ICBO: International Conference on Biomedical Ontology
July 28-30, 2011 · Buffalo, NY, USA

Applying Rigidity to Standardizing OBO Foundry Candidate Ontologies

A. Patrice Seyed, Stuart C. Shapiro

Department of Computer Science and Engineering, University at Buffalo, State University of New York, USA

Abstract. The Open Biomedical Ontology (OBO) Foundry initiative is a collaborative effort for developing interoperable, science-based ontologies. OBO uses the Basic Formal Ontology (BFO) as its upper ontology. Ontologies developed for OBO use include some that have been ratified, and others holding the status of candidate. There are no formal, principled criteria that a candidate ontology must meet for ratification. To help address this problem, we propose a formal integration between Rigidity, a major component of OntoClean's approach to quality assurance of ontologies, and BFO's theory of types. This work augments ongoing efforts to build software designed to evaluate and standardize OBO Foundry candidate ontologies.

Keywords: ontology, criteria, OBO Foundry, BFO, OntoClean

1 Introduction

The Open Biomedical Ontology (OBO) Foundry[1] initiative is a collaborative effort for developing interoperable, science-based ontologies. A recently adopted principle for these ontologies is that they use the Basic Formal Ontology (BFO) [1] as their upper ontology. Some OBO Foundry ontologies have been ratified, and others hold the status of candidate. Rigidity is a major component of the OntoClean approach for detecting when the taxonomic relation is being used improperly [2]. A property is Rigid, if it is essential to all its instances; Non- Rigid, if non-essential to some instance; or Anti-Rigid, if non-essential to all instances.

BFO is only partially logically axiomatized [1]. Domain experts developing OBO Foundry candidate ontologies must regularly query BFO-trained ontologists in order to adhere to BFO's principles. Currently, there are no formal, principled criteria that a candidate ontology must meet for ratification. To address this problem, we propose a formal integration between OntoClean's theory of Rigidity and BFO's theory of *types*. We also propose an approach for evaluating OBO Foundry candidate ontologies based on this integration.

[1] http://www.obofoundry.org

2 Formal Theory of Classes

OntoClean uses *properties* as its categorical unit, which are the intension, or meaning, of general terms. BFO uses *types*, which are defined as that in reality to which the general terms of science refer (B. Smith, personal communication). We unify property and types under *class*. In what follows, we assume a first-order, sorted logic.

Although there are many theories of existence, we introduce a relation, ***exists_at***(x,t), which is non-committal and means that, under a certain ontological theory, object x is within its domain and x's existence spans some time, t. Everything exists at some time:

Axiom 1. $\forall x \exists t(\boldsymbol{exists\_\ at}(x,t))$

member_of(x,A,t) means that object x satisfies the definition of class A at t. With the ***member_of***(x,A,t) relation, there is no commitment about the nature of A. Therefore, membership at a time *does not* presuppose that existence spans that time:

Axiom 2. $\neg\forall xt(\exists A\ \boldsymbol{member\_of}(x,A,t)) \rightarrow \boldsymbol{exists\_\ at}(x,t))$

A particular class might or might not satisfy the unary predicate ***Instantiated***, which means there is some member of A at t

that exists at t:

Definition 1. $\textbf{\textit{Instantiated}}(A) =_{def}$
$\exists xt(\textbf{\textit{member\_of}}(x,A,t) \wedge \textbf{\textit{exists\_at}}(x,t))$

If a class does not have any members at any time, it satisfies the predicate ***Empty***:

Definition 2. $\textbf{\textit{Empty}}(A) =_{def}$
$\neg\exists xt(\textbf{\textit{member\_of}}(x,A,t))$

Empty (*Full_Eye_Transplant*) holds because no such procedure has been performed yet.

If a class has as members only those objects that exist at all times at which they are members, it satisfies the predicate ***Members_Exist***:

Definition 3. $\textbf{\textit{Members\_Exist}}(A) =_{def}$
$\forall xt(\textbf{\textit{member\_of}}(x,A,t) \rightarrow \textbf{\textit{exists\_at}}(x,t))$

Assuming a class Animal is defined to have as members animals only at times they are alive, ***Members_Exist*** (*Animal*) holds.[2]

3 Reformulating Rigidity

Rigidity has been defined in terms of S5 modal logic. As part of our integration, we provide just the underlying intuitions of those modal formalisms, prior to reformulating Rigidity in our formal system. Each object that has a Rigid property has that property at all times at which the object exists. We formalize this in terms of classes, instead of properties, by the predicate ***Rigid***:

Definition 4. $\textbf{\textit{Rigid}}(A) =_{def}$
$\forall x(\exists t(\textbf{\textit{member\_of}}(x,A,t))$
$\rightarrow \forall t_1 (\textbf{\textit{exists\_at}}(x,t_1) \rightarrow \textbf{\textit{member\_of}}(x,A,t_1)))$

Rigid (*Person*) means that all members of the class *Person* are people at all times at which they exist.

As an amendment to the original formulation of Rigid, [3] proposes that Rigid properties are only instantiated by actually existing objects. We have captured this intuition separately from Rigid, under the ***Members_Exist*** predicate. Also, because unexemplifiable properties are trivially Rigid, [3] constrains the theory (as suggested by [4] and [5]) to properties for which there exists some instance. We have separately defined this notion, also, under the ***Instantiated*** predicate.

Non-Rigid is the negation of Rigid, which we apply for our class formulation under the predicate ***Non-Rigid***:

Definition 5. $\textbf{\textit{Non-Rigid}}(A) =_{def} \neg\textbf{\textit{Rigid}}(A)$

Assuming that a person is a member of Student only while a registered student, ***Non-Rigid*** (*Student*) holds.

Anti-Rigid is true of a property, if, for every object that has that property, it is *possible* that it does not have that property at some time. An object may have an Anti-Rigid property at all times at which it exists. BFO is not concerned with what could have been, but rather what has been or currently is; therefore, Anti-Rigid is irrelevant to our theory.

4 Integrating Rigidity with BFO Theory of Types

The objects of BFO's domain are partitioned into *particulars* and *types*. Particulars are entities confined to specific spatial, spatiotemporal, or temporal regions (e.g., a specific grasshopper in front of me, its life, or the time interval that its life spans, respectively). Under BFO's theory, existence of a particular is based on it being observable at some level of granularity and/or causal by some scientifically-based measure. Numbers, for example, do not exist in BFO. ***Type*** (A) means that A is a class that meets the criteria for being a type, which we provide in what follows.

Not all classes thought to be types satisfy our reformulation of Rigid, for example *Embryo* and *Fetus*. If an organism maintains its identity through its development from an embryo into a fetus, then both classes are Non-Rigid. If these classes are not in fact types, then our axiomatization is clear. However, whether these sorts of classes are types or not is still debated by the OBO Foundry community; therefore, this issue remains unresolved. For the purposes of our method, we exclude these controversial classes from our domain; hence, types satisfy ***Rigid***:

Axiom 3. $\forall A(\textbf{\textit{Type}}(A) \rightarrow \textbf{\textit{Rigid}}(A))$

[2] For organisms we equate existence with living.

Types must also be instantiated [6]:

Axiom 4. $\forall A(\textbf{\textit{Type}}(A) \rightarrow \textbf{\textit{Instantiated}}(A))$

Types are therefore non-empty:[3]

Theorem 1. $\forall A(\textbf{\textit{Type}}(A) \rightarrow \neg\textbf{\textit{Empty}}(A))$

Another criterion for every class that is a type is that every member of the class at a time exists at that time. Therefore, every type satisfies ***Members_Exist***:

Axiom 5. $\forall A(\textbf{\textit{Type}}(A) \rightarrow \textbf{\textit{Members\_Exist}}(A))$

instance_of(x,A,t) means that particular x is an instance of type A at time t. If a general term refers to a class that is a BFO type, then each of the members of the class instantiates the type:

Definition 6. $\textbf{\textit{instance\_of}}(x,A,t) =_{def} \textbf{\textit{member\_of}}(x,A,t) \wedge \textbf{\textit{Type}}(A)$

While there is no restriction on what objects can be members of a class, particulars, not types, are instances of a type:

Axiom 6. $\forall AB(\textbf{\textit{Type}}(A) \wedge \textbf{\textit{Type}}(B) \rightarrow \neg\exists t(\textbf{\textit{instance\_of}}(A,B,t)))$

A class which satisfies ***Instantiated*** but not *Members_Exist* satisfies the predicate ***Partial***:

Definition 7. $\textbf{\textit{Partial}}(A) =_{def} \textbf{\textit{Instantiated}}(A) \wedge \neg\textbf{\textit{Members\_Exist}}(A)$

isa (A,B) means that all instances of type A are instances of type B:

Definition 8. $\textbf{\textit{isa}}(A,B) =_{def} \forall xt(\textbf{\textit{instance\_of}}(x,A,t) \rightarrow \textbf{\textit{instance\_of}}(x,B,t))$

isa is a relation between types:

Axiom 7. $\forall AB(\textbf{\textit{isa}}(A,B) \rightarrow \textbf{\textit{Type}}(A) \wedge \textbf{\textit{Type}}(B))$

It is the "backbone" BFO relation for scientific classification, i.e., building taxonomies. ***isa*** is provably reflexive, transitive, and anti-symmetric.

Under OntoClean's modal formulations, no Anti-Rigid property is a parent of a Rigid property (although a Rigid property may have a Non-Rigid parent, and vice versa). Although Anti-Rigid is irrelevant to our theory, by our reformulation of Non-Rigid, no ***Non-Rigid*** class is part of an ***isa*** hierarchy:

Theorem 2. $\forall A(\textbf{\textit{Non-Rigid}}(A) \rightarrow \forall B(\neg\textbf{\textit{isa}}(A,B) \wedge \neg\textbf{\textit{isa}}(B,A)))$

disa (A,B) ('d' for "direct") means there is no other type "in between" A and B in the ***isa*** hierarchy:

Definition 9. $\textbf{\textit{disa}}(A,B) =_{def} \textbf{\textit{isa}}(A,B) \wedge A \neq B \wedge \forall C(\textbf{\textit{isa}}(A,C) \wedge \textbf{\textit{isa}}(C,B) \rightarrow C=A \vee C=B)$

disa is provably irreflexive, intransitive, and asymmetric, and ***isa*** is its transitive closure.

The root type of the BFO upper ontology is *Entity*; *Continuant* and *Occurrent* are its subtypes. Continuants (e.g., a heart) can exist fully at different time instants, while occurrents (e.g., the process of a heart beating) unfold over time.

Following Aristotle's division of objects into substances and accidents, the two subtypes of *Continuant* are *IndependentContinuant* (*IC*) and *DependentContinuant* (*DC*), respectively. (For reasons of space, we omit treatment of the *DC* subtype *GenericallyDependentContinuant* (*GDC*), and restrict our discussion to the *DC* subtype *SpecificallyDependentContinuant* (*SDC*).) The shape of a specific cell instantiates *SDC*, and "depends on" a specific cell, which instantiates *IC*. ***depends_on*** (x,y,t) means that the specifically dependent continuant x exists at t only if the independent continuant y exists at t:[4]

Axiom 8. $\forall xy(\exists t(\textbf{\textit{depends\_on}}(x,y,t)) \rightarrow \forall t_1(\textbf{\textit{exists\_at}}(x,t_1) \rightarrow \textbf{\textit{exists\_at}}(y,t_1)))$

It also means that x cannot migrate to another independent continuant:

[3] Informal proofs corresponding to the theorems presented here are provided at http://www.cse.buffalo.edu/~apseyed/icbo2011proofs.pdf.

[4] This relation is frequently given as 'inheres' in the BFO literature.

Axiom 9. $\forall xy(\exists tt_1\ \textbf{\textit{depends\_on}}(x,y,t) \wedge \textbf{\textit{depends\_on}}(x,z,t_1)) \rightarrow y=z)$

Depends_On (A,B) means that for every instance of A there is some instance of B where the former instance depends on the latter:

Definition 10. $\textbf{\textit{Depends\_On}}(A,B) =_{def}$
$\forall xt(\textbf{\textit{instance\_of}}(x,A,t) \rightarrow$
$\exists y(\textbf{\textit{instance\_ of}}(y,B,t) \wedge \textbf{\textit{depends\_on}}(x,y,t)))$

If we assume the class *Student* has as members people at times at which they have the role of student, *Student* satisfies *Non-Rigid* and is not a type. However if the class is "re-conceived" as having as members individual student roles that are instances of *SDC*, then the class does satisfy *Type* and ***Depends_On*** *(Student*, *Person*) holds. At each time t at which some person x is a student, there exists some y that is a student role and is dependent on x:

$\exists y(\textbf{\textit{instance\_of}}(y,Student,t) \wedge \textbf{\textit{depends\_on}}\ (x,y,t))).$

BFO's theory of types is also committed to the *Disjointness Principle*,[5] that two types have no instances in common unless one is a subtype of the other:

Axiom 10. $\forall AB(\exists xt(\textbf{\textit{instance\_of}}(x,A,t) \wedge \textbf{\textit{instance\_ of}}(x,B,t)) \rightarrow \textbf{\textit{isa}}(A,B) \vee \textbf{\textit{isa}}(B,A))$

The *Single Inheritance Principle* follows, that no type has more than one direct supertype:

Theorem 3. $\forall AB(\textbf{\textit{disa}}(A,B) \rightarrow \forall C(\textbf{\textit{disa}}(\mathrm{A},C) \leftrightarrow C=B))$

A version of this principle is advocated by [7] for primitive class hierarchies, in order to keep ontologies modular. The Disjointness Principle assists in maintaining the ontological partioning of types into *DC*, *IC*, and *Occurrent*. Candidates (i.e., classes proposed as types in an OBO Foundry candidate ontology) conceived such that they that violate the Disjointness or Single Inheritance principles do not satisfy **Type**.

We propose that the subtyping relation between upper ontology types is ***disa***, based on the assumption that the types of BFO's upper ontology fall within a finite domain. If additional types are added, then it is a different ontology.

We also define a relation ***disjoint_from*** which holds between types A and B iff A and B do not share any instances at any time:

Definition 11. $\textbf{\textit{disjoint\_from}}(A,B) =_{def}$
$\forall xt(\textbf{\textit{instance\_of}}(x,A,t) \rightarrow \neg\ \textbf{\textit{instance\_ of}}(x,B,t))$

Axiom 11. $\forall AB(\textbf{\textit{disjoint\_from}}\ (A,B) \rightarrow \textbf{\textit{Type}}(A) \wedge \textbf{\textit{Type}}(B))$

We can show that for two direct subtypes of a third type, if the two types are not identical, then they are disjoint:

Theorem 4. $\forall AB((\exists C(\textbf{\textit{disa}}(A,C) \wedge \textbf{\textit{disa}}(B,C)) \wedge A\neq B) \rightarrow \textbf{\textit{disjoint\_from}}(A,B))$

We can also prove that sibling BFO upper ontology types (e.g., *Continuant* and *Occurrent*) and, more generally, any types not related by ***isa***, are disjoint types:

Theorem 5. $\forall AB((\textbf{\textit{Type}}(A) \wedge \textbf{\textit{Type}}(B)) \rightarrow (\textbf{\textit{isa}}(A,B) \vee \textbf{\textit{isa}}(B,A)) \oplus \textbf{\textit{disjoint\_from}}(A,B))$

5 Applying Rigidity and Other Type Criteria to Standardizing Candidate Types

Isolating violations of the Disjointness Principle will assist a modeler in determining if their candidates are types. These violations follow the pattern:

$$\textbf{\textit{isa}}\ (A,B) \wedge \textbf{\textit{isa}}\ (A,C)$$

where it does not hold that:

[5] Our work is based on BFO version 1.1, which we consider stable and "frozen" for our research. Recent work [8] indicates this principle only applies to the asserted isa hierarchy. This topic remains under debate.

$$\textbf{\textit{isa}}\ (B,C) \vee \textbf{\textit{isa}}\ (C,B)$$

which can be inferred under closed-world reasoning, or, it may be that the negation of both disjuncts holds. The potential ontology changes that alleviate this violation include:

1. ***isa*** (*B*,*C*) or ***isa*** (*C*,*B*) holds.
2. ***isa*** (*A*,*B*) or ***isa*** (*A*,*C*) is removed, including the choice that ***isa*** is changed to another relation, e.g., ***Depends_On***).
3. *A* is partitioned into multiple candidates, some of which are subtypes of *B* and some of *C*.

One reason for solution #1 is that one (or both) of the disjuncts holds, but the disjunct(s) has not been specified yet by the modeler. A common reason for solution #2 is that one candidate, *B*, is a type, and the other, *C*, is a ***Non-Rigid*** class. #3 is appropriate if a candidate is evaluated to have as members instances of disjoint upper ontology types (which we term ***Heterogeneous***).

We aim to assist a modeler in creating an ontology that does not violate the Disjointness Principle, by preemptively addressing the modeling choices #1, #2, and #3 above. We present a decision tree (see Figure 1)[6] that assists a modeler in evaluating whether a candidate is a type according to criteria provided in the previous section (satisfying ***Instantiated***, ***Members_Exist***, and ***Rigid***) and, if not, assists in redefining the candidate such that it is consistent with BFO. We assume that a modeler presents her candidates one at a time to a procedure that uses the decision tree to classify each in turn. A candidate that satisfies any combination of ***Empty***, ***Partial***, ***Heterogeneous***, or ¬***Members_Exist*** satisfies ¬***Type*** and requires further inspection and re-conceptualization for it to satisfy ***Type***.

In Figure 1, the descriptions of the answer choices for Question 2 correspond to more commonly modeled types under *IC*, *DC,* and *Occurrent*, namely *MaterialEntity*, *SDC*, and *Process*. This approach excludes rarely modeled types from our evaluation work, such as *SpatialRegion*, *TemporalRegion*, and *SpatioTemporalRegion*.[7] There are certain other types, (e.g., *GDC*) that will appear in an expanded version of the tree, in future work. The classification of candidates under domain-level types already classified via applications of the decision tree will also be addressed in future work.

5.1 Use Case

Figure 2 shows two candidates, their assumed definitions, and a modeler's response to each question presented to her.[8] The candidate's class label is used within Question 1; for example, for Candidate 1, the question asked is "What is an example of a compound?" Because Question 7 is reached and answered by "no", *Compound* is a type under our approach. Because of the answer "a" given for Question 2, *Compound* is classified under BFO's *MaterialEntity* type.

For Candidate 2, because the answer given for Question 6 is "yes", Reactant satisfies ¬***Type***, because it satisfies ***Non-Rigid***. Question 8 attempts to confirm if the modeler's class definition implicitly refers to some specifically dependent continuant. If this question is answered with "yes", then the candidate is a type if re-conceived as a subtype of *SDC*. Question 9 is asked to determine the classification of the members of A (as it was originally conceived) under *MaterialEntity*. In this case, the modeler chose *Compound*, and as a result ***Depends_on*** (*Reactant*, *Compound*) is asserted. Figure 3 shows the resulting ontology portion, where the upper ontology types are shaded.

Our use case addresses the modeling of Rigid and Non-Rigid candidates, and how a Non-Rigid candidate can be redefined such that it is consistent with BFO, proactively preventing violations of the Disjointness Principle. To extend our use case, if a third candidate, *Sodium Chloride*, were introduced, then the modeler would provide the same answers as given for *Compound*, and *Sodium*

[6] Redundant subtrees for Question 2 choices a, b, or c are combined. Variables that represent modeler-input terms appear in square brackets.

[7] That the modeling of these types is rare is apparent in the OBO Foundry's Ontology for Biomedical Investigations (see http://purl.obolibrary.org/obo/obi.owl).

[8] For illustrative purposes, we exclude elements from our notion of reactant.

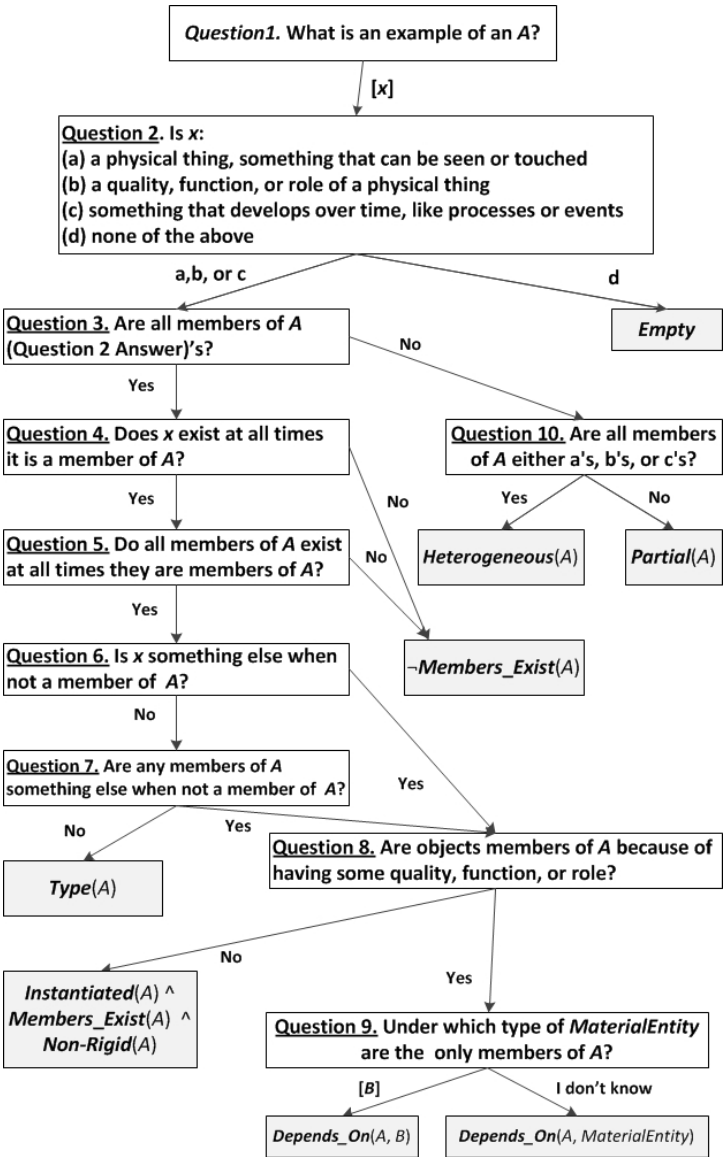

Figure 1. Decision Tree for Standardizing a Candidate Type

Question	Candidate 1: *Compound* **Assumed Definition:** "A substance consisting of two or more different elements combined in a fixed ratio" [10].	Candidate 2: *Reactant* **Assumed Definition:** "The electron donor in a redox reaction" [10].
1	"sodium chloride in this container"	"sodium chloride in this container"
2	a	a
3	yes	yes
4	yes	yes
5	yes	yes
6	no	yes
7	no	-
8	-	yes
9	-	*Compound*
10	-	-

Figure 2. Responses to Questions in Decision Tree

Chloride will be subtyped under *MaterialEntity* (the modeler will be able to subtype *Sodium Chloride* under *Compound* in a future version of the decision tree). Subtyping *Sodium Chloride* under the *SDC* subtree, where Reactant is subtyped, will simply not be presented as an option.

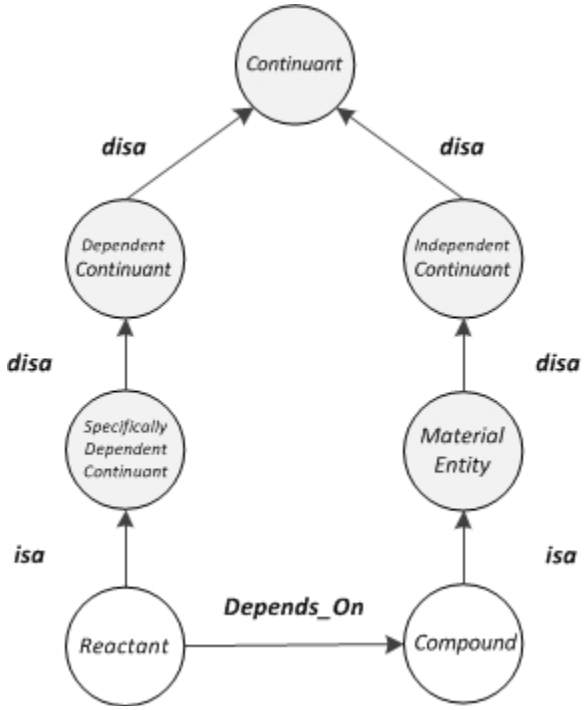

Figure 3. Ontology Portion After Evaluation

6 Conclusion and Future Work

The notion of class covers both OntoClean's notion of property and BFO's notion of *type*. A class might or might not satisfy ***Instantiated***, ***Empty***, ***Heterogeneous***, ***Partial***, ***Members_Exist***, ***Rigid***, or ***Non-Rigid***, the latter two capturing the intuitions of Rigidity within our formal theory of classes. BFO's notion of type is captured by a class that satisfies ***Instantiated, Members_Exist***, and ***Rigid***. A domain modeler who wants her ontology to be ratified for OBO use and thus BFO-compliant must show that the candidate types of her ontology are indeed types by these criteria.

In the future, we will address whether the Disjointness Principle should be enforced for only what is considered the "primitive" backbone of an ontology. We will also expand the decision tree to address Non-Rigid classes that refer to some material entity (e.g., *Endocrine System*) or process (e.g., *Fertilization*), where their members are conceived as the parts or participants, respectively.

Acknowledgments

We would like to thank William J. Rapaport, Alan Ruttenberg, Barry Smith, and the anonymous reviewers for their comments on previous drafts.

References

1. Spear, A.: Ontology for the Twenty First Century (2007)
2. Welty, C. and Guarino N.: Supporting ontological analysis of taxonomic relationships. *Data and Knowledge Engineering*. vol. 39, no. 1, 51–74, (2001)
3. Welty, C. and Andersen. W.: Towards OntoClean 2.0: A framework for Rigidity. *Applied Ontology*. vol. 1, no. 1, 107–116, IOS Press, Amsterdam (2005)
4. Andersen, W. and Menzel, C.: Modal rigidity in the OntoClean methodology. *FOIS* (2004)
5. Carrara, M.: Identity and Modality in OntoClean. *Applied Ontology*. no. 1, vol. 1., 128–139 (2004)
6. Smith, B.: The Logic of Biological Classification and the Foundations of Biomedical Ontology. *International Conference on Logic, Methodology and Philosophy of Science*. Elsevier-North-Holland (2003)
7. Rector, A.: Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL. *KCAP* (2003)
8. Smith, B. and Ceusters W.: Ontological realism: A Methodology for coordinated evolution of scientific ontologies. *Applied Ontology*. vol. 5, no. 3, 139–188 IOS Press (2010)
9. Campbell N. A., Reece, J. B.: *Biology*, 8th Edition. Pearson, University of Chicago press (1990)