README.md

ENVRI-FAIR Ontology Training

This training material was first prepared for the ENVRIweek 2020 and its training event for data center staff on *Terminologies for ENVRIs: Why, What & How*.

In this hands-on session we will learn how to construct ontologies using the Web Ontology Language by example of the Environment Ontology (ENVO). Specifically, we will describe the ENVO term lightning strike as an environmental system process. We create our small ontology with Protégé, which we first need to install.

Requirements

Please download, install and start Protégé on your local computer.

Ontology

Having started Protégé, you can describe your ontology by giving it an IRI, a version, arbitrary annotations and specifying what other ontologies need to be imported. Importing ontologies is useful if you want to reuse existing ontologies, for instance to extend ontologies with your own classes and properties. For instance, ENVO provides the class atmospheric observatory system which ACTRIS could extend with more specific observatories.

For the purpose here, we (direct) import the Dublin Core Elements 1.1 properties, specifically the OWL version at `https://protege.stanford.edu/plugins/owl/dc/protege-dc.owl`. This gives us additional annotation properties which we can use to annotate our ontology.

Let's give our ontology the IRI `http://vocab.envri.eu/` and version IRI `http://vocab.envri.eu/1.0.0`. It is good practice to also annotate your ontology to add creators, contributors, etc. The Dublin Core annotation properties we just imported support this. Let's thus add a few ontology annotations, in particular `dc:creator` and `dc:date`.

Classes

Following ENVO, the task here is to describe lightning strike as an environmental system process. Let's create some required classes. To do so, we navigate to the `Entities` and `Classes` tab in Protégé. You will see the top class `owl:Thing`. All your classes are sub classes of `owl:Thing`. Using the right button on your mouse, you can now create sub classes and thereby create a class hierarchy. Let's start with the class *environmental system process*. When creating a new class you need to provide a name. This can be a random string, an identifier such as a UUID, or a more intelligible name. ENVO names the class *environmental system process* with the ID `ENVO_02500000`. For the purpose here, we name this class simply "environmental system process". You will see that Protégé will generate an IRI for the given name.

We can now annotate our class. It is typically useful to provide a label for your classes, especially if you use some unintelligible name. You can annotate classes in the corresponding panel. As for ontology annotations, Protégé provides you a list of known annotation properties from which you can choose. Let's annotate our newly created class with an `rdfs:label` with literal value "environmental system process". You can also specify a language to state that this label is in English. This mmechanism allows for label translations. If you visit the ENVO class environmental system process you will notice that there is also comment. Include this comment as a class annotation using the property `rdfs:comment`. Similarly to the label, you can also specify the language of the comment.

Great, now we repeat these steps for the following classes: *atmospheric process*, *atmospheric lightning* and *lightning strike*. They form a class hierarchy. Lightning strike is thus a sub class of atmospheric lightning, which is a sub class of atmospheric process, which is a sub class of environmental system process. Provide labels for the classes and include the ENVO comment.

Descriptions

Now that we have described our class hierarchy for lightning strike as an environmental system process we should further describe the semantics of these classes. For lightning strike we included the comment

> A lightning process during which electrostatic discharge occurs between a cloud and an object on a planetary surface, or a planetary surface itself.

This amounts to a human readable description (definition) of lightning strike. The Web Ontology Language enables formal (i.e., machine readable) descriptions of classes. These are classes "class descriptions" or "axioms".

About lightning strike, ENVO states the following:

```
has participant some (planetary surface or material entity and adjacent to
some planetary surface)
```

This is a formal description of the latter part of the comment, namely "object on a planetary surface, or a planetary surface itself". Some planetary surface or some object on a planetary surface are *participants* in the lightning process. We thus need the `has participant` property. This is, specifically, a so called *object* property.

Let's add the `has participant` object property to our ontology. To do so, navigate to the `Entities` and `Object properties` tab in Protégé. You will see there the top object property `owl:topObjectProperty`. All your object properties are sub properties of `owl:topObjectProperty`. Using the right button on your mouse, you can now create the sub property `has participant`. Similarly to classes, we can annotate object properties. Add a sensible `rdfs:label` to your property.

There is an additional object property used in the ENVO description of lightning strike, namely `adjacent to`. Include this property as a sub property of `owl:topObjectProperty`.

We are still not quite done. As we can see, the ENVO description of lightning strike includes two additional classes, namely planetary surface and material entity. We need to include them in our ontology as well. In our ontology, material entity is a sub class of `owl:Thing` and planetary surface is a sub class of material entity.

Now we can express the formal description of lightning strike and its participants. For this, select the lightning strike class in Protégé and create an additional sub class axiom using the class expression editor. Type:

```
'has participant' some ('planetary surface' or 'material entity' and
'adjacent to' some 'planetary surface')
```

Now we can proceed with formal descriptions of other classes. Let's look at atmospheric lightning. In addition to being a sub class of atmospheric process, ENVO describes atmospheric lightning as a sub class of the electrostatic discharge process having at least one participant cloud and plasma as output. We already have included the `has participant` object property but we lack the classes `electrostatic discharge process`, `cloud` and `plasma` as well as the additional object property `has output`. By now you already know how to include them into your ontology. Make sure to classify these new terms into the existing hierarchy following ENVO.

Having added these classes and the additional property, we can now describe atmospheric lightning further. Select the `atmospheric lightning` class and added three additional sub class of axioms.

- Use the class hierarchy for `electrostatic discharge process`
- Use the object restriction creator for the participating cloud

- Use again the object restriciton creator for the output plasma.

These steps are repeated for all classes.

Individuals

So far we have specified our so called "domain of interest", that is lightning strikes as environmental system processes. We have described the terms we need to talk about this domain of interest.

We can now use these terms to make assertions about individuals, i.e. concrete lightning strikes that occurred in time and space. These are lightning strike *instances*.

Select the class `lightning strike` and add an instance. This is done by adding a new individual. You can name it as you wish, say `a lightning strike`. Annotate the individual with a suitable label. Now we can further relate our lightning strike following the class descriptions. Our lightning strike has two participants, at least one cloud and some planetary surface or material entity on some planetary surface and has some plasma output.

Let's add `a tree` as an individual material entity and `a forest floor` as a planetary surface. Now we can make the following two object property assertions:

```
a tree is adjacent to a forest floor
a lightning strike has partipant a tree
```

Let's add `a cloud` as an individual cloud and make the following property assertion:

```
a lightning strike has participant a cloud
```

Finally, let's add `a plasma` as an individual plasma and make the following property assertion:

```
a lightning strike has output a plasma
```

Data properties

In addition to object properties, we can also specify data properties. The difference is that the value of a data property is a literal, e.g. a number, string, date, etc. while the value of an object property is an individual.

Our forest floor planetary surface obviously has a polygon perimeter which could be encoded as a string of coordinates. Similarly, our tree is located at some latitude and longitude coordinates. These literals can be represented as data property assertions. Indeed, in a database about lightning strikes such information is important and ground the *symbols* used here (e.g., `a tree`) into the physical world.