Research Report

Web Scales

Taxonomies

Name: Saurabh M. Nair
Matriculation number: 221100437
Email id: saurabhnair1999@uni-koblenz.de

April 2022 – October 2022

1. Introduction

Ontology is a formal way of describing knowledge that consists of a set of perceptions for a specific field and a relationship between them. Hence, to achieve this, we need to initially develop a list of classes, attributes, and relationships among them.[1]
It is important to develop a domain ontology so that one can share a general idea about the domain. It can be suitable for further research in the specific domain. Not only for a brief overview of the domain but also for a clear understanding of the hierarchy and relationship between classes. It is very useful for users to navigate from one ontology to another one easily. Even it is easy to add other related features so that it can keep growing.[2]
Firstly, one needs to relate the same ontologies which make it easy to understand as only related classes and their relationships are together. In addition, this domain ontology makes the ontologies dynamic. Hence, it is easy to apply new emerging ontologies to it.
A **class hierarchy** in computer science is a classification of object types, denoting objects as the instantiations of class inter-relating the various classes by relationships such as "inherits", "extends", "is an abstraction of" and, "an interface definition".[6]
Taxonomical class hierarchy means the arrangement of classes in descending order in a top-down approach of a specific domain.[3] E.g., one parent class may contain zero or many subclasses which are also known as child classes. The class hierarchy takes almost everything into the account in a domain ontology that makes it clear to know its structure. It also makes it easy to understand the subclass and parent class relationship. Furthermore, it also makes it clear to understand the relation of one class of ontology with another. [3] In addition to this, we can even determine the sibling of the respective class. For creating a model of occupation as an ontology, firstly the classes and subclasses must be found followed by finding their relationship with each other so that the hierarchy of classes can be created. In such a way model for any ontology can be created.

2. State of the art

a. Top Down and Bottom-up approach

In the bottom-up approach, firstly collect all the concepts which come under that domain. After this only one can determine the relation of one concept with another. That is finding the parent class, siblings' class, and so on. In the way around at first, corpora of domain-related text are obtained via some statistical methods and then these are arranged in the hierarchical structure of the class.
In contrast, top-down approach the developer can decide the root class of the domain ontology. Using this root class, the developer extracts the related child classes or subclasses. Later on, the relationships are built upon these classes.
In the top-down approach start with the higher hierarchical classes and populate them with the classes with the narrow concept. From the previous schema or from the queries or from the corpora one must sort them by grouping them into hierarchical order.[4]

Hence, In the top-down development process, general concepts are defined first, and then the subsequent specializations are defined. In the bottom-up development process, specific classes are defined first and then we group these classes into more general concepts.[7]
The top-down approach can be used as a strategy for identifying concepts. The ontology can be developed and evaluated by the protégé tool query.[8] Visualization these ontologies can be even done using neo4j. Web scraping is also one of the way to get or expand the existing ontology.

b. Advantages and Disadvantages of both approches

Both approaches are unique in their own way but with every pros there exist cons too. The bottom-up approach is more systematic, predictable, more linear, and less chaotic but it also needs a lot of domain knowledge and also takes high consumption of time as one needs to find the concepts that belong to the domain, and then the relationship with classes is being constructed.
On the other part, in the top-down approach, one can just take the root class and expand it. It doesn't need the developer to be specialized in the field of that specific field. It is easy for beginners to just know about the basics of that domain and then they can expand it easily. Furthermore, the quality of ontology is not systematic because the developer doesn't have full knowledge about that domain.
If a developer has a systematic top-down view of the domain, then it may be easier to use the top-down approach.[5]

3. Working Algorithm

To begin with, I have selected one specific domain with which I was somewhat familiar i.e. occupation. Then tried to understand the domain. What is occupation? How is it different from other domains and what are all things I have to keep in mind while gathering the subclasses and their relationships
I have used various resources to gather the subclasses, classes, and their relations. The root node (Occupation) was taken as a primary node to find its subnodes the same way these subnodes became the root node which was extended using the various websites manually.[9][10][11][12][13]
One of the subdomain in occupation is software engineer which I have extended automatically using web scraping.

a. Importing dependencies

To begin with I have imported all the required libraries (dependencies) which are useful for data aggregation and web scraping. BeautifulSoap is a python library that is used to pull data from HTML or XML documents. BeautifulSoap is built using python. Pandas are used for the manipulation of datasets. And NumPy is a fundamental package for scientific computing.

b. Web scraping

The software engineer occupation class was expanded using web scraping.
URLs are from a different website[9][10] that contains the subclasses of the software engineer occupation class.

A software engineer is an occupation that comes under engineering occupation.
Web scrapping was implemented using BeautifulSoap library. Using the request module, I have imported all the content from the HTML page in an HTML format. Later on, I tried to find certain tags or divisions tags that can lead me to the required content. After acquiring the required information from the web page data frame was created using the Pandas library where the column represented the name of the subclass and its superclass respectively.

c. Data Aggregation

Once both datasets are created data aggregation takes place where the removal of duplicate values takes place. Additionally, data is preprocessed too where the spaces are replaced with "_".

d. Saving the data In Postgres

Once the data is preprocessed, then one can just open the OWL file which is created by protégé and append this data into it. At last, when the data will be appended to the OWL file then it will get saved as "occupation_10.08.owl".

4. Evaluation

Total number of checked subclass and disjoint class axioms = 265
Total number of approved axioms = 261

$$\text{Precision} = \frac{\text{Total number of checked subclass and disjoint class axioms}}{\text{Total number of approved axioms}} * 100$$

$$= \frac{261}{265} * 100 = 98.49\%$$

5. References

[1] (n.d.). *Knowledge Graph in the Enterprise*. Onotext [Whitepaper]. https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/

[2] Noy, N. F., & McGuinness, D. L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford, CA, 94305. https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

[3] Jin, Z. (2018). *Environment Modeling-Based Requirements Engineering for Software Intensive Systems*. Elsevier Inc. https://www.sciencedirect.com/book/9780128019542/environment-modeling-based-requirements-engineering-for-software-intensive-systems#book-description

[4] Condon, D. N., Wood, D., Mõttus, R., Booth, T., Costantini, G., Greiff, S., Johnson, W., Lukaszewski, A., Revelle, W., Wright, A. G. C., Ziegler, M., & Zimmermann, J. (2019). Bottom Up Construction of a Personality Taxonomy. *European Journal of Psychological Assessment*, *36*(6). https://doi.org/19.01.2021

[5] Ontology. (2022, November 1). In *Wikipedia*. https://en.wikipedia.org/wiki/Ontology

[6] Class hierarchy. (2022, April 3). In *Wikipedia*. https://en.wikipedia.org/wiki/Class_hierarchy

[7] [Web article] Parlar, S. (2019). *Ontologies: In Detail*. https://medium.com/. https://medium.com/analytics-vidhya/ontologies-in-detail-2916f9226133

[8] Aminu, E. F., Oyefolahan, I. O., Abdullahi, M. B., & Salaudeen, M. T. (2020). A Review on Ontology Development Methodologies for Developing Ontological Knowledge Representation Systems for various Domains. *Modern Education and Computer Science Press*, 28-39. https://doi.org/ 10.5815/ijieeb.2020.02.05, 2020

[9]**https://job-descriptions.org/jobs-in-systems-analysis-and-programming.html**

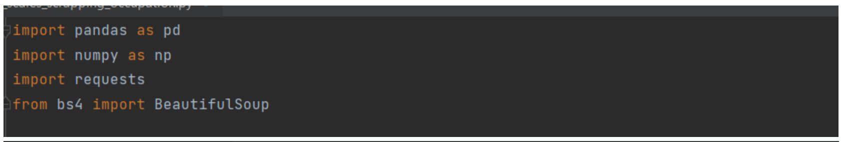
[10] **https://www.careerplanner.com/Job-Descriptions-DOT/C.cfm**

[11] United States Army enlisted rank insignia. (2022, October 30). In *Wikipedia*. https://en.wikipedia.org/wiki/United_States_Army_enlisted_rank_insignia [12] Army ranks and insignia of India. (2022, November 4). In *Wikipedia*. https://en.wikipedia.org/wiki/Army_ranks_and_insignia_of_India

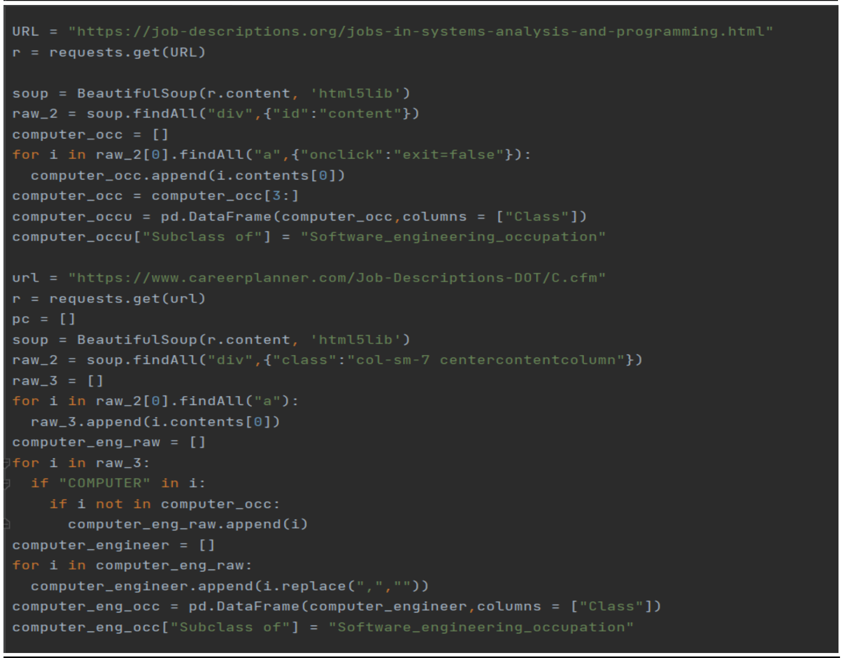
[13] Army ranks and insignia of the Russian Federation. (2022, August 16). In *Wikipedia*. https://en.wikipedia.org/wiki/Army_ranks_and_insignia_of_the_Russian_Federation

6. Appendices:

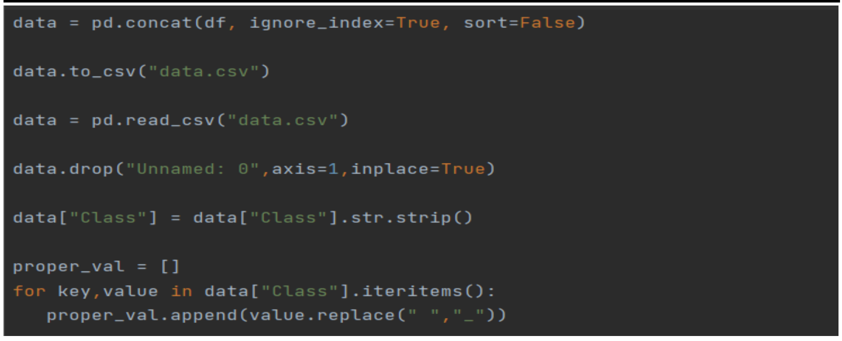
Working algorithm(Python):



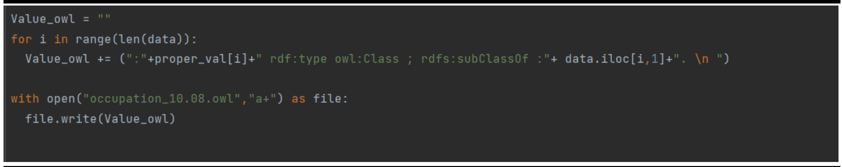
```python
import pandas as pd
import numpy as np
import requests
from bs4 import BeautifulSoup
```



```python
URL = "https://job-descriptions.org/jobs-in-systems-analysis-and-programming.html"
r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib')
raw_2 = soup.findAll("div",{"id":"content"})
computer_occ = []
for i in raw_2[0].findAll("a",{"onclick":"exit=false"}):
  computer_occ.append(i.contents[0])
computer_occ = computer_occ[3:]
computer_occu = pd.DataFrame(computer_occ,columns = ["Class"])
computer_occu["Subclass of"] = "Software_engineering_occupation"

url = "https://www.careerplanner.com/Job-Descriptions-DOT/C.cfm"
r = requests.get(url)
pc = []
soup = BeautifulSoup(r.content, 'html5lib')
raw_2 = soup.findAll("div",{"class":"col-sm-7 centercontentcolumn"})
raw_3 = []
for i in raw_2[0].findAll("a"):
  raw_3.append(i.contents[0])
computer_eng_raw = []
for i in raw_3:
  if "COMPUTER" in i:
    if i not in computer_occ:
      computer_eng_raw.append(i)
computer_engineer = []
for i in computer_eng_raw:
  computer_engineer.append(i.replace(",",""))
computer_eng_occ = pd.DataFrame(computer_engineer,columns = ["Class"])
computer_eng_occ["Subclass of"] = "Software_engineering_occupation"
```

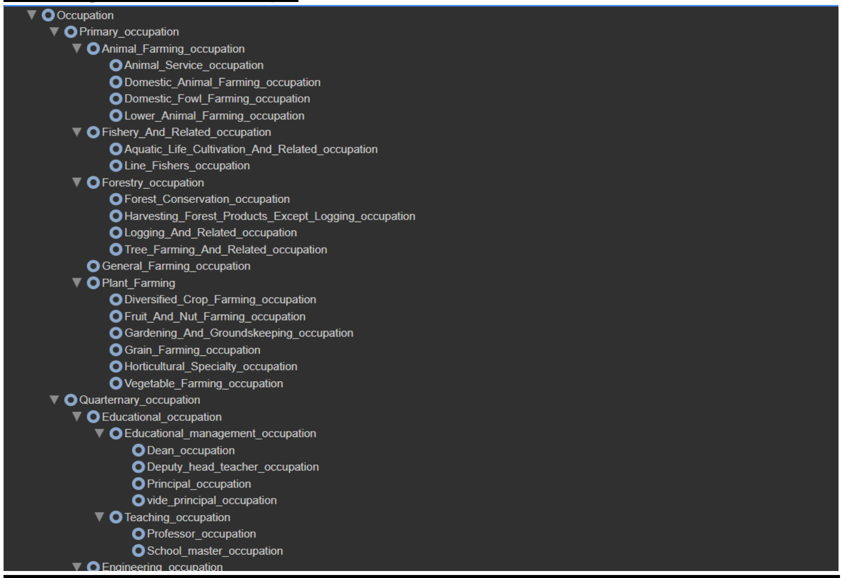


```python
data = pd.concat(df, ignore_index=True, sort=False)

data.to_csv("data.csv")

data = pd.read_csv("data.csv")

data.drop("Unnamed: 0",axis=1,inplace=True)

data["Class"] = data["Class"].str.strip()

proper_val = []
for key,value in data["Class"].iteritems():
   proper_val.append(value.replace(" ","_"))
```

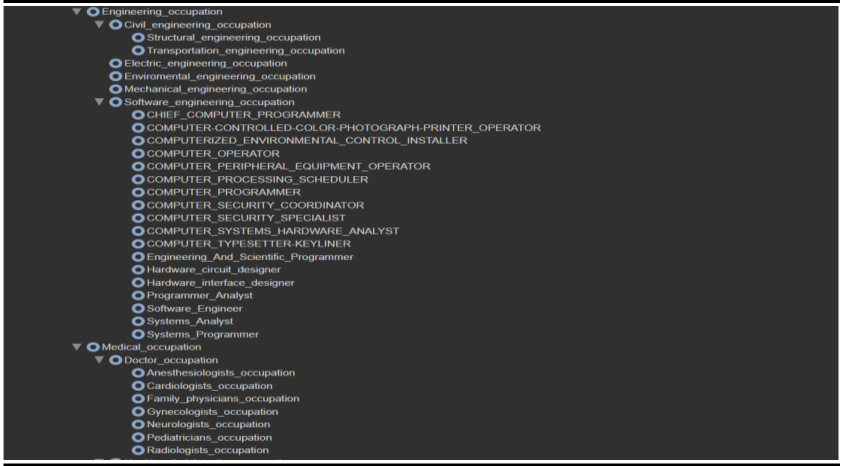


```python
Value_owl = ""
for i in range(len(data)):
  Value_owl += (":"+proper_val[i]+" rdf:type owl:Class ; rdfs:subClassOf :"+ data.iloc[i,1]+". \n ")

with open("occupation_10.08.owl","a+") as file:
  file.write(Value_owl)
```

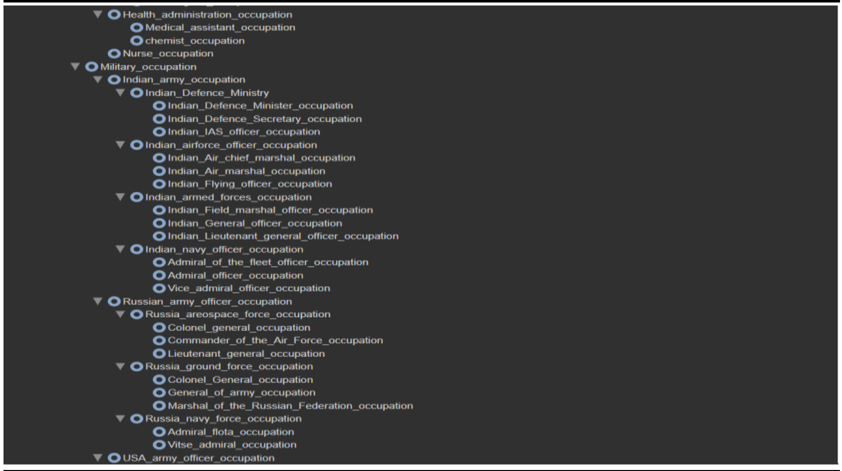
Protégé(Taxonomy)



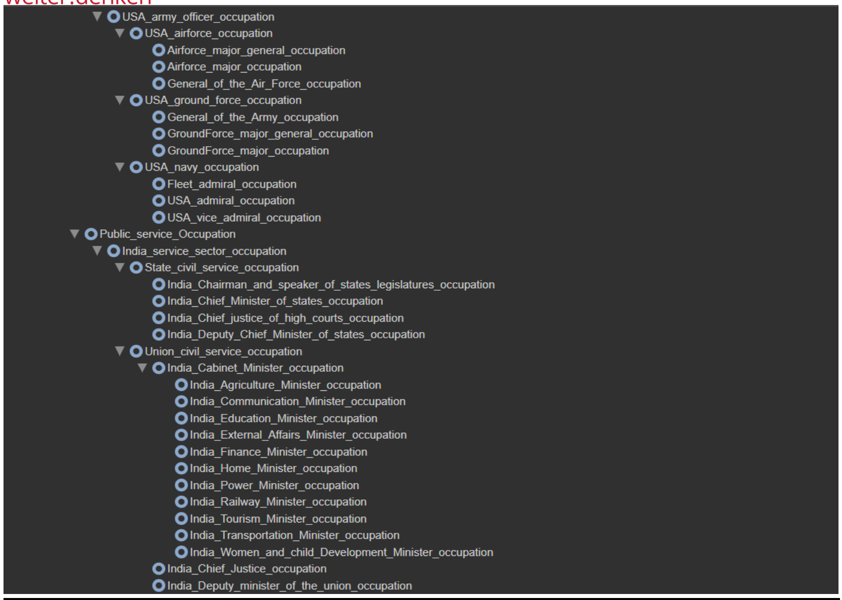
```
Occupation
  Primary_occupation
    Animal_Farming_occupation
      Animal_Service_occupation
      Domestic_Animal_Farming_occupation
      Domestic_Fowl_Farming_occupation
      Lower_Animal_Farming_occupation
    Fishery_And_Related_occupation
      Aquatic_Life_Cultivation_And_Related_occupation
      Line_Fishers_occupation
    Forestry_occupation
      Forest_Conservation_occupation
      Harvesting_Forest_Products_Except_Logging_occupation
      Logging_And_Related_occupation
      Tree_Farming_And_Related_occupation
    General_Farming_occupation
    Plant_Farming
      Diversified_Crop_Farming_occupation
      Fruit_And_Nut_Farming_occupation
      Gardening_And_Groundskeeping_occupation
      Grain_Farming_occupation
      Horticultural_Specialty_occupation
      Vegetable_Farming_occupation
  Quarternary_occupation
    Educational_occupation
      Educational_management_occupation
        Dean_occupation
        Deputy_head_teacher_occupation
        Principal_occupation
        vide_principal_occupation
      Teaching_occupation
        Professor_occupation
        School_master_occupation
    Engineering_occupation
```



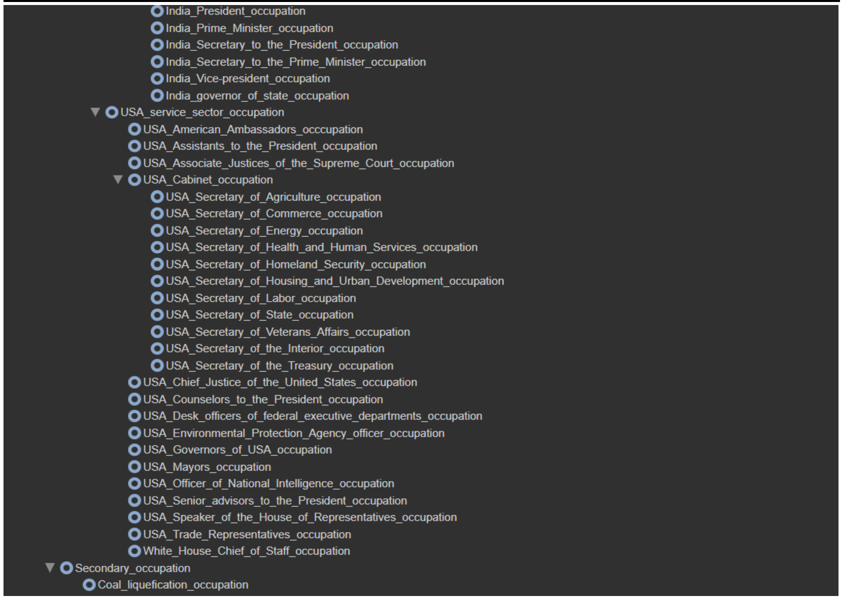
```
    Engineering_occupation
      Civil_engineering_occupation
        Structural_engineering_occupation
        Transportation_engineering_occupation
      Electric_engineering_occupation
      Enviromental_engineering_occupation
      Mechanical_engineering_occupation
      Software_engineering_occupation
        CHIEF_COMPUTER_PROGRAMMER
        COMPUTER-CONTROLLED-COLOR-PHOTOGRAPH-PRINTER_OPERATOR
        COMPUTERIZED_ENVIRONMENTAL_CONTROL_INSTALLER
        COMPUTER_OPERATOR
        COMPUTER_PERIPHERAL_EQUIPMENT_OPERATOR
        COMPUTER_PROCESSING_SCHEDULER
        COMPUTER_PROGRAMMER
        COMPUTER_SECURITY_COORDINATOR
        COMPUTER_SECURITY_SPECIALIST
        COMPUTER_SYSTEMS_HARDWARE_ANALYST
        COMPUTER_TYPESETTER-KEYLINER
        Engineering_And_Scientific_Programmer
        Hardware_circuit_designer
        Hardware_interface_designer
        Programmer_Analyst
        Software_Engineer
        Systems_Analyst
        Systems_Programmer
    Medical_occupation
      Doctor_occupation
        Anesthesiologists_occupation
        Cardiologists_occupation
        Family_physicians_occupation
        Gynecologists_occupation
        Neurologists_occupation
        Pediatricians_occupation
        Radiologists_occupation
```



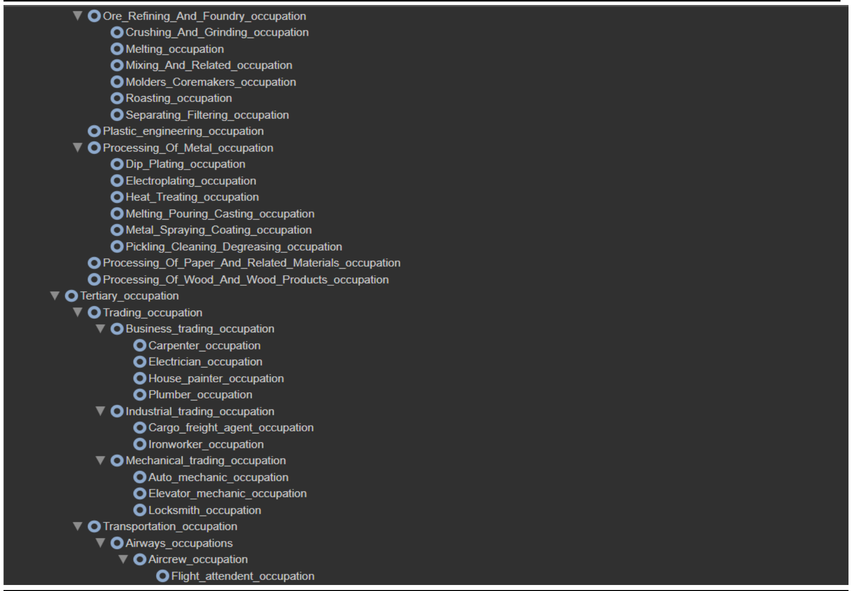
```
      Health_administration_occupation
        Medical_assistant_occupation
        chemist_occupation
      Nurse_occupation
    Military_occupation
      Indian_army_occupation
        Indian_Defence_Ministry
          Indian_Defence_Minister_occupation
          Indian_Defence_Secretary_occupation
          Indian_IAS_officer_occupation
        Indian_airforce_officer_occupation
          Indian_Air_chief_marshal_occupation
          Indian_Air_marshal_occupation
          Indian_Flying_officer_occupation
        Indian_armed_forces_occupation
          Indian_Field_marshal_officer_occupation
          Indian_General_officer_occupation
          Indian_Lieutenant_general_officer_occupation
        Indian_navy_officer_occupation
          Admiral_of_the_fleet_officer_occupation
          Admiral_officer_occupation
          Vice_admiral_officer_occupation
      Russian_army_officer_occupation
        Russia_areospace_force_occupation
          Colonel_general_occupation
          Commander_of_the_Air_Force_occupation
          Lieutenant_general_occupation
        Russia_ground_force_occupation
          Colonel_General_occupation
          General_of_army_occupation
          Marshal_of_the_Russian_Federation_occupation
        Russia_navy_force_occupation
          Admiral_flota_occupation
          Vitse_admiral_occupation
      USA_army_officer_occupation
```



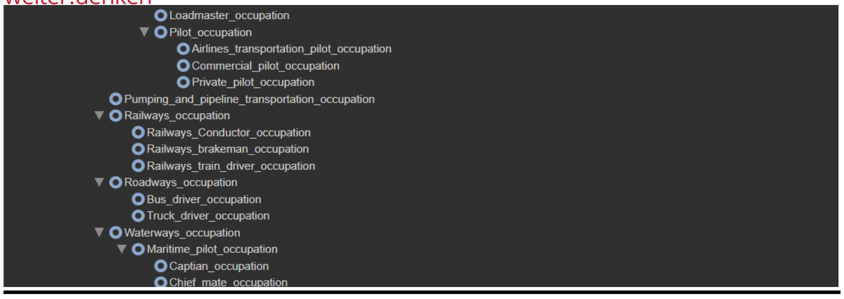
```
    USA_army_officer_occupation
      USA_airforce_occupation
        Airforce_major_general_occupation
        Airforce_major_occupation
        General_of_the_Air_Force_occupation
      USA_ground_force_occupation
        General_of_the_Army_occupation
        GroundForce_major_general_occupation
        GroundForce_major_occupation
      USA_navy_occupation
        Fleet_admiral_occupation
        USA_admiral_occupation
        USA_vice_admiral_occupation
  Public_service_Occupation
    India_service_sector_occupation
      State_civil_service_occupation
        India_Chairman_and_speaker_of_states_legislatures_occupation
        India_Chief_Minister_of_states_occupation
        India_Chief_justice_of_high_courts_occupation
        India_Deputy_Chief_Minister_of_states_occupation
      Union_civil_service_occupation
        India_Cabinet_Minister_occupation
          India_Agriculture_Minister_occupation
          India_Communication_Minister_occupation
          India_Education_Minister_occupation
          India_External_Affairs_Minister_occupation
          India_Finance_Minister_occupation
          India_Home_Minister_occupation
          India_Power_Minister_occupation
          India_Railway_Minister_occupation
          India_Tourism_Minister_occupation
          India_Transportation_Minister_occupation
          India_Women_and_child_Development_Minister_occupation
        India_Chief_Justice_occupation
        India_Deputy_minister_of_the_union_occupation
```



```
        India_President_occupation
        India_Prime_Minister_occupation
        India_Secretary_to_the_President_occupation
        India_Secretary_to_the_Prime_Minister_occupation
        India_Vice-president_occupation
        India_governor_of_state_occupation
    USA_service_sector_occupation
      USA_American_Ambassadors_occcupation
      USA_Assistants_to_the_President_occupation
      USA_Associate_Justices_of_the_Supreme_Court_occupation
      USA_Cabinet_occupation
        USA_Secretary_of_Agriculture_occupation
        USA_Secretary_of_Commerce_occupation
        USA_Secretary_of_Energy_occupation
        USA_Secretary_of_Health_and_Human_Services_occupation
        USA_Secretary_of_Homeland_Security_occupation
        USA_Secretary_of_Housing_and_Urban_Development_occupation
        USA_Secretary_of_Labor_occupation
        USA_Secretary_of_State_occupation
        USA_Secretary_of_Veterans_Affairs_occupation
        USA_Secretary_of_the_Interior_occupation
        USA_Secretary_of_the_Treasury_occupation
      USA_Chief_Justice_of_the_United_States_occupation
      USA_Counselors_to_the_President_occupation
      USA_Desk_officers_of_federal_executive_departments_occupation
      USA_Environmental_Protection_Agency_officer_occupation
      USA_Governors_of_USA_occupation
      USA_Mayors_occupation
      USA_Officer_of_National_Intelligence_occupation
      USA_Senior_advisors_to_the_President_occupation
      USA_Speaker_of_the_House_of_Representatives_occupation
      USA_Trade_Representatives_occupation
      White_House_Chief_of_Staff_occupation
Secondary_occupation
  Coal_liquefication_occupation
```



```
  Ore_Refining_And_Foundry_occupation
    Crushing_And_Grinding_occupation
    Melting_occupation
    Mixing_And_Related_occupation
    Molders_Coremakers_occupation
    Roasting_occupation
    Separating_Filtering_occupation
  Plastic_engineering_occupation
  Processing_Of_Metal_occupation
    Dip_Plating_occupation
    Electroplating_occupation
    Heat_Treating_occupation
    Melting_Pouring_Casting_occupation
    Metal_Spraying_Coating_occupation
    Pickling_Cleaning_Degreasing_occupation
  Processing_Of_Paper_And_Related_Materials_occupation
  Processing_Of_Wood_And_Wood_Products_occupation
Tertiary_occupation
  Trading_occupation
    Business_trading_occupation
      Carpenter_occupation
      Electrician_occupation
      House_painter_occupation
      Plumber_occupation
    Industrial_trading_occupation
      Cargo_freight_agent_occupation
      Ironworker_occupation
    Mechanical_trading_occupation
      Auto_mechanic_occupation
      Elevator_mechanic_occupation
      Locksmith_occupation
  Transportation_occupation
    Airways_occupations
      Aircrew_occupation
        Flight_attendent_occupation
```



```
        Loadmaster_occupation
      ▼ Pilot_occupation
          Airlines_transportation_pilot_occupation
          Commercial_pilot_occupation
          Private_pilot_occupation
  Pumping_and_pipeline_transportation_occupation
▼ Railways_occupation
      Railways_Conductor_occupation
      Railways_brakeman_occupation
      Railways_train_driver_occupation
▼ Roadways_occupation
      Bus_driver_occupation
      Truck_driver_occupation
▼ Waterways_occupation
    ▼ Maritime_pilot_occupation
          Captian_occupation
          Chief_mate_occupation
```

Evaluation Table

Axioms	Logical Test	Result
Subclass axiom for the class “Aquatic_Life_Cultivation_And_Related_occupation”	Test 1 "Every Aquatic_Life_Cultivation_And_Related_occupation is a Fishery_And_Related_occupation"	+
Disjoint class axiom of the class "Aquatic_Life_Cultivation_And_Related_occupation"	Test 2 "No Aquatic_Life_Cultivation_And_Related_occupation is Line_Fishers_occupation "	-
Subclass axiom for the class “Line_Fishers_occupation”	Test 1 "Every Line_Fishers_occupation is a Fishery_And_Related_occupation"	+
Disjoint class axiom of the class "Line_Fishers_occupation"	Test 2 "No Line_Fishers_occupation is Aquatic_Life_Cultivation_And_Related_occupation "	+
Subclass axiom for the class “Animal_Service_occupation”	Test 1 "Every Animal_Service_occupation is a Animal_Farming_occupation"	-
Disjoint class axiom of the class "Animal_Service_occupation"	Test 2 "No Animal_Service_occupation is Domestic_Animal_Farming_occupation "	+
Disjoint class axiom of the class "Animal_Service_occupation"	Test 2 "No Animal_Service_occupation is Domestic_Fowl_Farming_occupation "	+
Disjoint class axiom of the class "Animal_Service_occupation"	Test 2 "No Animal_Service_occupation is Lower_Animal_Farming_occupation "	+
Subclass axiom for the class “Domestic_Animal_Farming_occupation”	Test 1 "Every Domestic_Animal_Farming_occupation is a Animal_Farming_occupation"	+
Disjoint class axiom of the class "Domestic_Animal_Farming_occupation"	Test 2 "No Domestic_Animal_Farming_occupation is Animal_Service_occupation "	+
Disjoint class axiom of the class "Domestic_Animal_Farming_occupation"	Test 2 "No Domestic_Animal_Farming_occupation is Domestic_Fowl_Farming_occupation "	-
Disjoint class axiom of the class "Domestic_Animal_Farming_occupation"	Test 2 "No Domestic_Animal_Farming_occupation is Lower_Animal_Farming_occupation "	+
Subclass axiom for the class “Domestic_Fowl_Farming_occupation”	Test 1 "Every Domestic_Fowl_Farming_occupation is a Animal_Farming_occupation"	+
Disjoint class axiom of the class "Domestic_Fowl_Farming_occupation"	Test 2 "No Domestic_Fowl_Farming_occupation is Domestic_Animal_Farming_occupation "	-

Disjoint class axiom of the class "Domestic_Fowl_Farming_occupation"	Test 2 "No Domestic_Fowl_Farming_occupation is Animal_Service_occupation "	+
Disjoint class axiom of the class "Domestic_Fowl_Farming_occupation"	Test 2 "No Domestic_Fowl_Farming_occupation is Lower_Animal_Farming_occupation "	+
Subclass axiom for the class "Lower_Animal_Farming_occupation"	Test 1 "Every Lower_Animal_Farming_occupation is a Animal_Farming_occupation"	+
Disjoint class axiom of the class "Lower_Animal_Farming_occupation"	Test 2 "No Lower_Animal_Farming_occupation is Domestic_Animal_Farming_occupation "	+
Disjoint class axiom of the class "Lower_Animal_Farming_occupation"	Test 2 "No Lower_Animal_Farming_occupation is Animal_Service_occupation "	+
Disjoint class axiom of the class "Lower_Animal_Farming_occupation"	Test 2 "No Lower_Animal_Farming_occupation is Domestic_Fowl_Farming_occupation "	+
Subclass axiom for the class "Professor_occupation"	Test 1 "Every Professor_occupation is a Teaching_occupation"	+
Disjoint class axiom of the class "Professor_occupation"	Test 2 "No Professor_occupation is School_master_occupation "	+
Subclass axiom for the class "School_master_occupation"	Test 1 "Every School_master_occupation is a Teaching_occupation"	+
Disjoint class axiom of the class "School_master_occupation	Test 2 "No School_master_occupation is Professor_occupation "	+
Subclass axiom for the class "Structural_engineering_occupation"	Test 1 "Every Structural_engineering_occupation is a Civil_engineering_occupation"	+
Disjoint class axiom of the class "Structural_engineering_occupation"	Test 2 "No Structural_engineering_occupation is Transportation_engineering_occupation "	+
Subclass axiom for the class "Transportation_engineering_occupation"	Test 1 "Every Transportation_engineering_occupation is a Civil_engineering_occupation"	+
Disjoint class axiom of the class "Transportation_engineering_occupation"	Test 2 "No Transportation_engineering_occupation is Structural_engineering_occupa	+
Subclass axiom for the class "Captian_occupation"	Test 1 "Every Captian_occupation is a Maritime_pilot_occupation"	+
Disjoint class axiom of the class "Captian_occupation"	Test 2 "No Captian_occupation is Chief_mate_occupation "	+
Subclass axiom for the class "Chief_mate_occupation"	Test 1 "Every Chief_mate_occupation is a Maritime_pilot_occupation"	+
Disjoint class axiom of the class "Chief_mate_occupation"	Test 2 "No Chief_mate_occupation is Captian_occupation "	+
Subclass axiom for the class "Bus_driver_occupation"	Test 1 "Every Bus_driver_occupation is a Roadways_occupation"	+
Disjoint class axiom of the class "Bus_driver_occupation"	Test 2 "No Bus_driver_occupation is Truck_driver_occupation "	+

Subclass axiom for the class “Truck_driver_occupation”	Test 1 "Every Truck_driver_occupation is a Roadways_occupation"	+
Disjoint class axiom of the class "Truck_driver_occupation"	Test 2 "No Truck_driver_occupation is Bus_driver_occupation "	+
Subclass axiom for the class “Railways_train_driver_occupation”	Test 1 "Every Railways_train_driver_occupation is a Railways_occupation"	+
Disjoint class axiom of the class "Railways_train_driver_occupation"	Test 2 "No Railways_train_driver_occupation is Railways_brakeman_occupation "	+
Disjoint class axiom of the class "Railways_train_driver_occupation"	Test 2 "No Railways_train_driver_occupation is Railways_Conductor_occupation "	+
Subclass axiom for the class “Railways_brakeman_occupation”	Test 1 "Every Railways_brakeman_occupation is a Railways_occupation"	+
Disjoint class axiom of the class "Railways_brakeman_occupation"	Test 2 "No Railways_brakeman_occupation is Railways_train_driver_occupation "	+
Disjoint class axiom of the class "Railways_brakeman_occupation"	Test 2 "No Railways_brakeman_occupation is Railways_Conductor_occupation "	+
Subclass axiom for the class “Railways_Conductor_occupation”	Test 1 "Every Railways_Conductor_occupation is a Railways_occupation"	+
Disjoint class axiom of the class "Railways_Conductor_occupation"	Test 2 "No Railways_Conductor_occupation is Railways_train_driver_occupation "	+
Disjoint class axiom of the class "Railways_Conductor_occupation"	Test 2 "No Railways_Conductor_occupation is Railways_brakeman_occupation "	+
Subclass axiom for the class “Private_pilot_occupation”	Test 1 "Every Private_pilot_occupation is a Pilot_occupation"	+
Disjoint class axiom of the class "Private_pilot_occupation"	Test 2 "No Private_pilot_occupation is Commercial_pilot_occupation "	+
Disjoint class axiom of the class "Private_pilot_occupation"	Test 2 "No Private_pilot_occupation is Airlines_transportation_pilot_occupation "	+
Subclass axiom for the class “Commercial_pilot_occupation”	Test 1 "Every Commercial_pilot_occupation is a Pilot_occupation"	+
Disjoint class axiom of the class "Commercial_pilot_occupation"	Test 2 "No Commercial_pilot_occupation is Airlines_transportation_pilot_occupation "	+
Disjoint class axiom of the class "Commercial_pilot_occupation"	Test 2 "No Commercial_pilot_occupation is Private_pilot_occupation "	+
Subclass axiom for the class “Airlines_transportation_pilot_occupation”	Test 1 "Every Airlines_transportation_pilot_occupation is a Pilot_occupation"	+
Disjoint class axiom of the class "Airlines_transportation_pilot_occupation"	Test 2 "No Airlines_transportation_pilot_occupation is Private_pilot_occupation "	+
Disjoint class axiom of the class "Airlines_transportation_pilot_occupation"	Test 2 "No Airlines_transportation_pilot_occupation is Commercial_pilot_occupation "	+
Subclass axiom for the class “Auto_mechanic_occupation”	Test 1 "Every Auto_mechanic_occupation is a Mechanical_trading_occupation"	+

Disjoint class axiom of the class "Auto_mechanic_occupation"	Test 2 "No Auto_mechanic_occupation is Elevator_mechanic_occupation "	+
Disjoint class axiom of the class "Auto_mechanic_occupation"	Test 2 "No Auto_mechanic_occupation is Locksmith_occupation "	+
Subclass axiom for the class "Elevator_mechanic_occupation"	Test 1 "Every Elevator_mechanic_occupation is a Mechanical_trading_occupation"	+
Disjoint class axiom of the class "Elevator_mechanic_occupation"	Test 2 "No Elevator_mechanic_occupation is Locksmith_occupation "	+
Disjoint class axiom of the class "Elevator_mechanic_occupation"	Test 2 "No Elevator_mechanic_occupation is Auto_mechanic_occupation "	+
Subclass axiom for the class "Locksmith_occupation"	Test 1 "Every Locksmith_occupation is a Mechanical_trading_occupation"	+
Disjoint class axiom of the class "Locksmith_occupation"	Test 2 "No Locksmith_occupation is Auto_mechanic_occupation "	+
Disjoint class axiom of the class "Airlines_transportation_pilot_occupation"	Test 2 "No Locksmith_occupation is Elevator_mechanic_occupation "	+
Subclass axiom for the class "USA_admiral_occupation"	Test 1 "Every USA_admiral_occupation is a USA_navy_occupation"	+
Disjoint class axiom of the class "USA_admiral_occupation"	Test 2 "No USA_admiral_occupation is USA_vice_admiral_occupation "	+
Disjoint class axiom of the class "USA_admiral_occupation"	Test 2 "No USA_admiral_occupation is Fleet_admiral_occupation "	+
Subclass axiom for the class "Fleet_admiral_occupation"	Test 1 "Every Fleet_admiral_occupation is a USA_navy_occupation"	+
Disjoint class axiom of the class "Fleet_admiral_occupation"	Test 2 "No Fleet_admiral_occupation is USA_admiral_occupation "	+
Disjoint class axiom of the class "Fleet_admiral_occupation"	Test 2 "No Fleet_admiral_occupation is USA_vice_admiral_occupation "	+
Subclass axiom for the class "USA_vice_admiral_occupation"	Test 1 "Every USA_vice_admiral_occupation is a USA_navy_occupation"	+
Disjoint class axiom of the class "USA_vice_admiral_occupation"	Test 2 "No USA_vice_admiral_occupation is Fleet_admiral_occupation "	+
Disjoint class axiom of the class "USA_vice_admiral_occupation"	Test 2 "No USA_vice_admiral_occupation is USA_admiral_occupation "	+
Subclass axiom for the class "Indian_Air_chief_marshal_occupation"	Test 1 "Every Indian_Air_chief_marshal_occupation is a Indian_airforce_officer_occupation"	+
Disjoint class axiom of the class "Indian_Air_chief_marshal_occupation"	Test 2 "No Indian_Air_chief_marshal_occupation is Indian_Air_marshal_occupation "	+
Disjoint class axiom of the class "Indian_Air_chief_marshal_occupation"	Test 2 "No Indian_Air_chief_marshal_occupation is Indian_Flying_officer_occupation "	+
Subclass axiom for the class "Indian_Air_marshal_occupation"	Test 1 "Every Indian_Air_marshal_occupation is a Indian_airforce_officer_occupation"	+
Disjoint class axiom of the class "Indian_Air_marshal_occupation"	Test 2 "No Indian_Air_marshal_occupation is Indian_Air_marshal_occupation "	+

Disjoint class axiom of the class "Indian_Air_marshal_occupation"	Test 2 "No Indian_Air_marshal_occupation is Indian_Flying_officer_occupation "	+
Subclass axiom for the class “Indian_Flying_officer_occupation”	Test 1 "Every Indian_Flying_officer_occupation is a Indian_airforce_officer_occupation"	+
Disjoint class axiom of the class "Indian_Flying_officer_occupation"	Test 2 "No Indian_Flying_officer_occupation is Indian_Air_chief_marshal_occupation "	+
Disjoint class axiom of the class "Indian_Flying_officer_occupation"	Test 2 "No Indian_Flying_officer_occupation is Indian_Air_marshal_occupation "	+
Subclass axiom for the class “Dean_occupation”	Test 1 "Every Dean_occupation is a Educational_management_occupation"	+
Disjoint class axiom of the class "Dean_occupation"	Test 2 "No Dean_occupation is Deputy_head_teacher_occupation "	+
Disjoint class axiom of the class "Dean_occupation"	Test 2 "No Dean_occupation is Principal_occupation "	+
Disjoint class axiom of the class "Dean_occupation"	Test 2 "No Dean_occupation is vice_principal_occupation "	+
Subclass axiom for the class “Deputy_head_teacher_occupation”	Test 1 "Every Deputy_head_teacher_occupation is a Educational_management_occupation"	+
Disjoint class axiom of the class "Deputy_head_teacher_occupation"	Test 2 "No Deputy_head_teacher_occupation is Dean_occupation "	+
Disjoint class axiom of the class "Deputy_head_teacher_occupation"	Test 2 "No Deputy_head_teacher_occupation is Principal_occupation "	+
Disjoint class axiom of the class "Deputy_head_teacher_occupation"	Test 2 "No Deputy_head_teacher_occupation is vice_principal_occupation "	+
Subclass axiom for the class “Principal_occupation”	Test 1 "Every Principal_occupation is a Educational_management_occupation"	+
Disjoint class axiom of the class "Principal_occupation"	Test 2 "No Principal_occupation is Dean_occupation "	+
Disjoint class axiom of the class "Principal_occupation"	Test 2 "No Principal_occupation is Deputy_head_teacher_occupation "	+
Disjoint class axiom of the class "Principal_occupation"	Test 2 "No Principal_occupation is vice_principal_occupation "	+
Subclass axiom for the class “vice_principal_occupation”	Test 1 "Every vice_principal_occupation is a Educational_management_occupation"	+
Disjoint class axiom of the class "vice_principal_occupation"	Test 2 "No vice_principal_occupation is Dean_occupation "	+
Disjoint class axiom of the class "vice_principal_occupation"	Test 2 "No vice_principal_occupation is Deputy_head_teacher_occupation "	+
Disjoint class axiom of the class "vice_principal_occupation"	Test 2 "No vice_principal_occupation is Principal_occupation "	+
Subclass axiom for the class “India_Chairman_and_speaker_of_states_legislatures_occupation”	Test 1 "Every India_Chairman_and_speaker_of_states_le	+

	gislatures_occupation is a State_civil_service_occupation"	
Disjoint class axiom of the class "India_Chairman_and_speaker_of_states_legislatures_occupation"	Test 2 "No India_Chairman_and_speaker_of_states_legislatures_occupation is India_Chief_Minister_of_states_occupation "	+
Disjoint class axiom of the class "India_Chairman_and_speaker_of_states_legislatures_occupation"	Test 2 "No India_Chairman_and_speaker_of_states_legislatures_occupation is India_Chief_justice_of_high_courts_occupation "	+
Disjoint class axiom of the class "India_Chairman_and_speaker_of_states_legislatures_occupation"	Test 2 "No India_Chairman_and_speaker_of_states_legislatures_occupation is India_Deputy_Chief_Minister_of_states_occupation "	+
Subclass axiom for the class "India_Chief_Minister_of_states_occupation"	Test 1 "Every India_Chief_Minister_of_states_occupation is a State_civil_service_occupation"	+
Disjoint class axiom of the class "India_Chief_Minister_of_states_occupation"	Test 2 "No India_Chief_Minister_of_states_occupation is India_Chairman_and_speaker_of_states_legislatures_occupation "	+
Disjoint class axiom of the class "India_Chief_Minister_of_states_occupation"	Test 2 "No India_Chief_Minister_of_states_occupation is India_Chief_justice_of_high_courts_occupation "	+
Disjoint class axiom of the class "India_Chief_Minister_of_states_occupation"	Test 2 "No India_Chief_Minister_of_states_occupation is India_Deputy_Chief_Minister_of_states_occupation "	+
Subclass axiom for the class "India_Chief_justice_of_high_courts_occupation"	Test 1 "Every India_Chief_justice_of_high_courts_occupation is a State_civil_service_occupation"	+
Disjoint class axiom of the class "India_Chief_justice_of_high_courts_occupation"	Test 2 "No India_Chief_justice_of_high_courts_occupation is India_Chairman_and_speaker_of_states_legislatures_occupation "	+
Disjoint class axiom of the class "India_Chief_justice_of_high_courts_occupation"	Test 2 "No India_Chief_justice_of_high_courts_occupation is India_Chief_Minister_of_states_occupation "	+

Disjoint class axiom of the class "India_Chief_justice_of_high_courts_occupation"	Test 2 "No India_Chief_justice_of_high_courts_occupation is India_Deputy_Chief_Minister_of_states_occupation "	+
Subclass axiom for the class “India_Deputy_Chief_Minister_of_states_occupation”	Test 1 "Every India_Deputy_Chief_Minister_of_states_occupation is a State_civil_service_occupation"	+
Disjoint class axiom of the class "India_Deputy_Chief_Minister_of_states_occupation"	Test 2 "No India_Deputy_Chief_Minister_of_states_occupation is India_Chairman_and_speaker_of_states_legislatures_occupation "	+
Disjoint class axiom of the class "India_Deputy_Chief_Minister_of_states_occupation"	Test 2 "No India_Deputy_Chief_Minister_of_states_occupation is India_Chief_Minister_of_states_occupation "	+
Disjoint class axiom of the class "India_Deputy_Chief_Minister_of_states_occupation"	Test 2 "No India_Deputy_Chief_Minister_of_states_occupation is India_Chief_justice_of_high_courts_occupation "	+
Subclass axiom for the class “Carpenter_occupation”	Test 1 "Every Carpenter_occupation is a Business_trading_occupation"	+
Disjoint class axiom of the class "Carpenter_occupation"	Test 2 "No Carpenter_occupation is Electrician_occupation "	+
Disjoint class axiom of the class "Carpenter_occupation"	Test 2 "No Carpenter_occupation is House_painter_occupation "	+
Disjoint class axiom of the class "Carpenter_occupation"	Test 2 "No Carpenter_occupation is Plumber_occupation "	+
Subclass axiom for the class “Electrician_occupation”	Test 1 "Every Electrician_occupation is a Business_trading_occupation"	+
Disjoint class axiom of the class "Electrician_occupation"	Test 2 "No Electrician_occupation is Carpenter_occupation "	+
Disjoint class axiom of the class "Electrician_occupation"	Test 2 "No Electrician_occupation is House_painter_occupation "	+
Disjoint class axiom of the class "Electrician_occupation"	Test 2 "No Electrician_occupation is Plumber_occupation "	+
Subclass axiom for the class “House_painter_occupation”	Test 1 "Every House_painter_occupation is a Business_trading_occupation"	+
Disjoint class axiom of the class "House_painter_occupation"	Test 2 "No House_painter_occupation is Carpenter_occupation "	+
Disjoint class axiom of the class "House_painter_occupation"	Test 2 "No House_painter_occupation is Electrician_occupation "	+
Disjoint class axiom of the class "House_painter_occupation"	Test 2 "No House_painter_occupation is Plumber_occupation "	+
Subclass axiom for the class “Plumber_occupation”	Test 1 "Every Plumber_occupation is a Business_trading_occupation"	+

Disjoint class axiom of the class "Plumber_occupation"	Test 2 "No Plumber_occupation is Carpenter_occupation "	+
Disjoint class axiom of the class "Plumber_occupation"	Test 2 "No Plumber_occupation is Electrician_occupation "	+
Disjoint class axiom of the class "Plumber_occupation"	Test 2 "No Plumber_occupation is House_painter_occupation "	+
Subclass axiom for the class “Primary_occupation”	Test 1 "Every Primary_occupation is a Occupation"	+
Disjoint class axiom of the class "Primary_occupation"	Test 2 "No Primary_occupation is Quarternary_occupation "	+
Disjoint class axiom of the class "Primary_occupation"	Test 2 "No Primary_occupation is Secondary_occupation "	+
Disjoint class axiom of the class "Primary_occupation"	Test 2 "No Primary_occupation is Tertiary_occupation "	+
Subclass axiom for the class “Quarternary_occupation”	Test 1 "Every Quarternary_occupation is a Occupation"	+
Disjoint class axiom of the class "Quarternary_occupation"	Test 2 "No Quarternary_occupation is Primary_occupation "	+
Disjoint class axiom of the class "Quarternary_occupation"	Test 2 "No Quarternary_occupation is Secondary_occupation "	+
Disjoint class axiom of the class "Quarternary_occupation"	Test 2 "No Quarternary_occupation is Tertiary_occupation "	+
Subclass axiom for the class “Secondary_occupation”	Test 1 "Every Secondary_occupation is a Occupation"	+
Disjoint class axiom of the class "Secondary_occupation"	Test 2 "No Secondary_occupation is Primary_occupation "	+
Disjoint class axiom of the class "Secondary_occupation"	Test 2 "No Secondary_occupation is Quarternary_occupation "	+
Disjoint class axiom of the class "Secondary_occupation"	Test 2 "No Secondary_occupation is Tertiary_occupation "	+
Subclass axiom for the class “Tertiary_occupation”	Test 1 "Every Tertiary_occupation is a Occupation"	+
Disjoint class axiom of the class "Tertiary_occupation"	Test 2 "No Tertiary_occupation is Primary_occupation "	+
Disjoint class axiom of the class "Tertiary_occupation"	Test 2 "No Tertiary_occupation is Quarternary_occupation "	+
Disjoint class axiom of the class "Tertiary_occupation"	Test 2 "No Tertiary_occupation is Secondary_occupation "	+
Subclass axiom for the class “Anesthesiologists_occupation”	Test 1 "Every Anesthesiologists_occupation is a Doctor_occupation"	+
Disjoint class axiom of the class "Anesthesiologists_occupation"	Test 2 "No Anesthesiologists_occupation is Cardiologists_occupation "	+
Disjoint class axiom of the class "Anesthesiologists_occupation"	Test 2 "No Anesthesiologists_occupation is Family_physicians_occupation "	+
Disjoint class axiom of the class "Anesthesiologists_occupation"	Test 2 "No Anesthesiologists_occupation is Gynecologists_occupation "	+
Disjoint class axiom of the class "Anesthesiologists_occupation"	Test 2 "No Anesthesiologists_occupation is Neurologists_occupation "	+

Disjoint class axiom of the class "Anesthesiologists_occupation"	Test 2 "No Anesthesiologists_occupation is Pediatricians_occupation "	+
Disjoint class axiom of the class "Anesthesiologists_occupation"	Test 2 "No Anesthesiologists_occupation is Radiologists_occupation "	+
Subclass axiom for the class "Cardiologists_occupation"	Test 1 "Every Cardiologists_occupation is a Doctor_occupation"	+
Disjoint class axiom of the class "Cardiologists_occupation"	Test 2 "No Cardiologists_occupation is Anesthesiologists_occupation "	+
Disjoint class axiom of the class "Cardiologists_occupation"	Test 2 "No Cardiologists_occupation is Family_physicians_occupation "	+
Disjoint class axiom of the class "Cardiologists_occupation"	Test 2 "No Cardiologists_occupation is Gynecologists_occupation "	+
Disjoint class axiom of the class "Cardiologists_occupation"	Test 2 "No Cardiologists_occupation is Neurologists_occupation "	+
Disjoint class axiom of the class "Cardiologists_occupation"	Test 2 "No Cardiologists_occupation is Pediatricians_occupation "	+
Disjoint class axiom of the class "Cardiologists_occupation"	Test 2 "No Cardiologists_occupation is Radiologists_occupation "	+
Subclass axiom for the class "Family_physicians_occupation"	Test 1 "Every Family_physicians_occupation is a Doctor_occupation"	+
Disjoint class axiom of the class "Family_physicians_occupation"	Test 2 "No Family_physicians_occupation is Anesthesiologists_occupation "	+
Disjoint class axiom of the class "Family_physicians_occupation"	Test 2 "No Family_physicians_occupation is Cardiologists_occupation "	+
Disjoint class axiom of the class "Family_physicians_occupation"	Test 2 "No Family_physicians_occupation is Gynecologists_occupation "	+
Disjoint class axiom of the class "Family_physicians_occupation"	Test 2 "No Family_physicians_occupation is Neurologists_occupation "	+
Disjoint class axiom of the class "Family_physicians_occupation"	Test 2 "No Family_physicians_occupation is Pediatricians_occupation "	+
Disjoint class axiom of the class "Family_physicians_occupation"	Test 2 "No Family_physicians_occupation is Radiologists_occupation "	+
Subclass axiom for the class "Gynecologists_occupation"	Test 1 "Every Gynecologists_occupation is a Doctor_occupation"	+
Disjoint class axiom of the class "Gynecologists_occupation"	Test 2 "No Gynecologists_occupation is Anesthesiologists_occupation "	+
Disjoint class axiom of the class "Gynecologists_occupation"	Test 2 "No Gynecologists_occupation is Cardiologists_occupation "	+
Disjoint class axiom of the class "Gynecologists_occupation"	Test 2 "No Gynecologists_occupation is Family_physicians_occupation "	+
Disjoint class axiom of the class "Gynecologists_occupation"	Test 2 "No Gynecologists_occupation is Neurologists_occupation "	+
Disjoint class axiom of the class "Gynecologists_occupation"	Test 2 "No Gynecologists_occupation is Pediatricians_occupation "	+

Disjoint class axiom of the class "Gynecologists_occupation"	Test 2 "No Gynecologists_occupation is Radiologists_occupation "	+
Subclass axiom for the class “Neurologists_occupation”	Test 1 "Every Neurologists_occupation is a Doctor_occupation"	+
Disjoint class axiom of the class "Neurologists_occupation"	Test 2 "No Neurologists_occupation is Anesthesiologists_occupation "	+
Disjoint class axiom of the class "Neurologists_occupation"	Test 2 "No Neurologists_occupation is Cardiologists_occupation "	+
Disjoint class axiom of the class "Neurologists_occupation"	Test 2 "No Neurologists_occupation is Family_physicians_occupation "	+
Disjoint class axiom of the class "Neurologists_occupation"	Test 2 "No Neurologists_occupation is Gynecologists_occupation "	+
Disjoint class axiom of the class "Neurologists_occupation"	Test 2 "No Neurologists_occupation is Pediatricians_occupation "	+
Disjoint class axiom of the class "Neurologists_occupation"	Test 2 "No Neurologists_occupation is Radiologists_occupation "	+
Subclass axiom for the class “Pediatricians_occupation”	Test 1 "Every Pediatricians_occupation is a Doctor_occupation"	+
Disjoint class axiom of the class "Pediatricians_occupation"	Test 2 "No Pediatricians_occupation is Anesthesiologists_occupation "	+
Disjoint class axiom of the class "Pediatricians_occupation"	Test 2 "No Pediatricians_occupation is Cardiologists_occupation "	+
Disjoint class axiom of the class "Pediatricians_occupation"	Test 2 "No Pediatricians_occupation is Family_physicians_occupation "	+
Disjoint class axiom of the class "Pediatricians_occupation"	Test 2 "No Pediatricians_occupation is Gynecologists_occupation "	+
Disjoint class axiom of the class "Pediatricians_occupation"	Test 2 "No Pediatricians_occupation is Neurologists_occupation "	+
Disjoint class axiom of the class "Pediatricians_occupation"	Test 2 "No Pediatricians_occupation is Radiologists_occupation "	+
Subclass axiom of the class "Radiologists_occupation"	Test 1 "Every Tertiary_occupation is a Doctor_occupation"	+
Disjoint class axiom of the class "Radiologists_occupation"	Test 2 "No Radiologists_occupation is Anesthesiologists_occupation "	+
Disjoint class axiom of the class "Radiologists_occupation"	Test 2 "No Radiologists_occupation is Cardiologists_occupation "	+
Disjoint class axiom of the class "Radiologists_occupation"	Test 2 "No Radiologists_occupation is Family_physicians_occupation "	+
Disjoint class axiom of the class "Radiologists_occupation"	Test 2 "No Radiologists_occupation is Gynecologists_occupation "	+
Disjoint class axiom of the class "Radiologists_occupation"	Test 2 "No Radiologists_occupation is Neurologists_occupation "	+
Disjoint class axiom of the class "Radiologists_occupation"	Test 2 "No Radiologists_occupation is Pediatricians_occupation "	+

Subclass axiom for the class “Indian_Field_marshal_officer_occupation”	Test 1 "Every Indian_Field_marshal_officer_occupation is a Indian_armed_forces_occupation"	+
Disjoint class axiom of the class "Indian_Field_marshal_officer_occupation"	Test 2 "No Indian_Field_marshal_officer_occupation is Indian_General_officer_occupation "	+
Disjoint class axiom of the class "Indian_Field_marshal_officer_occupation"	Test 2 "No Indian_Field_marshal_officer_occupation is Indian_Lieutenant_general_officer_occupation "	+
Subclass axiom for the class “Indian_General_officer_occupation”	Test 1 "Every Indian_General_officer_occupation is a Indian_armed_forces_occupation"	+
Disjoint class axiom of the class "Indian_General_officer_occupation"	Test 2 "No Indian_General_officer_occupation is Indian_Lieutenant_general_officer_occupation "	+
Disjoint class axiom of the class "Indian_General_officer_occupation"	Test 2 "No Indian_General_officer_occupation is Indian_Field_marshal_officer_occupation "	+
Subclass axiom for the class “Indian_Lieutenant_general_officer_occupation”	Test 1 "Every Indian_Lieutenant_general_officer_occupation is a Indian_armed_forces_occupation"	+
Disjoint class axiom of the class "Indian_Lieutenant_general_officer_occupation"	Test 2 "No Indian_Lieutenant_general_officer_occupation is Indian_Field_marshal_officer_occupation "	+
Disjoint class axiom of the class "Indian_Lieutenant_general_officer_occupation"	Test 2 "No Indian_Lieutenant_general_officer_occupation is Indian_General_officer_occupation "	+
Subclass axiom for the class “Coal_liquefication_occupation”	Test 1 "Every Coal_liquefication_occupation is a Secondary_occupation"	+
Disjoint class axiom of the class "Coal_liquefication_occupation"	Test 2 "No Coal_liquefication_occupation is Ore_Refining_And_Foundry_occupation "	+
Disjoint class axiom of the class "Coal_liquefication_occupation"	Test 2 "No Coal_liquefication_occupation is Plastic_engineering_occupation "	+
Disjoint class axiom of the class "Coal_liquefication_occupation"	Test 2 "No Coal_liquefication_occupation is Processing_Of_Paper_And_Related_Materials_occupation "	+
Disjoint class axiom of the class "Coal_liquefication_occupation"	Test 2 "No Coal_liquefication_occupation is Neurologists_occupation "	+
Disjoint class axiom of the class "Coal_liquefication_occupation"	Test 2 "No Coal_liquefication_occupation is Processing_Of_Wood_And_Wood_Products_occupation "	+

Subclass axiom for the class “Ore_Refining_And_Foundry_occupation”	Test 1 "Every Ore_Refining_And_Foundry_occupation is a Secondary_occupation"	+
Disjoint class axiom of the class "Ore_Refining_And_Foundry_occupation"	Test 2 "No Ore_Refining_And_Foundry_occupation is Coal_liquefication_occupation "	+
Disjoint class axiom of the class "Ore_Refining_And_Foundry_occupation"	Test 2 "No Ore_Refining_And_Foundry_occupation is Plastic_engineering_occupation "	+
Disjoint class axiom of the class "Ore_Refining_And_Foundry_occupation"	Test 2 "No Ore_Refining_And_Foundry_occupation is Processing_Of_Paper_And_Related_Materials_occupation "	+
Disjoint class axiom of the class "Ore_Refining_And_Foundry_occupation"	Test 2 "No Ore_Refining_And_Foundry_occupation is Neurologists_occupation "	+
Disjoint class axiom of the class "Ore_Refining_And_Foundry_occupation"	Test 2 "No Ore_Refining_And_Foundry_occupation is Processing_Of_Wood_And_Wood_Products_occupation "	+
Subclass axiom for the class “Plastic_engineering_occupation”	Test 1 "Every Plastic_engineering_occupation is a Secondary_occupation"	+
Disjoint class axiom of the class "Plastic_engineering_occupation"	Test 2 "No Plastic_engineering_occupation is Coal_liquefication_occupation "	+
Disjoint class axiom of the class "Plastic_engineering_occupation"	Test 2 "No Plastic_engineering_occupation is Ore_Refining_And_Foundry_occupation "	+
Disjoint class axiom of the class "Plastic_engineering_occupation"	Test 2 "No Plastic_engineering_occupation is Processing_Of_Paper_And_Related_Materials_occupation "	+
Disjoint class axiom of the class "Plastic_engineering_occupation"	Test 2 "No Plastic_engineering_occupation is Neurologists_occupation "	+
Disjoint class axiom of the class "Plastic_engineering_occupation"	Test 2 "No Plastic_engineering_occupation is Processing_Of_Wood_And_Wood_Products_occupation "	+
Subclass axiom for the class “Processing_Of_Metal_occupation”	Test 1 "Every Processing_Of_Paper_And_Related_Materials_occupation is a Secondary_occupation"	+
Disjoint class axiom of the class "Processing_Of_Metal_occupation"	Test 2 "No Processing_Of_Paper_And_Related_Materials_occupation is Coal_liquefication_occupation "	+

Disjoint class axiom of the class "Processing_Of_Metal_occupation"	Test 2 "No Processing_Of_Paper_And_Related_Materials_occupation is Ore_Refining_And_Foundry_occupation "	+
Disjoint class axiom of the class "Processing_Of_Metal_occupation"	Test 2 "No Processing_Of_Paper_And_Related_Materials_occupation is Plastic_engineering_occupation "	+
Disjoint class axiom of the class "Processing_Of_Metal_occupation"	Test 2 "No Processing_Of_Paper_And_Related_Materials_occupation is Neurologists_occupation "	+
Disjoint class axiom of the class "Processing_Of_Metal_occupation"	Test 2 "No Processing_Of_Paper_And_Related_Materials_occupation is Processing_Of_Wood_And_Wood_Products_occupation "	+
Subclass axiom for the class “Processing_Of_Paper_And_Related_Materials_occupation”	Test 1 "Every Processing_Of_Paper_And_Related_Materials_occupation is a Secondary_occupation"	+
Disjoint class axiom of the class "Processing_Of_Paper_And_Related_Materials_occupation"	Test 2 "No Neurologists_occupation is Coal_liquefication_occupation "	+
Disjoint class axiom of the class "Processing_Of_Paper_And_Related_Materials_occupation"	Test 2 "No Neurologists_occupation is Ore_Refining_And_Foundry_occupation "	+
Disjoint class axiom of the class "Processing_Of_Paper_And_Related_Materials_occupation"	Test 2 "No Neurologists_occupation is Plastic_engineering_occupation "	+
Disjoint class axiom of the class "Processing_Of_Paper_And_Related_Materials_occupation"	Test 2 "No Neurologists_occupation is Processing_Of_Paper_And_Related_Materials_occupation "	+
Disjoint class axiom of the class "Processing_Of_Paper_And_Related_Materials_occupation"	Test 2 "No Neurologists_occupation is Processing_Of_Wood_And_Wood_Products_occupation "	+
Subclass axiom for the class “Processing_Of_Wood_And_Wood_Products_occupation”	Test 1 "Every Processing_Of_Wood_And_Wood_Products_occupation is a Secondary_occupation"	+
Disjoint class axiom of the class "Processing_Of_Wood_And_Wood_Products_occupation"	Test 2 "No Processing_Of_Wood_And_Wood_Products_occupation is Coal_liquefication_occupation "	+
Disjoint class axiom of the class "Processing_Of_Wood_And_Wood_Products_occupation"	Test 2 "No Processing_Of_Wood_And_Wood_Products_occupation is Ore_Refining_And_Foundry_occupation "	+

Disjoint class axiom of the class "Processing_Of_Wood_And_Wood_Products_occupation"	Test 2 "No Processing_Of_Wood_And_Wood_Products_occupation is Plastic_engineering_occupation "	+
Disjoint class axiom of the class "Processing_Of_Wood_And_Wood_Products_occupation"	Test 2 "No Processing_Of_Wood_And_Wood_Products_occupation is Processing_Of_Paper_And_Related_Materials_occupation "	+
Disjoint class axiom of the class "Processing_Of_Wood_And_Wood_Products_occupation"	Test 2 "No Processing_Of_Wood_And_Wood_Products_occupation is Neurologists_occupation "	+
Subclass axiom for the class “Educational_occupation”	Test 1 "Every Educational_occupation is a Quarternary_occupation"	+
Disjoint class axiom of the class "Educational_occupation"	Test 2 "No Educational_occupation is Engineering_occupation "	+
Disjoint class axiom of the class "Educational_occupation"	Test 2 "No Educational_occupation is Medical_occupation "	+
Disjoint class axiom of the class "Educational_occupation"	Test 2 "No Educational_occupation is Military_occupation "	+
Disjoint class axiom of the class "Educational_occupation"	Test 2 "No Educational_occupation is Public_service_Occupation "	+
Subclass axiom for the class “Engineering_occupation”	Test 1 "Every Engineering_occupation is a Quarternary_occupation"	+
Disjoint class axiom of the class "Engineering_occupation"	Test 2 "No Engineering_occupation is Educational_occupation "	+
Disjoint class axiom of the class "Engineering_occupation"	Test 2 "No Engineering_occupation is Medical_occupation "	+
Disjoint class axiom of the class "Engineering_occupation"	Test 2 "No Engineering_occupation is Military_occupation "	+
Disjoint class axiom of the class "Engineering_occupation"	Test 2 "No Engineering_occupation is Public_service_Occupation "	+
Subclass axiom for the class “Medical_occupation”	Test 1 "Every Medical_occupation is a Quarternary_occupation"	+
Disjoint class axiom of the class "Medical_occupation"	Test 2 "No Medical_occupation is Educational_occupation "	+
Disjoint class axiom of the class "Medical_occupation"	Test 2 "No Medical_occupation is Engineering_occupation "	+
Disjoint class axiom of the class "Medical_occupation"	Test 2 "No Medical_occupation is Military_occupation "	+
Disjoint class axiom of the class "Medical_occupation"	Test 2 "No Medical_occupation is Public_service_Occupation "	+
Subclass axiom for the class “Military_occupation”	Test 1 "Every Military_occupation is a Quarternary_occupation"	+
Disjoint class axiom of the class "Military_occupation"	Test 2 "No Military_occupation is Educational_occupation "	+

Disjoint class axiom of the class "Military_occupation"	Test 2 "No Military_occupation is Engineering_occupation "	+
Disjoint class axiom of the class "Military_occupation"	Test 2 "No Military_occupation is Medical_occupation "	+
Disjoint class axiom of the class "Military_occupation"	Test 2 "No Military_occupation is Public_service_Occupation "	+
Subclass axiom for the class “Public_service_Occupation”	Test 1 "Every Public_service_Occupation is a Quarternary_occupation"	+
Disjoint class axiom of the class "Public_service_Occupation"	Test 2 "No Public_service_Occupation is Educational_occupation "	+
Disjoint class axiom of the class "Public_service_Occupation"	Test 2 "No Public_service_Occupation is Engineering_occupation "	+
Disjoint class axiom of the class "Public_service_Occupation"	Test 2 "No Public_service_Occupation is Medical_occupation "	+
Disjoint class axiom of the class "Public_service_Occupation"	Test 2 "No Public_service_Occupation is Military_occupation "	+