Upper Ontologies

`A limited number of very high-level concepts that appear constantly.`

1. Introduction

There are many different types of knowledge. However, in almost of the cases as humans being we tend to characterize any given knowledge by two **mental constructions**:

- Categories.
- Objects.

Therefore, we can assume that an object belongs to a category. We will use them to express any kind of information.

2. Upper Ontologies

`Definition`
An ontology is a formal, explicit description of a domain of interest.

A simpler definition expresses an ontology as: an attempt to write down our mental structure when we deal within a specific knowledge. It has four main peculiarities:

- 1st **formal**. An ontology should be written using a language with a clear and non-ambiguous semantic. A correct ontology should not leave space to ambiguity; in this case, we cannot use **Natural Language** since that it's the primary cause of ambiguity.
- 2nd **explicit**. The information should be available or, in the worst case, derivable in a finite time.
- 3rd **description**. It should provide us interesting informations.
- 4th **domain**. Such description, it should be related to some topic of interest.

As humans being, we always make use of ontologies to describe our mental structure about a specific domain or topic of interest; at the same time, we tend to organize our own mental structures through two notions:

- **Categories**.
- **Objects**.

Example

Let's us suppose the following problem:
`the new university director wants to organize all the employees, with the only purpose of the salary management`

From the description of the problem we can already determine which is the domain of the ontology: **the management of the salaries**.

In addition, we could imagine that the university employees belong to:

- Administratives group.

- Technicians group.
- Researches group.

Given our goal (salary management), we have to define which are the properties that have a meaning in the ontology representation. For example, since we have the notion of *employee* a more general category of it should be `Person`. However, this new possible category is out of the scope, we should not care about the `Person` entity.

Categories can be more or less general (person vs. researcher) and, often, these categories can be organized following a hierchical order, like the subclass and superclass interpretation in Object-Oriented programming. Generally speaking, this distinction between categories describes also the **depth** of the ontology used. If we are handling within the **more general categories** usually we say that we are dealing with **upper ontologies**.

`Definition`
An **upper ontology** is an ontology that focuses on the more general domain.

At least an upper ontology has two characteristics:

- It should be applicable to any specific domain; since that it's the more general one, if we cannot derive this behavior the ontology used is not an upper ontology.
- We should be able to combine different general concepts without incurring in inconsistencies.

3. Categories

It is commonly accepted that humans represent the knowledge in terms of categories and objects belonging to. Part of the mental structure relies on categories and the same time the goals of the knowledge can be either **instance-specific** or **category-driven**.

But, why do we need categories? One possible answer would be: categories allow us to predict how the future will evolve or what will happen in the future. Therefore, given the categories, we may recognize new objects classifying them by the known categories. The act of classify a new object is a **prediction**.

In First Order Logic, we have two possible alternatives to make a prediction:

- 1st representing **categories as predicates**. Following this approach we end up in the Second Order Logic, not practicable.
- 2nd through **reification**, representing **categories as objects**. This is possible by the notion of `membership` and `subclass`.

By the way, there's one main problem: what is it the right language to represent categories, instances and their relationships? This query defines one of the most difficult task because, in the computational field, we have adopted the most simple approach: determine categories and their entities by the mathematical notion of set.

Therefore, the notion of `membership` is defined by the $\in$ symbol used through sets and, indeed, the notion of `subclass` is described by the $\subset$ symbol.

Example

Given the object `aa123bb`, we know that `aa123bb` belongs to the category `Car`:

$$aa123bb \in Car.$$

Any member of the category `Car` is also a member of the category `Vehicle`:

$$Car \subset Vehicle.$$

Usually, each member of a category enjoys some **properties** (called **necessity properties**):

$$(x \in Car) \rightarrow hasWheels(x).$$

Members of a category can be recognized by some properties (named **sufficiency properties**):

$$hasPlate(x) \land hasWheels(x) \rightarrow x \in Car.$$

Additionally, categories can be seen as properties of other categories:

$$Car \in VehicleType.$$

Categories relate each other by the subclass relation, but in some cases given a set of categories S, the categories theirselves can be disjointed. Formally, the disjoint relation is described as follows:

$$Disjoint(S) \Leftrightarrow (\forall c_1, c_2 \in S \land c_1 \neq c_2 \rightarrow c_1 \cap c_2 = \emptyset).$$

So, basically, we are saying that the two categories c_1 and c_2 do not share the same objects.

Subcategories might also cover all the possible categories of the parent category. Formally, given a category c and a set S of categories, S is an **exhaustive decomposition** (meaning that we can cover the general category c by all its subcategories in S) of c if:

$$ExhaustiveDecomposition(S, c) \rightarrow (\forall i \in c \Leftrightarrow \exists c_2, c_2 \in S \land i \in c_2).$$

If a category can be decomposed in more categories and each of them is disjointed from the others, then we have a **partition**. Given a category c, a partition of the category c occurs when:

- We have **disjointness** between subcategories of c.
- We can describe an **exhaustive decomposition** of c.

Formally, given a category c and a set S of subcategories, S is a **partition** of c if:

$$Partition(S, c) \Leftrightarrow Disjoint(S) \land ExhaustiveDecompotion(S, c).$$

4. Physical Composition

Physical composition is another discipline based around the study of when something is a part of a more large entity. This discipline is called **meriology**, that studies the relation between object and its parts (note: we are assuming categories like object representations).

Example

- A shopping bag contains three pieces of bread, ...

- A car is composed of an engine, four wheels, five seats, ...
- A laptop is made of a motherboard, a cpu, a keyboard, a screen, ...

Sometimes between the parts of an object could be a **structural relation**. This feature may help us to distinguish objects in the physical composition discipline:

- If it exists, the relation is named **PartOf**. The PartOf relation enjoys some properties, as:
  - 1st **Transitivity**. $PartOf(x, y) \land PartOf(y, z) \rightarrow PartOf(x, z)$.
  - 2nd **Reflexivity**. $PartOf(x, x)$.
- If it does not exist, the relation is named **BunchOf**. The BunchOf relation aims to define objects in terms of composition of other **countable** objects.