The Foundational Model of Anatomy in OWL: experience and perspectives

Christine Golbreich[1], Songmao Zhang[2], Olivier Bodenreider[3]

[1]LIM, University Rennes 1, 35043 Rennes, France
Christine.Golbreich@univ-rennes1.fr

[2]Institute of Mathematics, Chinese Academy of Sciences, Beijing 100080, P.R. China
smzhang@math.ac.cn

[3]U.S. National Library of Medicine
8600 Rockville Pike, MS 43, Bethesda, Maryland 20894, USA
olivier@nlm.nih.gov

Abstract. This paper reports our experience with OWL for the Foundational Model of Anatomy (FMA). We show that converting the FMA from Protégé into OWL DL was possible, with most features of the original FMA captured. The conversion relies on translation and enrichment rules, implemented with flexible options. Unsurprisingly, reasoning with OWL proved to be a real challenge, due to the sheer size and complexity of the FMA. As the entire FMA in OWL DL raised inference problems hard to solve in terms of time and memory, an incremental approach was adopted. A number of various smaller versions that Racer could handle were successfully tested. Some inconsistencies were identified and some classes reclassified. The analysis of the results obtained so far shows the benefits of representing the FMA in OWL and, more generally, the usefulness of DLs reasoning techniques for large-scale biomedical ontologies shared on the Web.

1 Introduction

As OWL is now the W3C recommended standard for ontologies, converting frame-based ontologies to OWL becomes an important need. Once converted to OWL, ontologies currently developed with frames become virtually interoperable with other ontologies and can be used as resources for the Semantic Web. Also of interest is OWL higher expressiveness and the powerful reasoning services associated to its underlying description logic. There is now a large and increasing library of ontologies developed in Protégé [10]. This trend of conversion can already be noted in biomedicine, where large terminologies are migrated to OWL e.g. the Medical Subject Headings (MeSH) [7], the Gene Ontology™ [8] and the National Cancer Institute Thesaurus [9]. The frame-based ontology under study is the Digital Anatomist Foundational Model of Anatomy (FMA), which was converted from

Protégé 2.1 to OWL DL. The FMA is the most complete ontology of human canonical anatomy [2]. The version used in this study, dated of July 2004, contains 70,169 concepts and more than 1.5 million relations. The FMA was selected for several reasons. From a biomedical perspective, anatomy plays a prominent role in biomedicine. As its authors claim, the FMA is "a *reference* ontology in biomedical informatics for correlating different views of anatomy, aligning existing and emerging ontologies in bioinformatics ..." [2]. Anatomy, together with Gene and Disease reference ontologies, constitute the backbone of the future Semantic Web for Life Sciences. From a knowledge representation perspective, evaluating OWL and DL inference techniques for a large-scale biomedical ontology such as the FMA was attractive. Indeed, the sheer size of the FMA makes it a real challenge for DLs reasoning techniques and OWL tools, both for editors (e.g., Protégé OWL) and reasoners (e.g., Racer). As a main goal of the FMA conversion to OWL was to evaluate the advantages of DLs over frames and the possible benefits obtained from reasoning with OWL, the language selected for the conversion is OWL DL, in contrast to OWL Full proposed in [4]. Indeed, OWL DL provides completeness and decidability of the interesting reasoning problems (satisfiability and subsumption) supporting consistency checking and automatic classification. OWL DL reasoners are available e.g. Racer, Pellet, while OWL Full is undecidable, offers no computational guarantees and lacks any suitable reasoner.

The method used to automatically convert the FMA from Protégé 2.1 into OWL DL is first presented (§2). The experience of reasoning with OWL and its results are next reported (§3). The choices of conversion and future perspectives for the FMA are then discussed. Lessons learnt from this experiment may be generalized for large-scale ontologies of the Semantic Web (§4).

2 Conversion to OWL DL

As DLs and frames share the same object paradigm, it might be thought that converting a Protégé ontology into OWL is straightforward and could be achieved by a simple export function mapping Protégé primitives to OWL constructs. But, the export function from Protégé to OWL did not work for the FMA, neither in one step (i.e., directly), nor in two steps (i.e., from database to text then to OWL). Besides, even if it had worked, it would be ineffective, mainly for two reasons:

First, migrating a frame-based ontology to OWL requires not only a syntactic "translation", but also a semantic "enrichment". Indeed, *property restrictions* such as `allValuesFrom` or `someValuesFrom` contained in the OWL axioms cannot be directly derived from the original frame representation, where they are not specified. Additionally, classification strongly relies on the classes *logical* definitions. A reasoner (e.g., Racer) can only automatically classify classes under "defined" classes[1] – i.e., classes with at least one necessary and sufficient condition. Necessary *and*

[1] except if a property has a domain (or range) that is a primitive class, which can coerce classes to be reclassified under the primitive class that is the domain or range of the property (§4).

sufficient conditions cannot be derived directly from the Protégé model, because in frames, all slots at class with a specified range or value, are considered as a set of necessary conditions. Specifying *defined classes* is a major "enrichment" of the ontology.

The second reason is that the FMA in Protégé makes an extensive use of metaclasses[2], which are not allowed in OWL DL. Each anatomical entity is modeled both as a metaclass and as an instance of a metaclass. This was the "technical solution for enabling the selective inheritance of attributes" in Protégé [2] (see §4). For example, Heart is defined as a metaclass, subclass of Organ_with_cavitated_organ _parts, itself subclass of Organ, and as its instance. At the meta level, Heart inherits all the slots, facets, characteristics (range, cardinality, inverse etc.) of its superclassses, For example, Heart inherits from Organ the slot bounded_by with multiple values allowed in Surface_of_organ, the slot arterial_supply with multiple values allowed in the classes Artery, Arteriole_Arterial_plexus or Set_of_arteries, the slot venous_drainage with multiple values in the class Subdivision_of_venous_tree_organ or Organ_part_tree_structure, etc. (Table 1 Annex). But at the class level, the own slots of Heart are assigned particular values e.g., bounded_by is filled with Surface_of_heart, arterial_supply with Right_coronary_artery and Left_coronary_artery etc. Directly translating metaclasses into OWL would lead to OWL Full, instead of OWL DL. Simply removing metaclasses as suggested in [4] would not be satisfactory neither, since all the knowledge encoded at the metaclasses would be lost.
Therefore, we defined our own method of conversion, which aims at providing the desired enrichments and at capturing the knowledge encoded at metaclasses differently.

2.1 Method of conversion

The migration was achieved from the CLIPS files. The conversion relies on *translation* and *enrichment* rules, implemented with flexible options. The conversion rules are detailed in the Annex.

Translation draws on the structural correspondence between Protégé and OWL constructs. The Protégé class taxonomy defined at meta level is translated into an OWL subclass hierarchy. Template slots defined at the top level are translated into OWL properties with the same features as those specified in Protégé, i.e. same range, inverse, cardinality, etc., simply mapping each of them to the corresponding OWL primitive (Fig. 1). For example, the Protégé single slots 'has_mass' or 'has_boundary', defined with type SYMBOL, allowed values FALSE TRUE, and cardinality 0 1, are simply translated into an owl:DatatypeProperty, with range datatype Boolean, and declared to be an owl: FunctionalProperty. The Protégé multislot constitutional_part defined with type SYMBOL, allowed parents Physical_anatomical_entity and inverse slot constitutional_part_of,

[2] A metaclass is a class whose instances are themselves classes

is translated into an `owl:ObjectProperty` with `(rdfs:range rdf:resource="#Physical_anatomical_entity")` and inverse `(owl:inverseOf rdf:resource="#constitutional_part_of")`.

Protégé slot	OWL property
Type INTEGER, FLOAT, STRING	`DatatypeProperty` with range datatype integer, float and string
Type SYMBOL with allowed values T or F	`DatatypeProperty` with range datatype boolean
Type SYMBOL with allowed values (not T nor F)	`ObjectProperty` with range the enumerated class of all the allowed individuals
Type SYMBOL with allowed parents	`ObjectProperty` with range the union of all the allowed classes
Type INSTANCE with allowed classes	

Fig. 1 Some translation rules for slots

Enrichment, in contrast, introduces new logical features. The enrichment rules were designed to reflect the original underlying principles of the FMA model. Some enrichment rules and the rationale behind them are presented next.

– **Property restrictions**: the choice between universal and existential property restrictions is mainly based on the distinct role of template and own slots in Protégé.

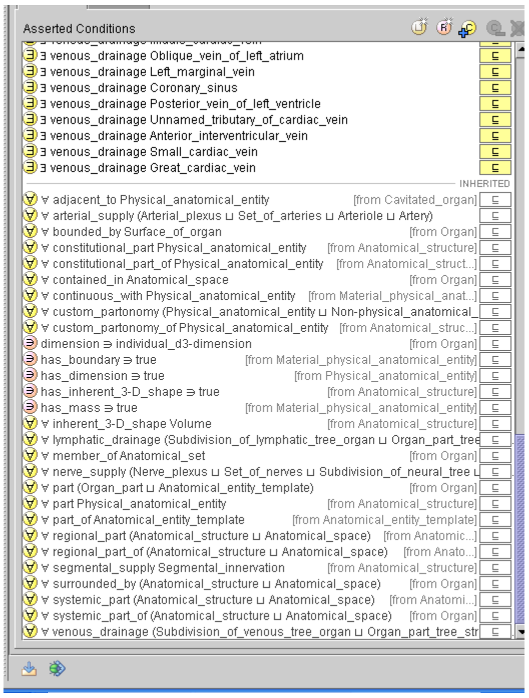
Template slots "specify which slot each member of a class shall have and what the restrictions (facets) on the values of these slots shall be" [2]. Template slots with their constraints are inherited by the subclasses and the instances. Therefore, allowed parents or allowed classes specified for a template slot at metaclass, are converted into *universal* property restrictions `(owl:allValuesFrom)`. In contrast, according to the FMA principle of "canonical anatomy" [2], when a class instantiates a metaclass, the specific values assigned to a template slot inherited as own slot describe the typical canonical structure of the particular anatomical entity in terms of relations that should necessary exist, e.g. in terms of the existing parts composing an organ. Therefore, they are converted into *existential* property restrictions `(owl:someValuesFrom)` (Fig. 2).

Protégé template slot at metaclass	OWL property restriction
with allowed parents or allowed classes C_i	`owl:allValuesFrom` constraint to the union of all the allowed classes C_i enforced on the property when applied to the class
with an allowed value	`owl:hasValue` constraint to the specified value enforced on the property when applied to the class
Protégé own slot at class	`owl:someValuesFrom` constraint to the class C, enforced on the property when applied to the class
with a specific class C as value assigned	
with a specific datatype value assigned	`owl:hasValue` constraint to the specific value, enforced on the property when applied to the class

Fig. 2 Some enrichment rules

For example, the multislot `bounded_by` of the metaclass `Organ` with allowed-parents `Surface_of_organ` is converted into the universal restriction (∀ `bounded_by Surface_of_organ)` on the property `bounded_by` of `Organ`, that is next inherited by its subclass `Heart`. But when `Heart` inherits `bounded_by` as an own slot assigned with the value `Surface_of_heart`, it is converted into the existential restriction (∃ `bounded_by Surface_of _heart`).

Similarly, `venous_drainage` is restricted by a universal restriction inherited from its superclasses, but when `Heart` inherits `venous_drainage` as an own slot assigned with the values `Oblique_vein_of_left_atrium`, `Left_marginal_vein`, `Coronary_sinus`, `Posterior_vein_of_left_ventricle`, `Unnamed_tributary_of_cardiac_vein`, `Anterior_interventricular_vein`, `Small_cardiac_vein` etc. they are converted into `owl:someValuesFrom` restrictions specifying the value constraints on the property for the class `Heart` (Fig.3).



```
Asserted Conditions
∃ venous_drainage Oblique_vein_of_left_atrium
∃ venous_drainage Left_marginal_vein
∃ venous_drainage Coronary_sinus
∃ venous_drainage Posterior_vein_of_left_ventricle
∃ venous_drainage Unnamed_tributary_of_cardiac_vein
∃ venous_drainage Anterior_interventricular_vein
∃ venous_drainage Small_cardiac_vein
∃ venous_drainage Great_cardiac_vein
                                                                    INHERITED
∀ adjacent_to Physical_anatomical_entity                    [from Cavitated_organ]
∀ arterial_supply (Arterial_plexus ⊔ Set_of_arteries ⊔ Arteriole ⊔ Artery)
∀ bounded_by Surface_of_organ                                        [from Organ]
∀ constitutional_part Physical_anatomical_entity   [from Anatomical_structure]
∀ constitutional_part_of Physical_anatomical_entity   [from Anatomical_struct...]
∀ contained_in Anatomical_space                                      [from Organ]
∀ continuous_with Physical_anatomical_entity   [from Material_physical_anat...]
∀ custom_partonomy (Physical_anatomical_entity ⊔ Non-physical_anatomical_
∀ custom_partonomy_of Physical_anatomical_entity   [from Anatomical_struc...]
dimension ∋ individual_d3-dimension                                  [from Organ]
has_boundary ∋ true                 [from Material_physical_anatomical_entity]
has_dimension ∋ true                         [from Physical_anatomical_entity]
has_inherent_3-D_shape ∋ true                     [from Anatomical_structure]
has_mass ∋ true                     [from Material_physical_anatomical_entity]
∀ inherent_3-D_shape Volume                       [from Anatomical_structure]
∀ lymphatic_drainage (Subdivision_of_lymphatic_tree_organ ⊔ Organ_part_tree
∀ member_of Anatomical_set                                           [from Organ]
∀ nerve_supply (Nerve_plexus ⊔ Set_of_nerves ⊔ Subdivision_of_neural_tree ⊔
∀ part (Organ_part ⊔ Anatomical_entity_template)                     [from Organ]
∀ part Physical_anatomical_entity                 [from Anatomical_structure]
∀ part_of Anatomical_entity_template       [from Anatomical_entity_template]
∀ regional_part (Anatomical_structure ⊔ Anatomical_space)   [from Anatomic...]
∀ regional_part_of (Anatomical_structure ⊔ Anatomical_space)   [from Anato...]
∀ segmental_supply Segmental_innervation          [from Anatomical_structure]
∀ surrounded_by (Anatomical_structure ⊔ Anatomical_space)            [from Organ]
∀ systemic_part (Anatomical_structure ⊔ Anatomical_space)   [from Anatomi...]
∀ systemic_part_of (Anatomical_structure ⊔ Anatomical_space)         [from Organ]
∀ venous_drainage (Subdivision_of_venous_tree_organ ⊔ Organ_part_tree_str
```

Fig. 3 Restrictions asserted or inherited on properties of the OWL class `Heart`

– **Equivalent class definition**: a "defined" class has at least one necessary and sufficient condition. At this preliminary step, one slot p is manually selected, and a class A having values $B_1,\ldots,\ B_n$ assigned to its own slot p, is defined as equivalent to the conjunction of all the existential value restrictions on p to the classes B_i and of metaclass and superclass of A (after some optimization). As aggregated objects are often described in terms of their parts and as meronymic relationships play a particularly important role in the FMA, it was chosen to define anatomical entities in terms of their parts. At this first step, the property "constitutional part" was selected, resulting in 570 defined classes. For example, the class `Heart` is defined by:
`Heart ≡ Organ_with_cavitated_organ_parts` ⊓ (∃ `contitutional_part Wall_of_heart`) ⊓ (∃ `Organ_with_cavitated_organ_parts Cavity_of_left_atrium`) ⊓ (∃ `contitutional_part … ) …` ⊓ (∃ `contitutional_part … )` (Fig. 4).

The choice of the property 'constitutional part' was partly motivated by a size issue: constitutional part is well populated in FMA, compared for instance to 'custom partonomy', thus is computationally more significant. But, such a definition is not "semantically" satisfying for all classes, as all the classes cannot be uniformly defined solely in terms of their constitutional parts (the same parts may belong to different structures). But at this step, the priority was to test if Racer classification could be run. Different definitions for the different subtrees, and more complex expressions combining several properties will be next investigated (§ 4).

Metaclasses are converted into ordinary OWL DL classes. Subclass relation between metaclasses and metaclass instantiation are both translated into OWL `subClassOf` axioms. According to the previous rules, range restrictions of a template slot defined at metaclass are represented by *universal* property restrictions, while structural own slots with values assigned at class by *existential* property restrictions. In order to respect “selective inheritance”, own slots such as name, identifiers e.g. `UWDAID, author` etc., with values assigned at class, are converted to OWL *annotations*, preventing their propagation to their instances or subclasses. Each entity of the FMA is thus represented by an OWL DL class, its metaclass and instance definitions been merged. Fig. 4 shows the class `Heart` with its `equivalentClass` definition and its `subClassOf` axioms including universal or existential restrictions, derived from the original metaclass and class definitions of `Heart` in Protégé.

Heart in Protégé

Metaclass

```
(defclass Heart
  (is-a Organ_with_cavitated_organ_parts)
                   …
)
```

Instance of Metaclass

```
([Heart]
  of Organ_with_cavitated_organ_parts
  (constitutional_part
          Wall_of_heart
          Cavity_of_left_atrium
          Cavity_of_right_ventricle
          Cavity_of_left_ventricle
          Right_coronary_artery
          Left_coronary_artery
           …
  (bounded_by
          Surface_of_heart)
  (arterial_supply
          Right_coronary_artery
          Left_coronary_artery)
           …
)
```

Heart in OWL DL

```
<owl:Class rdf:ID="Heart">
 <owl:equivalentClass>
   <owl:Class>
     <owl:intersectionOf rdf:parseType="Collection">
     <owl:Class rdf:about=
      "#Organ_with_cavitated_organ_parts"/>
     <owl:Restriction>
      <owl:onProperty
       rdf:resource="#constitutional_part" />
      <owl:someValuesFrom
        rdf:resource="#Wall_of_heart" />
     </owl:Restriction>
     …
 </owl:equivalentClass>
 <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#bounded_by"/>
      <owl:someValuesFrom
       rdf:resource="#Surface_of_heart"/>
   </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
    <owl:Restriction>
     <owl:onProperty rdf:resource="#arterial_supply"/>
      <owl:someValuesFrom
       rdf:resource="#Right_coronary_artery" />
    </owl:Restriction>
   …
 </owl:Class>
```

Fig. 4 Class Heart in OWL DL, derived from Protégé metaclass and class definitions

The other conversion rules are given in the Annex.

Aware of the arbitrariness of some of these choices, the enrichment rules were designed and implemented with flexible options. This flexibility permitted to automatically generate various OWL files with different flavors, size and computational complexity. Moreover, these options can be easily modified, which is key to the incremental approach adopted for reasoning (§3.1).

2.2 Results

Ignoring laterality distinctions, i.e. classes differing from their parents only by laterality, a subset of about 40,000 concepts and their slot values were extracted for conversion, i.e. 57% of the 70,000 concepts of the original FMA. Applied to this subset, the conversion process described earlier resulted in about 117,000 frames, including 40,000 OWL named classes. More precisely, there are 187 properties and 85 individuals specified in this file. 20 properties correspond to annotation, 19 to datatype and 148 to object properties. There are 107,238 `subClassOf` axioms (38,772 from taxonomy and 3,378 from metaclass instantiation), 39337 classes where 559 are defined by `equivalentClass` axioms. OWL constructors `allValuesFrom`, `someValuesFrom`, `hasValue`, `oneOf`, `unionOf`, `FunctionalProperty`, `SymmetricProperty`, `InverseOf` all occur in the OWL file resulting from the conversion (available at http://mor.nlm.nih.gov/pubs/supp/2005-owled-cg/FMA-constitutionalPartForNS.owl) It took about 15mn to load the FMA OWL file in Protégé OWL in a Windows XP PC with 4GB memory (1h30 with 512Mb).

3 Reasoning with OWL

Reasoning with OWL proved to be a real challenge, due to the sheer size and complexity of the FMA. As the entire FMA in OWL DL raised inference problems hard to solve in terms of time and memory, an incremental approach was adopted.

3.1 Incremental approach

We used Racer (Version 1.7) with the OWL files generated by the conversion process to investigate consistency checking and automatic classification. Launched from Protégé-OWL, the classification failed. Running Racer directly from Rice, we experienced problems related to memory limitation (4GB). Since Racer could not handle the entire FMA OWL file (in fact restricted to 2/3 of the whole FMA), as suggested by the Racer authors, we decided to test smaller versions so as to reduce the size and time issues and investigate eventual errors, adding more features incrementally. First, a FMA OWL version with all classes but without any properties was checked to test if the taxonomy alone could be successfully classified. Then, we added equivalent class definitions using only one property to test if the ontology with defined classes could pass Racer. Next, we successively introduced, step by step, object properties, annotation properties, datatype properties, and finally object properties used for attributed slots. When properties are introduced in partial versions, the conversion rules described previously are applied. For example, a small version where the object property `bounds` and its inverse `bounded_by` are introduced, includes for each class having these properties specified, the subclass axioms containaing the corresponding existential and universal restrictions of the properties `bounds` and `bounded_by`.

3.2 Results

Racer passed the first test: the classification of the FMA OWL version without any properties was successful, taking about 25 minutes with 512Mo on a Pentium 4. Then, the classification with "defined" classes described by the conjunction of the existential restrictions on the `constitutional_part` or `custom_partonomy` property as necessary and sufficient condition was also successful. Next, various versions were generated with all classes but containing a limited number of properties. Depending on the properties introduced, the tests were successful or not. Some results are summarized below:

Reasoning with Racer (version 1.7) was *successful* for the following partial versions:

- Ontology with only the class hierarchy defined but without any property.
- Ontology with defined classes (based on `constitutional_part`).
- Ontology with defined classes (based on `constitutional_part` and `constitutional_part_of`).
- Ontology with defined classes and annotation properties.
- Ontology with defined classes, annotation properties, and all datatype properties.
- Ontology with defined classes and primitive classes with restrictions on the property `branch_of` in subClassOf axioms.
- Ontology with defined classes and primitive classes with restrictions on the property `arterial_supply` in subClassOf axioms.
- Ontology with defined classes and primitive classes with restrictions on the property `2D_part` in subclass axioms.
- Ontology with defined classes and primitive classes with restrictions on the property `bounds` and its inverse `bounded_by` in subclass axioms.
- Ontology with defined classes and primitive classes with restrictions on properties `dimension` and `has_physical_state` in subclass axioms.
- Ontology with primitive classes with restrictions on attributed slot `location` and all slots used in `location` (e.g., `related_object`, etc.).
- Ontology with primitive classes with restrictions on attributed slot `attributed_part` and all slots used in `attributed_part` (e.g., related_part, etc.)

Reasoning with Racer (version 1.7) *failed* for:

- Ontology with defined classes and annotation properties, added with primitive classes with restrictions on all the object properties in subclass axioms.
- Ontology with defined classes and primitive classes with restrictions on all object properties in subclass axioms.
- Ontology with primitive classes with restrictions on the property `branch_of` and on its inverse in subclass axioms.
- Ontology with Subclass axioms with restrictions on the property `continuous_with`, declared symmetric.

The reasons for failure are not easy to analyze. For instance, Racer was successful with equivalent class axioms and subclass axioms with restrictions on the property `bounds` and its inverse `bounded_by`, but failed for subclass axioms with restrictions on the property `branch_of` and its inverse `branch`, while it was successful with `branch_of` alone (without its inverse). This experience shows that the sheer size of the FMA is not the only issue. The results of reasoning with OWL are related to several factors. The complexity of the generated OWL ontology, due to the OWL constructors used (e.g. `oneOf`), the presence of `inverseOf` axioms or "global" axioms, and the algorithms implemented and optimization techniques of the reasoners, are certainly critical issues for the FMA.

3.3 Benefits

Although problems with computational resources occurred for reasoning with the whole FMA in OWL DL, Racer could handle various less complex versions, which enabled to detect *inconsistencies* in the original FMA and to *reclassify* some classes.

No inconsistencies were found in the first versions, but when datatype properties were added several inconsistencies were identified. 113 classes were identified as unsatisfiable by Racer because of opposite boolean values:

– *Inconsistencies from conflicts between metaclass and class definitions in Protégé.* A class assigned with a boolean value in its own slot and which inherits the opposite value from its superclasses, is unsatisfiable in OWL. For example, `Zone_of _cell` is unsatisfiable (hence, all its subclasses) because its own slot `has_mass` was assigned `false` at instance (converted to the restriction `has_mass:false`) while this single-slot had value `true` at its superclass `Material_physical_anatomical_entity` (converted to `has_mass:true`). Other inconsistencies were revealed from the inconsistency of the metaclass and class definitions of an entity. A class *A* subclass of *B* and instance of *C* in FMA, where *B* and *C* have opposite values for a boolean datatype property, e.g. `has_mass`, is unsatisfiable in OWL. For example, `Compartment_subdivision` is defined as a subclass of `Anatomical_cluster`, which is a subclass of `Material_physical_anatomical _entity` (`has_mass:true`). On the other hand, `Compartment_subdivision` is an instance of `Anatomical_space`, which is a subclass of `Non-material_physical_anatomical_entity` (`has_mass:false`).

– *Inconsistencies from global and local conflicting domain or range.* `rdfs:range` (resp. `domain`) restrictions are global. Thus if p has class A' as domain and B' as range, and A has a property p with range B, then B must be a subclass of B' and A must be a subclass of A'. Conflicting definitions of global and local ranges or domains lead to inconsistencies in OWL. For example, `Surface_of_wrist` is unsatisfiable because '2D_part' has an existential restriction to `Anatomic_snuff_box` which is a subclass of `Material_physical_anatomical_entity` (`has_mass:true`), while the range of “2D_part” is `Non-material_physical_anatomical_entity` (`has_mass:false`). These inconsistencies exhibit modeling errors in the original Protégé FMA.

Racer also reclassified some classes. In the ontology including defined classes based on the `constitutional_part` property, 286 classes of the asserted hierarchy were moved within the inferred hierarchy, and some classes were identified to be equivalent. For example, as the two sibling classes `Wall_of_biatrial_part_of_heart` and `Wall_of_biventricular_part_of_heart`, have the same constitutional parts[3] in the original FMA, they became equivalent for this definition. However, the equivalence did not hold anymore when adding other restrictions to these definitions. For example, adding restrictions on the property `constitutional_part_of`, enables to differentiate the two classes, as they are parts of different wholes: `Wall_of_biventricular_part_of_heart` is a `constitutional_part_of` `Biventricular_part_of_heart`, while `Wall_of_biatrial_part_of_heart` is a `constitutional_part_of` `Biatrial_part_of_heart`. Thus, although most of the reclassifications were related to the class definitions in terms of their constitutional parts, it nevertheless shows the power of reasoning with OWL DL.

In conclusion, the results obtained so far show the benefits of OWL DL for the FMA. First, checking the logical consistency of the FMA enabled to find errors that would have probably been missed otherwise. Second, automatically computing the classification hierarchy is another advantage for such a large ontology. As the FMA has been under development at the University of Washington since 1994 and is still evolving, such services are useful for quality assurance purposes.

4 Discussion and perspectives

Converting a large part of the FMA from Protégé into OWL DL was possible. [4] proposes to translate the entire FMA in OWL Full and to delete the metaclasses of the original Protégé FMA so as to use OWL DL in application contexts, arguing that “an OWL-DL representation is possible, but requires to give up some of the original features”. In contrast, we converted a large part of the entire FMA into OWL DL with all the knowledge encoded at its metaclasses, and our conversion still complies with OWL DL constraints in particular, a class is not at the same time an individual. All the direct subclasses, superclasses, template slots, slot-constraints, that were defined in Protégé metaclasses are translated, using OWL DL constructs and axioms. The main transformation that permitted to use OWL DL, is the deletion of the Protégé higher order structure. It was achieved in replacing metaclass instantiations by subclass axioms ([A] of B in Protégé is converted to a `subClassOf` axiom $A \sqsubseteq B$ in OWL). This did not introduce significant changes, because the class and the metaclasses hierarchies were integrated in the original model: “except for its root, all

[3] Fibroelastic_connective_tissue_of_endocardium Fibrocollagenous_sheath_of_cardiac_muscle_tissue Fibroelastic_connective_tissue_of_epicardium

concepts in the Anatomy Taxonomy are subclass of a superclass and also an instance of a metaclass" [2]. In fact, this metaclass construction was introduced in Protégé for different purposes presented in [2] [3]:

(1) First, to model each anatomical entity as a "set of sets", (e.g., `Vertebra` as a set of different types of vertebrae: cervical, thoracic, lumbar, themselves sets of other sets e.g., first,.., fifth lumbar vertebra). A first order language as OWL DL cannot capture this feature. However, the use of the representation of an anatomical entity as a "set of sets" is quite limited in Protégé. In fact, the "members of each of these collections are represented in Protégé as *subclasses* of Vertebra" [2] e.g., "the class `Vertebra` subsumes different collection of vertebrae, cervical, thoracic, and lumbar vertebra", which are further refined into more specialized subclasses.

(2) The metaclass construction has another purpose, "to enforce slot value restrictions" [3]. In frames, a slot inherited can only be refined to subclasses of its initial range. For example, when `Cervical_Vertebra` inherits from `Vertebra` the slot `part_of` with range `Vertebral_Column`, its range must be a subclass of `Vertebral_Column`. Metaclasses were intended to enforce restrictions to other classes, such as class `Cervical_Vertebral_Column`, which is not a subclass of `Vertebral_Column` in the FMA model, but `part_of` it. Thus, thanks to metaclass instantiation, the wanted values are assigned to own slots at class (§ 2.1). This artefact is no more needed in OWL, since it is possible to use `subClassOf` axioms instead, e.g., `∃ part_of Vertebral_Column` for the class `Vertebra` and `∃ part_of Cervical_Vertebral_Column` for its subclass `Cervical_Vertebra`, although `Cervical_Vertebral_Column` is not subsumed by `Vertebral_Column`.

(3) Metaclasses were also intended to specify multiple values specific to each class e.g., specifying that a `Vertebra` has parts `Body_of_vertebra`, `Vertebral_arch`, `Bone_of_vertebra`, etc. In OWL this can be captured by several restrictions such as `(∃ part_of Body_of_vertebra) ⊓ (∃ part_of Vertebral_arch) ⊓ (∃ part_of Bone_of_vertebra)` etc.

(4) Finally, metaclasses are used for specifying metadata such as `name`, `author`, `authority`, `UWDAID`, etc. Assigning vlaues to these "non structural" own slots at metaclass instantiation prevents them from being propagated to their instances or subclasses. In OWL this can be done thanks *annotations*.
In conclusion, thanks to OWL's higher expressiveness, most intended meanings of the Protégé metaclasses can be captured, with the exception of "set of sets", which does not represent a significant loss in our opinion, considering their use in Protégé.

As far as we know, the NCI Thesaurus was one of the largest file in Protégé OWL so far. But it is much smaller and exhibits less complexity than the FMA in OWL. The NCI Thesaurus contains 53,000 frames, including 34,000 classes, 100 properties and 9,000 conditions, while the original FMA contains 70,000 concepts and the converted subset 117,000 frames, including 40,000 OWL classes, 187 properties and about 110,000 axioms. NCI was converted to OWL Lite, while the FMA is represented in OWL DL. No defined class, no `hasValue`, `allValuesFrom` restrictions, nor `unionOf` or enumerated classes `oneOf` are specified in the NCI, while they all occur in the FMA OWL file.

The size and complexity of the FMA in OWL make it a real challenge for DLs systems. It showed that, with the current state of the art of DL inference technology, it might generate inference problems that are hard to solve in terms of time and space resources. Indeed, the main problem was computational. Some optimizations were achieved to reduce the complexity. For example, it was necessary to step down the number of disjunctions generated by the conversion for the domain of properties, which caused Racer – would have any inference system – to run into space problems Interestingly, after optimization, two classes remain in the domain of `location` instead of 1,618 originally [12]. Difficulties also occurred for inverse with existential restrictions. However, Racer could handle various less complex versions of the FMA in OWL DL, detect inconsistencies, and reclassify classes. This experiment was done with Racer version 1.7. As Racer evolves – for example its authors are currently working on optimizations that address the issue of inverse roles – it is worthwhile to make tests with next versions, and also to evaluate the performance of other OWL DL reasoners.

In the future, we would like to improve the current conversion process and to remove some of its limitations:

– First, we suggest adding *disjointness axioms* between sibling primitive classes. Ideally, a classification satisfies the so-called "jointly exhaustive and pairwise disjoint" rule. The inconsistencies reported §3 are mainly based on opposite values of a boolean datatype property and their propagation, but disjointness axioms will most probably lead to identifying more inconsistencies in the FMA.

– Second, we propose using *qualified cardinality restrictions*. We converted structural own slots values by existential property restrictions, mainly for two reasons. On the one hand, the assumption that if a class *A* has a slot *p* filled with values $B_1, B_2 \ldots B_n$ in Protégé (e.g., constitutional part), it means that for every individual of *A*, *p* has at least one value of each class B_i. On the other hand we were faced to the expressiveness limitation of OWL, which does not support qualified cardinality restrictions. However, for example defining restrictions "has Part `someValuesFrom` B_1" and "hasPart `someValueFrom` B_2" is weaker than "hasPart exactly one B_1 and one B_2", as it does not prevent from having several parts of the same B_i. If OWL was extended with qualified cardinality restrictions, more precise definitions might be provided.

– Thirdly, we suggest completing our current class definitions by *closure axioms* [11]. Indeed, existential property restrictions, as well as qualified cardinality restrictions, do not prevent from having values from an unwanted class to be assigned to a given property. For example, adding `allValuesFrom` restrictions to the class $B_1 \sqcup B_2$ would prevent values from B_3 but not from B_1 or B_2 to be assigned as parts, and would coerce values to come *only* from B_1 or B_2.
Qualified cardinality and closure axioms would allow to more faithfully reflect the FMA authors definitions. For example, the equivalent class definition `Left_lung` ≡ `Lung` ⊓ (= 1 `regional_part Upper_lobe_of_left_lung`) ⊓ (= 1 `regional_part Lower_lobe_of_left_lung`) ⊓ (∀ `regional_part Upper_lobe_of_left_lung` ⊔ `Upper_lobe_of_left_lung`) ⊓ etc.

would enable to define a Left Lung as having exactly one left upper lobe, one left lower lobe and only those two lobes as regional parts, or a Right Lung as having exactly one right upper lobe, one middle lobe and one right lower lobe and only those three lobes, reflecting the definitions from the Protégé FMA:

```
([Left_lung] of Lung
   (definition “Lung which consists of the left upper
     lobe and left lower lobe” )
   (regional_part
         Upper_lobe_of_left_lung
         Lower_lobe_of_left_lung)
   …)

([Right _lung] of Lung
   (definition “Lung which consists of the right upper
     lobe, middle lobe and right lower lobe”)
   (regional_part
         Upper_lobe_of_right_lung
         Middle_lobe_of_lung
         Lower_lobe_of_right_lung)
   …)
```

– A major issue is the specification of the *defined classes*. Several possible options might be considered [5]: (1) each class has a single definition, which includes the conjunction of all the qualified property restrictions derived from the values of its own structural slots and attributed relations; (2) each class has a set of several equivalent definitions (3) each class has one preferred definition, the other conditions being simply necessary; (4) there are no a priori “defined” classes but only primitive classes, all axioms expressing only necessay conditions. As the FMA is a “shared reference ontology”, it might be considered that its representation in OWL DL is a first formal specification, to be further refined into more detailed formal specifications for each application, thanks to relevant equivalent class axioms. Currently the biggest challenge for the FMA is certainly the specification of reliable class definitions. Equivalent conditions – single or multiple, default or optional – must be defined in close collaboration with the FMA authors, based on “semantically” correct expressions supporting the unique identification of anatomical entities. For the moment, only one property, `constitutional_part` or `custom_partonomy`, was selected for the equivalent class definitions. This may perhaps be relevant for Organ, while other anatomical entities like Organ part, Cell, or Tissue etc. need different criteria of identification. As all the anatomical entities do not share the same definition, different expression templates should be specified for the different subtrees, e.g. Organ, Cell, etc. Conversion rules should be improved to support arbitrary combinations of properties, constructors, and cardinality restrictions, so as to build specific expressions suited to each subtree.

At this first stage, the conversion aimed at capturing the Protégé FMA model as faithfully as possible, in order to evaluate its original properties. In the future, in addition to the above proposals, we suggest to introduce some changes in the model. For example, the OWL classes used for the Protégé attributed relations might be specified by n-ary relations in an external base related to the ontology. New classes might be introduced such as `Venous_drainage`, `Arterial_Supply` for improving consistency and factorizing reasons. Enumerated classes might be approximated otherwise etc.

An interesting point of discussion is about the choice of OWL DL versus OWL Full for large-scale ontologies such as the FMA, and more generally for domain versus application Web ontologies. The main lessons learnt from this experience is that a possible option for large-scale domain ontologies such as the FMA, designed as “sharable reference ontologies”, is to represent them in OWL DL with only primitive classes, but a library of optional usual class equivalent definitions been provided together. As each particular application, may have different needs, it will remain on the user responsibility to select predefined definitions from the library or to build his own definitions, so as to refine and customize the ontology according to his own needs. For example, the brain MRI images application [6] requires defining some anatomical structures, e.g. gyri, from their boundaries, while another application may need to focus on parts. The advantage of this solution is twofold. First, it would concretely implements the notion of a “Semantic Web reference ontology” specified independently of applications. Second, it allows still benefiting of DLs reasoning services such as consistency checking and classification for both the general reference ontology and the more customized ones. The results inferred from reasoning, even with partial versions, are fruitful to improve the consistency and classification of the global reference ontology. As it is crucial to guarantee the correctness of a reference ontology sharable on the Web, it is more advantageous to convert large domain ontologies to OWL DL than to OWL Full, in spite of some computational issues.

5 Conclusion

Converting the whole FMA from its original frame-based representation into the first order language OWL DL was possible, while capturing most features of the original model in Protégé. Reasoning with OWL proved to be a real challenge, because of the sheer size and complexity of the FMA in OWL. The entire FMA raised computational problems hard to solve in terms of time and space resources, but after some optimizations, various smaller versions were successfully tested with Racer. Several inconsistencies were revealed in the original modeling of the FMA. Some classes of the asserted hierarchy were reclassified; some classes were identified to be equivalent. Although most of them were related to the class definitions in terms of their constitutional parts, it nevertheless shows the power of reasoning with OWL DL. Thus, the results obtained so far demonstrate the advantages of OWL over frames for large-scale domain ontologies such as the FMA and help suggest future additional possible improvements of the FMA. This experiment is only a first step, the conversion rules are still being improved and refined. The resulting ontologies are being tested with RacerPro™ [4] and other reasoners may also be used. Issues with DL reasoner scalability should be carefully investigated.

[4] http://www.franz.com/products/racer/

Acknowledgments

This research was supported in part by the Intramural Research Program of the National Institutes of Health (NIH), National Library of Medicine (NLM), while Songmao Zhang and Christine Golbreich were visiting scholars at the Lister Hill National Center for Biomedical Communications, NLM, NIH. Our thanks to Volker Haarslev and Ralf Möller for their valuable advice and help on DLs and Racer, to Cornelius Rosse, José Mejino and Todd Detwiler for the FMA, and to other helpful recommendations.

References

1. OWL Web Ontology Language 1.0 Reference. http://www.w3.org/TR/owl-ref/
2. Rosse C, Mejino JL, Jr. A reference ontology for biomedical informatics: the Foundational Model of Anatomy. J Biomedical Informatics 2003; 36(6):478-500.
3. Noy N, Musen MA, Mejino JL and Rosse C. Pushing the Envelope: Challenges in a Frame-Based Representation of Human Anatomy. Data and Knowledge Engineering Journal; 2002 48(3):335-359.
4. Dameron O, Rubin DL and Musen AM. Challenges in Converting Frame-Based Ontology into OWL: the Foundational Model of Anatomy Case-Study. Proc. AMIA Annual Symposium 2005: (in press).
5. Golbreich C., Zhang S., Bodenreider O., Migrating the FMA from Protégé to OWL. 8th Protégé Intern. Conf., Madrid 2005:108-111 (Extended version http://www.med.univ-rennes1.fr/lim/doc_115.pdf)
6. Golbreich C., Bierlaire O., Dameron O. , Gibaud B. What reasoning support for Ontology and Rules? the brain anatomy case study. Proceedings of the workshop "OWL Experiences and Directions", Nov 11-12, 2005, Galway, Ireland 2005: electronic proceedings: http://CEUR-WS.org
7. Soualmia L, Golbreich C, Darmoni S. Representing the MeSH in OWL: Towards a semi-automatic migration. Proceedings of the KR 2004 Workshop on Formal Biomedical Knowledge Representation 2004:81-87, available at http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-102/soualmia.pdf
8. Wroe CJ, Stevens R, Goble CA, Ashburner M. A methodology to migrate the Gene Ontology to a description logic environment using DAML OIL. Pacific Symposium Biocomputing 2003:624-35
9. Golbeck J, Fragoso G, Hartel F, Hendler J, Oberthaler J, Parsia B. The National Cancer Institute's thesaurus and ontology. Journal of Web Semantics 2003;1(1)
10. Noy N. F. Sintek M, Decker S., Crubezy M, Fergerson R. W., & Musen M. A.. Creating Semantic Web Contents with Protege-2000. IEEE Intelligent Systems 16(2): 60-71, 2001.
11. Rector A, Drummond N, Horridge M, Rogers J, Knublauch H, Stevens R, Wang H, Wroe C. OWL Pizzas: Practical Experience in Teaching OWL-DL: Common Errors and Common Patterns. European Conference on Knowledge Acquisition (EKAW-2004). 63-81. 2004.
12. Zhang S, Bodenreider O, Golbreich C. Experience in reasoning with the Foundational Model of Anatomy in OWL DL. Proc. Pacific Symposium on Biocomputing 2006: (in press).

ANNEX

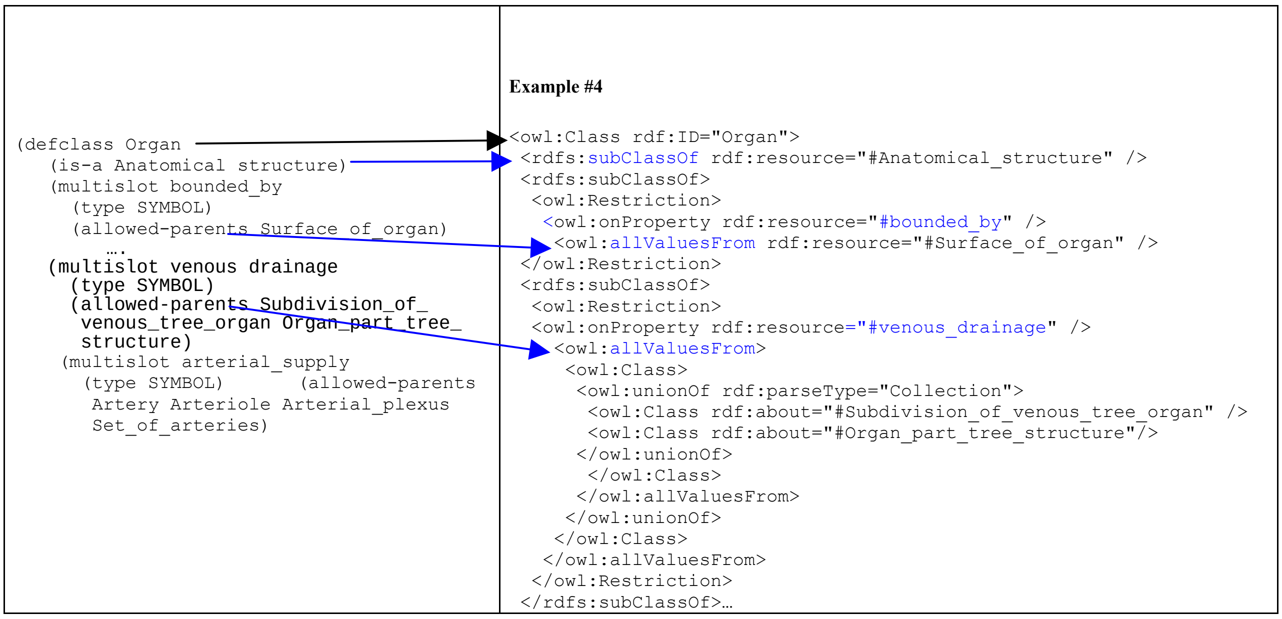
Conversion rules

The different rules depend on whether information is stored at metaclass or class level. They capture all the knowledge defined in the original Protégé model, either at metaclasses, classes, or instances, while respecting the original principles of the FMA.

1. Class information. Classes and properties – stored at (meta)class level in Protégé – are converted to OWL classes and properties with specified domain (`rdfs:domain`) and range (`rdfs:range`). The following property characteristics are translated to OWL corresponding constructs: inverse to `owl:inverseOf`, logical characteristics, i.e. transitive, symmetric to `owl:TransitiveProperty`, `owl:SymmetricProperty`, property cardinality and values restrictions to `owl:FunctionalProperty`, `owl and hasValue`.In practice, rules are the following:

– **Top level slots**, specified in Protégé to save top-level slot information, are converted to `DatatypeProperty` or `ObjectProperty` with the relevant range and restrictions, according to their definition. For example, (1) a top-level slot with type SYMBOL and allowed-values TRUE FALSE is converted to a `DatatypeProperty` with `range #Boolean` e.g., `has_boundary` (Example #1 **Table 1**), (2) a top-level slot with type SYMBOL with allowed-values different from TRUE FALSE is converted to an `ObjectProperty` with an enumerated class (`oneOf`{allowed-values}) as range, (3) a top-level slot with type SYMBOL with allowed-classes (allowed-parents) is converted to an `ObjectProperty` with the union of the allowed (meta)classes as `range`, e.g., the `range` of the `multislot venous_drainage is the union of #Subdivision_of_venous_tree_organ and Organ_part_tree_structure` (Example #2), (4) a top-level slot with type INSTANCE is converted to an `ObjectProperty`, etc.
Single-slot with cardinality 0 1 is converted to `FunctionalProperty` (Example #1).
Inverse-slot. If top level slot have "inverse-slot", it is converted to SymmetricProperty or inverseOf. If the inverse value is itself, it is SymmetricProperty with range assigned to its domain, else it is inverseOf. Thus, for example, the top level slot `has boundary` is converted to a `DatatypeProperty` with `range #boolean`, with a `FunctionalProperty` restriction, the multislot `bounded by` is converted to an `ObjectProperty` with `range #Physical_anatomical_entity`, and `inverse #bounds` (Example #3).

FMA in Protégé	FMA in OWL DL
Top level slot (defclass %3ACLIPS_TOP_LEVEL_SLOT_CLAS (single-slot has boundary (type SYMBOL) (allowed-values FALSE TRUE) (cardinality 0 1) (create-accessor read-write)) (multislot venous drainage (type SYMBOL) (allowed-parents Subdivision_of_ venous_tree_organ Organ_part_tree_ structure) (multislot bounded by (type SYMBOL) (allowed-parents Physical_ anatomical_entity) (inverse-slot bounds) (create-accessor read-write))	**Example #1** <owl:DatatypeProperty rdf:ID="has_boundary"> <rdfs:domain rdf:resource="#Physical_anatomical_entity"/> <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/> </owl:DatatypeProperty> **Example #2** <owl:ObjectProperty rdf:ID="venous_drainage"> <rdfs:range> <owl:Class> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="#Subdivision_of_venous_tree_organ" /> <owl:Class rdf:about="#Organ_part_tree_structure" /> </owl:unionOf> </owl:Class> </rdfs:range>
Class slots (defclass Physical anatomical entity (single-slot has_boundary (type SYMBOL) (allowed-values FALSE TRUE) (cardinality 0 1) (create-accessor read write))) (defclass Anatomical structure (multislot bounded_by (type SYMBOL) (allowed-parents Physical_anatomical_entity) (inverse-slot bounds) (create-accessor read-write))	**Example #3** <owl:ObjectProperty rdf:ID="bounded_by"> <owl:inverseOf rdf:resource="#bounds" /> <rdfs:domain> <owl:Class> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="#Anatomical_space" /> <owl:Class rdf:about="#Anatomical_structure" /> <owl:Class rdf:about="#Anatomical_line" /> <owl:Class rdf:about="#Anatomical_surface" /> </owl:unionOf> </owl:Class> </rdfs:domain> <rdfs:range rdf:resource="#Physical_anatomical_entity"/> </owl:ObjectProperty>



```
(defclass Organ
   (is-a Anatomical structure)
   (multislot bounded_by
     (type SYMBOL)
     (allowed-parents Surface_of_organ)
       ….
  (multislot venous drainage
    (type SYMBOL)
    (allowed-parents Subdivision_of_
     venous_tree_organ Organ_part_tree_
     structure)
   (multislot arterial_supply
     (type SYMBOL)      (allowed-parents
      Artery Arteriole Arterial_plexus
      Set_of_arteries)
```

Example #4

```
<owl:Class rdf:ID="Organ">
 <rdfs:subClassOf rdf:resource="#Anatomical_structure" />
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#bounded_by" />
    <owl:allValuesFrom rdf:resource="#Surface_of_organ" />
 </owl:Restriction>
 <rdfs:subClassOf>
  <owl:Restriction>
  <owl:onProperty rdf:resource="#venous_drainage" />
    <owl:allValuesFrom>
     <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Subdivision_of_venous_tree_organ" />
       <owl:Class rdf:about="#Organ_part_tree_structure"/>
      </owl:unionOf>
       </owl:Class>
      </owl:allValuesFrom>
     </owl:unionOf>
    </owl:Class>
   </owl:allValuesFrom>
  </owl:Restriction>
 </rdfs:subClassOf>…
```

Table 1 Examples of conversion rules for top level and template slots

– **Slots at class** enable to define the `domain` of an OWL property and to refine its value constraints: if p is slot of class A1, then A1 becomes the domain of p e.g. `#Physical_anatomical_entity` becomes the domain of `has_boundary` (Example #1); if the same slot p occurs in class A2, then the domain of p is the union of A1 and A2 (e.g. the domain of `bounded_by` in Example #3); Optimization of domain c1 ⊔ c2… ⊔ cn has been done: if ci is descendant of another class according to two levels of is-a, then ci is removed from the domain (reducing so "arterial supply" domain from 4007 classes to 4!).
Allowed-parents, allowed–classes, value define the allowed values of properties at class. They are converted to necessary conditions expressing value constraint on the property for this class: if p is slot of class A specified with allowed-parents or allowed-classes (resp. with value), then p is converted by a necessary condition at class A expressing value constraints for p by `owl:allValuesFrom` the union class of all its allowed-parents or allowed-classes e.g. `allowed-parents Surface_of_organ` (Example #4) (resp. by hasValue).
Is-a is converted into subsumption axioms (subClassOf): A is-a B (if B is not USER nor :STANDARD-CLASS or :STANDARD-SLOT or RELATION) is converted to A subClassOf B (resp. A is-a B1 B2 is converted to `subClassOf B1 ∩ subClassOf B2`), e.g. `Organ is-a Anatomical_structure` (Example #4).[5]

[5] Additional examples are provided at http://mor.nlm.nih.gov/pubs/supp/2006-psbsz/2006-psb-sz-supp.pdf

2. Instance information. The values of class own slots – specified at instance level in Protégé to store data specific to a class – are converted either into OWL values of annotation properties or into existential restrictions on the class properties. In practice, the rules are the following:

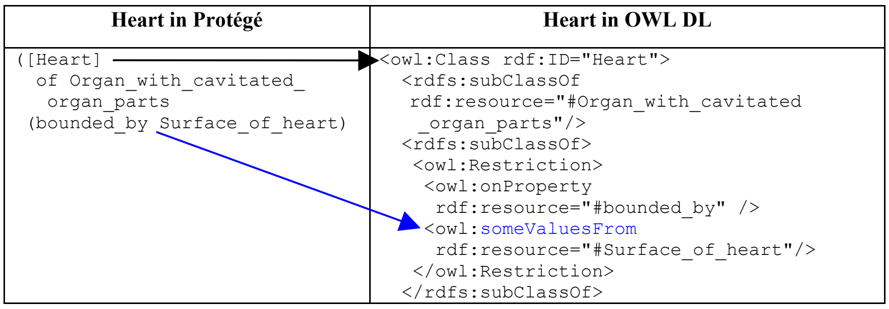
- **Non structural slots**. In Protégé non structural slots such as 'preferred name', 'synonyms', 'UWDAID', 'definition', '`author`' etc., are defined as slots of metaclasses. When classes instantiate the metaclasses, they become own slots assigned with values specific to each class, which are thus not propagated to their instances or subclasses [3]). For example, 'UWDAID' is an annotation whose value for Heart is 7088 (Table 2). We used annotations on classes instead, which are allowed in OWL-DL. Properties designed as AnnotationProperty have been manually selected. They include "UWDAID", "author", "authority", "modification", "name_", "Date_entered_modified", "TA_ID", "definition", "Modified_by", "Latin_name_TA", "UMLS_ID", "Outdated_meaning" "Other_Latin_equivalents", "Source", "View", "Abbreviation" "English_equivalent".

```
< owl:AnnotationProperty rdf:ID="UWDAID">
  <rdf:type
   rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
</owl:AnnotationProperty>

<owl:Class rdf:ID="Heart">
 <UWDAID>7088</UWDAID>
```

Table 2 Conversion of a non structural own slot into an `AnnotationProperty`

- **Structural slots**. Another use of metaclasses in Protégé is for "structural" slots, such as `part_of`, `custom_partonomy`, `bounded_by`, `arterial_supply`, etc. It enables to specify each class for "canonical" anatomy thanks to the particular values assigned to its own slots, which are thus not propagated. For example, a "canonical" `Heart` is specified as having as custom partonomy exactly a `Right_atrium`, a `Left_atrium`, a `Right_ventricle`, a `Left_ventricule`, as being `bounded_by` exactly a `Surface_of_heart`. Structural own slots are converted to necessary (or necessary and sufficient) conditions at class A expressing `someValuesFrom` constraints for p to the union of all the classes assigned to p. For example (`bounded_by Surface_of_heart`) is converted to a `someValuesFrom` restriction on property `#bounded_by`, expressing that any instance of heart is necessarily bounded at least by one `#Surface_of_heart` (Table 3).



Heart in Protégé

```
([Heart]
  of Organ_with_cavitated_
   organ_parts
 (bounded_by Surface_of_heart)
```

Heart in OWL DL

```
<owl:Class rdf:ID="Heart">
  <rdfs:subClassOf
   rdf:resource="#Organ_with_cavitated
    _organ_parts"/>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty
     rdf:resource="#bounded_by" />
    <owl:someValuesFrom
     rdf:resource="#Surface_of_heart"/>
   </owl:Restriction>
  </rdfs:subClassOf>
```

Table 3 Conversion of a structural own slot into an existential property restriction

– **Attributed relations**. The values of attributed relations are represented in OWL by nested class generated in following the same conversion rules as for classes. For example `attributed_part` is an attributed relation which allowed values are instances of class `Part_of_relationship_value` e.g. `fm-live_12491, fm-live_12492 etc.` They are converted to `someValuesFrom` restrictions on the property `#attributed_part`

Heart in Protégé	Heart in OWL

`([Heart]`
`  of Organ_with_`
`  cavitated_organ_parts`
`(attributed_part`
` [fm-live_12491]`
` [fm-live_12492]`
` [fm_live_17313]`
` [fm_live_17314]`
` [fm_live_17315]`
` [fm_live_17316]`
` [fm_live_17317]`
` [fm_live_17318]`
` [fm_live_17319]`
` [fm_live_17320]`
` [fm_live_17321]`
` [fm_live_17322]`
` [fm_live_17323])`
 	
`where`
 	
`([fm-live_12491] of Organ`
`  _subdivision_part_of_`
`  relationship_value`
`(anatomical_arbitrary`
`  Arbitrary)`
`(partition Partition_2)`
`(related part Right_side`
`  _of_heart)`
`(shared_unshared`
`  Unshared))`
	`<rdfs:subClassOf>`
	` <owl:Restriction>`
	`  <owl:onProperty rdf:resource="#attributed_part" />`
	`  <owl:someValuesFrom>`
	`  <owl:Class rdf:ID="fm-live_12491"> <!-- nested class from [fm-live_12491] -->`
	`<rdfs:subClassOf`
	`rdf:resource="#Organ_subdivision_part_of_relationship_value" /> <!-- From of_name in PINS -->`
	`<rdfs:subClassOf>`
	` <owl:Restriction>`
	` <owl:onProperty rdf:resource="#anatomical_arbitrary"/>`
	` <owl:hasValue rdf:resource="#individual_Arbitrary" />`
	` </owl:Restriction>`
	`</rdfs:subClassOf>`
	`<rdfs:subClassOf>`
	` <owl:Restriction>`
	`  <owl:onProperty rdf:resource="#partition" />`
	`  <owl:hasValue`
	`rdf:resource="#individual_Partition_2"/>`
	` </owl:Restriction>`
	`</rdfs:subClassOf>`
	`<rdfs:subClassOf>`
	` <owl:Restriction>`
	`  <owl:onProperty rdf:resource="#related_part" />`
	`  <owl:someValuesFrom`
	`rdf:resource="#Right_side_of_heart"/>`
	` </owl:Restriction>`
	`</rdfs:subClassOf>`
	`<rdfs:subClassOf>`
	` <owl:Restriction>`
	`  <owl:onProperty rdf:resource="#shared_unshared" />`
	`  <owl:hasValue rdf:resource="#individual_Unshared" />`
	` </owl:Restriction>`
	`</rdfs:subClassOf>`
	`    </owl:Class> <!-- end of nested class for instance [fm-live_12491] -->`
	`   </owl:someValuesFrom>`
	`  </owl:Restriction>`
	` </rdfs:subClassOf>`
	`<rdfs:subClassOf>`
	`…`
	`</owl:Class>`

Table 4 Conversion of attributed relations

– **Instantiation of metaclasses** is converted by subsumption axioms: [A] of B is converted to an axiom subClassOf B for A[6] e.g., the axiom `subClassOf Organ_with_cavitated_organ_parts` for class `Heart` Table 3.

[6] Except for “[A] of A”, “[A] of B ” when “A is-a B ”, “[A] of B ” when “A is-a C” and C is a descendant of B, for which optimizations prevent the generation of useless axioms.