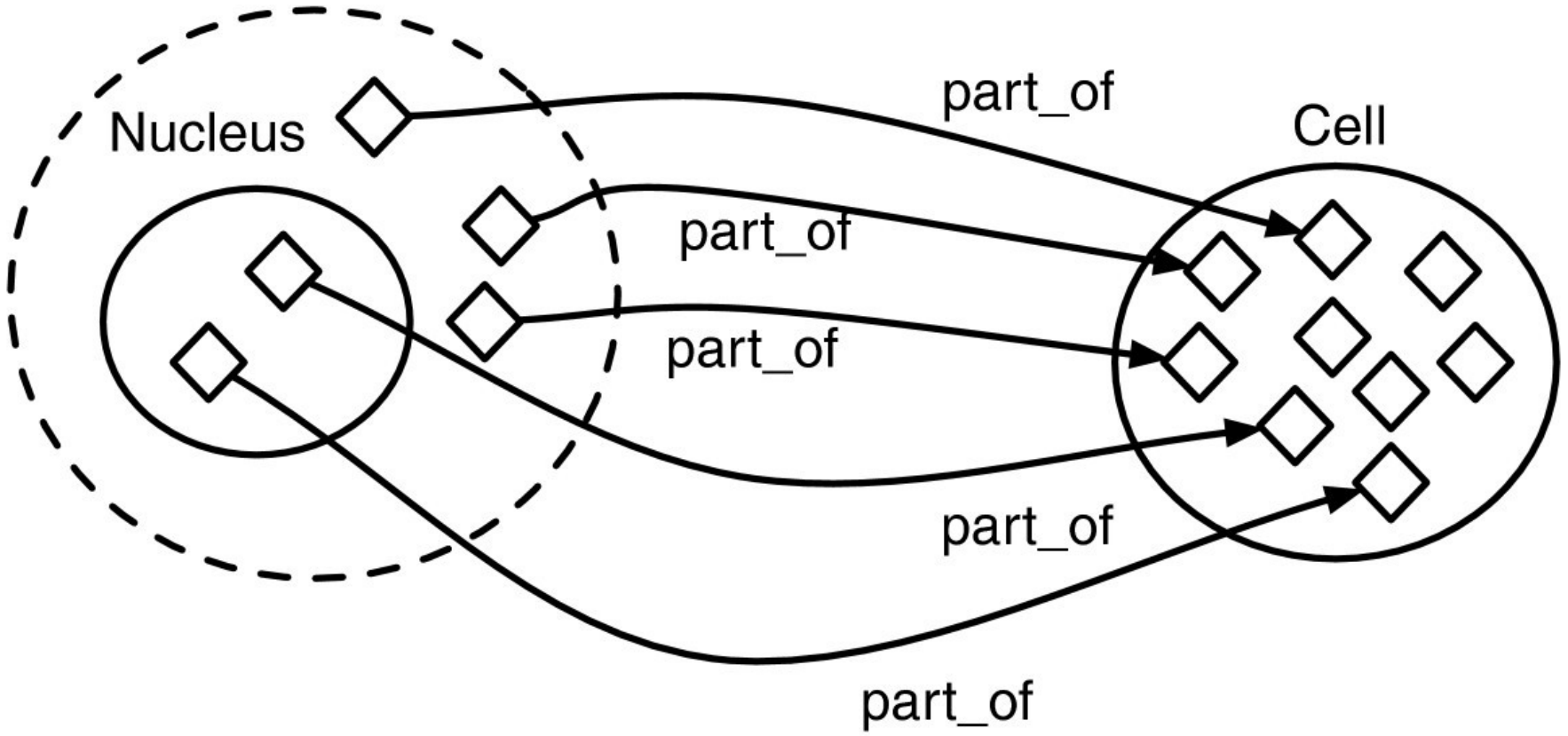
- Baseada em lógica de descrição (***DL***);
- Permite a **classificação automática** da hierarquia;
- Permite checar **inconsistências** na ontologia;

OWL – Full

- Expressividade é mais importante do que a decidibilidade;

Componentes da ontologia

- Uma ontologia OWL é composta pelos seguintes elementos: **Indivíduos**, **propriedades** e **classes**.

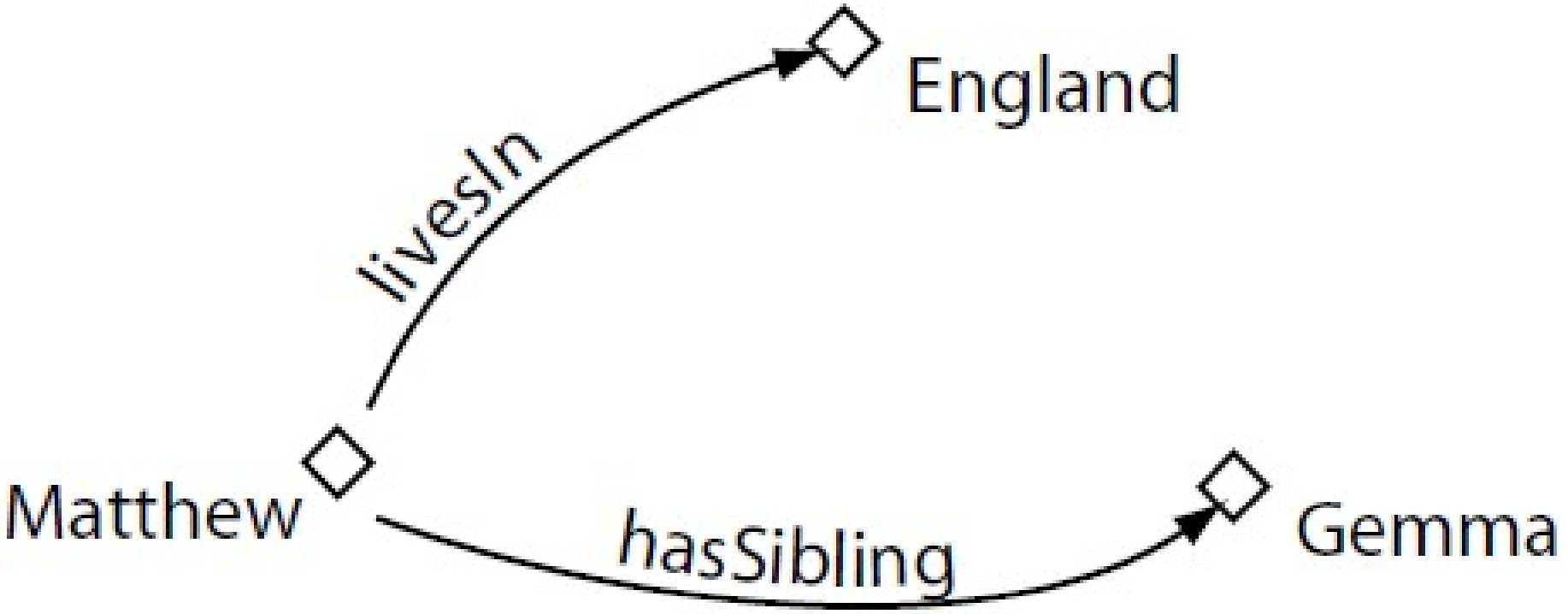


Indivíduos

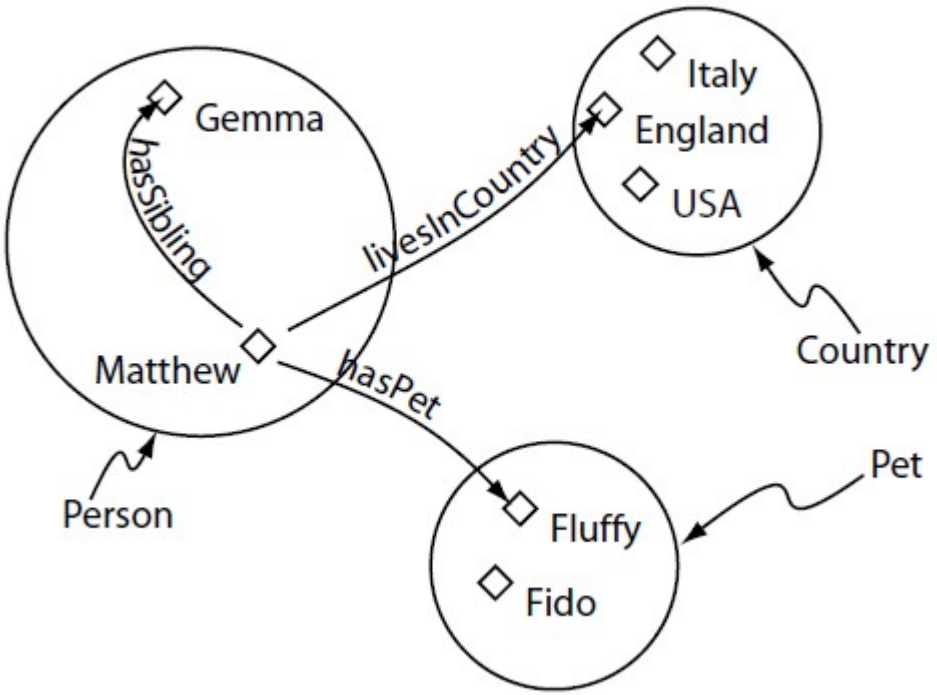
- Indivíduos são os objetos do domínio.

Propriedades

- Em **OWL**, propriedades representam as **relações** entre classes e **atributos**.



Classes



- Representação concreta de um **conceito** ou entidade.
- **Conjuntos** que **podem conter** indivíduos.

Tarefa #1
Criar um novo projeto

- Crie um novo projeto “**OWL / RDF Files**” que utilize o dialeto “**OWL DL**” para a construção de uma **ontologia de pizzas**.

Tarefa #2
Criar classes básicas

- Crie as classes **Pizza**, **CoberturaPizza** e **MassaPizza**. Assegure elas sejam subclasses de **owl:Thing.**

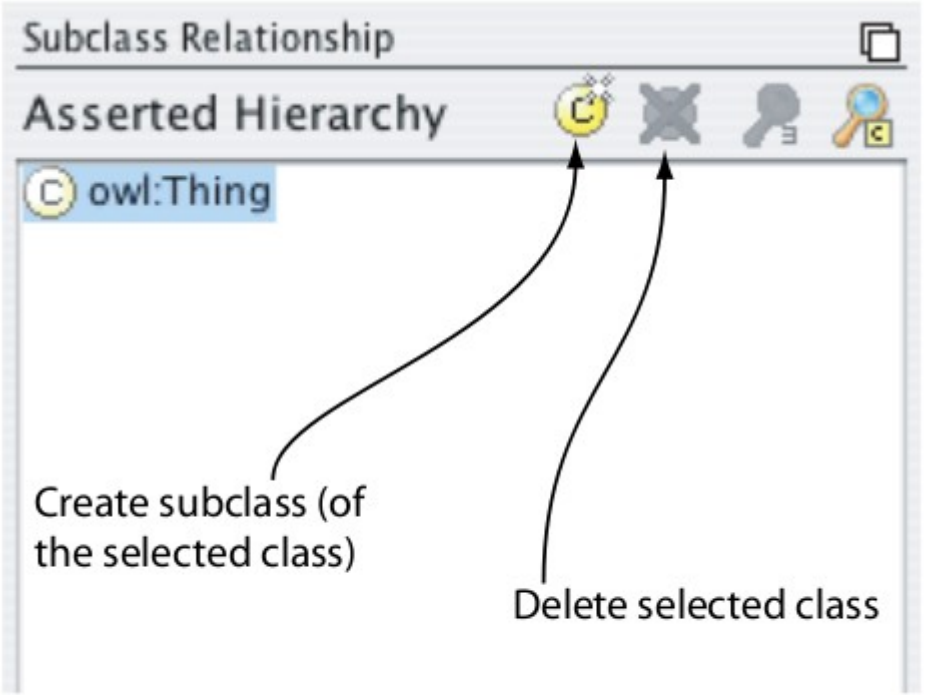


Figure 4.2: The Class Hierarchy Pane

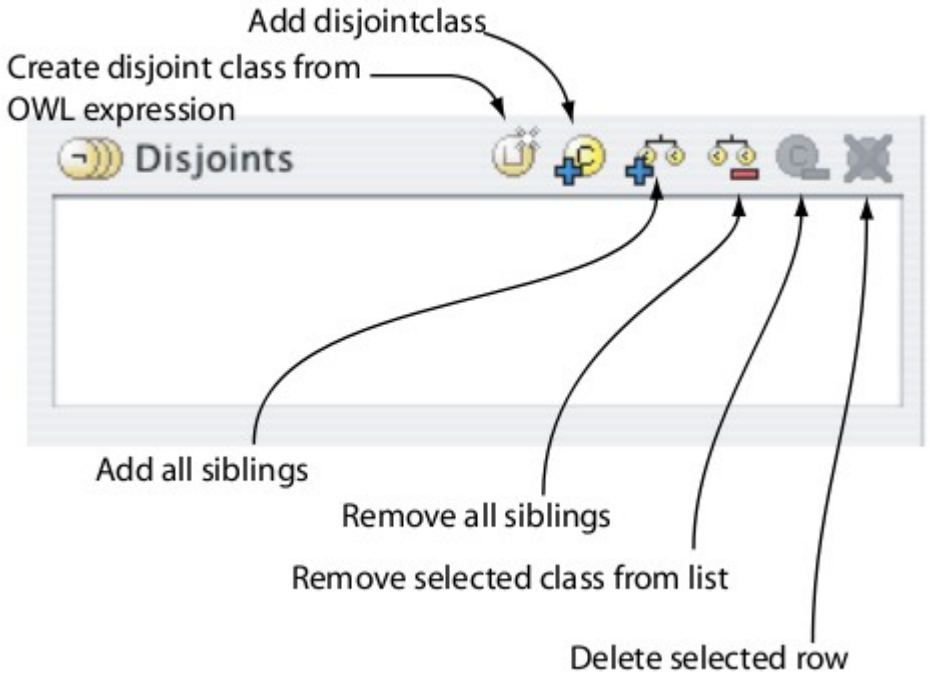
Disjunção entre classes

- Classes que **não são ditas como diferentes** podem ser consideradas **equivalentes** ou com interseção diferente de vazio. (hipótese de **mundo aberto**)

- Definição: **classes disjuntas** são aquelas que não possuem interseção (não podem ter indivíduos em comum).

Tarefa #3
Definir classes básicas como disjuntas

- Faça com que as classes **Pizza**, **CoberturaPizza** e **MassaPizza** sejam disjuntas.



Tarefa #4

Adicionar subclasses das classes básicas

- Crie as seguintes subclasses de **MassaPizza**: **MassaFina**, **MassaGrossa**.

- Crie as seguintes subclasses de **CoberturaPizza**: **CoberturaCarne**, **CoberturaVegetal**, **CoberturaQueijo**, **CoberturaFrutosDoMar**.

- Adicione classes a essas subclasses. *(exemplos: CoberturaPepperoni, CoberturaCalabreza, CoberturaFrango, CoberturaGafanhoto, CobeturaTomate, CoberturaOliva, CoberturaPimenta, CoberturaPimentaVermelha, CoberturaPimentaVerde, CoberturaMuzarela, CoberturaParmesão, CoberturaCamarão, CoberturaLeãoMarinho)*

Propriedades

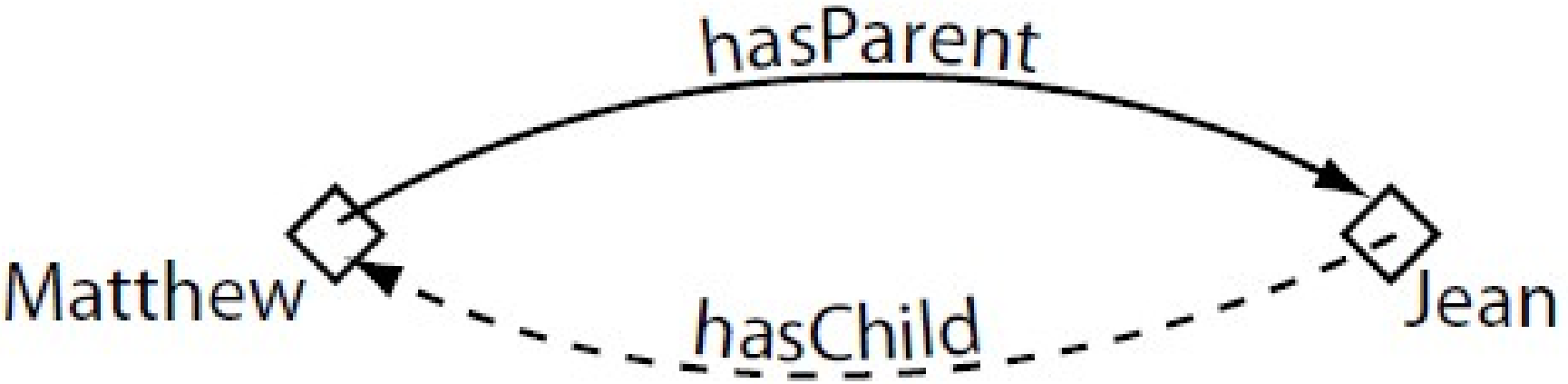
- ***object properties***
  - Relação binária entre indivíduos.
- ***datatype properties***
  - Relação entre um indivíduo e um tipo predefinido.
- ***annotation properties***
  - Adicionar informações sobre classes, objetos, indivíduos...

Tarefa #5
Criar propriedades

- Crie uma propriedade de **objeto** chamada **temIngrediente.**
- Crie as propriedades **temMassa** e **temCobertura**, ambas subpropriedades de **temIngreditente**.

Propriedade Inversa

- Toda propriedades pode ter uma propriedade inversa correspondente.

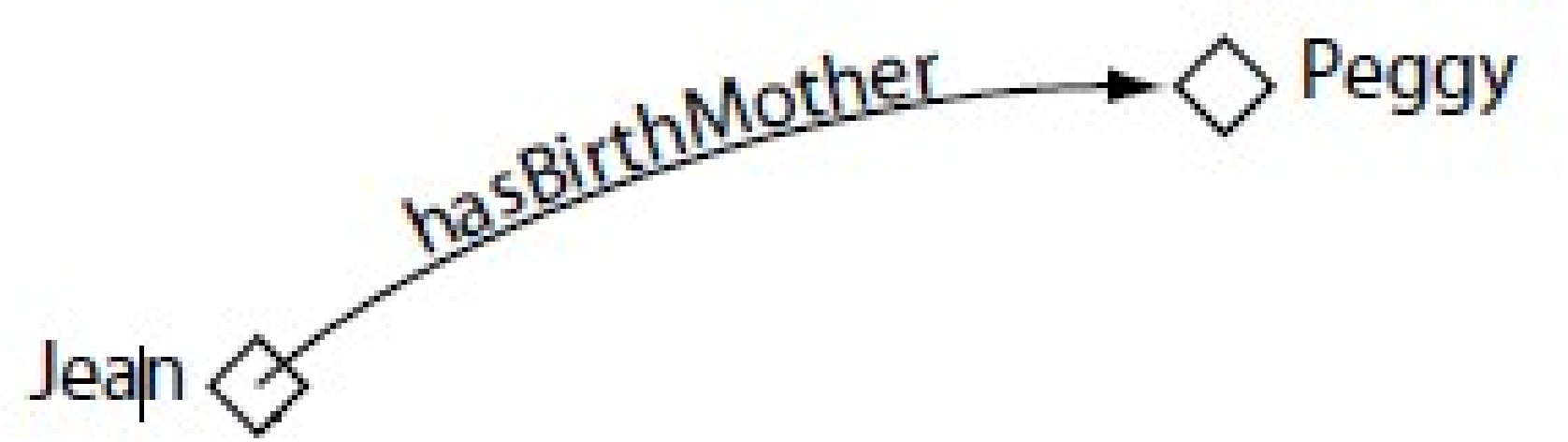


Tarefa #6
Criar propriedades inversas

- Crie as propriedades **ehIngredienteDe**, **ehMassaDe**, **ehCoberturaDe** como sendo as propriedades inversas de **temIngrediente**, **temMassa** e **temCobertura**, respectivamente.

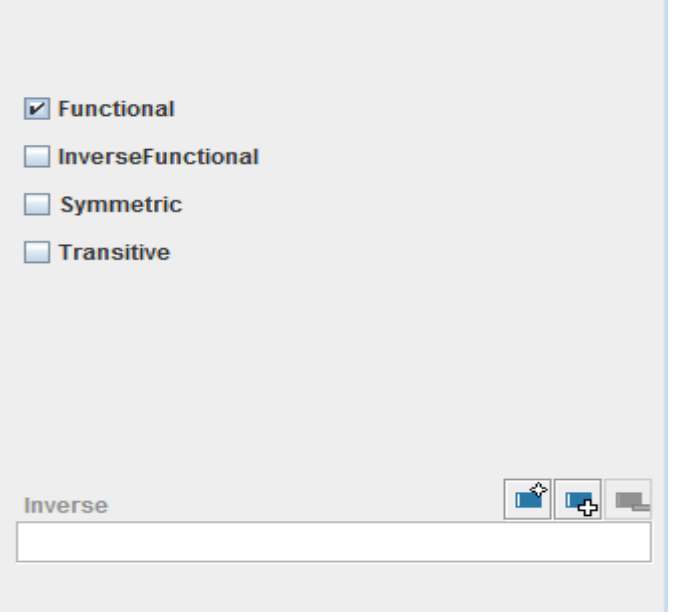
Propriedade Funcional

- Se uma propriedade é dita funcional para dado indivíduo, ele pode se relacionar a **apenas um** outro indivíduo a partir dela.



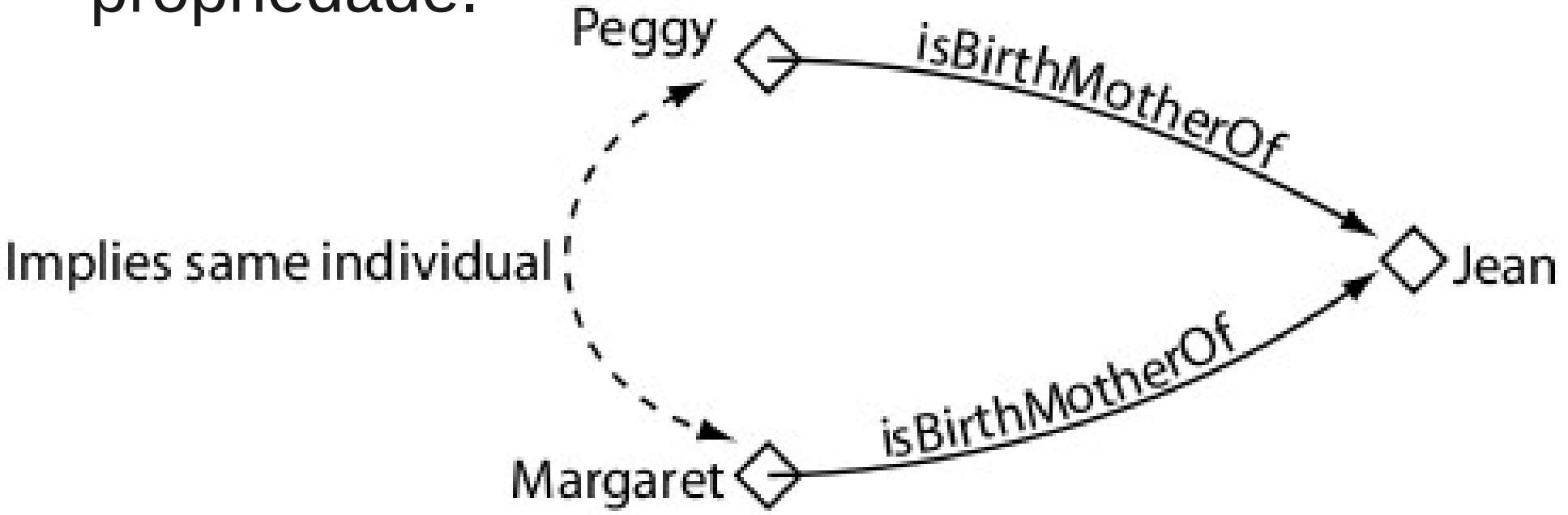
Tarefa #7
Criar propriedade funcional

- Faça com que a propriedade **temMassa** seja funcional.



Propriedade Funcional Inversa

- Se uma propriedade é funcional inversa significa que a relação **inversa é funcional**.
- Para um dado indivíduo, só pode **haver um outro indivíduo relacionado a ele** através da propriedade.



Tarefa #8
Criar propriedade funcional inversa

- Faça com que a propriedade **ehMassaDe** seja uma propriedade funcional inversa.

Propriedade Transitiva

- Se a propriedade é transitiva e **a** e **b** se relacionam por ela assim como **b** e **c,** então é possível inferir que **a** e **c** também se relacionam.

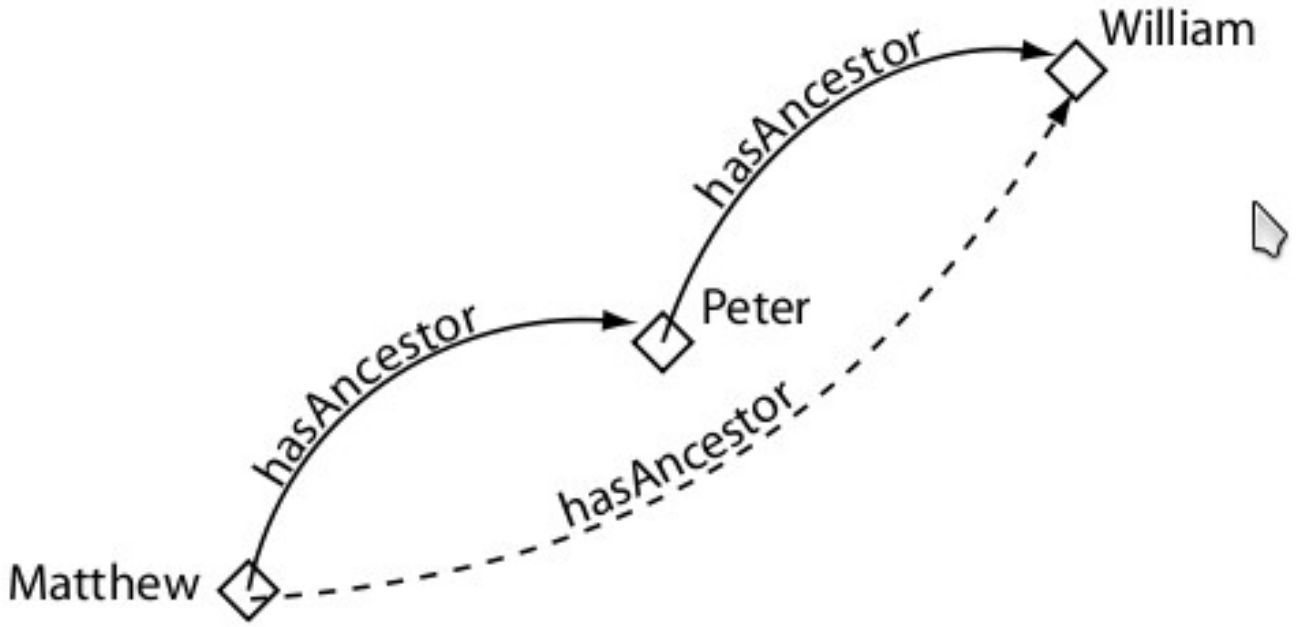


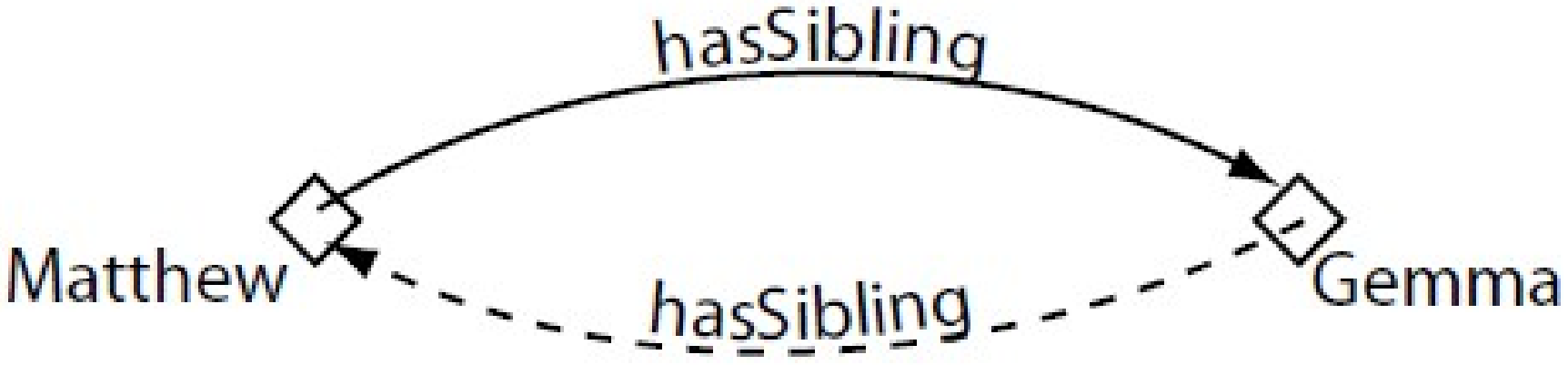
Figure 4.21: An Example Of A Transitive Property: hasAncestor

Tarefa #9
Criar propriedade transitiva

- Faça com que as propriedades **temIngrediente** e **ehIngredienteDe** sejam transitivas.

Propriedade Simétrica

- Uma relação binária é simétrica se qualquer **aRb implica em bRa**.



Tarefa #10
Criar propriedade simétrica

- Crie a propriedade simétrica **temGostoParecido** entre duas coberturas de pizza.

Domínio e subdomínio de propriedades

- Podem ser definidos para propriedades **entre indivíduos**.
- Não são interpretados como restrições, e sim como **axiomas**.
- Quando multiplas classes compoẽm o subdomínio ou o domínio de uma relação, então é considerada a **união** dessas classes.

Tarefa #11

Definir domínio e subdomínio das propriedades

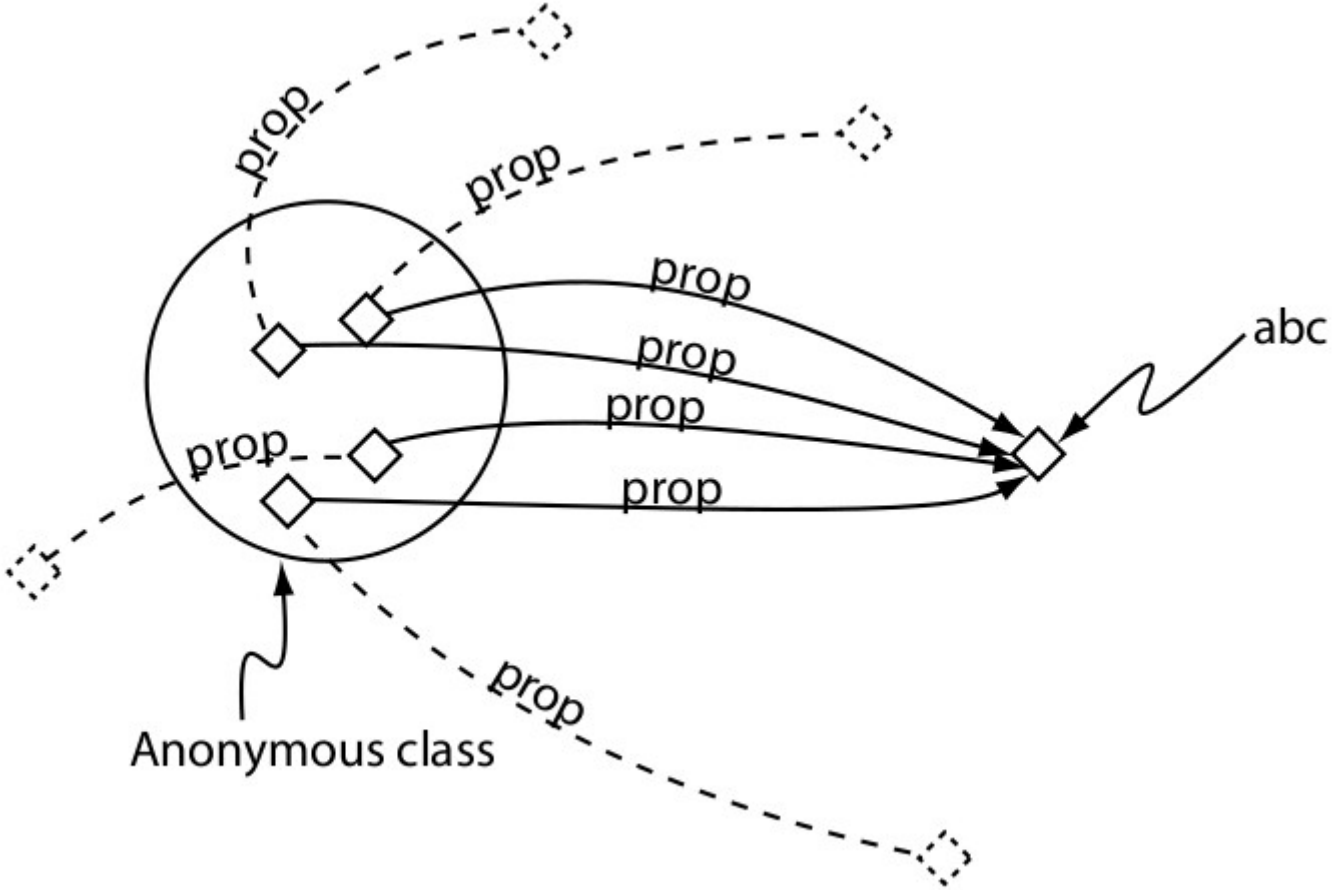
- Defina que a propriedade **temCobertura** tem como domínio **Pizza** e como subdomínio **CoberturaPizza.**

Restrições

- São definidas para restringir quais são os indivíduos pertencentes a uma classe.
- Os três principais tipos de restrição são:
  - Restrições com **quantificadores** (∃ e ∀)
  - Restrições de **cardinalidade**
  - Restrições de **valor**

Restrições de valor

- Descreve um conjunto de indivíduos que se relacionam com um outro **indivíduo específico (valor).**

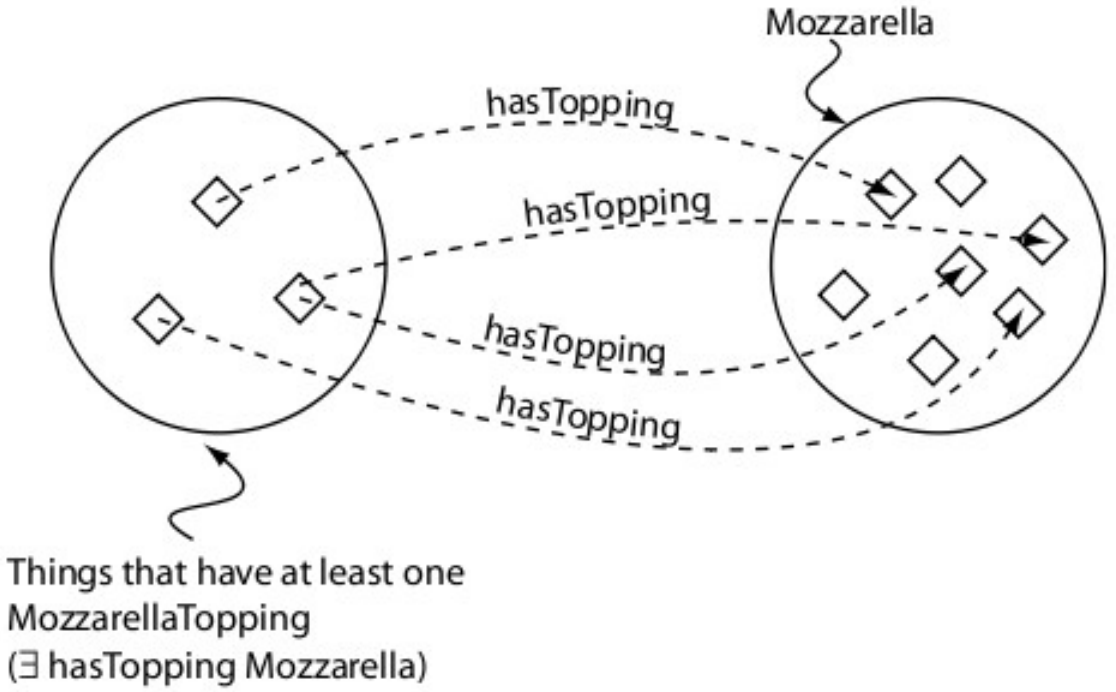


Restrições de cardinalidade

- **Restrição de Cardinalidade Mínima**
  - Especifica se um indivíduo tem **“pelo menos”** uma quantidade N de relações usando a propriedade P
- **Restrição de Cardinalidade Máxima**
  - Especifica se um indivíduo tem **“no máximo”** uma quantidade N de relações usando a propriedade P
- **Restrição exata da Cardinalidade**
  - Especifica se um indivíduo tem **“exatamente”** uma quantidade N de relações usando a propriedade P

Restrições com quantificadores

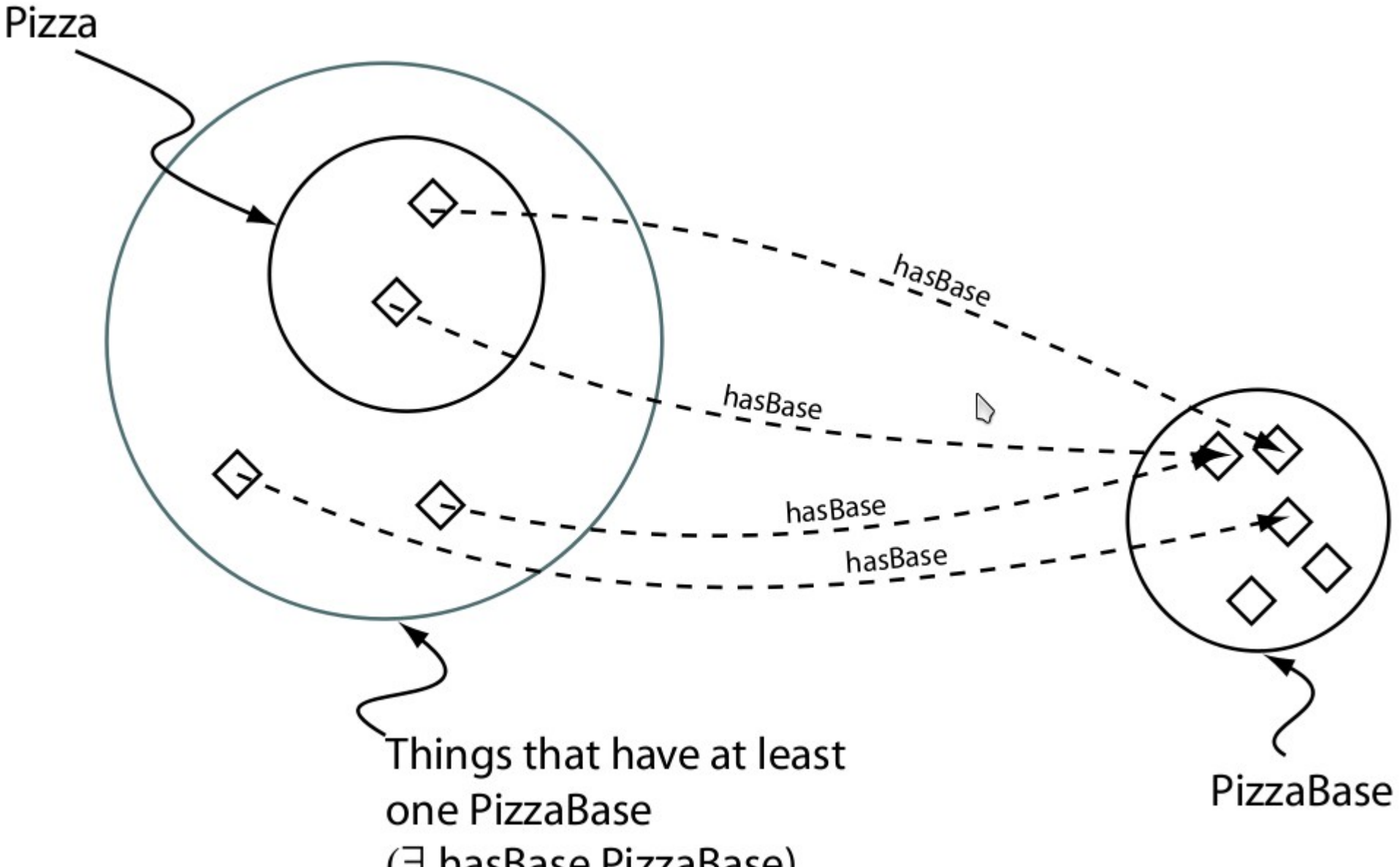
- **Quantificador Existencial (∃)**
  - Pode ser lido como ***"pelo menos um"*** ou ***"vários"***
- **Quantificador Universal (∀)**
  - Pode ser lido como ***"somente"***



Tarefa #12

definir restrições nas classes básicas

- Defina que a toda **Pizza** tem que ter uma **MassaPizza**.
- Crie a **Pizza PizzaMuzarela** com sendo uma **Pizza** que possui apenas **CoberturaMuzarela**.

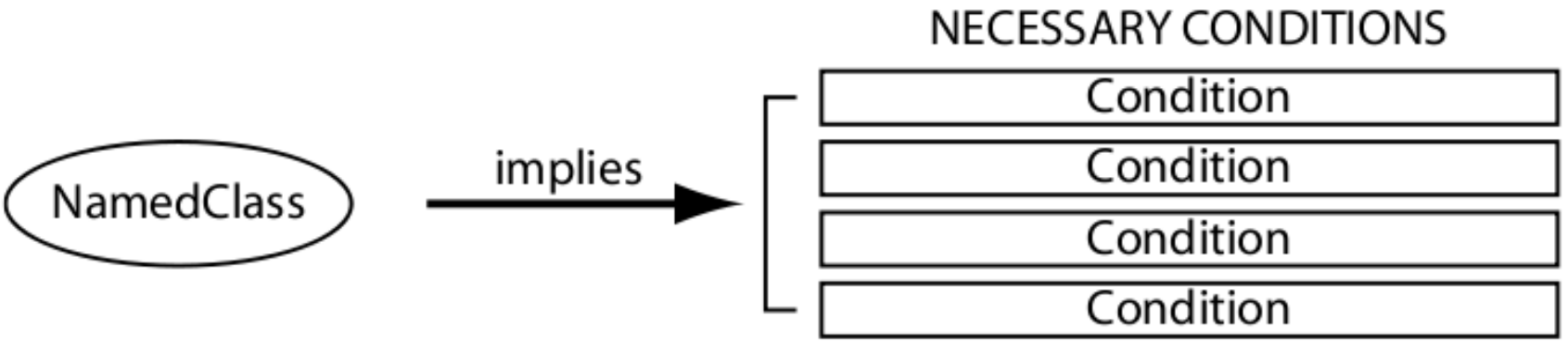

Pizza
hasBase
hasBase
hasBase
hasBase
Things that have at least
one PizzaBase
(∃ hasBase PizzaBase)
PizzaBase

Descrição de classes

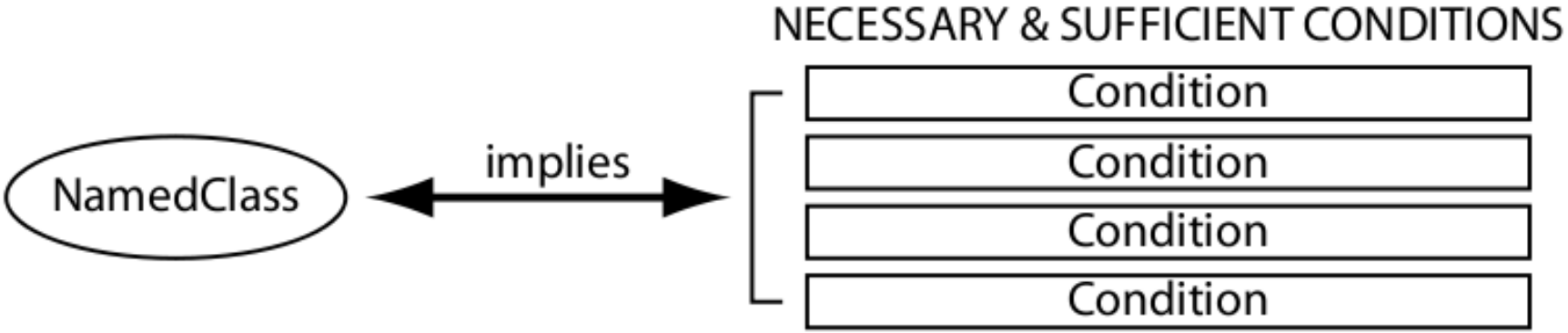
- A **descrição** de uma classe é feita quando apenas define-se as **condições necessárias** para que um indivíduo faça parte dela.
- São chamadas **Classes Primitivas** (ou parciais)
- Condição necessária: *"Se algo é membro dessa classe, então é necessário que ele respeite estas condições."*

Definição de classes

- A **definição** de uma classe é feita quando as **condições necessárias e suficiêntes** são descritas.
- São chamadas de **Classes Definidas** (ou completas)
- Condição necessária e suficiênte: *“Se algo respeitas essas condições, então ele é um membro desta classe.”*



If an individual is a member of 'NamedClass' then it must satisfy the conditions. However if some individual satisfies these necessary conditions, we cannot say that it is a member of 'Named Class' (the conditions are not 'sufficient' to be able to say this) - this is indicated by the direction of the arrow.



If an individual is a memeber of 'NamedClass' then it must satisfy the conditions. If some individual satisfies the condtions then the individual must be a member of 'NamedClass' - this is indicated by the double arrow.

Tarefa #13

Definir uma classe

- **Defina** uma classe **PizzaDeQueijo** como sendo qualquer **Pizza** que tenha uma cobertura **CoberturaQueijo**.

Raciocinador

- Infere **conseqüências lógicas** a partir de um conjunto de fatos ou **axiomas.**
- **Checagem de consistência**
  - Checa se alguma classe não pode ser instânciada ou se é contraditória.
- **Reclassificação da ontologia**
  - Reorganiza a hierarquia de classes a partir das classes definidas.
- **Infere tipos**
  - Computa equilvalência entre classes a partir das propriedades.
- **Suporte a regras**
  - SWRL...

Última Tarefa
Defina as seguintes pizzas

- **PizzaComQueijo**
  - Possui alguma combertura de queijo.
- **PizzaInteressante**
  - Possui 3 ou mais coberturas.
- **PizzaDeCarne**
  - Possui pelo menos uma cobertura com carne
- **Margherita**
  - Tem cobertura de Mozarela e Tomate
- **QuatroQueijos**
  - Tem quatro coberturas de queijo
- **PizzaNãoVegetariana**
  - Não é uma pizza vegetariana
- **PizzaItaliana**
  - Tem pais de origem a Itália e tem massa fina
- **PizzaVegetariana**
  - Não tem cobertura de peixe nem de carne.