OWL Ontologies in SNePS *
SNeRG Technical Note 41

Jonathan Bona
Department of Computer Science and Engineering
201 Bell Hall
University at Buffalo, The State University of New York
Buffalo, NY 14260-2000
jpbona@cse.buffalo.edu

January 4, 2008

1 Introduction

Many ontology editing and reasoning systems can import and export entire ontologies as OWL (Web Ontology Language) files, and many existing ontologies are implemented in OWL. Our work on the interdisciplinary psychiatric ontologies project involves the use of SNePS to reason about medical ontologies. The main subject of this Technical Note is the SNePS OWL Importer (SNOWLi) - a tool which imports OWL files into SNePS [Shapiro and The SNePS Implementation Group, 2007].

The task of importing an OWL file into SNePS can be subdivided into two general tasks: parsing the OWL file, and creating a SNePS representation of the file's contents. In this paper we briefly introduce OWL and discuss the system used to process the OWL files. We then describe the SNePS representation and conclude with instructions for using SNOWLi from within SNePS.

2 Background

2.1 OWL and RDF

OWL is the Web Ontology Language. It is actually a family of three different languages: OWL Lite, OWL DL, and OWL Full (listed here in order of increasing expressibility) [McGuinness and van Harmelen, 2004]. We are concerned mainly with OWL DL, which is a description logic. All versions of OWL are based on RDF.

RDF is the Resource Description Framework [Klyne and Carroll, 2004]. It is based on the Extensible Markup Language, XML [Bray et al., 2006]. An RDF file can be viewed as a sequence of statements about resources. Each statement is a triple with a subject, an object, and a predicate. Each resource has a Uniform Resource Identifier, or URI, which uniquely identifies it. Every resource in OWL/RDF is either a class or a property. This includes classes such as *rdfs:Resource, rdfs:Class, owl:Class, and rdfs:Property* and properties such as *rdfs:subClassOf and rdf:type.*

*This work is supported by the grant "UB Task Force for ontology-based IT support for large scale field studies in Psychiatry" from the John R. Oishei Foundation. The Foundation's mission is to enhance the quality of life for Buffalo area residents by supporting education, healthcare, scientific research and the cultural, social, civic and other charitable needs of the community. The Foundation was established in 1940 by John R. Oishei, founder of Trico Products Corporation.

2.2 OWL Example from Basic Formal Ontology

The Basic Formal Ontology (BFO) is a top-level ontology that has been implemented in OWL. The following is BFO's definition of the resource *bfo:Entity* as an *owl:Class*. It is taken from the OWL implementation of Basic Formal Ontology [Stenzhorn et al., 2007]. Line numbers are not a part of the original definition but have been added here to aid discussion.

```
1:     <owl:Class rdf:about="&bfo;Entity">
2:         <rdfs:label rdf:datatype="&xsd;string">entity</rdfs:label>
3:         <owl:unionOf rdf:parseType="Collection">
4:             <owl:Class rdf:about="&snap;Continuant"/>
5:             <owl:Class rdf:about="&span;Occurrent"/>
6:         </owl:unionOf>
7:     </owl:Class>
```

The definition of this *owl:Class* begins on line 1 here with the opening of the `<owl:Class .. >` tag, and ends on line 7 with the closing tag, `</owl:Class>`. The `rdf:about` attribute is used within the `owl:Class` tag to provide the namespace and id which will uniquely identify this resource. The value of the attribute, `"&bfo;Entity"` indicates that this resource has the id `Entity` in the `bfo` namespace.

Line 2 defines an `rdfs:label` whose XML Schema datatype is `xsd:string`. The value of this label is the string `"entity"`. This label provides a human-friendly way of representing the resource's name/id.

Briefly, lines 3-6 say that *bfo:Entity* is the union of two other classes: *snap:Continuant* and *span:Occurrent*.

Another OWL example taken from a human disease ontology is given in Section 5 of this report.

2.3 AllegroGraph

Franz Inc's AllegroGraph [Franz Incorporated, 2007] is a system that manipulates RDF graphs. It will take as input an RDF file, parse that file, and build an indexed database of triples representing the statements in that file. AllegroGraph provides interfaces to this database which can be accessed using any of several programming languages including Lisp.

As discussed above, the OWL implementation of Basic Formal Ontology contains the statement that a *bfo:Entity* is an *owl:Class*. This information is represented in the file as:

```
...
<owl:Class rdf:about="&bfo;Entity">
...
</owl:Class>
...
```

AllegroGraph will produce a subject/predicate/object triple in its database to represent this information. That triple can be printed as follows:

```
<<http://www.ifomis.org/bfo/1.1#Entity>,
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
  <http://www.w3.org/2002/07/owl#Class> >
```

The subject of this triple is *bfo:Entity* - the *Entity* resource in the *bfo* namespace. "bfo" is the short name for the namespace which has as its URI (Uniform Resource Identifier) `http://www.ifomis.org/bfo/1.1`. The URI `http://www.ifomis.org/bfo/1.1#Entity`, which appears as the subject of the triple printed above, uniquely identifies the *bfo:Entity* resource.

The predicate of this triple is the property *rdf:type*. As with the subject, the full URI for the property is shown here. It is a combination of the URI for the *rdf* namespace (`http://www.w3.org/1999/02/22-rdf-syntax-ns`) and the name of the resource ("type").

The object of the triple is the class *owl:Class*. The URI for the *owl* namespace is `http://www.w3.org/2002/07/owl`. The name of the resource within the namespace is "Class".

This triple represents the statement that the resource *bfo:Entity* is an instance of the class *owl:Class*.

2.3.1 Usage of AllegroGraph by SNOWLi

SNOWLi uses AllegroGraph to parse an input file, producing a database of triples. It then translates each triple into an appropriate SNePSLOG representation and asserts it to SNePS using SNePSLOG's *tell* function.

3 SNePSLOG Representation

3.1 Resources and Namespaces

When an OWL file is imported using SNOWLi, every resource encountered in the file has asserted about it in SNePS that it is a resource, that it has a particular id and a particular namespace, and that its namespace has a particular URI. This information is represented using two proposition-valued terms:

```
;;; The proposition that [res]  is a resource with id [id] in namespace [ns]
Resource( res, id, ns)
;;; The proposition that [ns] is a namespace with URI [uri]
Namespace( ns, uri)
```

For the first argument to `Resource`, we use the standard short name for the resource: the name of the resource's namespace, followed by a colon, followed by the id within that namespace of the resource.

For example, the *rdf:type* property is represented as follows:

```
Resource(rdf:type,type,rdf).
Namespace(rdf,"http://www.w3.org/1999/02/22-rdf-syntax-ns#").
```

These are the propositions that `rdf:type` is a resource with id `type` in the *rdf* namespace, and that `rdf` is a namespace with the URI: `http://www.w3.org/1999/02/22-rdf-syntax-ns#`.

3.2 Triples as Propositions

Each subject/predicate/object triple is translated into a SNePSLOG representation of the proposition it represents and asserted in SNePS. Each property that appears as the predicate of a triple has its own unique predicate in SNePSLOG.

The SNePSLOG representation of the proposition that the *rdf:type* of *bfo:Entity* is *owl:Class* is:

```
rdf:type(bfo:Entity,owl:Class).
```

Relevant information about each participating resource (and its namespace) is asserted prior to its being used as part of any other assertion.

3.3 SNePSLOG Mode 2

Predicates for OWL/RDF Properties are automatically generated using SNePSLOG Mode 2. Unlike Mode 3, Mode 2 does not require frames for predicates to be defined before using them. Unlike Mode 1, Mode 2 represents each relation using a unique set of slot names.

In SNePSLOG Mode 2, frames are automatically generated when they are used for the first time. The labels of arcs participating in the frame follow the convention outlined on page 58 of the SNePS 2.7 User Manual [Shapiro and The SNePS Implementation Group, 2007]:

"In this mode, every proposition of the form P(x1, . . . , xn) is represented by a node of the form $\{<|relP|, \{P\}>, <|rel-arg\#P1|, \{x1\}>, ..., <|rel-arg\#Pn|, \{xn\}>\}$"

Thus the proposition *rdf:type(bfo:Entity,owl:Class)* is represented by a frame with a slot labelled "|rel rdf:type|" filled by *rdf:type*, a slot labelled "|rel-arg#rdf:type1|" filled by *bfo:Entity*, and a slot labelled "|rel-arg#rdf:type2|" filled by *owl:Class*.

```
: |rdf:type|(|bfo:Entity|,|owl:Class|).

  WFF1!:  rdf:type(bfo:Entity,owl:Class)
 CPU time : 0.00

: %(describe m1)
(M1! ( REL rdf:type rdf:type)
        (REL-ARG#rdf:type1 bfo:Entity)
        (REL-ARG#rdf:type2 owl:Class))

  WFF1!:  rdf:type(bfo:Entity,owl:Class)
 CPU time : 0.00
```

This version of SNOWLi cannnot be used with SNeRE because SNeRE requires SNePSLOG to be used in Mode 3. One way of handling this would be to have the Lisp code for SNOWLi generate and define a unique frame for each property it encounters before using actually using it.

4 Using SNOWLi

This section describes how to load SNOWLi and use it with SNePS. The functions that comprise SNOWLi are in the package `snowli`. The `import-owl` function is used to import OWL files. It is defined in both the `snowli` and `snepslog` packages.

These instructions for using SNOWLi assume that SNePS is already running. Before SNOWLi's `import-owl` function can be used, the `snowli` file must be loaded.

4.1 Location on CSE Servers

SNOWLi is currently located on CSE Department servers in the directory
`/projects/snwiz/Libraries/`

This directory has a subdirectory `Examples/owl-files/`, which contains several OWL files that can be used for testing/demonstrating SNOWLi. The `disease_ontology.owl` file is quite large. Loading it into SNOWLi fails due to resource limitations on the server. Any of the other OWL files in the `Examples/owl-files/` directory should work.

4.2 Loading SNOWLi

To load SNOWLi from the Lisp prompt, load the `snowli` file:

```
cl-user(1): (load "/projects/snwiz/snowli")
```

This file defines the *snowli* package and loads the other files in the package. It also initializes AllegroGraph and loads SNePS if SNePS is not already running. SNePS will not be re-loaded if it is already running.

When SNOWLi has finished loading, you will be returned to the Lisp prompt. Type `"(snepslog)"` to go into SNePSLOG.

As shown in the demo (Appendix A), SNOWLi can be loaded from within SNePSLOG by calling the *cl::load* function:

```
: ^(cl:load "/projects/snwiz/snowli")
```

4.3 import-owl

The *import-owl* function takes as its only argument the name of the file to import. It loads and parses the file using AllegroGraph and then converts the resulting triples to their SNePSLOG representation, asserting each resulting proposition to SNePS via the tell/ask interface.

To import a file with the filename "example.owl", from within SNePSLOG, type:

```
: ^(import-owl "example.owl"}
```

Once *import-owl* has returned, each of the statments contained in the original OWL/RDF file is represented as one or more propositions in SNePS.

4.4 Demo File

The file *owl-demo.snepslog* in the `/projects/snwiz/Libraries/Examples/` directory demonstrates the basic functionality of the tool. To use it, start SNePS and enter SNePSLOG. Type

```
: demo "/projects/snwiz/Libraries/Examples/owl-demo.snepslog"
```

Appendix A is the transcript of a run of this demo file.

5 OWL Example from Human Disease Ontology

The OWL excerpt shown in this section is taken from the OBO (Open Biomedical Ontologies) Foundry's Human Disease domain ontology. The OBO Foundry aims to assemble a collection of interoperable biomedical ontologies, and has made a number of such ontologies available for download from their website (obo-foundry.org) [Smith et al., 2007].

OBO has its own flat file format in which ontologies are expressed. There is also a standard mapping from OBO to OWL, and various tools for performing this translation[1]. This section presents and discusses excerpts from an OBO human disease ontology that is available for dowload as a translated OWL file from `http://www.berkeleybop.org/ontologies/obo-all/disease_ontology/disease_ontology.owl`.

The XML-based OWL file is much larger than the OBO flat file from which it is derived. The relevant portion of the original OBO file is provided for comparison in Appendix B.

5.1 Definition of the term *gallbladder disease* in *disease_ontology.owl*

The following OWL text is a part of the definition of the term *gallbladder disease* in the disease ontology. The full translation of this term into an *owl:Class* is nearly five hundred lines long. The original file does not have line numbers.

```
 33:  <owl:Class rdf:about="http://purl.org/obo/owl/DOID#DOID_0000000">
 34:    <rdfs:label xml:lang="en">gallbladder disease</rdfs:label>
 35:    <oboInOwl:hasOBONamespace>mixed_disease_ontology</oboInOwl:hasOBONamespace>
 36:    <oboInOwl:hasDbXref>
 37:      <oboInOwl:DbXref>
 38:        <rdfs:label>GeneRIF:14567398</rdfs:label>
 39:        <oboInOwl:hasURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
 40:            http://purl.org/obo/owl/GeneRIF#GeneRIF_14567398
 41:         </oboInOwl:hasURI>
 42:      </oboInOwl:DbXref>
 43:    </oboInOwl:hasDbXref>
 44 - 523:          ....
524:  </owl:Class>
```

[1] `http://www.bioontology.org/wiki/index.php/OboInOwl:Main_Page`

Line 33 uses *rdf:about* to specify the id of this *owl:Class* as `http://purl.org/obo/owl/DOID#DOID_0000000`. The portion of the URI preceding the '#' is the URI for the *DOID* namespace. *DOID_0000000* will uniquely identify gallbladder disease in the *DOID* namespace.

Line 34 uses *rdfs:label* to provide a human-readable label ("gallbladder disease") for *DOID_0000000*.

Line 35 says that the *gallbladder disease* term is in the OBO namespace *mixed_disease_ontology*

Lines 36-43 use the *DbXref* relation in the *oboInOwl* namespace to say that *gallbladder disease* has an analogous term in another ontology (a cross-reference).

Lines 37-42 describe the cross-referenced term.

Line 38 gives the cross-referenced term a label: "GeneRIF:14567398".

Lines 39-41 specify the URI for *GeneRIF:14567398*.

Line 39 says that *GeneRIF:14567398* has a URI and that the URI's XML Schema datatype is *anyURI*.

Line 40 gives *GeneRIF:14567398*'s URI as `http://purl.org/obo/owl/GeneRIF#GeneRIF_14567398`.

5.2 *gallbladder_disease* in SNePSLOG

This section shows the relevant propositions that result when the excerpt shown above of the *gallbladder disease* portion of the human disease OWL file is imported into SNePS.

```
p1:    rdf:type(http://purl.org/obo/owl/DOID#DOID_0000000,owl:Class)
p2:    rdfs:label(http://purl.org/obo/owl/DOID#DOID_0000000,gallbladder disease)
p3:    oboInOwl:hasOBONamespace(http://purl.org/obo/owl/DOID#DOID_0000000,mixed_disease_ontology)

p4:    oboInOwl:hasDbXref(http://purl.org/obo/owl/DOID#DOID_0000000,anon1)
p5:    rdf:type(anon1,oboInOwl:DbXref)
p6:    rdfs:label(anon1,GeneRIF:14567398)
p7:    oboInOwl:hasURI(anon1,http://purl.org/obo/owl/GeneRIF#GeneRIF_14567398)
```

p1 comes from line 33 in the OWL file and says that *DOID_0000000* is an *owl:Class*.

p2 comes from line 34 in the OWL file and says that *DOID_0000000* has the label *gallbladder disease*.

p3 comes from line 35 in the OWL file and says that *DOID_0000000* has the OBO namespace *mixed_disease_ontology*.

p4 comes from line 36 in the OWL file and says that *DOID_0000000* has a cross reference, which is here called *anon1*. The cross reference is treated by AllegroGraph as an anonymous class because, while a label is specified for it, no id is specified. SNOWLi generates a unique term for each anonymous class it encounters.

p5 comes from line 37 in the OWL file and says that *anon1*'s type is *oboInOwl:DbXref*.

p6 comes from line 38 in the OWL file and says that *anon1* has the label *GeneRIF:14567398*.

p7 comes from lines 39, 41 in the OWL file and says that *anon1* has the URI `http://purl.org/obo/owl/GeneRIF#GeneRIF_14567398`.

5.2.1 Propositions about Resources and Namespaces

The following propositions were also asserted by SNOWLi while importing the gallbladder information from the OWL file. They describe the various resources and namespaces mentioned in the file.

```
r1:    Resource(oboInOwl:hasDefaultNamespace,hasDefaultNamespace,oboInOwl)
r2:    Resource(owl:Class,Class,owl)
r3:    Resource(oboInOwl:hasOBONamespace,hasOBONamespace,oboInOwl)
r4:    Resource(oboInOwl:hasDbXref,hasDbXref,oboInOwl)
r5:    Resource(oboInOwl:DbXref,DbXref,oboInOwl)
r6:    Resource(oboInOwl:hasURI,hasURI,oboInOwl)
r7:    Resource(rdf:type,type,rdf)
r8:    Resource(rdfs:label,label,rdfs)

n1:    Namespace(bfo,http://www.ifomis.org/bfo/1.1#)
```

```
n2:    Namespace(owl,http://www.w3.org/2002/07/owl#)
n3:    Namespace(xsd,http://www.w3.org/2001/XMLSchema#)
n4:    Namespace(xs,http://www.w3.org/2001/XMLSchema#)
n5:    Namespace(oboInOwl,http://www.geneontology.org/formats/oboInOwl#)
n6:    Namespace(rdf,http://www.w3.org/1999/02/22-rdf-syntax-ns#)
n7:    Namespace(rdfs,http://www.w3.org/2000/01/rdf-schema#)
```

Appendix A: Demo

This transcript shows the basic functionality of SNOWLi. It has been edited only to remove irrelevent and lengthy output produced by Lisp when loading files.

After importing the owl file, *list-wffs* displays around 200 wffs. Most of these were edited out and replaced with an elipsis.

The owl file loaded by this demo (`FirstPOExample.owl`) was created by Professor Stuart Shapiro as a course demonstration using the Protégé[Gennari et al., 2002] system with OWL Plugin. The file defines several subclasses of the class *Animal* and the relations between them, as well as some individuals that instantiate the classes. For instance, it contains the information that: *Person* is a subclass of *Animal*, that *Man* and *Woman* are subclasses of *Person*, that *male* and *female* are instances of the class *Sex*, and that the relation *hasSex* holds between *Woman* and *female*.

```
: demo "/projects/snwiz/Libraries/Examples/owl-demo.snepslog"

File /shared/projects/snwiz/Libraries/Examples/owl-demo.snepslog is now the source of input.

 CPU time : 0.00

: ;;; Load snowli
^(cl:load "/projects/snwiz/Libraries/snowli")

...

 CPU time : 2.58

:
;;; Import a file
^( import-owl "/projects/shapiro/CSE663/ProtegeKBs/FirstProtegeOwlExample/FirstPOExample.owl")

 CPU time : 2.79

:
;;; At this point, all the statements in the OWL file
;;; have been imported and asserted in SNePS

list-wffs
  wff193!:  rdf:type(http://www.owl-ontologies.com/Ontology1193158209.owl,owl:Ontology)
  wff192!:  Resource(owl:Ontology,Ontology,owl)
  wff191!:  rdf:type(Ontology1193158209:Parent,owl:Class)

  ...

  wff2!:  Namespace(Ontology1193158209,http://www.owl-ontologies.com/Ontology1193158209.owl#)
  wff1!:  Namespace(rdfs,http://www.w3.org/2000/01/rdf-schema#)


 CPU time : 0.03

:
;;; Which things are Persons?
rdf|:|type(?x,Ontology1193158209|:|Person)?
```

```
  wff159!:  rdf:type(Ontology1193158209:Sally,Ontology1193158209:Person)
  wff124!:  rdf:type(Ontology1193158209:Pete,Ontology1193158209:Person)
  wff39!:  rdf:type(Ontology1193158209:Lucy,Ontology1193158209:Person)
  wff34!:  rdf:type(Ontology1193158209:Tom,Ontology1193158209:Person)

 CPU time : 0.00

:
;;; Which Resources have the id Sally
Resource(?x,Sally,?y)?

  wff26!:  Resource(Ontology1193158209:Sally,Sally,Ontology1193158209)

 CPU time : 0.00

:
;;; For all x
;;; If x's rdf:type is Person (in this ontology)
;;; AND x is a Resource with id p in namespace y,
;;; THEN p is the name of a person
all(x,p,y)( {rdf|:|type(x,Ontology1193158209|:|Person), Resource(x, p, y)} &=> PersonName(p)).

  wff194!:  all(y,p,x)({Resource(x,p,y),rdf:type(x,Ontology1193158209:Person)} &=> {PersonName(p)})

 CPU time : 0.00

:
;;; What are the names of Persons?
PersonName(?p)?

  wff198!:  PersonName(Sally)
  wff197!:  PersonName(Pete)
  wff196!:  PersonName(Lucy)
  wff195!:  PersonName(Tom)

 CPU time : 0.01

:


 CPU time : 0.00

:

End of /shared/projects/shapiro/PsychO/SNOWLi/owl-demo.snepslog demonstration.


 CPU time : 5.41

:
```

Appendix B: *gallbladder disease* in OBO file *disease_ontology.obo*

The following is an excerpt from the OBO file `disease_ontology.obo`. The OWL version of this file is used in the discussion and examples in Section 5 of this report, starting on page 5.

gallbladder disease is the first term defined in the file and its definition ends on line 91. Line numbers do not appear in the original file. Lines 10-89 are omitted here. They are similar to line 9 in that they all are cross references for *gallbladder disease*.

```
 1:    format-version: 1.0
 2:    default-namespace: mixed_disease_ontology
 3:    saved-by: ozborn
 4:    auto-generated-by: Human Disease Ontology Tools, version 0.8 Beta
 5:
 6:    [Term]
 7:    id: DOID:0000000
 8:    name: gallbladder disease
 9:    xref: GeneRIF:14567398
 ...
90:    is_a: DOID:27 ! Digestive System Disorders
91:    is_a: DOID:43 ! disease of organ with cavitated organ parts
```

References

[Bray et al., 2006] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., Yergeau, F., and Cowan, J. (2006). *Extensible Markup Language (XML) 1.1 (Second Edition)*. Published online on August 16th, 2006 at `http://www.w3.org/TR/2006/REC-xml11-20060816/`.

[Franz Incorporated, 2007] Franz Incorporated (2007). *AllegroGraph 2.2.4: Introduction*. Published online at `http://agraph.franz.com/allegrograph/doc/lisp/agraph-introduction.html`.

[Gennari et al., 2002] Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubzy, M., Eriksson, H., Noy, N. F., and Tu, S. W. (2002). The evolution of protégé: An environment for knowledge-based systems development.

[Klyne and Carroll, 2004] Klyne, G. and Carroll, J. J. (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax*. Published online on February 10th, 2004 at `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

[McGuinness and van Harmelen, 2004] McGuinness, D. L. and van Harmelen, F. (2004). *OWL Web Ontology Language Overview*. Published online on February 10th, 2004 at `http://www.w3.org/TR/2004/REC-owl-features-20040210/`.

[Shapiro and The SNePS Implementation Group, 2007] Shapiro, S. C. and The SNePS Implementation Group (2007). *SNePS 2.7 User's Manual*. Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY. Available as `http://www.cse.buffalo.edu/sneps/Manuals/manual27.pdf`.

[Smith et al., 2007] Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., the OBI Consortium, Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., H, R., Scheuermann, Shah, N., Whetzel, P. L., and Lewis, S. (2007). The obo foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11).

[Stenzhorn et al., 2007] Stenzhorn, H., Spear, A., Grenon, P., and Ruttenberg, A. (2007). *Basic Formal Ontology (BFO)*. Published online on September 18th, 2007 at `http://www.ifomis.org/bfo/1.1`.