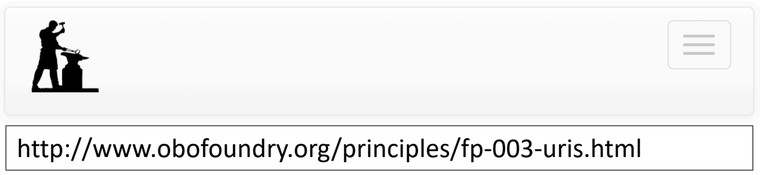
A Python Utility for Working with OBO Foundry Terms

Jonathan P. Bona
Department of Biomedical Informatics,
University of Arkansas for Medical Sciences, Little Rock, AR

OBO Principles regarding term identifier conventions require ontology terms to use numeric local term identifiers, and forbid local identifiers that "consist of labels or mnemonics meaningful to humans."

This requirement can make it hard to work directly with the identifiers. To write by hand (or in code) an assertion that a certain individual is an instance of UBERON:lung', one must know that its URI is:
`http://purl.obolibrary.org/obo/UBERON_0002048`


http://www.obofoundry.org/principles/fp-003-uris.html

Principle: URI/Identifier Space (principle 3)

Each class and relation (property) in the ontology must have a unique URI identifier. The URI should be constructed from a base URI, a prefix that is unique within the Foundry (e.g. GO, CHEBI, CL) and a local identifier (e.g. 0000001). The local identifier should not consist of labels or mnemonics meaningful to humans. Additional information is available at http://www.obofoundry.org/id-policy

This poster describes a simple utility that allows programmers to import representations of an OBO Foundry ontologies as Python classes, and then use those Python classes within a program to refer to terms in the ontology by their labels rather than managing the URIs explicitly.

In Python software that we have written in the past to transform instance data into semantic representations with OBO Foundry ontologies, we started out managing the associations between ontology labels and URIs with ad-hoc mappings involving only terms that were of immediate interest.

```
uberons = {'larynx' :
           'http://purl.obolibrary.org/obo/UBERON_0001737',
           'nasopharynx' :
           'http://purl.obolibrary.org/obo/UBERON_0001728',
           'hypopharynx' :
           'http://purl.obolibrary.org/obo/UBERON_0001051',
           'oropharynx' :
           'http://purl.obolibrary.org/obo/UBERON_0001729',
           'glottis' :
           'http://purl.obolibrary.org/obo/UBERON_0002486',
           'sinus' :
           'http://purl.obolibrary.org/obo/UBERON_0001825',
           'anatomical entity' :
           'http://purl.obolibrary.org/obo/UBERON_0001062',
           'colon' :
           'http://purl.obolibrary.org/obo/UBERON_0001155',
           'oral cavity' :
           'http://purl.obolibrary.org/obo/UBERON_0000167'}
```

For example, we might maintain a Python dictionary object for all the UBERON terms used in our project and then use the Python expression `uberons['lung']` to retrieve the URI for that term.

Among the disadvantages of this approach is the need to first look up individual terms using external resources and then copy their labels and URIs into source code. This manual step is clunky, error-prone, and difficult to reuse from one project to the next.

To solve this issue our tool generates a Python class for each ontology. For each term defined in an ontology, the ontology's Python class has an attribute named using the term's label, with underscores substituted for spaces where applicable. The value of a term's attribute is a string representation of its URI.

Following the running example, an import statement at the top of a Python source file brings in a Python class for `UBERON` that allows one to get the URI for any term in the UBERON ontology. The shows an interactive Python session using the URIs for two terms.

```
from obo import UBERON
>>> UBERON.lung
'http://purl.obolibrary.org/obo/UBERON_0002048'
>>> UBERON.lobe_of_liver
'http://purl.obolibrary.org/obo/UBERON_0001113'
```

This tool simplifies using terms from OBO Foundry ontologies within Python programs. We have used it so far with a handful of ontologies. It is under active development. Planned improvements include handling imports within ontology files, and automating the generation of Python classes for each ontology when the ontologies are updated.

A draft implementation is available in a public GitHub repository:
https://github.com/jonathanbona/obof-py