Data Management Report

Master Degree in Data Science

Project 2: Construction of an ontology

16 January 2020

- **Candidate 1: Christian Riccio P37000002**
- **Candidate 2: Giacomo Matrone P37000011**

Brief Introduction of Ontologies

What is an ontology? An ontology is a formal explicit description of a domain of interest. Ontologies are used to capture knowledge about some domain of interest. An ontology describes the concepts in the domain and also the relationships that hold between those concepts. The main carachteristics are:

- ***Description***: is a form of representing a knowledge;
- ***Formal***: symbolic and mechanized;
- ***Explicit***: a list of piece of knowledge;
- ***Domain***: restricted to a particular sector of knowledge.

So, an ontology describe common words and concept used for describing a domain of interest. An object of this kind is composed of:

- ***Classes*** which represents the concepts of a domain of interests;
- ***Relations*** between these such classes;
- ***Property*** which every single concept own;
- ***Restriction*** on properties.

By the construction of the classes is possible to define instances that represents objects of the real world. Such instances also get the same properties of the class which they belongs to, by inheritance. So, in the end, we can say that **“an ontology describes the concepts in the domain and also the relationships that hold between those concepts”**.

Schematically, the concept of ontology can be represented by a vector that contains those five elements: ***ontology=(C, R, F, I, A)*** ,where:

- ***C*** stands for ***concepts*** and is the set of objects that we want to illustrate;
- ***R*** stands for the ***relations*** between those objects;
- ***F*** indicate the set of ***functions*** established inside the concepts;
- ***I*** stands for the ***instances***, basically are the objects of the real world;
- ***A*** stands for ***axioms*** that are propositions that are always true.

About Protégé

The tool used to construct and edit ontologies is the software Protégé created by Stanford University. It is a Java-based software and gave the possibility to menage ontologies in different format: in particular the two most known are `rdfs` and `owl` (Ontology Web Language).

The stages of ontology design, are:

- Determining the domain and the goal of the ontology;
- Focusing the principal concepts (named as Key concepts) of what to describe;
- Defining constrains on properties;
- Constructing instances;
- Assigning values to properties for all the instances created.

We can classify the part of an ontology by considering the following concepts:

- (1) **Individuals:** represent objects in the domain in which we are interested;
- (2) **Properties:** are binary relations on individuals since they link to individuals together. Such properties are limited to having a single value and so to being *functional*;
- (3) **Classes:** are view as a set that contain individuals, also class may be organized in *subclasses*;

The empty ontology contains one class called `Thing`. Such class is the one that represents the set containing all individuals.

The project

The development of our work concernes on our personal view about the concept of *System of Transport*. So, starting with it our Thing class represent such system.

Our classes and sub-classes are:

- *Air*:
    1. Airlift
    2. Airline
- *Ground*:
    1. Cargo
    2. Passengers
- *Sea*:
    1. Cruise
    2. Shipping
- *Employees*:
- *Infrastructures*:
    1. Airports
    2. Ports
    3. Railways
    4. Roads
- *Managers*
- *Payment_method*:
    1. Online
    2. Offline
- *Private*
- *Pubblic*
- *Trips*
- *Users*:
    1. Companies
    2. Person

Firs of all, is needed to diversify classes between eachother, and in particular strarting from the 3 types of transports we impose the constrains that Air,Ground and Sea are `Disjoint` eachother. The same for the classes referred to Employees and Managers. Considering sub-classes of Infrastrucures (i.e. Aiports,Ports, Railways,Roads), it comes out that they are disjoint eachother too. Private is disjoint from Public and at the end the two sub-classes of Users (i.e. Companies and Person) are disjoint from eachother as well.

The next step has been to define properties that exsist between classes, i.e relationships between classes. Infact this properties links individuals. Such properties express a particular behaviour needed to specify the link between individuals, in a certain way this thing is seen like a *feature* of the property. For example, focusing on our work, the property *"can_be_private"* refers to the domain Private and links the classes: Airlift,Cruise, Shipping, Ailiner, Cargo, Passengers and such property is Asymmetric and Irreflexive and, also, is the inverse of "*can_be_public*".

Furthermore, different classes need to be disjoint eachother simply because a kind of object cannot be in the same moment an another one, so instances are uniques.

At the end we had focused our attention on the `DataProperty` to then work on Datatype. Datatype properties link an individual to an XML Schema Datatype value or an rdf literal. In other words, they describe relationships between an individual and data values, hence a datatype property can be used to relate an individual to a concrete data value that may be typed or untyped.

In [ ]: