EQUIVALENT VERSUS SUBCLASSOF

HENRIETTE HARMSE

In creating their first ontology, there are at least two aspects of `EquivalentTo` and `SubClassOf` that perplex users. The first is when to use `EquivalentTo` and when to use `SubClassOf`. The second problem is best illustrated by the following example:

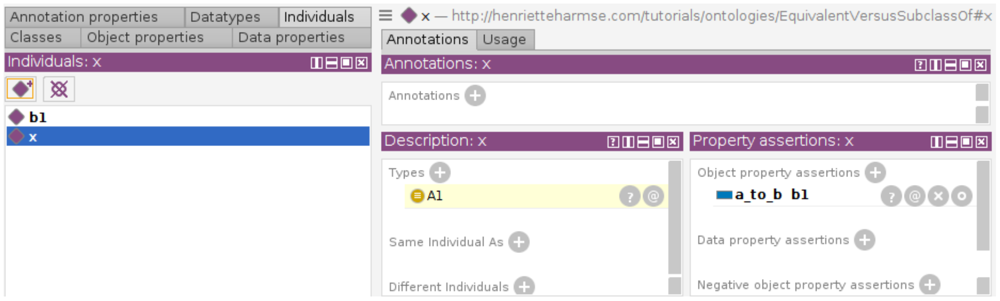
```
ObjectProperty: a_to_b

Class: A1
   EquivalentTo: (a_to_b some B)

Class: A2
   SubClassOf: (a_to_b some B)

Class: B

Individual: b1
   Types:
       B

Individual: x
   Facts:
       a_to_b  b1
```

When running a reasoner on this example, the individual `x` is inferred to be of type `A1`. What perplex users sometimes is that `x` is not inferred to be of type `A2` as well.

1. The difference between `EquivalentTo` and `SubClassOf`

The first thing to be aware of wrt `equivalentTo` is that

Date: 21st December 2021.



b1
x

x — http://henrietteharmse.com/tutorials/ontologies/EquivalentVersusSubclassOf#x

Types
A1

Object property assertions
a_to_b b1

FIGURE 1. `x` is inferred to be of type `A1`

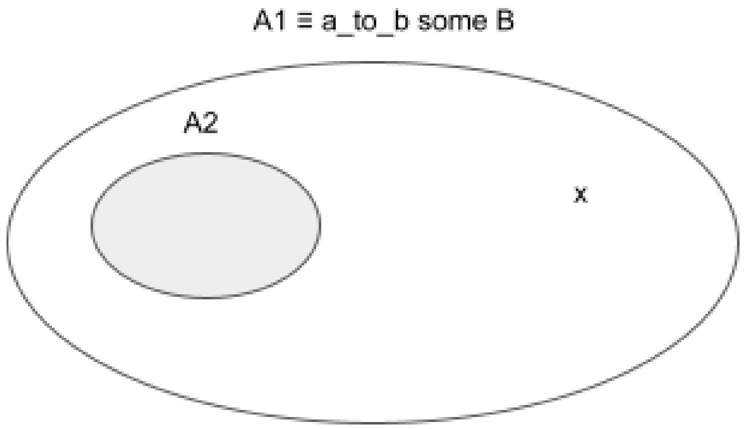


FIGURE 2. A2 and x wrt a_to_b some B

```
Class: C
    EquivalentTo: D
```

is an abbreviation for

```
Class: C
    SubClassOf: D

Class: D
        SubClassOf: C
```

The semantics of `SubClassOf` is subset. Thus, the above states that the set `C` is a subset of the set `D` and the set `D` is a subset of the set `C`. Which means that the sets `C` and `D` are exactly the same set. We say they are equivalent.

Note that if I know that the classes `C1` and `C2` are both subclasses of class `C`, there is nothing more I can say about how class `C1` relates to class `C2`. This is a bit like knowing that bicycles and trucks are both vehicles - I can say nothing more about how bicycles relate to trucks beyond knowing that they are both vehicles.

2. Back to our initial example

Understanding the semantics of `EquivalentTo` we can see that indeed the individual `x` is an instance of `A1`. Understanding the semantics of `SubClassOf` helps us to understand why `x` is not inferred to be of type `A2`. We know that `A2` is a subclass of `a_to_b some B` and that `x` is an instance of `a_to_b some B`, but there is nothing that can force the reasoner to infer that `x` is necessarily an instance of the class `A2`. This is illustrated in the next figure.

```
Class: C1
SubClassOf: C

Class: C2
SubClassOf: C
```

What can we say about how `C1` and `C2` relates to each other? Absolutely nothing. All we know about `C1` and `C2` is that they are both subclasses of `C`. It is a bit like knowing a bicycle and a truck are both vehicles. That does not make a bicycle a truck or a truck a bicycle, though both are vehicles.

github

References