A Method for Evaluating and Standardizing Ontologies

A. Patrice Seyed
Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY, USA
apseyed@buffalo.edu

1 Introduction

For my thesis work I am developing a method for evaluating and standardizing ontologies based on an integration of the Basic Formal Ontology (BFO) and OntoClean. BFO serves as the upper ontology for the domain ontologies of the Open Biomedical Ontologies (OBO) Foundry[1]. The OBO Foundry initiative is a collaborative effort for developing interoperable, science-based ontologies. OntoClean is an approach for the quality assurance of ontologies, and helps a modeler detect when the subsumption relation is used improperly.

Ontologies developed for OBO use include some that have been ratified, and others holding the status of "candidate". To maintain consistency between ontologies, it is important to establish formal principled criteria that a candidate ontology must meet for ratification. The formalisms that result from our integration will serve as criteria an OBO Foundry candidate ontology must satisfy in order to be ratified. The formalisms will also serve as a constraints within a prototype of an ontology editor that interactively asks a modeler questions that helps alleviate constraint violations.

The first step in the integration is the unification of the categorical units of OntoClean and BFO. OntoClean uses *properties*, which are the intension of general terms. BFO uses *types*, "generalizations about the structure, order, and regularity that exists in nature that experiments and observations make possible" [Spear, 2007, p.16]. We unify *property* and *type* under the categorical unit *class*, since each has a corresponding class. **member_of**(x,A,t) means that the object x satisfies the definition of class A at t. **exists_at**(x,t) means that under a certain ontological theory, object x is within its domain, and it's existence spans t.

OntoClean requires that a modeler characterize their properties with respect to the notions of *Identity*, *Unity*, and *Rigidity*.[2] *Identity* holds for properties for which there is a way to uniquely identify its instances. *Unity* holds for properties for which their instances are integral wholes. *Rigidity* is a notion that characterizes a property as either: Rigid, essential to all its instances; Non-Rigid, non-essential to some instance; or Anti-Rigid, non-essential to all instances. In what follows we describe the completed work on integrating the Rigidity component of OntoClean with BFO's theory of types.

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

[2]Dependence will be addressed within the scope of Rigidity in the dissertation.

2 Integrating Rigidity with the BFO Theory of Types

Each object that has a Rigid property has that property at all times at which the object exists. We formalize this under classes instead of properties by the unary relation **Rigid**. **Rigid**(*Person*) means that all members of the class *Person* are people at all times at which they exist. Types are not generally characterized as Rigid in BFO because for classes like *Fetus* which some consider a type, the existence conditions are debatable. For the purposes of our method we exclude these kinds of classes from our domain, hence types satisfy the **Rigid** relation.

Because unexemplifiable properties are trivially Rigid, the theory was later constrained to properties for which there exists some instance [Welty and Andersen, 2005, p.108]. We define this intuition more strictly: **Instantiated**(A) means there is some member of A at t which exists at t. Types also satisfy this criterion [Smith, 2003, p. 6]. Also, Rigid properties are only instantiated by actually existing objects. For classes, **Exists**(A) means that for A its members exist at all times at which they are members. This intuition is essential to BFO; all types satisfy this criterion.

Non-Rigid is presented as the negation of OntoClean's notion of Rigid, which we apply for our class formulation, under the unary relation **Non-Rigid**. **Non-Rigid**(*Student*) means a member of the class *Student* exists at a time at which he is not a student. Given that types (under the restriction given) are Rigid, it follows that no Non-Rigid class is a type.

The semantics of Anti-Rigid are such that an object may have an Anti-Rigid property at all times it exists, with the possibility that it could have not had that property while existing. Because a non-modal interpretation of Anti-Rigid cannot capture its intuition, we do not reformulate Anti-Rigid. For modeling BFO ontologies, we can now use **Rigid** and **Non-Rigid**, as a replacement for OntoClean's modal definitions of Rigidity.

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). **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, but also to what is a BFO type, then each of the members of the class instantiates the type. BFO considers **instance_of**, not **member_of**, the most basic relation for constructing ontologies.

Particulars that instantiate a type are in some sense a part of a whole; types exist in their corresponding particulars [Smith, 2003, p.6], so there is a dependency between a type and its particulars. (This may be more controversial for types that are artifactual, however.) With the **member_of**(x,A,t) relation, there is no commitment about the nature of A. Classes may be members of a class, but only particulars are instances of a type.

isa(A,B) means that all instances of A are instances of B. It is the "backbone" BFO relation for scientific classification, i.e., building taxonomies. **isa** is provably reflexive, transitive, and anti-symmetric. **disa**(A,B) ('d' for "direct") means there is no other type "in between" A and B in the **isa** hierarchy. **disa** is provably irreflexive, intransitive, and asymmetric, and **isa** is its transitive closure. The root type of the BFO upper ontology is *Entity* and is directly subtyped by *Continuant* and *Occurrent*. Continuants (e.g., a heart) are continuous with respect to time, while occurrents (e.g., process of a heart beating) are bound with respect to time. If an occurrent instantiates a type, it instantiates that type for all time.

BFO's theory of types commits to the *Disjointness Principle*, that two types have no instances in common unless one is a subtype of the other. The *Single Inheritance Principle* follows, that no type has more than one direct supertype. From the definition of **disa** and the Disjointness Principle, it follows that two types which have an instance and a direct supertype in common are identical. Candidate types (i.e., classes proposed as types in an OBO Foundry candidate ontology) that violate either principle are not types.

2.1 Dependency

BFO is influenced by Aristotle's division of particulars into substances and accidents. BFO refers to these under the types *IndependentContinuant* and *DependentContinuant*, which disjointly fall under *Continuant*. An example of an independent is a specific chunk of wood. An example of a dependent is the texture of that chunk of wood. The **depends_on**(x,y,t) relation means that the dependent x depends on the independent y at a time t. Relative to our example, the texture is given the status of a particular in BFO, but it only exists when the chunk of wood exists, and cannot migrate from one chunk to another.

There are other type-level relations besides **isa** which, instead of subsumption, describe relationship patterns between particulars of two types. **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 instance. For example, a function of mitochondria is to supply cellular energy.

3 Applying Rigidity and Disjointness to BFO-Compliant Domain Modeling

Our reformulation of Rigidity is useful as a modeling construct for BFO and OBO because a Non-Rigid candidate type is frequently a class whose members are independents and the class definition implicitly refers to dependents. In the context of an evaluation method, the notion of Rigid and Non-Rigid candidates is useful together with the Disjointness Principle.

Violations of the Disjointness Principle follow the pattern A is a subtype of B and C, where under closed world reasoning, B is not a subtype of C, and vice versa. One reason for the violation is that B and C are types, and one is a subtype of the other, but it has not been specified yet by the modeler. Another possibility is that one candidate, B, is a type, and the other, C, is Non-Rigid. Let's assume that A is the candidate *Molecule* and C is the candidate *Reactant*. We assume that *Reactant* has the informal definition "A starting material in a chemical reaction." In this case members of *Reactant* are molecules and the class definition implicitly refers to other particulars which BFO considers roles. Because reactant roles are only played by molecules, **Depends_On**(*Reactant*,*Molecule*) is the appropriate representation for the relationship, instead of **isa**(*Molecule*,*Reactant*).

4 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**, **Exists**, **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**, **Exists**, 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 future work we will integrate OntoClean's other components (i.e., Unity and Identity) with BFO. We will also more closely tie the current work to the ontology modeling process. In doing so we will treat our formalisms as a foundation for developing a phased, question-asking method that assists a modeler in determining if a candidate type satisfies **Instantiated**, **Exists**, **Rigid**, or **Non-Rigid**. Ofttimes ontology literature is obfuscated by philosphical lingo. We take a more intuitive approach in our method by asking the modeler questions using examples they provide for what they want to classify. Ultimately, the method we develop will provide a modeler assistance in evaluating candidate types. For those candidate types that do not satisfy criteria needed to be a type, the method will assist the modeler in determining how the candidate type can be formulated in a manner that is consistent with BFO's theory.

References

Barry Smith. The Logic of Biological Classification and the Foundations of Biomedical Ontology. In D. Westerstahl, editor, *International Conference on Logic, Methodology and Philosophy of Science*. Elsevier-North-Holland, 2003.

Andrew Spear. Ontology for the twenty first century: An introduction with recommendations. Technical report, University at Buffalo, 2007.

Christopher Welty and William Andersen. Towards OntoClean 2.0: A framework for Rigidity. *Applied Ontology*, 1(1):107–116, 2005.