OO FEATURES THAT OWL LACKS

HENRIETTE HARMSE

At some level OWL seem to be very similar to UML, but there are important differences that you have to keep in mind. In this post I detail features you will have to do without coming from a coding or object orientation perspective when you start using OWL. These are:

1. classes do not have attributes,
2. classes do have class variables (but **not** in the way you think),
3. you have to live without methods,
4. there are no interfaces, and
5. neither are there any abstract classes.

1. Classes do not have Attributes

Programmers are used to classes that have attributes or properties. This is also refered to as instance/member variables of the class. In OWL classes do not have properties. Rather, properties can be used to model relations that exist between classes and data types. You can however state that a class is a subclass of something that has some property. I.e.,

```
Class: Course
  SubClassOf:
    hasSubject some Subject
```

which states that a `Course` is a subclass of things that have some `Subject`. I explain this here and here.

2. Classes do have class variables (but **not** in the way you think)

In programming class variables are values that describe the class rather than instances of the class. In OWL this is achieved trough `annotations`. There exist pre-defined annotation properties and you can define your own annotation properties. Interestingly annotation properties can be specified for any class, individual or axiom, as well as the ontology itself. The reasoner cannot reason about annotations, it is used for purely

3. There are no Methods nor Interfaces

This may seem like silly a comment, but is worth making it explicit: "OWL does not have methods". Why not? Well, OWL is not a programming language nor a modelling language for designing software. It is a conceptual modelling language. Can you model methods in OWL? Yes, I have done it here to find software modelling heuristic violations, but it is not trivial. However, even if you can define the conceptual notion of a method, there is no way to execute methods in OWL. Again, because it is not a programming language.

Date: 15th April 2018.

Well, since there are no methods in OWL, there are no interfaces either. In programming the idea is that an interface defines the signature of an interaction, usually in terms of method signatures, without specifying how the methods are actually implemented. This allows for the same interface to have different implementations.

4. Neither are there Abstract Classes

Abstract classes in programming are used to enable programmers to only implement a subset of the interfaces specified, thereby forcing subclasses to do the implementation. However, again since interfaces do not exist, and there are no methods, it does not make sense for OWL to support abstract classes.

What about the case where in programming an abstract class is defined that only consist of member/instance variables? In programming this has the effect that no instances of this class can be created directly. The only way an instance can be created is by creating an instance of a subclass of the abstract class. Importantly these instances created of the subclass are still instances of the abstract class.

In OWL there is **no** way to force that individuals cannot be created of a class, but only for its subclasses. The closest to not being able to create individuals for a class is the class ‘owl:Nothing‘, but that really means the class has zero individuals, which is not what is meant by an abstract class in programming.

5. Conclusion

In this post I detailed some object oriented features programmer may feel OWL lacks. However, OWL is a conceptual modelling language with reasoning capability. Thinking about that causes one to realize it makes as little sense to say OWL lacks some OO feature as saying that Java or C# lacks reasoning capability.

References