Basics: Creating an Ontology

Credit

This assignment is identical to the one in the City uni course.

Ontology Modelling (Task OWL)

Create a small ontology that models the domain of Pizza_data. Use Protégé to develop the ontology. Save the ontology into turtle format (e.g., .ttl) and rdf/xml format (e.g., .owl).

Guidelines:

Subtask OWL.1: Discussion and motivation of the different modelling choices in the report (30%).

Subtask OWL.2: Change namespace for the ontology e.g., http://www.city.ac.uk/ds/inm713/your_name/ (5%).

Subtask OWL.3: Create a prefix (e.g., your initials ejr:) for the defined namespace (5%).

Subtask OWL.4: The ontology should abstract knowledge about the domain, valid for the data at hand but potentially valid for other datasets (e.g., tables). The ontology should, in principle, only contain concepts, object and data properties. Tip: you should difference between the general concept Margherita_Pizza and the margherita pizza served at "Pizzeria da Mario" (10%).

Subtask OWL.5: Organize classes into a hierarchy (15%).

Subtask OWL.6: Create local scope property restrictions or global scope ones (i.e., domain and range) as necessary (15%).

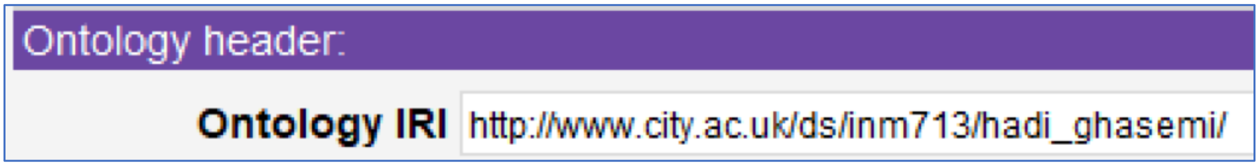
Subtask OWL.7: Create appropriate property characteristics (10%).

Subtask OWL.8: Create a label (e.g., name or synonym) and a comment (e.g., meaning of the concept) for each created entity (5%).

Subtask OWL.9: Create or reuse an annotation property creator and annotate/indicate the ontology has been created by you (5%).

The given dataset pertains to pizzas served in restaurants across the United States. The small version of the dataset for this lesson includes columns such as **name**, **address**, **city**, **postcode**, **state**, **categories**, **menu item, item value**, **currency**, and **item description**. Based on these columns classes like **restaurant**, **city**, **state**, **country**, **menu item**, and **currency** can be considered for the ontology model, while the **address** and **price** can be included as **literals** in the knowledge graph. Additionally, further thought should be given to identifying pizza categories, types, and descriptions. One simple way is to not consider pizza types as separate entities and have pizza descriptions associated with the pizza using a property, "has_description".

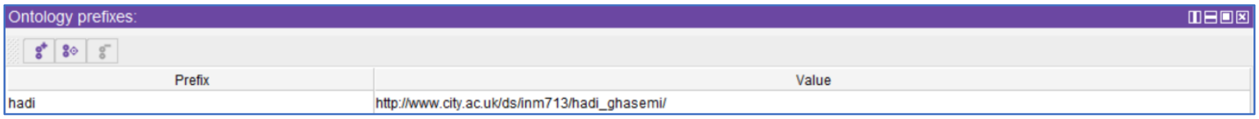
To begin the **namespace** of this ontology, it can be defined as follows:



Ontology header:

Ontology IRI http://www.city.ac.uk/ds/inm713/hadi_ghasemi/

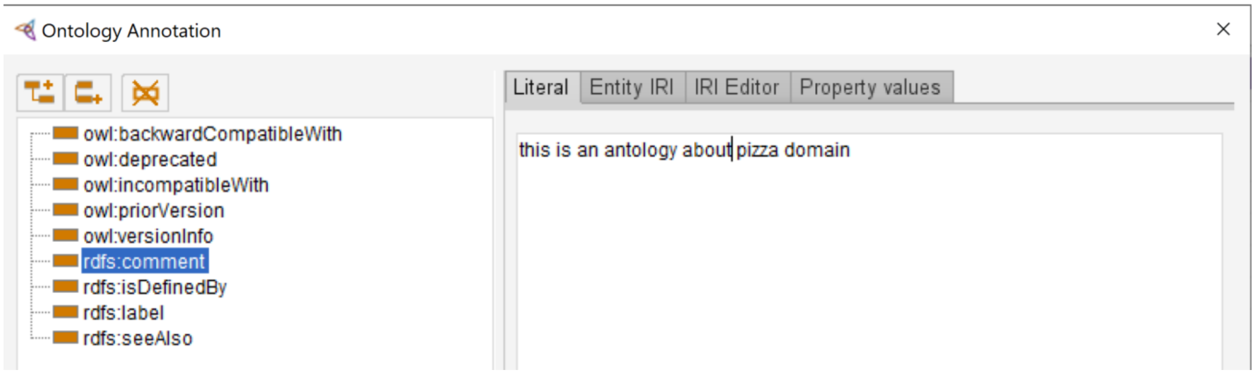
The prefix for this namespace can be defined as follows:



Ontology prefixes:

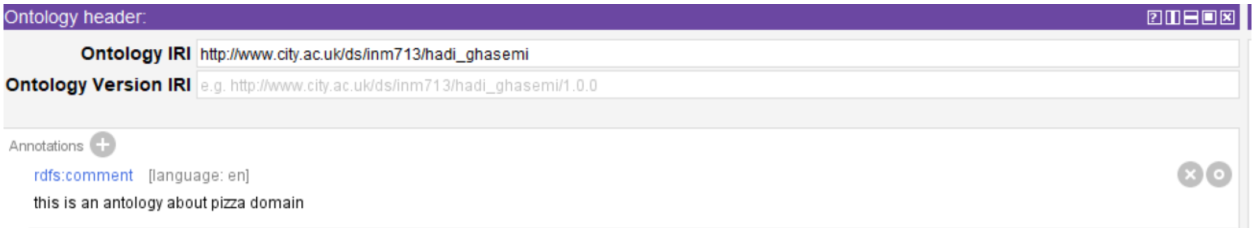
Prefix	Value
hadi	http://www.city.ac.uk/ds/inm713/hadi_ghasemi/

Further explanation about the ontology has been added in the annotation section as comments.



Ontology Annotation

this is an antology about pizza domain



Ontology header:

Ontology IRI http://www.city.ac.uk/ds/inm713/hadi_ghasemi

Ontology Version IRI e.g. http://www.city.ac.uk/ds/inm713/hadi_ghasemi/1.0.0

Annotations

rdfs:comment [language: en]

this is an antology about pizza domain

The initial idea was to define the classes and their subclasses hierarchically from top to bottom as follows: pizza is a subclass of the menu, and the menu is a subclass of the restaurant, and the restaurant is a subclass of the city, and so on.

Hierarchy:

- ❖ Country
    - ❖ Province
        - ❖ City
            - ❖ Restaurant
                - ❖ Menu
                    - ❖ Pizza

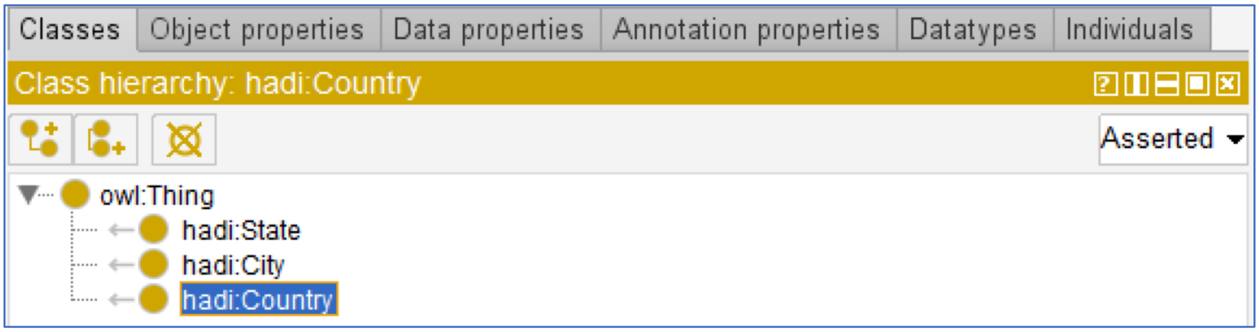
Where the hierarchy reflects the hierarchical relationships between classes, where a Country contains Provinces, Provinces contain Cities, and Cities contain Restaurants. Menus are associated with Restaurants, and Pizzas are included in Menus.

However, since we want to create an ontology in the domain of restaurants, the idea was implemented differently, and this hierarchy was not considered in it.

Classes

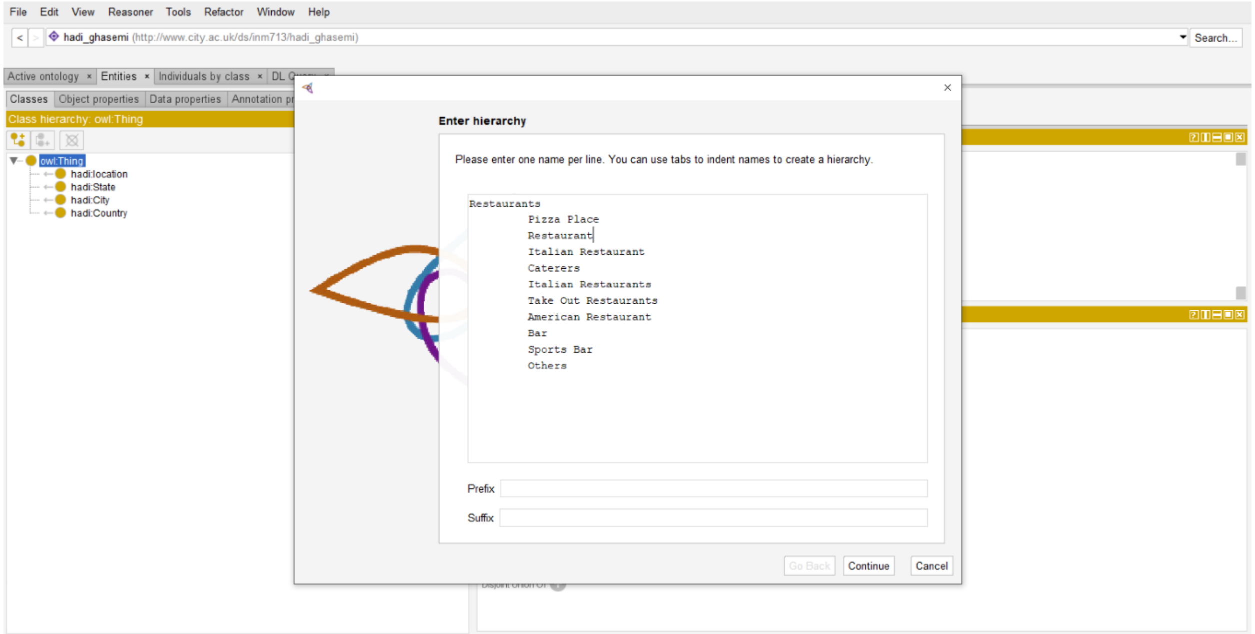
- Restaurant: Represents a category of restaurant.
- Pizza: Represents a type of pizza.
- Menu item: Represents a menu item.
- Country: Represents a country.
- State: Represents a province/state.
- City: Represents a city.
- Currency: Represents a Currency as dollar.
- Location: Represents a Building or store which has address and postal code.

Three classes of city, state, and country were created at first.



Then different types of restaurants and pizzas are added as subclasses of restaurant and pizza respectively.

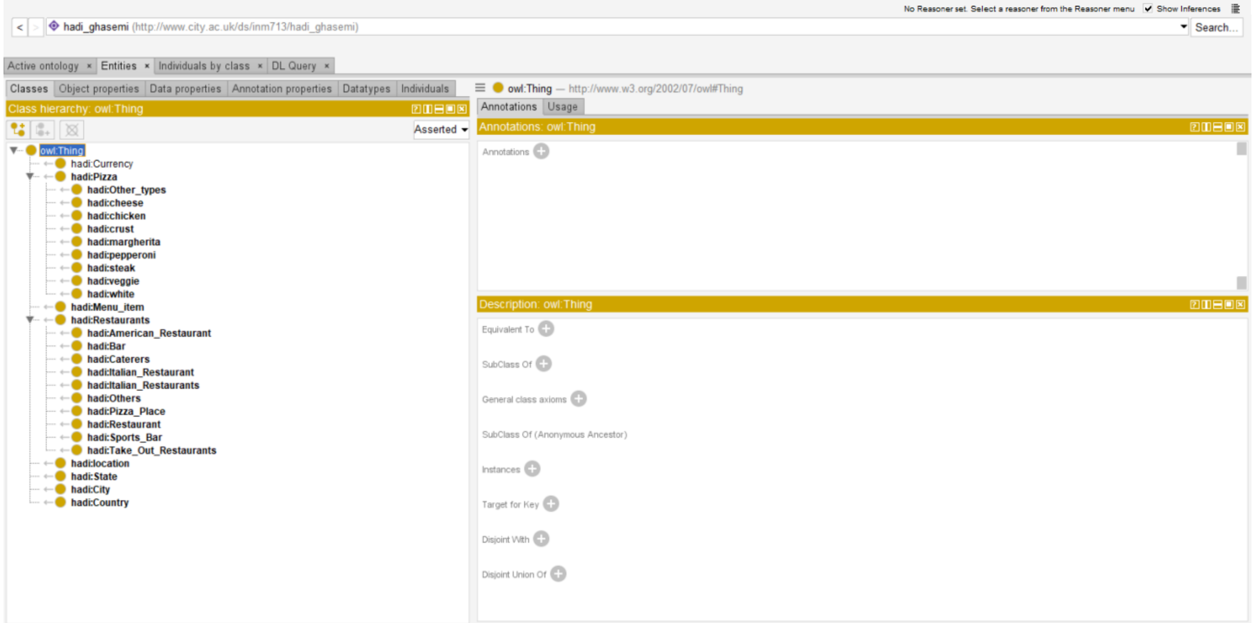
Finally, these classes and subclasses are constructed for ontology.



Please enter one name per line. You can use tabs to indent names to create a hierarchy.

```
Restaurants
        Pizza Place
        Restaurant
        Italian Restaurant
        Caterers
        Italian Restaurants
        Take Out Restaurants
        American Restaurant
        Bar
        Sports Bar
        Others
```

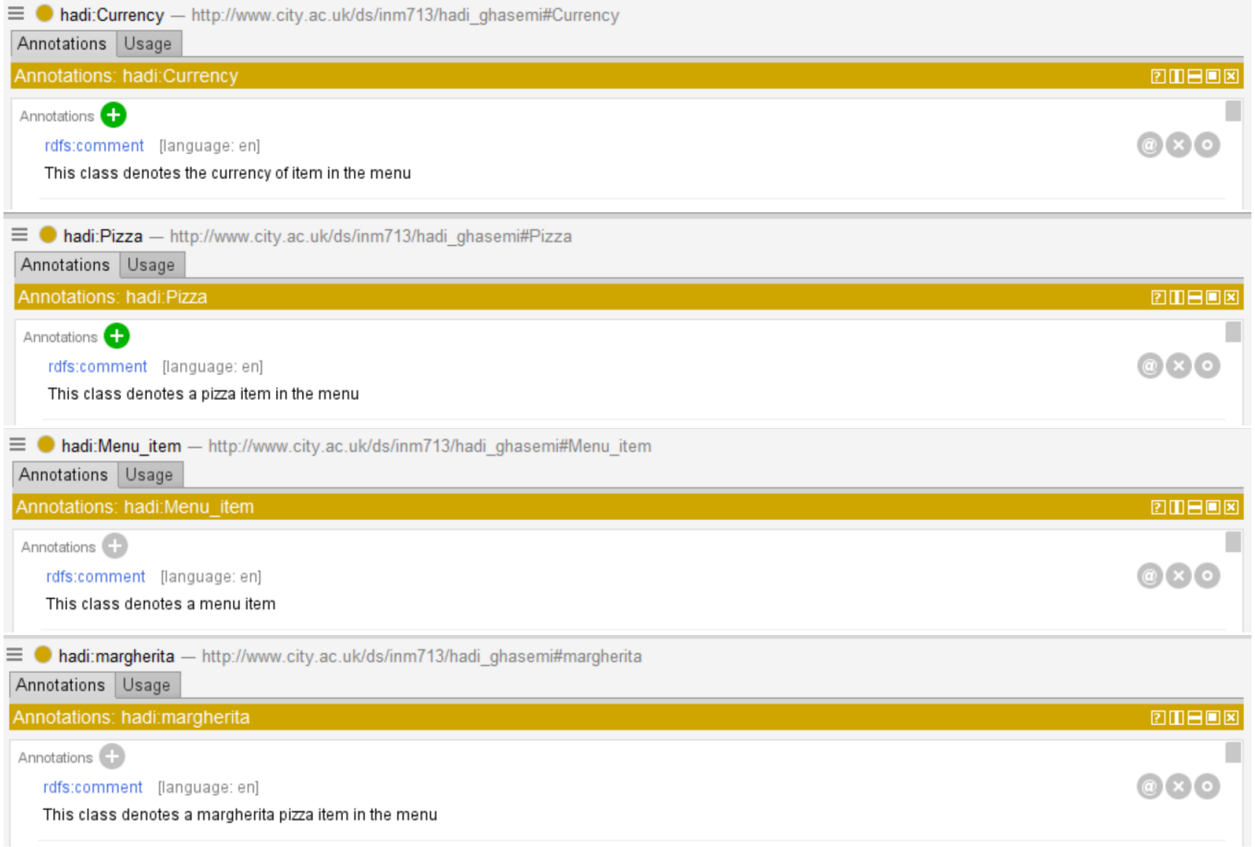
```
owl:Thing
    hadi:location
    hadi:State
    hadi:City
    hadi:Country
```



```
owl:Thing
    hadi:Currency
    hadi:Pizza
        hadi:Other_types
        hadi:cheese
        hadi:chicken
        hadi:crust
        hadi:margherita
        hadi:pepperoni
        hadi:steak
        hadi:veggie
        hadi:white
    hadi:Menu_item
    hadi:Restaurants
        hadi:American_Restaurant
        hadi:Bar
        hadi:Caterers
        hadi:Italian_Restaurant
        hadi:Italian_Restaurants
        hadi:Others
        hadi:Pizza_Place
        hadi:Restaurant
        hadi:Sports_Bar
        hadi:Take_Out_Restaurants
    hadi:location
    hadi:State
    hadi:City
    hadi:Country
```

Based on the given task requirement to create an ontology that is valid not only for the Pizza_data dataset but also potentially for other datasets, I have considered the concept of "Location" for other types of Building and also we have Menu item as an item of Menu which allows us to have non pizza food in the restaurant and the type of menu item is not restricted to pizza. This allows for the possibility of adding non-Pizza food items to the knowledge graph if they are present in the menu. Additionally, to create a distinction between general concepts and specific instances, such as differentiating between "Margherita Pizza" as a general concept and the specific instance of "Margherita Pizza" served at a specific restaurant, I have defined subclasses for different types of pizzas under the "Pizza" class. This way, pizzas served at a restaurant are connected to the respective restaurant via the menu, and they are also connected to the specific pizza type class. This allows for distinguishing them based on both the restaurant and the pizza type.

Then an annotation is written for each of the classes.



hadi:Currency — http://www.city.ac.uk/ds/inm713/hadi_ghasemi#Currency

rdfs:comment [language: en]

This class denotes the currency of item in the menu

hadi:Pizza — http://www.city.ac.uk/ds/inm713/hadi_ghasemi#Pizza

rdfs:comment [language: en]

This class denotes a pizza item in the menu

hadi:Menu_item — http://www.city.ac.uk/ds/inm713/hadi_ghasemi#Menu_item

rdfs:comment [language: en]

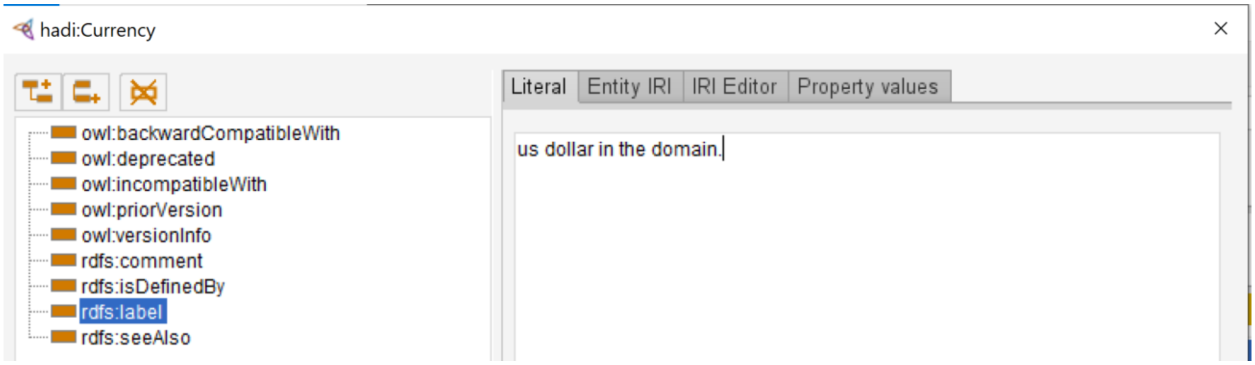
This class denotes a menu item

hadi:margherita — http://www.city.ac.uk/ds/inm713/hadi_ghasemi#margherita

rdfs:comment [language: en]

This class denotes a margherita pizza item in the menu

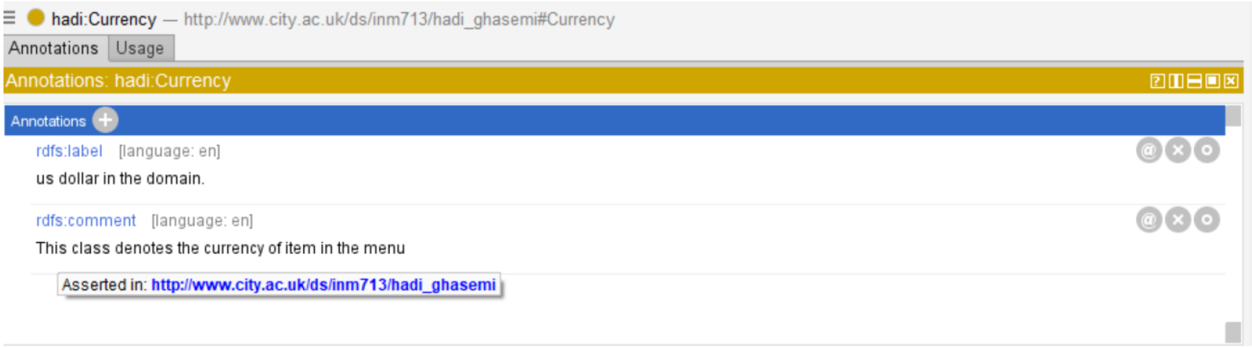
Also, labels are added to the classes.



hadi:Currency

- owl:backwardCompatibleWith
- owl:deprecated
- owl:incompatibleWith
- owl:priorVersion
- owl:versionInfo
- rdfs:comment
- rdfs:isDefinedBy
- rdfs:label
- rdfs:seeAlso

us dollar in the domain.



hadi:Currency — http://www.city.ac.uk/ds/inm713/hadi_ghasemi#Currency

rdfs:label [language: en]

us dollar in the domain.

rdfs:comment [language: en]

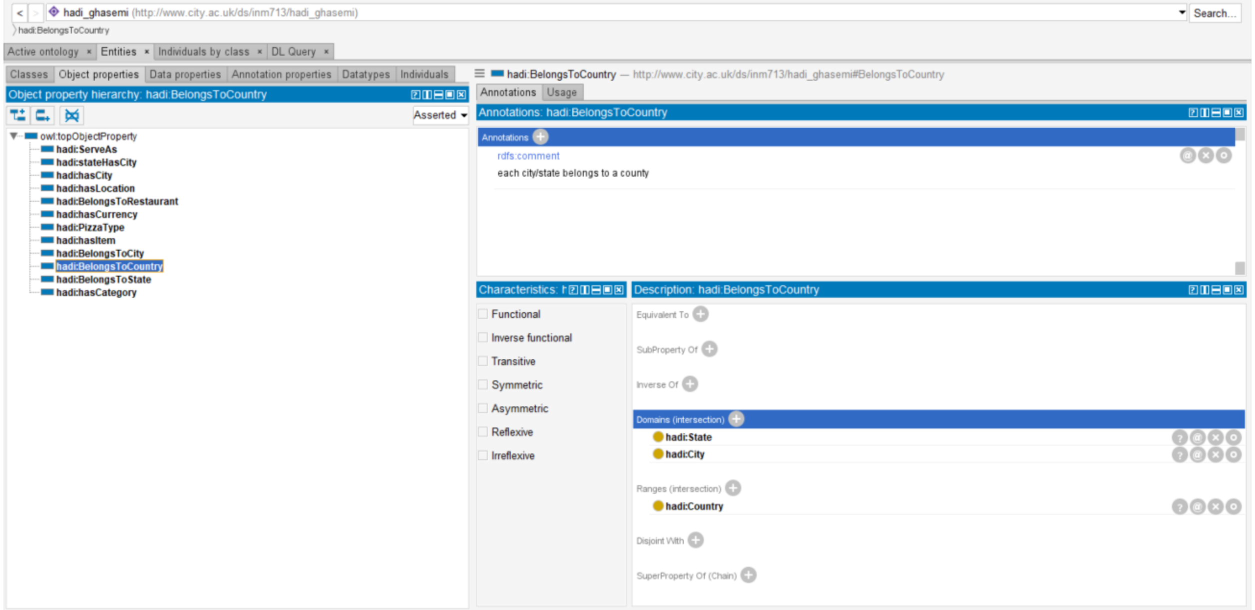
This class denotes the currency of item in the menu

Asserted in: http://www.city.ac.uk/ds/inm713/hadi_ghasemi

Object Properties

Object properties have been used to establish relationships between classes, such as linking a Restaurant to its menus, its location, and the city to which it belongs.

Then object properties define as this:



hadi_ghasemi (http://www.city.ac.uk/ds/inm713/hadi_ghasemi)

hadi:BelongsToCountry

Object property hierarchy: hadi:BelongsToCountry

- owl:topObjectProperty
  - hadi:ServeAs
  - hadi:stateHasCity
  - hadi:hasCity
  - hadi:hasLocation
  - hadi:BelongsToRestaurant
  - hadi:hasCurrency
  - hadi:PizzaType
  - hadi:hasItem
  - hadi:BelongsToCity
  - hadi:BelongsToCountry
  - hadi:BelongsToState
  - hadi:hasCategory

hadi:BelongsToCountry — http://www.city.ac.uk/ds/inm713/hadi_ghasemi#BelongsToCountry

Annotations: hadi:BelongsToCountry

Annotations

rdfs:comment
each city/state belongs to a county

Characteristics: h

□ Functional
□ Inverse functional
□ Transitive
□ Symmetric
□ Asymmetric
□ Reflexive
□ Irreflexive

Description: hadi:BelongsToCountry

Equivalent To

SubProperty Of

Inverse Of

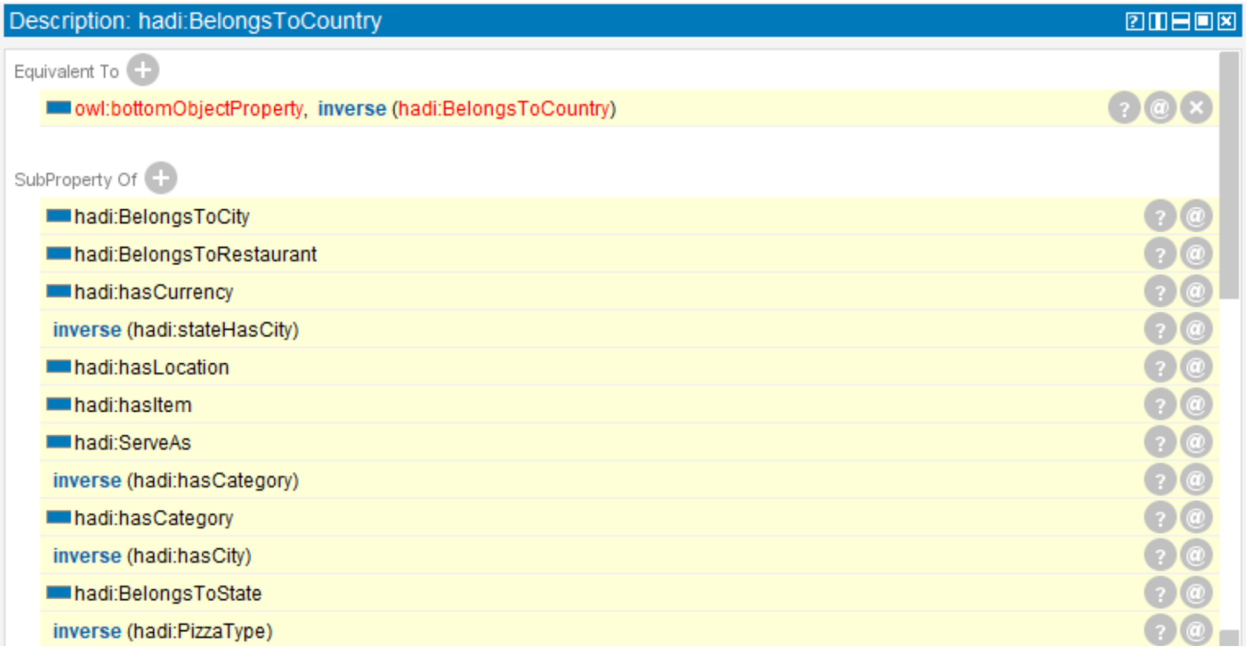
Domains (intersection)
- hadi:State
- hadi:City

Ranges (intersection)
- hadi:Country

Disjoint With

SuperProperty Of (Chain)

Now we use a reasoner to discover misplaced relationships.



Description: hadi:BelongsToCountry

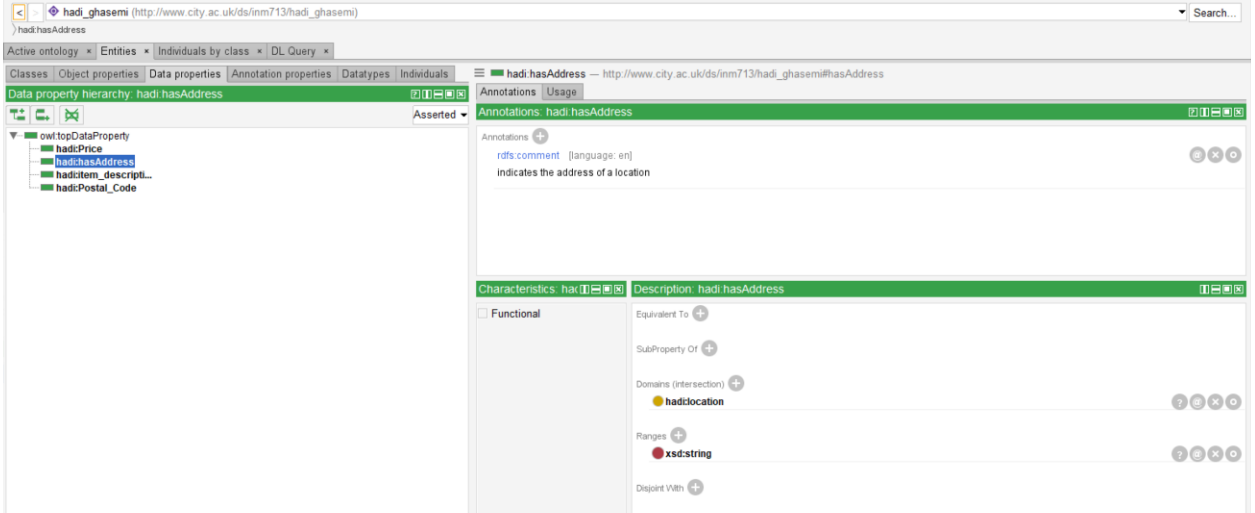
Equivalent To
- owl:bottomObjectProperty, inverse (hadi:BelongsToCountry)

SubProperty Of
- hadi:BelongsToCity
- hadi:BelongsToRestaurant
- hadi:hasCurrency
- inverse (hadi:stateHasCity)
- hadi:hasLocation
- hadi:hasItem
- hadi:ServeAs
- inverse (hadi:hasCategory)
- hadi:hasCategory
- inverse (hadi:hasCity)
- hadi:BelongsToState
- inverse (hadi:PizzaType)

Data Properties

Data properties have been selected to capture the attributes associated with each class, such as the address, categories, coordinates, URL, price range, and postal code.

Next we will create data properties as below:



hadi_ghasemi (http://www.city.ac.uk/ds/inm713/hadi_ghasemi)

hadi:hasAddress

owl:topDataProperty
- hadi:Price
- hadi:hasAddress
- hadi:item_descripti...
- hadi:Postal_Code

hadi:hasAddress — http://www.city.ac.uk/ds/inm713/hadi_ghasemi#hasAddress

Annotations
rdfs:comment [language: en]
indicates the address of a location

□ Functional

Equivalent To

SubProperty Of

Domains (intersection)
hadi:location

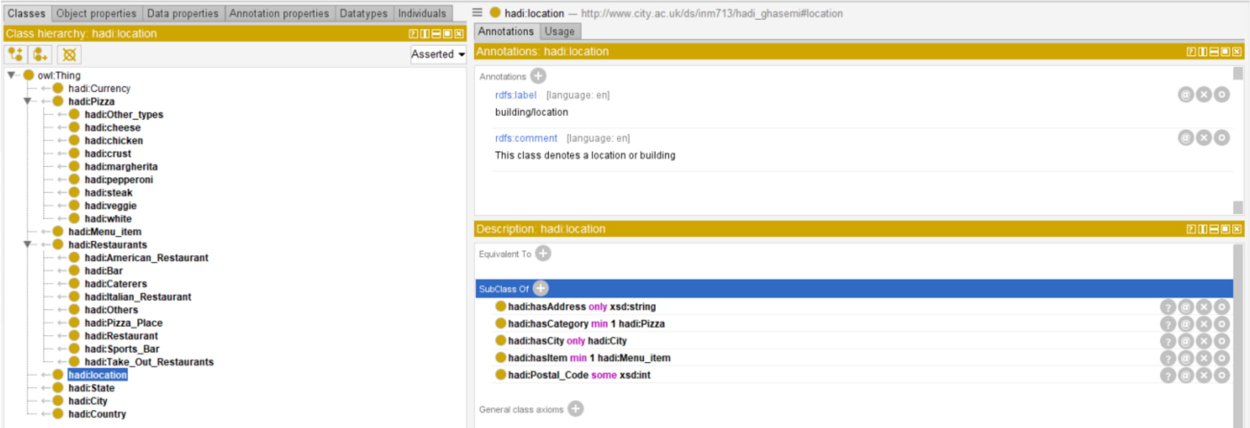
Ranges
xsd:string

Disjoint With

Restrictions:

Based on the EDA analysis performed on the dataset using pandas, it was observed that some restaurants lack postal codes. Therefore, the cardinality of the property associated with it should be at least 0 and at most 1.

Furthermore, each restaurant can have multiple categories. For example, a restaurant may fall into categories such as "pizza place," "American restaurant," and "bar." Therefore, the cardinality of the property associated with these categories should be at least 1 and can be infinite. Additionally, there are no restrictions on the number of items available in a restaurant's menu.



owl:Thing
- hadi:Currency
- hadi:Pizza
  - hadi:Other_types
  - hadi:cheese
  - hadi:chicken
  - hadi:crust
  - hadi:margherita
  - hadi:pepperoni
  - hadi:steak
  - hadi:veggie
  - hadi:white
- hadi:Menu_item
- hadi:Restaurants
  - hadi:American_Restaurant
  - hadi:Bar
  - hadi:Caterers
  - hadi:Italian_Restaurant
  - hadi:Others
  - hadi:Pizza_Place
  - hadi:Restaurant
  - hadi:Sports_Bar
  - hadi:Take_Out_Restaurants
- hadi:location
- hadi:State
- hadi:City
- hadi:Country

hadi:location — http://www.city.ac.uk/ds/inm713/hadi_ghasemi#location

Annotations
rdfs:label [language: en]
building/location

rdfs:comment [language: en]
This class denotes a location or building

Equivalent To

SubClass Of
- hadi:hasAddress only xsd:string
- hadi:hasCategory min 1 hadi:Pizza
- hadi:hasCity only hadi:City
- hadi:hasItem min 1 hadi:Menu_item
- hadi:Postal_Code some xsd:int

General class axioms

For the remaining properties, both the minimum and maximum cardinality should be 1.