Foundations of Semantic Web Technologies

Tutorial 5

Dörthe Arndt

WS 2022/23

Exercise 5.1. Consider the data below relating to creatures that appear in the works of H.P. Lovecraft (among other authors).

```
@base <http://example.org/> .
@prefix ex:  <http://ex.org/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

ex:Cthulhu ex:knownAs "Cthulhu" , "The Great Dreamer" ;
ex:father ex:Nug ;
ex:createdBy ex:HPLovecraft ;
ex:offspring ex:Cthylla , ex:Nctosa , ex:Nctolhu ;
ex:partOf ex:GreatOldOnes .

ex:Hastur a ex:GreatOldOne ;
ex:knownAs "Hastur" , "The King in Yellow" .

ex:YogSothoth ex:knownAs "Yog-Sothoth" ;
ex:partOf ex:OuterGods ;
ex:offspring ex:Nug .

ex:LaviniaWhateley a ex:FictionalCharacter ;
ex:memberOf ex:FirstUnitedChurchOfCthulhu ;
ex:givenName "Lavinia Whateley" .

ex:WilburWhateley a ex:FictionalCharacter ;
ex:memberOf ex:FirstUnitedChurchOfCthulhu ;
ex:father ex:YogSothoth ;
ex:mother ex:LaviniaWhateley .

ex:Nyarlathotep ex:createdBy ex:HPLovecraft ;
ex:partOf ex:OuterGods .

ex:Azathoth ex:createdBy ex:HPLovecraft ;
ex:partOf ex:OuterGods .

ex:Necronomicon a ex:Book ;
ex:writtenBy ex:AbdulAlhazred , ex:MadArab ;
ex:describes ex:Cthulhu , ex:YogSothoth , ex:Hastur ;
ex:createdBy ex:HPLovecraft .

ex:CthulhuMythos a ex:FictionalUniverse .

ex:GreatOldOnes a ex:Group .

ex:OuterGods a ex:Group .

ex:FirstUnitedChurchOfCthulhu a ex:ReligiousCult .
```

```
        ex:LovecraftianDeity ex:partOf ex:CthulhuMythos .
```

As in exercise 4.4, we want to add additional OWL triples to derive new data. You can either use `http://rdfplayground.dcc.uchile.cl` or `http://ppr.cs.dal.ca:3002/n3/editor/s/8DkcyTy6`. If you choose the second option, just add the data and the axioms you write bolow the rules which are already stated.

For the first option, you need to copy and paste the from data above to either the text filed on the left. On the right-hand side, you should add RDFS/OWL definitions and axioms that allow to infer what is specified in the question (note that you cannot add the required data explicitly; it must be inferred through the requested RDFS/OWL definitions). The definitions you add should accumulate. Note that you can define nested definitions as follows, which will count as one axiom (it states that: the class of *DCC students who are also masters or PhD students* is a sub-class-of the class of *entities that have a supervisor who is a professor on the DCC staff*):

```
        [ owl:intersectionOf ( :DCCStudent [ owl:unionOf ( :MastersStudent :PhDStudent ) ] ) ]
        rdfs:subClassOf
        [ owl:someValuesFrom [ owl:intersectionOf ( :Professor :DCCStaff ) ] ;
                owl:onProperty :supervisor ] .
```

In this (rather complex) example we used `rdfs:subClassOf` (rather than `owl:equivalentClass`) since other types of students may have DCC professors as supervisors. As another example relating to the data provided about Lovecraftian characters that you can copy and paste into the system to get started, consider:

```
        @prefix ex:  <http://ex.org/>.
        @prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
        @prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
        @prefix owl:    <http://www.w3.org/2002/07/owl#> .

        [ owl:hasValue ex:FirstUnitedChurchOfCthulhu ; owl:onProperty ex:memberOf ]
        rdfs:subClassOf [ owl:hasValue ex:Cthulhu ; owl:onProperty ex:worships ] .
```

This defines that any member of the *First United Church of Cthulhu* worships *Cthulhu*, but not vice versa (entities that worship *Cthulhu* may not necessarily be in the *First United Church of Cthulhu*). We will see that this infers:

```
        ex:LaviniaWhateley ex:worships ex:Cthulhu .
        ex:WilburWhateley ex:worships ex:Cthulhu .
```

You might also see that *Lavinia Whateley* and *Wilbur Whateley* are now instances of blank nodes; these blank nodes refer to the class you defined as `[ owl:hasValue ex:Cthulhu ; owl:onProperty ex:worships ]`, i.e., the class of entities that worship *Cthulhu* (such blank nodes can be ignored). If we also, hypothetically, wanted to infer, vice versa, that any entity that worships *Cthulhu* is in the *First United Church of Cthulhu*, we could replace `rdfs:subClassOf` in the definition with `owl:equivalentClass` (but we don't want to infer this).

Extend this example with RDFS/OWL axioms to answer the following:

(a) Add one axiom to define that anyone who is part of the *Great Old Ones* group is an instance of the *Great Old One* class, and vice-versa, inferring that:

```
                ex:Hastur ex:partOf ex:GreatOldOnes .
                ex:Cthulhu a ex:GreatOldOne .
```

(b) Add one axiom to state that all *offspring* of an instance of *Great Old One* are also instances of *Great Old One*, inferring that:

```
                ex:Cthylla a ex:GreatOldOne .
                ex:Nctosa a ex:GreatOldOne .
                ex:Nctolhu a ex:GreatOldOne .
```

(c) Add one axiom to state that all those that are part of the *Outer Gods* or are part of the *Great Old Ones* are all instances of *Lovecraftian Deity* (but not vice versa), inferring that:

```
ex:Cthulhu a ex:LovecraftianDeity .
ex:Hastur a ex:LovecraftianDeity .
ex:YogSothoth a ex:LovecraftianDeity .
...
```

(d) Add one axiom to state that anyone with a father or mother that is a *Lovecraftian Deity* is a *Supernatural Being*, inferring that:

```
ex:WilburWhateley a ex:SupernaturalBeing .
```

(e) Add one axiom to define that a *Lovecraftian Deity* created by *H.P. Lovecraft* is part of the *Cthulhu Mythos* (but not vice versa).

```
ex:Cthulhu ex:partOf ex:CthulhuMythos .
ex:Nyarlathotep ex:partOf ex:CthulhuMythos .
...
```

(f) Add one axiom to state that any *Book* created by *H.P. Lovecraft* was authored by *at most* one entity.[1] This should infer:

```
ex:AbdulAlhazred owl:sameAs ex:MadArab .
ex:MadArab owl:sameAs ex:AbdulAlhazred .
...
```

Exercise 5.2. In our last OWL-exercise, we will look at more complex entailments involving existentials, disjunctions and counting. For this we will use a reasoner called HermiT[2]: an OWL 2 DL reasoner based on a tableau rather than rules. HerMiT will find all possible entailments and will halt on any valid input, but (unlike the previous rule-based reasoner) may reject ontologies with features that may lead to undecidability.

Navigate to `http://cc7220.dcc.uchile.cl/lab06/` to find a simple interface with the following default data loaded for you:

```
@prefix ex:  <http://ex.org/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

ex:divisor a owl:ObjectProperty .

ex:value a owl:DatatypeProperty , owl:FunctionalProperty .

ex:Number a owl:Class .
ex:OddNumber a owl:Class .
ex:EvenNumber a owl:Class .
ex:PrimeNumber a owl:Class .
ex:CompositeNumber a owl:Class .
ex:UnitNumber a owl:Class .
ex:PowerOfTwoNumber a owl:Class .

ex:N1 a ex:UnitNumber ; ex:value 1 .
```

[1]The property *created by* indicates that Lovecraft created the book for his fiction, while the property *written by* indicates the author of the book according to his fiction.

[2]`http://www.hermit-reasoner.com/`

```
ex:One a ex:UnitNumber .

ex:N2 a ex:Number ; ex:value 2 .

ex:N3 a ex:OddNumber ; ex:value 3 .

ex:N4 a ex:Number , ex:PowerOfTwoNumber ; ex:value 4 ;
ex:divisor ex:N2 .

ex:N5 a ex:Number ; ex:value 5 .

ex:N6 a ex:EvenNumber ; ex:value 6 ;
ex:divisor ex:N3 ;  .

ex:N7 a ex:PrimeNumber ; ex:value 7 .

ex:N8 a ex:Number ; ex:value 8 ;
ex:divisor ex:N4 .

ex:N15 a ex:OddNumber ; ex:value 15 ;
ex:divisor ex:N5 , ex:N3 .
```

You can simply write your solutions at the bottom of the data (there is only one input form).

We use `ex:divisor` to indicate a factor of a number. We consider numbers to be positive (non-zero) integers.

You should add RDFS/OWL definitions and axioms that allow to infer the given data. Other (reasonable) inferences not listed may also arise. Your answers should only define axioms on classes and properties. Axioms should accumulate from question to question. You can invent new classes or new properties, but this reasoner requires that classes (that do not have explicit instances in the data) are declared as such, and that properties are declared as datatype properties (taking literal values), or object properties (taking IRI or blank node values). So (though not necessary) if you wish to add a class such as `ex:NonPrimeNumber` and a new (object) property such as `ex:multiple`, be sure to add:

```
ex:NonPrimeNumber a owl:Class .
ex:multiple a owl:ObjectProperty .
```

When we say "axioms" it can be one or more (not necessarily multiple). There might be multiple equivalent ways to answer a question. Also please note that some triples that are entailed will not be shown (the set of entailed triples would be infinite). Of particular note is that, for some reason, the reasoner does not print `owl:sameAs` inferences (rather you can see that the two terms will be used interchangeably in the results).

(a) Add axioms to state that all numbers have 1 (`ex:N1`) and themselves as divisors. These axioms should infer that:

```
ex:N2 ex:divisor ex:N1 , ex:N2 .
ex:N4 ex:divisor ex:N1 , ex:N4 .
...
```

(b) Add axioms to state that an even number is a number with 2 (`ex:N2`) as a divisor, while an odd number is a number that is not even. These axioms should infer that:

```
ex:N2 a ex:EvenNumber .
ex:N3 a ex:Number .
ex:N4 a ex:EvenNumber .
ex:N6 a ex:Number ; ex:divisor ex:N2 .
ex:N15 a ex:Number .
```

***Note**: we have nothing to prove that something is an odd number yet since (under the Open World Assumption) it is possible that any number has 2 as a divisor, just that we have not stated it in the data.*

(c) Add axioms to state that a composite number is a number with more than two divisors, a prime number is a number with precisely two divisors, and a unit number is a number with precisely one divisor. (The counts of divisors include the number itself and 1.) These axioms should infer that:[3]

```
ex:N1 a ex:Number , ex:OddNumber .
ex:One a ex:Number .
ex:N2 ex:divisor ex:One .
...
ex:N4 a ex:CompositeNumber .
ex:N6 a ex:CompositeNumber .
ex:N7 a ex:OddNumber .
ex:N8 a ex:CompositeNumber .
ex:N15 a ex:CompositeNumber .
```

Note*: There are a number of very indirect conclusions pseudo-magically appearing here: try to figure out why each such triple is entailed (e.g., how do we know that `ex:N1` and `ex:One` are the same, how do we know that 1 is odd, how do we know that 7 is odd, how do we know that 4, 6, 8 and 15 are composite, etc.?). But still, we cannot yet infer any new primes: under the Open World Assumption, numbers (other than the unit number) may have other divisors that we have not yet included in the data: we cannot yet prove that any number has precisely two divisors to conclude that it is prime.*

(d) Add axioms to state that an even number is a number that has at least one even number as a divisor, that any composite number has a prime divisor (not vice versa: we know that prime numbers have themselves as a divisor), and that a number is a power of 2 if and only if it has precisely one divisor that is odd. This should infer:

```
ex:N1 a ex:PowerOfTwoNumber .
ex:N2 a ex:PrimeNumber , ex:PowerOfTwoNumber .
ex:N5 a ex:OddNumber .
ex:N8 a ex:EvenNumber .
```

Note*: Again we get some indirect inferences: for example, how do we know that 1 is a power of two, why do we now infer that 5 is odd when we said nothing in our axioms about odd numbers, and how do we know that 2 is a power of 2 and also prime?*

If you look through the results, you'll find some other interesting conclusions. Of course here we used numbers as an example, but these rich types of inferences could also be applied for zebroids, medical treatments, car parts, magnetic fields, etc. Note that if you try your solutions in RDF Playground, you will miss inferences because the (OWL 2 RL/RDF) rules that RDF Playground uses are incomplete. On the other hand, if you define that divisor is transitive, then the HermiT reasoner used here will reject the input as it does not allow for defining cardinalities on transitive properties (in order to ensure decidability).

[3]We should infer that `ex:N1 owl:sameAs ex:One`, but the reasoner seems to not show `owl:sameAs` inferences, maybe for conciseness.