A METHOD FOR EVALUATING AND STANDARDIZING ONTOLOGIES

by

Ali Patrice Seyed

December 23, 2011

A dissertation submitted to the
Faculty of the Graduate School of
the University at Buffalo, State University of New York
in partial fulfillment of the requirements for the
degree of
Doctor of Philosophy

Department of Computer Science and Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

I have many people to thank for helping through the Ph.D. process. My wife, Kelly: thank you for your love and patience with those countless nights and weekends of my "dissertating". I would like to also thank my father, mother, and sister for their love and support.

Thank you Dr. Shapiro for investing your time and energy in me these past four years. Our logic and ontology discussions have truly shaped how I now approach ontology research and also life–that's the truly valuable part about rigourous, logical thinking. Clearly just a "thank you" here is inadequate, but I hope to thank you furthermore by representing SNeRG, the Computer Science and Engineering Department, and the University at Buffalo well in all my future ventures.

I would like to thank Dr. Rapaport for his time and energy also, and working through proofs with me, as I laid down the foundations for my work. Thank you also for the handful of stories from which so many important lessons have emerged. Thank you Dr. Smith for the valuable exchanges that helped reinforce important aspects of ontology research. Thank you to my committee as a whole for commenting on the many revisions that lead to this point of completion.

I would like to thank all of the members of the SNeRG research lab, especially Jonathan Bona for the valuable user testing on the Wizard plugin application, Michael Prentice for hand-checking some of my early fitch-style proofs, and also thank you both for being good friends. Thank you to Andre Andrade for the valuable user testing also.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

Abstract

The Open Biomedical Ontology (OBO) Foundry initiative is a collaborative effort for developing interoperable, science-based ontologies. The Basic Formal Ontology (BFO) serves as the upper ontology for the domain-level ontologies of OBO. BFO is an upper ontology of types as conceived by defenders of realism. Among the ontologies developed for OBO use, there are those which have been ratified, and those currently 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. Members of the OBO Foundry have expressed interest in using OntoClean to help construct BFO compliant domain ontologies. OntoClean is a system that decomposes the notion of sortal into criteria for deciding when subsumption can hold between classes. OntoClean primarily includes three components, based on the notions of Rigidity, Identity, and Unity. The methodology for integrating the OntoClean and BFO approaches to constructing consistent ontologies has been clarified by this dissertation.

A formal integration between BFO, the Relation Ontology of BFO (RO), and OntoClean is given. The informal aspects and differing formalisms within and between the theories are analyzed and integrated within the axioms and theorems of one first-order system put forth in the dissertation. To set the foundation for this work, the categorical units of *type* and *property* of BFO and OntoClean, respectively, are unified under *class*. The modal logic axioms that OntoClean's theory of Rigidity is expressed within are interpreted and

reformulated in our system, where axioms connecting it with BFO's categorical unit *type* is given. Central to this work is the axiom that a type is a class that is Rigid, i.e., a class whose definition is fundamental to the existence of the members of the class.

A unity criterion for a class is a way in which certain parts of a member of the class are related such that they form the whole member. Our reformulation of this work focuses on the notion of the underlying unifying relation. As opposed to the informal approach taken by OntoClean, we express the notion that a class is *unified under* a relation as a meta-predicate defined by a definition schema. An identity criterion for a class is a test by which a member of the class can be re-identified. However as given in OntoClean, the notion of *identity criteria* is ontologically ambiguous. A formalization is given that provides a mapping to processes during which identity is confirmed. The reformulations of Unity and Identity are discussed in terms of *Material Entity* subtypes of BFO's theory.

The integration work and resulting formal system affords a theoretically sound ontological foundation. A method for evaluating and standardizing candidate OBO Foundry ontologies is developed atop this foundation, where the method focuses on BFO's integration with OntoClean's notion of Rigidity. The method is given as a decision tree algorithm that evaluates one class at a time as they are introduced into an ontology, asking a prospective modeler questions that map to specific integration axioms.

Finally, an implementation of the decision tree is provided in the form of an interactive Wizard plugin for the ontology editor Protégé. In an iterative approach informal user testing was applied to improve the questions and error messages. The plugin serves to facilitate ontologists and subject matter experts in making explicit what is implicit in, or unclearly specified for, the classes they choose to introduce into an ontology. Ultimately though, the integration axioms serves as a software platform-independent foundation for future software designed to evaluate and standardize candidate domain ontologies for the OBO Foundry.

Part I

Introduction

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

Ontology research today involves interdisciplinary collaborations, where contributions are drawn from researchers in the fields of philosophy (e.g., metaphysics, epistemology), linguistics, mathematics, logic, computer science, and the natural sciences. In philosophy, an ontology is "a system of categories accounting for a certain vision of the world" (Guarino, 1998, p. 2) and is independent of a particular language. In computer science or ontological engineering, an ontology is a engineering artifact that is a constituted of a set of terms and predicates that have an intended meaning or denotation, and are used to describe a certain reality (Guarino, 1998, p. 2). An ontology in this sense is commonly expressed in a formal system that includes a clear syntax and semantics, as well as rules for deduction.

A domain ontology (in the philosophical sense) is an ontology of a specific area of interest (e.g., genomes, psychiatry). Domain ontologies are also formalized at different levels of granularity (e.g., micro-organism, macro-organism, population, geopolitics). An upper ontology is constituted of primitive categories (e.g., object, quality, process) and relations (e.g., subsumption, inherence, participation) that can be used as a framework for any domain ontology.

The purpose of this thesis is to develop a method for evaluating and standardizing domain ontologies based on how their categories relate to those of an upper ontology and to metacategories. We use ‘category’ as an informal term to refer to a class, property, or type. Each of these is defined in Chapter 2. By ‘metacategory’, we mean categories of categories, that is, categories with categories as members. Upper ontology categories (e.g., Object, Process) are those that are the most general and useful as a framework for defining any domain-specific category (e.g., Hummingbird, Process of Hummingbird Pollination). The metacategories discussed in this dissertation are based on an analysis of the notion of sortal. Sortals are categories for which there is a way to distinguish between, identify, and count their instances. For example, Identity is a meta-category of every category (e.g., Human) for which there is a way to uniquely identify each of its instances. In other words, the objects that belong to the Identity category are in fact categories whose objects are uniquely identifiable.

A formal language in which an ontology is expressed is syntactically composed of predicates and terms. A predicate is a symbol that, when applied to any number of terms as arguments, forms an atomic formula. The predicates and terms are used to denote relations and entities, respectively. Predicates and terms, along with connectives and quantifiers, are used to compose formulas, which according to the syntax of a formal language, may or may not be well-formed.

The World Wide Web Consortium (W3C) defines the intension of a class as the set of criteria that individuals must satisfy in order to be its members.[1] It is often the case, as in Description Logics (Baader et al., 2003), that unary predicates are used to denote classes. A fundamental aspect of ontology research is how predicates representing classes are logically specified in an ontology. How a predicate is logically related to another should reflect the actual relationships between the individuals the represented classes are intended to include.

Often predicates, terms, and variously constructed formalisms are documented with a natural language description of what each is intended to rep-

[1]http://www.w3.org/TR/owl-features/

resent. If formalized in a W3C standard specification language (e.g., RDF or a member of the OWL family[2]) these descriptions appear in 'rdfs:comment' XML tags. Because logical formalisms are limited in some way or other in even the most expressive logics, it is important to an ontology's utility that any assumptions underlying the modeling are explicated in these descriptions.

Our evaluation method is based on two foundations; first, on Basic Formal Ontology (BFO) (Smith, 1998; Grenon, 2003a,b; Spear, 2007), an upper ontology which uses the notion of types as its categories. BFO is conceived by defenders of realism. Realist ontologies are constructed with the aim of reflecting reality as it is, without being influenced by how it is conceived. Second, it is based on the OntoClean method (Guarino and Welty, 2001, 2002, 2004; Welty and Guarino, 2001), which primarily decomposes the notion of sortal into criteria for clarifying when subsumption can hold between properties. A property ϕ subsumes a property ψ if ϕ "generalizes" ψ. In other words, any object that has the property ψ at a time must have the property ϕ at the same time. For example *being an organism* subsumes *being a human.* Although distinct notions, types and properties are similar in that they have corresponding classes. 'Type', 'property', and 'class' are defined in Chapter 2 and analysed in Section 5.

BFO is an upper ontology that has been widely adopted in biomedical research (Smith et al., 2007). BFO has a highly active user list, and both its content and its applications have been and will continue to be subjected to thorough critical review. However BFO is only partially logically axiomatized. This leads domain experts using BFO as an upper ontology for developing their domain ontologies to regularly query the ontologists who are developing BFO, in order to adhere to its principles.

OntoClean is an approach for quality assurance of ontologies, as it is a method for detecting where the subsumption relation is being used improperly (Welty and Guarino, 2001). OntoClean was orginally intended to be applied to

[2]http://www.w3.org/TR/owl-ref/

ontologies composed of properties. Properties, in the sense we are using this term here, are the meanings of certain linguistic expressions; these expressions are predicates, sentences with the subject noun phrase removed. It is a property which is an interpretation of a predicate, or more formally, provides a mapping from a predicate to domain objects that serve as referents or extensions of the predicate. These domain objects are said to "have" the property. For example if we say that some thing is a ball and red, we are in effect saying it has the property *being a ball* and *being red*, or for simplicity, has the property *ball* and *red*. By the OntoClean approach, x having the property ϕ also means that x instantiates ϕ. Property ϕ subsumes property σ if and only if every x that has the property σ also has the property ϕ.

OntoClean employs certain features for describing properties. The understanding of these features is primarily based on an analysis of the notion of sortals. Sortals are categories for which there is a way to distinguish between, identify, and count their members. Note that categories which are sortals, by the original intent for the term 'sortal', have as members material objects. OntoClean requires categorizing properties by assigning to them certain features which they possess, referred to as 'metaproperties', based on the notions of Rigidity, Identity, Unity, and Dependence. For each metaproperty, there are implications upon if the subsumption relation can hold between properties. Inconsistencies derived after a modeler classifies properties by metaproperties serves to reveal modeling mistakes.

Identity is a metaproperty of those properties that are sortals. Identity holds for every property for which there is a way to uniquely identify each of its instances. For example, Identity holds for the property 'being a human'. A criterion for identity of this property's instances is based on the possibility of fingerprinting. Take also for example the property 'being an airline passenger'. The criterion for identity of its instances is based on having flown on a specific, unique flight. Therefore a particular that is identified as two passengers may be one human. Because the criterion for identity of 'being a human' does not apply to ‘being an airline passenger’, the former property cannot subsume the latter. Because OntoClean includes explicit contraints like this one, whereas most other methods for constructing ontologies are merely guidelines (Little and Vizenor, 2006; Gangemi et al., 2001), OntoClean is an attractive approach for improving the quality of ontologies.

OntoClean’s constraints are useful to help a modeler check if she is using the subsumption relation incorrectly, but it does not address the specific assumptions that should be clarified in order for a modeler to successfully address the ontological issues surrounding a constraint violation. For instance, oftentimes the relation between properties is stated as subsumption where the implicit assumptions of the modeler would confirm it as a different relation. An OntoClean constraint violation that detected that subsumption is incorrect would be alleviated simply if the subsumption relation were removed, with no concern of its being more accurately changed instead. Our research addresses this problem, and also serves to bridge the gap between the evaluation of ontologies and an approach for standardization.

Although some ontology editing tools (e.g., WebODE, OntoEdit) (Fernandez-Lopez et al., 2002; Sure et al., 2003) have included OntoClean constraints in their software, there is no theoretically sound evaluation method developed within the framework of the OntoClean constraints—they indicate when violations occur and nothing more. This is analogous to a compiler that provides a list of error messages, or a spell checker that does not suggest possible corrections.

OntoClean is considered by influential figures in the ontology research community as a good approach to improving ontologies (Oltramari et al., 2002; Fernandez-Lopez et al., 2002). Members of the BFO developer and user communities have expressed interest in using OntoClean to help construct well-formed BFO-compliant domain ontologies, and we here show for the first time how this can be done. One common criticism of OntoClean is that “it is not always clear how to assign metaproperties, and that agreement between even experienced ontology designers is quite low” (Welty, 2006). Another criticism is that it is difficult and time-consuming to apply in a practical setting; this is particularly true for someone with no experience in formal ontology or philosophy, which is predominantly the case for domain experts.

An emerging approach towards the goal of data interoperability is establishing a standard semantics that is used for constructing ontologies appropriately. BFO and OntoClean try to reach this goal by following different strategies. BFO is an upper ontology and serves as a foundation for classifying domain categories. OntoClean provides constraints on subsumption, and is aimed at helping modelers of ontologies use the subsumption relation in a consistent, well-founded manner. By applying the notions of OntoClean to the perspective of BFO, we advance the aim of ontology quality assurance, and ultimately move closer to the goal of data interoperability.

Our method evaluates ontologies based on their level of adherence to BFO and OntoClean, and decides how an ontology should be altered based on this information. The method also considers what concise questions can be posed to the modeler in order to improve the ontology based on the evaluation. This will facilitate ontologists and subject matter experts in making explicit what is implicit in, or not clearly specified for, the nature of the categories of a domain they choose to include in their ontology. This will result in the creation of domain ontologies with more accurately and more clearly defined categories, which will promote their reuse and further the goal of a standard semantics for data interoperability.

Chapter 2

Clarification of Terms

Ontology research is performed by researchers working within several overlapping and collaborating disciplines. Because of this, polysemy of shared terms often exists and leads to miscommunication and ultimately confusion. (Not to mention, meanings of a term can migrate with changes in focus of a particular discipline.) The definitions provided are introduced for the purposes of this dissertation, but we believe that they each correspond to established usage within at least some of the sub-communities involved in ontology research.

An **entity** is anything that exists, which includes what is mind-dependent (e.g., my thoughts on the shape of bacteria) and what is mind-independent (e.g., the bacteria themselves), as well as what is general (e.g., blood types) and what is specific (e.g., this sample of blood)(Smith et al., 2006, p. 1). A **particular** is an entity that is confined to a single specific spatial, spatiotemporal, or temporal region (e.g., a specific grasshopper in front of me, its life, or the time interval of its life, respectively).

We, as people, generally use natural language to communicate, using sentences as the foundational unit. Within sentences we use nouns to refer to the entities about which we aim to convey some meaning. A **predicate$_{\mathbf{lin}}$** is that which results from removing the subject noun phrase from a sentence. An open predicate$_{\text{lin}}$ is a predicate$_{\text{lin}}$ whose application is not restricted to a finite number of particulars (Armstrong, 1980, p.14). Examples of open predicates$_{\text{lin}}$ are 'is round', 'is orange', and 'is a human'. If a predicate$_{\text{lin}}$ is not open, then it is closed, e.g., 'is the first person in space'. An open predicate$_{\text{lin}}$ with the copula removed is still usually considered an open predicate$_{\text{lin}}$, but here we mainly refer to it as a 'general term'. For example, 'Red' and 'Round' are general terms which can be applied for example to a specific red ball.

A **class** is an entity that is a collection of all and only the particulars to which a given general term applies (Smith et al., 2006). This definition (which we shall formalize later) commits a class to being an extensional entity, despite the fact that its definition is either extensional or intensional. The intension of a general term is commonly referred to as its **property**, which is said to be its meaning or sense (Quinton, 1957). OntoClean uses properties as its unit for classification.

A **type** (also called a universal) is "a generalization about the structure, order, and regularity that exists in nature that experiments and observations make posssible" (Spear, 2007, p.16). General terms such as 'molecule', 'cell', and 'photosynthesis' refer to types. A type exists at all times and in all places where particulars instantiating them exist (Spear, 2007). What distinguishes types and particulars is that the former is multiply instantiated while the latter is and can be instantiated only once. Types are repeatable (instances are non-repeatable). Note that an arbitrary collection of particulars (e.g., gifts I received for Christmas) can form a class but is not a type. This is debatable, as Lakoff (1990) suggests: in the Dyirbal language, women, fire and dangerous things do form a type. This notion of type does however diverge from the one put forth in this section.

Realist ontologies, the kind we focus on in this dissertation, include only those classes which are types for scientific classification. The defenders of ontological realism see types as what is discovered or discoverable through rigorous empirical research. We employ this sense of 'type' in what follows.

A **representation** is of or about something, and is intended to refer to some entity or entities external to the representation (Smith et al., 2006, p.2). Based on this loose definition, representations are mind-dependent (i.e., cognitive, e.g., ideas, thoughts, beliefs, symbols) or mind-independent (images, records, signs). (Symbols and signs from the perspective of Semiotics is out of the current scope.)

A **concretization** of a representation is a representational artifact that "serve[s] to make the cognitive representations existing in the minds of separate subjects publicly accessible in some enduring fashion" (Smith et al., 2006, p.3). Subsequent to the discovery of types, they are typically made comprehensible to wide audiences via some form of representation artifact, for example some sentence, or some publication (primarily electronic and/or printed format) that explicates them. Widely accessible forms include textbooks distributed in schools, as well as scientific journals.

The term 'predicate' also has a relevant meaning in logic (henceforth $\mathbf{predicate_{log}}$). An n-ary $\text{predicate}_{\text{log}}$ is a symbol that, with n terms as arguments, forms an atomic formula. The combination of a predicate applied to at least one variable term is referred to as an open (logical) sentence.

Properties are commonly represented by unary $\text{predicates}_{\text{log}}$ (e.g., **alive**(x)), binary relations are represented by binary $\text{predicates}_{\text{log}}$ (e.g., **taller_than**(x,y)), and in general n-ary relations are represented by n-ary $\text{predicates}_{\text{log}}$ (e.g., a ternary relation is **connection_between_flights**(x,y,z)). A unary $\text{predicate}_{\text{log}}$ chosen for representation is commonly intuitively chosen to correspond to a general term whose meaning is the intended property and applies to the intended class of particulars. For example one may choose the symbol 'dog' with the intent that its meaning corresponds to the property 'being a dog' and applies to the class of all dogs.

A **property** is the meaning of a predicate, and for practical domain modeling, is treated as a criterion that must be satisfied in order for a particular to be in its corresponding class. Not every property corresponds to a non-empty class; 'being a round square' does not. (However, some argue that a property that does not have any corresponding instances is not a property at all.) Two different properties (‘being an equiangular triangle’ and ‘being an equilateral triangle’) may have the same corresponding class. No two classes with different members correspond to the same property. Specializing a property (e.g., replacing ‘being a cell’ with ‘being a pyramidal cell’) either further restricts the number of particulars that have that property or has no effect, but can never increase the number.

The use of properties as the categorical unit for constructing ontologies often leads to the formation of arbitrary kinds of classes, such as those defined by a random collection of objects (e.g., utensils in your kitchen), not currently existing entities (e.g., the 100th President), and specific scaled measurements (e.g., the period of time taken to grow a collection of cells). Creating ontologies based on properties is an approach that does not require specialized training, but can lead to confusion as to what relations truly hold between its classes, and ultimately what an ontology formed on this basis is about. We give a more in-depth analysis of this problem in Section 5.

A **metaproperty** is a category of a property which serves to inform an ontology modeler about how the particulars of an ontology are conceived. ‘meta’ was chosen to prefix the term ‘property’ in order to emphasize that a metaproperty is a modeling tool and lies outside the domain of discourse. An example metaproperty is Rigid. A property is Rigid if and only if (henceforth iff) the entities to which the property applies must have that property. (Rigid is defined formally and discussed in greater detail in Section 5.6.) For instance, ‘being a human’ is a property that all humans must have, but ‘being a student’ is optional. Any entity that is a student can be (i.e., can exist) without being a student. Any entity that is a human, cannot *be* without being a human.

The metaproperties of OntoClean are defined in Section 3.5. Considering that ontologies based on BFO and those to which the OntoClean method is assumed to be applied are formed by the application of the notion of types and properties, respectively, we compare these as a starting point of our integration in Section 5.

Each class that is introduced as part of a domain ontology is a **candidate type** (henceforth 'candidate'), until it is "promoted" to a type (of BFO), which cannot occur if there are any theorems derivable about it that violate any BFO and/or OntoClean formalisms. In Part II we introduce constraints that if violated prevent this promotion from occurring, at least until they are alleviated.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Background

3.1 Ontology

As a discipline of philosophy, ontology deals with the nature of being or existence. When used as a common noun, 'ontology' is used to refer to a "classification of the entities of the world (physical objects, events, etc.)" (Cocchiarella, 2001). An ontology will often include other relations between the entities in its domain, such as part of relations. Ontologies are developed in the field of formal ontology, the "systematic, formal axiomatic development of the logic of all forms and modes of being" (Cocchiarella, 2001). We take the term 'formal' to mean an approach that leads to logical specification. The term 'ontology' has also been used in the field of artificial intelligence (AI), in a different sense. There, it is rather an engineering artifact that is usually expressed in a logical system. An ontology includes a list of terms and predicates, with descriptions of explicit assumptions made about the meaning of the terms and predicates (Guarino, 1998). These assumptions are normally formalized in first-order logic or a fragment thereof. The unary predicates refer to classes and the binary predicates refer to relations (Guarino, 1998), although this causes problems for relations that require time-indexing (see Section 3.4, on relations involving continuants). Classes form a subsumption hierarchy, where a subsumee has all the properties of its subsumers, along with additional properties. More elaborate ontologies include axioms to indicate relations between the classes other than subsumption, which further “constrains [the concept’s] intended interpretation” (Guarino, 1998). The term ‘ontology’ is now also associated with the field conceptual modeling, almost separated from its original use (Guarino, 1998). Guarino (1998) indicates a preference for using the common noun ‘ontology’ for the AI sense, which will be its use in this dissertation.

3.2 Ontological Engineering

Ontological engineering is a field that puts into practice the ontological study of domains of interest and logical formalizations about entities of those domains. It is “the set of activities that concern the ontology development process, the ontology life cycle, and the methodologies, tools, and languages for building ontologies”(Gómez-Pérez et al., 2004). It attempts to “codify the features of things based on the essence of things” (Gómez-Pérez et al., 2004). The goal of ontological engineering is to produce “reusable knowledge-based components and services” (Gómez-Pérez, 1995), which are beneficial to developers at the time at which they are designing software, because it results in a “shared application of domain knowledge using a common vocabulary” (Guarino, 1998).

A strong link between formal ontological theories and their practical application is vital to forming a basis for interoperability between disparate data sources. The current research takes a step forward in this direction. Our method for evaluating ontologies integrates multiple methods for domain modeling formalized as a set of constraints that enforce good practices, such that an ontology evaluated and standardized by the method will have a more solid foundation as a result, one rooted in formal ontological theory.

3.3 Domain Ontology

A domain ontology is the logical formalization of the objects, qualities, functions, roles, events, etc., and their respective relationships within a specific area of interest (or domain). The ontology is normally constructed by a domain expert, but ideally by a team of a domain expert and an ontologist. With recent advances in science and the resultant explosion of data—often in disparate sources—domain ontologies play a critical role in improving interoperability among information systems, improving communication between information systems and people (e.g., researchers, clinicians, software developers, etc.), and improving communication among people. Good domain ontologies include a collection of terms that are intended to refer exclusively to types, and that cover entities of the domain of interest. Each term includes a natural language definition that is concise yet expressive, resulting from careful analysis of what each term is intended to represent. Furthermore, how the terms relate by subsumption—forming the ontology backbone—and by any other relation is accurately formalized in logic. (Ideally, relationships that are too expressive for the logic used should still be expressed in natural language, and accompany the ontology as documentation.)

Single inheritance is the principle that no class has more than one immediate superclass. Disjointness is the principle that there is no overlap between classes, i.e., if they share members at all, one shares all its members with the other. In other words, if a particular instantiates two classes then one of the two classes must be a subclass of the other or they are identical. Single inheritance and disjointness are principles that good domain ontologies exhibit (Smith, 2003). Therefore, their hierarchies form a set of trees, where the nodes refer to classes that are disjoint from classes at the same level in the tree. Unfortunately, it is not only good domain ontologies that find widespread use (Schulz et al., 2006), and as a result, poorly constructed ontologies propagate problems that they were created to solve. BFO and OntoClean both encourage the creation of ontologies that follow the principles of single inheritance and disjointness, each with a different approach. Nevertheless, there is still work to be done to link these approaches to each other, and for practical use, which is addressed by this dissertation.

3.4 BFO

3.4.1 Introduction

BFO is an upper ontology, that is, an ontology that is domain neutral and designed to be useful as a framework for formalizing any domain. BFO is guided by realism, the philosophical position in which "reality and its constituents exist independently of our (linguistic, conceptual, theoretical, cultural) representations thereof" (Grenon, 2003c). BFO is an upper ontology of types as we have defined the term in Chapter 2. BFO has served as the upper ontology for several domain ontologies of the collaborative, science-based Open Biomedical Ontologies (OBO) Foundry effort (Smith et al., 2007), including the Phenotypic Quality ontology (PATO).[1] Because of its widespread use, especially in the biomedical sciences where ontologies fill an immediate need, and also because it was created to reflect reality, BFO is a good choice for evaluating ontologies.

Besides realism, the design of BFO presupposes perspectivalism, fallibilism, and adequatism. The position of perspectivalism admits that there exist multiple representations of reality that are equally good and can "capture different and important features of one and the same reality" (Spear, 2007). This includes the view of separating out particulars in the world at different levels of granularity. Fallibilism concedes the possibility that scientific theories may in fact be incorrect, given that reality "exists independently of our ways of conceptualizing it" (Spear, 2007). Finally, adequatism is the principle that an ontology representing a domain should remain true to the domain's fundamental elements, instead of abstracting to basic aspects of reality (Spear, 2007). For example, an ontology of biological phenomena must not only include such things as cells and organs but also populations (Spear, 2007). An ontology should describe the domain at the level of granularity that is necessary to be comprehensive.

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

3.4.2 Theory

BFO divides entities into continuants and occurrents following (Zemach, 1970). To reflect this distinction BFO's ontology of types includes as upper level types *Continuant* and *Occurrent*. (Henceforth we use BFO-rooted type labels (which we italicize) informally followed by the term 'particular' to refer to the particulars that instantiate a certain type, or simply informally use the type label to refer to a particular. For example, continuant particular and continuant are used interchangeably.) What differentiates these types is their relation to time. Continuants are entities that are continuous with respect to time and therefore do not have temporal parts. This implies and is implied by the equally fundamental notion that continuants are fully present in any instant of time in which they exist. For instance, a cell is a continuant. Occurrents are entities that are bound with respect to time or "unfold". This implies and is implied by the equally fundamental notion that occurrents have time as a part of them. (Instantaneous occurrents do not have a typing in BFO and are out of the current scope, but are discussed in (Bittner and Donnelly, 2004).) For instance, an occurrence of cell division is an occurrent. Temporal regions are occurrents whose parts consist solely of temporal regions.

In BFO, *Continuant* has subtypes *IndependentContinuant* and *Dependent-Continuant* (see figure 6.1). Dependent continuants, such as qualities, inhere in or are borne by independent continuants. By this we mean they exist insofar as they exist as a characteristic of some specific independent continuant. For example, the shape of a cell inheres in a specific cell. (The terms 'inhere in' and 'borne by' are further defined in Section 6.2.)

Continuant is also subtyped by *Spatial Region*. A spatial region is some part of space-time which is either not extended at all in time or does not extend past an instant of time (Bittner and Donnelly, 2004, p. 284). (BFO considers space-time as an absolute 4-dimensional entity.) An independent or dependent continuant "occupies" or is located at a spatial region for some time.

In BFO, *Occurrent* is subtyped by *ProcessualEntity* and *SpatiotemporalRe-*

gion (see 6.1). For processual entities we mainly focus on processes, for example the course of a disease or the life of a bumble bee. A spatio-temporal region is some part of space-time. A processual entity occupies a space-time region.

Occurrent is also subtyped by *TemporalRegion*. Temporal regions are taken to be "a neutral kind of time" (Spear, 2007, p. 62) and "indexes" both continuant entities existing in spatial regions and processual entities existing in space-time regions. ("A neutral kind of time" and "indexes" are admittedly unclear and vague terms, and the nature of time is not made entirely clear in the BFO literature.)

Trentelman (Personal Communication) however treats time as a projection of a time function that maps spatiotemporal regions onto a maximal temporal region (i.e., the timeline of the universe), and treats space as a projection of a space function which maps spatiotemporal regions onto a maximal spatial region (i.e, the space of the universe). Further investigation and theorizing on the nature of the universe falls outside of this dissertation's scope, as we maintain our course towards evaluating and standardizing ontologies with BFO as it is currently theorized, with minor formalization adjustments.

3.5 OntoClean

3.5.1 Introduction

OntoClean is a method that provides insight into how to properly construct an ontology of properties. Recall from Chapter 2 that a property has a corresponding class, and like types, properties have particulars as instances. In the OntoClean literature it is common to refer to a property by the general term of which it is the meaning (Guarino and Welty, 2009); this convention is occasionally followed (when convenient) in the current section. For example, the property 'being a human' is referred to as *Human*.

The originators of OntoClean, Guarino and Welty, describe the philosophical notions of identity and unity based on whether particulars of a property can be distinguished and counted, respectively. This is derived from Strawson (1959), who discusses sortals as categories with these characteristics. Guarino and Welty define identity and unity, as well as rigidity and dependence as metaproperties, because they are defined as properties of properties. However, we prefer to define metaproperties as characteristics of properties, where a characteristic picks out some aspect of all particulars that instantiate a property. We offer a full treatment of this perspective and its axiomatization in Part II.

OntoClean requires that a modeler classify the properties of an ontology by assigning to them the metaproperties just listed. For each metaproperty, there are constraints concerning when the subsumption relation can hold between properties, the violations of which are only apparent after the aforementioned classification is performed. Each metaproperty and its corresponding constraints are discussed in the following subsections. In the current chapter, we present OntoClean informally, as Guarino and Welty have in (Guarino and Welty, 2004, 2009). In Part II, we provide the corresponding formal account, discuss our interpretation of the work, and show how we leverage it towards a method for evaluating and standardizing candidate BFO domain ontologies.

3.5.2 Rigidity

The notion of rigidity includes the metaproperties Rigid, Non-Rigid, and Anti-Rigid. Each characterizes properties with respect to the essentialness of a property to its instances. A property is essential to an instance if it is a property the instance must have. For example, the property 'being human' is essential to a particular human, Peter, because he cannot exist other than as a human.

A property is Rigid iff the property is essential to all of its instances. 'being human' is a Rigid property. A property is Non-Rigid iff the property is not Rigid (i.e., there is some instance for which that property is not essential to it). By this definition, the property 'being a student' is Non-Rigid, because there is at least one human for which the property is inessential. As another example, the property 'being an organism of only one cell with only one nucleus' is Non-

Rigid because it is inessential for some instances (e.g., Caulerpa particulars, which originally have one nucleus, but as they develop have many (Jacobs, 1994)). Note that this property being essential for some instances, which it is for diatom particulars (i.e., diatoms are unicellular, mononucleate organisms throughout their existence) is irrelevant to the definition of Non-Rigid.

A property is Anti-Rigid iff it is inessential (i.e., optional) to every particular with that property. 'being a student' is a prototypical case, given that every person is not a student before enrollment or after graduation. The constraint that applies to Rigidity is that Anti-Rigid properties can only subsume other Anti-Rigid properties. This is formally discussed in Section 5.6.

Notions like Rigidity are not exclusive to realist ontologies. Andersen and Menzel (2004) suggest there are "non-existent" bacteria, i.e., a class defined by a description of a strand of bacteria that has never existed. A property based on such a category would exhibit the metaproperty of Rigid. Furthermore, fictitious worlds with elaborately described natural laws could possibly apply the notion of Rigidity as well. In a fairy tale world where a human can turn into a frog, the property 'being a human' is Non-Rigid, because there exists some human that can cease being a human and still exist. Currently BFO does not have a way of dealing with either of these kinds of categories, we address how it can in Part II.

3.5.3 Identity

The study of identity deals with being able to determine whether and how a particular can be re-identified (Guarino and Welty, 2009, p. 204). An identity criterion is a metaproperty for how particulars of a property are identifiable. Properties are used to define classes by some similarities their instances share. For example the property 'having DNA' is a property all humans share. In contrast, an identity criterion concerns how particulars that share a property are different from each other in such a way that they can be distinguished on that basis. Often identity criteria can be limited to those that are simply necessary, i.e., essential to an entity. Naturally if x and y do not share the same essential properties they are not identical. For example a necessary identity criterion for humans is “having the same DNA encoding”. This is not a sufficient identity criterion due to identical twins having the same DNA encoding.

Identity criteria are a topic that has been hotly debated in the field of philosophy. Such debates sometime involve a conflation of two distinct questions concerning some particular x: ‘What is it for this to be x?’ and ‘How can we know that this is x’? (Williamson, 1990). The former, a metaphysical question, addresses the nature of x. The latter, an epistemological question, addresses some characteristic by which x with a certain property can be told apart from y with the same property. OntoClean’s usage, which we consider in this section, seems to be in the latter sense, although there is no clear boundary between the ontological and epistemological issues at hand. In Chapter II we investigate both OntoClean’s and BFO’s perspectives on this topic.

A question that often leads to the discovery of a characteristic for identification is, given some x and some y: ‘Do x and y have the same ___?’. Since identity criteria are difficult to specify, necessary or sufficient conditions are considered (by OntoClean) acceptable as long as they are not trivial or circular. As a constraint, every property must inherit identity criteria of their subsuming properties. A property with an identity criterion that its subsuming type does not have must be Rigid, and is said to “supply” that identity criterion (Welty and Guarino, 2001). Otherwise, an identity criterion is inherited, and is said to “carry” it.

For example, *Organism* has the identity criterion ‘having the same DNA code’. If there is no subsumer of *Organism* with this identity criterion, then *Organism* must be Rigid (and carries it), which is the case. (This is not provable but rather an axiom assumed of the OntoClean work.) Its subproperty *Human* inherits this identity criterion, and has another identity criterion ‘having the same fingerprint pattern’, because every human can be uniquely identified through fingerprinting. Its immediate subsumer, *Organism* does not have this criterion, therefore *Human* must be Rigid, which it is. In this example *Organism* supplies the identity criterion 'having the same DNA code', while *Human* inherits this criterion (ergo carries it) and supplies the identity criterion 'having the same fingerprint pattern'. This example of two properties conforms to the constraint for an identity criterion, but would not if *Human* did not inherit the identity criterion 'having the same DNA code' from its subsumer property *Organism*, i.e., if humans could be not identified by DNA code.

3.5.4 Unity

Another OntoClean metaproperty is Unity, which holds for properties whose particulars are integral wholes. Thus for each particular satisfying a property with the metaproperty Unity, its parts and boundaries are connected in such a way that they are connected to all the other parts and to nothing else (Welty and Guarino, 2001, p.7). An example of a property without Unity is *Amount of Water*, because water can be scattered arbitrarily. In contrast, *Book* has Unity, since its particulars are whole objects. We can distinguish what is and is not a part of a book, for example pages, cover, etc. This is referred to by some as the count-mass distinction.

A unity criterion of a property is a specific condition that must hold among the parts of each of its particulars in order for the particular to be considered a whole. One could argue that a table with a portion removed is still a table, and therefore does not have a unity criterion, but this is not the case. The following passage explicates why by describing the nature of the property *Table*, emphasizing the importance of connectedness (Quine, 1981, pp.92–93):

> A table contains a graded multitude of nested or overlapping physical objects each of which embodies enough of the substance to have qualified as a table in its own right, but only in abstraction from the rest of the molecules. Each of these physical objects would qualify as a table, that is, if cleared of the overlapping and surrounding

molecules, but should not be counted as a table when still embedded in a further physical object.

A simplified way to determine whether a property has a unity criterion is by being able to answer the question of its instances 'how much/many ____ are there?' without a measuring unit (Lowe, 1989a). For example, 'how much coffees are there?' cannot be answered without assuming units (e.g., pounds, cups). In contrast, 'how many marbles are there?' can be. Although this is a useful device, there are more fine-grained distinctions that exist for describing unity criteria (e.g., for liquids with varied compositional structure (Bittner and Donnelly, 2007)). We will investigate this further, and provide formulae which will enrich the theory and strengthen its use for evaluation and correction.

There are three main kinds of unity, namely: topological, which deals with how an entity's parts are connected (e.g., a piece of coal); "morphological", a combination of topological and shape (e.g., a constellation); and functional, where the parts contribute to a function the whole performs (e.g., a hammer strikes a surface) (Welty and Guarino, 2001, p.10). Unity criteria are inherited downward in a subsumption hierarchy, like identity criteria. A property has Anti-Unity if every particular is not necessarily a whole (Welty and Guarino, 2001, p.10-11). Anti-Unity is also inherited downward in a subsumption hierarchy.

This dissertation investigates the best approach to logically formalize the ontological entities involved in a unity criterion and their relationships with BFO upper ontology types and RO relations. Since the research on OntoClean describes three kinds of unity, we can use these as a starting point for our work. This formalization has also not been investigated and performed previously, and is crucial in our effort of integrating BFO and RO with OntoClean.

3.5.5 Dependence

OntoClean includes as a metaproperty external dependence, primarily based on Simons (1987). (OntoClean refers to this metaproperty as dependence, but we refer to it as external dependence to avoid confusion with BFO's closely related notion.) A property x is externally dependent on a property y if the existence of x implies the existence of y, and y neither constitutes nor is a part of x (Welty and Guarino, 2001).

A property that is externally dependent (on some existing child) is *Parent*, since each of its particulars exist only if there is some corresponding *Child* particular (Welty and Guarino, 2001). A parent is externally dependent on a child because he or she cannot be a parent without having had a child. Also the property *Child* is externally dependent also, because every child is borne of his or her parents. Another example is *Satellite*—one such particular, Ganymede, exists as a satellite in relation to Jupiter (Simons, 1987). The constraint that applies to external dependence is that it is inherited downward in a subsumption hierarchy. The property *Sibling* inherits external dependence from *Relative* in the subsumption hierarchy. In the OntoClean literature, every example property that is given as an instance of external dependence is also an instance of Anti-Rigid. For our method we investigate this, and the relation between OntoClean's notion of dependence and BFO/RO.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

OntoClean-Related Work

4.1 Original OntoClean Application

The researchers behind OntoClean made an initial attempt at translating it into an application for evaluating ontologies (Welty and Guarino, 2001). They used CLASSIC (Brachman et al., 1991) to create what they describe as a question/answer system that performs consistency checking based on OntoClean. Each possible metaproperty is represented as a Description Logic (DL) concept in CLASSIC, with corresponding subsumption constraints (as described in Section 3.5) as necessary conditions on them. The domain properties are represented as individuals of the OntoClean metaproperties, based on assignments given by the modeler during use of the system. CLASSIC was not able to provide any subsumption reasoning services between the properties because they were represented by DL individuals, instead of DL concepts.

The following describes a typical interaction of the system and a modeler (Welty and Guarino, 2001):

[It] is designed to assist a modeler in choosing the appropriate metaprop-

> erties for their property by asking a series of questions. More general questions are asked first, such as “does the property carry identity?” and the modeler may respond yes, no, or unsure. If unsure about a metaproperty, more specific questions can be asked such as, “are instances of the property countable?” It is always possible to leave answers unknown, of course. In these cases the system cannot verify the correctness of those properties.

The system notifies the modeler when she makes a decision that violates any of the constraints of OntoClean, which includes that only Anti-Rigid properties can be a subproperty of Anti-Rigid properties. A significant issue with this implementation is that it generates an ABox (i.e., ground facts) version of the ontology that is not linked to the original ontology. The concepts of the modeler's ontology are individuals in the meta-ontology, and a method for mapping the two is not provided.

Although OntoClean has been used by those with experience in conceptual modeling, a major difficulty in using it is “what identity and unity criteria to apply to the properties of the domain” (Welty and Guarino, 2001). To address this problem, Welty and Guarino (2001) included several examples of common identity and unity criteria as subproperties of the existing metaproperties. If a modeler is unsure about when to indicate an identity or unity criterion, the application suggests these example criteria to help her determine which are appropriate for the properties she is modeling.

In order for the proposed evaluation and standardization method to be realizable as an application that can detect and correct violations pertaining to the principles of OntoClean and BFO, it should consider that specific questions, not unlike the ones asked above, be posed to the modeler. However, our approach will differ, since the questions that need to be asked should not pertain to just a category and its subsumer category, but also under which one of several disjoint categories of BFO a category is subsumed. For instance a modeler can be assisted by being asked a question that helps to decide whether a category is a

Continuant or *Occurrent*, and if a *Continuant*, whether it is a *DependentContinuant* or *IndependentContinuant*, and so forth (Simon and Smith, 2004). G&W did not connect their approach with an upper ontology as we do here, which will assist with additional constraints and ultimately standardization. We also investigate what it means for there to be several violations at once, and what kinds of questions can be asked to alleviate multiple violations at once.

4.2 WebODE

WebODE (Corcho et al., 2002) is an application for ontological engineering that includes reasoning capabilities. WedODE provides support for using Methontology (Fernandez et al., 1997), a method for building ontologies from scratch (Corcho et al., 2002). Corcho et al. (2002) claim a tight integration of OntoClean with Methontology. This is useful because Methontology by itself does not include design principles for the development of taxonomies or methods for isolating and correcting mistakes in taxonomies (Fernandez-Lopez et al., 2002), the latter of which is directly related to the current research. The built-in software that supports OntoClean in WebODE is called ODEClean. It uses the Ontology of Properties (Guarino and Welty, 2000a) as the base ontology, and includes the OntoClean constraints. This ontology is translated into Ciao Prolog automatically with WebODE, and ODEClean consults the types (as they define them) and Prolog axioms during evaluation of a domain ontology. Therefore the evaluation itself is more or less automated.

The top of the base ontology consists of *Type* and *Role*, which in the Ontology of Properties are labeled *sortal* and *non-sortal*, respectively. Also used is an ontology of particulars, the instances of which are in fact particulars of the domain. The metaproperty tagging of rigidity, identity, unity, and dependence are instance attributes inherited from the root class of the ontology of particulars, *property*. The WAB (WebODE axiom builder) module is used for formalizing the OntoClean subsumption constraint axioms (discussed in Section 3.5). Every concept is linked as an instance of the root class. Through the inheritance of the meta-attributes, the user can give values for each concept that is being built.

In WebODE, when an OntoClean axiom is violated, a pop-up error message window appears with the exact axiom violated and the two types involved. A novice modeler would have difficulty appropriately reacting to the message in the WebODE window without experience with OntoClean, because they would not have an idea of how best to correct it. Furthermore, the approach fails to take advantage of knowledge already available in the ontology, such as relations with upper ontology types, or knowledge that is applied to the design of an upper ontology like BFO, to help a modeler alleviate a violation. If it did, it could suggest how the error could be removed based on existing formalizations. Our method lays the groundwork for an application that not only evaluates the assignments of metaproperties with respect to constraint violations, but also changes an ontology—towards standardization in line with BFO—based on user feedback to certain questions.

4.3 OntoEdit

The closed-source Java-based application OntoEdit (Sure et al., 2003) focuses on the steps of ontology development: requirements specification, refinement, and evaluation. The initial phase, requirements specification, requires a modeler to describe an ontology domain and goal, as well as its use cases, and applications supported by it. This is performed using the OntoKick and Mind2Onto plugins of OntoEdit (Sure et al., 2003). During the refinement phase, the modeler extends the description of the ontology, which is completely formalized in the knowledge representation language F-Logic (Kifer et al., 1990). Ontobroker provides inferencing to support the process. Features of Ontobroker include support for namespaces (for modularization, naming ontologies), switching off sets of definitions for testing (i.e., reasoning over portions of an ontology), unique rule names, database connectors (for mapping DB tables into predicates), and an available API.

The evaluation phase includes observing the requirements specification and applying the OntoClean method. (Sure et al., 2003) use two building blocks of "a set of axioms that formalize definitions, constraints and guidelines given in OntoClean" and a meta-ontology of properties to have as a reference model when performing evaluations. In OntoEdit, the OntoClean constraints for rigidity, unity, identity, and dependence are formalized as F-Logic axioms. Using a plugin developed just for applying OntoClean, when once meta-relations have been used to tag concepts with OntoClean metaproperties, queries can be made to determine if any inconsistencies exist, using Ontobroker (Sure et al., 2003). Sure et al. (2003) indicate the next version of their plugin will have a dynamic GUI to tag concepts with metaproperties, and the results should guide the user through possible actions for fixing inconsistencies that were detected. The work of OntoEdit has been developed as a commercial product by the company Ontoprise, coupled with the Ontobroker inference engine and OntoKick plugin for requirements specification.

OntoEdit is an all-in-one software suite for ontology editing that has built in the OntoClean subsumption constraints and provides violation alerts. Unfortunately, there is no current work on guiding the user to removing culprits of the constraint violations. Further, there is no usage of an upper ontology for additional constraints or any motivation towards standardization, which are integral to our research.

4.4 TMEO Method

TMEO (Tutoring Methodology for the Enrichment of Ontologies) (Oltramari and Vetere, 2008) is a prototype software system that assists a user in classifying their classes based on the DOLCE upper ontology (Masolo et al., 2003). Although DOLCE was developed from a cognitive/linguistic perspective—which differs from BFO's realist perspective—it shares several features with BFO, including the top-level division of entities as continuants or occurrents. The developers of TMEO emphasize that they hide deep ontological issues from users because the system is intended for users of any experience level.

As an example, the system asks a user if an instance of *Glass* (1) has a spatial or temporal nature, (2) is something a human can sense, (3) is countable, (4) is an artifact. The questions correspond to different levels of specificity of the DOLCE ontology, and the system classifies the class in question, here *Glass*, based on the responses. In this case, if the user answers yes to the final question, the class is subsumed under *Artifact*, which is subsumed under the DOLCE class *Concrete*.

TMEO is interesting because, like the intentions behind this dissertation, it attempts to make the principles behind an upper ontology "accessible" to a modeler of a domain ontology while they are constructing it. TMEO however, unlike our work, does not apply an upper ontology as a standard by which domain ontologies can be more easily integrated, as is the case for the BFO and the OBO Foundry, and does not take advantage of OntoClean. Also unlike our work, there are no formalisms provided, and there is no discussion on how a reasoner can be used to correct inconsistent modeling choices given their framework. For our method we take into consideration the approach of helping a user in the process of identifying which is the appropriate superclass—among mutually disjoint child classes—of the class she is currently trying to model.

Part II

Integration

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

The Theory of Classes

5.1 Introduction

In the following chapters we perform a detailed analysis of the OntoClean meta-properties and explicate how they can be leveraged towards a method for evaluating and standardizing candidate BFO domain ontologies. As a starting point, we inspect and compare the classification units of OntoClean and BFO, respectively property and type.

The originators of OntoClean assume that properties, which they consider the meanings of expressions, are commonly used as the categorical unit to build ontologies because it requires no specialized training to consider them. Therefore in OntoClean a modeler is assumed to use properties as the basis for designating classes of an ontology. However under this approach properties are frequently treated as if isomorphic to classes. Some properties, for example 'being a round square', correspond to an empty class, therefore isomorphism is lost.

Building ontologies on the basis of predicates$_{\text{lin}}$ is a course-grained approach, as evidenced by the need for a method like OntoClean. For example, the property 'in close proximity to Earth' is used to classify objects. The property refers to an object external to the instance, the Earth. Therefore the property depends on a binary relation between spatially separate objects. For BFO, properties that are relational cannot correspond to a type, because types do not depend on objects other than their instances. By the BFO approach, binary relations hold between the pairs of particulars in question (we discuss one of these relations in Subsection 6.2) and should be represented as such. This promotes the practice for every particular that plays an important role in the domain being modeled, it has a typing.

5.2 Unifying Properties and Types

In order to integrate OntoClean with BFO, that is, apply the notions underlying OntoClean to the process of constructing BFO-compliant domain ontologies, we must "unify" their categorization units, properties and types, respectively. Every property and type has a corresponding class (although only properties may have empty classes). As discussed in Chapter 2, classes (except those which are empty) are composed of particulars. As mentioned, a general term applies to, or denotes, a class of particulars. We formally apply the category *class* in order to address the differing ontological perspectives committed to by BFO and OntoClean, that is, realism and a mixture of natural language semantics and epistemology, respectively.

For our formal system we consider the two most basic kinds of entities to be particulars and classes, as opposed to particulars and types. The latter division would be more appropriate for a formal system defined for ontologies that are assumed to already be BFO-compliant. However, we design our system for the evaluation of all classes, and the detection of those classes that are certainly not types (Smith et al., 2006). In what follows we put forth our formal system which reflects this.

In this chapter we analyze Rigidity as it is given in the formal system chosen by Guarino and Welty, and redefine the core ideas of Rigidity within our formal system, while also giving BFO's theory of types. In closing, we describe how this work is useful for evaluating candidate types and the eventual goal of constructing BFO-compliant ontologies.

5.3 Formal System

Traditional first-order logic has one kind of object, and models have a single domain. In other words, there is no syntactic difference between kinds of objects. With sorted logic the domain of objects is divided into what are called sorts, therefore distinguishing objects can be done in the formal system instead of through classes. Because this provides a benefit in terms of simpler formal expressions and for computation, we assume a sorted logic with identity.

In the current and all subsequent chapters, variables w, x, y, and z range over all objects; A, B, C, and D range over those objects that are classes; t, t_1, and t_2 range over those objects that are times; and $\mathbf{r}$ and $\mathbf{s}$ range over relations. The logical connectives $\neg$, $\wedge$, $\vee$, $\oplus$ $\rightarrow$, and $\leftrightarrow$ have the meanings of negation, conjunction, inclusive disjunction, exclusive disjunction, implication, and bi-implication, respectively. The binary relation $=$ is identity. A definition is expressed by putting what is being defined on the left of the binary relation, $=_{\text{def}}$, and to that which it is being defined as meaning on the right (definiendum and definiens, respectively). We provide axioms to capture the basic assumptions about our theory, define certain important predicates in terms of the theory, and use axioms and definitions to prove theorems. $\forall$ and $\exists$ represent universal and existential quantification. Leading universal quantifiers are generally omitted for readability in the text, and are assumed to have the widest possible scope.

In a definition, each argument serves as a meta-variable, standing for any term, which could be a variable or a constant term. In each case of a definition, within any well-formed formula besides the definition the definiendum can be replaced by the definiens, with the appropriate term substitution. Because we can always use the definiens in place of the definiendum, definitions are theoretically superfluous, but do make our formalisms more manageable. More importantly, a definition usually implies that the definiens is worth careful consideration, and the aggregate of definitions put forth in a theory embodies a choice of what is most important to the theory (Whitehead and Russell, 1957). Definitions provide easy summaries of information that we never need to look at again – thus they allow more complex thoughts to be lazily and easily transmitted and used in further and more ambitious research.

In **Chapter 7**, we introduce several notions that, in order to introduce them them into a formal system, require, instead of a first-order system, a second-order system or a first-order system that provides for second-order expressivity, which are both beyond our current scope. We therefore introduce *definition schemata*, labeled as such, for introducing several notions which, given our first-order system, are meta-logical, and are therefore given as meta-predicates.

For example if a meta-predicate $\mathbf{P}(x,r)$ is introduced x is a metavariable that represents any particular particular and r is a metavariable that represents any particular relation. As in any schemata, the constants which are applied, i.e., the constants that take the place of x and r, replace occurrences of x and r of the wffs in the definiens (i.e., right hand side) of the definition schema. Note that in all cases where a meta-predicate appears in the definiens of another definition schema or the right-hand side of an implication, it serves only as a "shortcut" for the definiens of the respective definition schema, with the appropriate substitutions. We list all axioms, definitions, definition schemata, and theorems in order of appearance in Appendix C.1 and by type in Appendix C.2.

5.4 Existence

Prior to providing our non-modal formulation of Rigid, it is necessary to discuss the topic of existence, a highly contentious philosophical topic. Existence is both intuitive and elusive, the boundary conditions for which are frequently debated. Kant said that "existence is not a predicate"—you cannot define things into existence (Kant, 1933). In other words, predicating 'exists' of an object does not add additional information in the same way 'is red' and 'is alive' does. Russell (1905) believed that sentences in which a general term denotes a non-existent are false, given his theory of descriptions.

Quine held that there are two kinds of existence: concrete, physical existence in the world (e.g., of Barack Obama), and abstract, non-physical existence (e.g., of the number 27) (Hirst, 1991). Meinong (1904) described his position in an oxymoron: there are objects of which it is true that there are no such objects, i.e., that they do not exist. Like Brentano, his teacher, Meinong proposed that every thought of an idea, such as the idea of a gold mountain, must be "directed toward" some object, and so all objects of thought have *being* in some sense, even if not real-world existence.

According to Rapaport (1978) there are two modes of predication, representing, respectively, the relation between Meinongian objects and their properties and, on the one hand, the relation between actual, physical objects and their properties, on the other. According to Parsons (1980) in an approach motivated by Meinong, existence is a property but not a categorical one (i.e., extranuclear), and as such it is not part of the nature of objects. Hirst (1991) suggests that quantifiers must sometimes scope over non-existent entities, but that even though existence can be predicated, it is neither a single predicate nor a predicate of an ordinary kind.

Although there is a wealth of theories of various kinds of existence, we introduce a relation, **exists_at**(x,t), which is uncommitted in this regard, and means that object x's existence spans t. For a certain ontological theory, if object x is within its domain, then it holds that it exists at some time:

Axiom 1. $\exists t(\mathbf{exists\_at}(x,t))$

Because it is presupposed that what 'x' denotes exists, if x does not exist at some time, there is some time at which it does exist.

5.5 Formal Theory of Classes

In order to integrate OntoClean's notion of Rigidity with BFO's theory of types, we must first "unify" their categorization units, properties and types, respectively. Every property and type has a corresponding class (although only propertie may have empty classes). As discussed in Chapter 2, classes (except those that are empty) are a collection of particulars. As mentioned, a general term applies to, or denotes, a class of particulars, We formally apply the notion of class in order to address the differing ontological perspectives committed to by OntoClean and BFO, that is, a mixture of natural language semantics with epistemology and realism, respectively.

We formalize **member_of**(x,A,t) to mean that the particular x falls under, or is a member of the class A at time t. x is a member of class A iff x satisfies the definition given for A. This formalism maintains first-order expressivity because classes are treated as first-order objects of the domain and are represented by terms, not predicates.

The subsumption relation between classes is **subclass_of**, and is defined by the **member_of** relation:

Definition 1. $$\mathbf{subclass\_of}(A,B) =_{\text{def}} \forall xt(\mathbf{member\_of}(x,A,t) \rightarrow \mathbf{member\_of}(x,B,t))$$

If two classes are identical then every time at which a particular is a member of one class, it is a member of the other:

Axiom 2. $$A{=}B \leftrightarrow (\mathbf{subclass\_of}(A,B) \wedge \mathbf{subclass\_of}(B,A))$$

subclass_of is therefore anti-symmetric. This axiom commits a class to being an extensional entity, despite the fact that a class definition is either exclusively extensional or intensional. For example, we can define a class *giraffe at Buffalo Zoo* by referring to each giraffe at Buffalo Zoo, or by giving the conditions only all giraffes at Buffalo Zoo satisfy. In either case, they are the same class. This informs our choice of the predicate 'member_of', which is to stress the extensional nature of the relationship we are modeling. The **subclass_of** relation is reflexive:

Theorem 1. $\textbf{subclass\_of}(A,A)$

Proof. Let x, A, and t be such that x is a member of A at t, which tautologically implies x is a member of A at t. By definition of **subclass_of** (**Definition 1**), $\textbf{subclass\_of}(A,A)$. □

The **subclass_of** relation is transitive:

Theorem 2. $\textbf{subclass\_of}(A,B) \wedge \textbf{subclass\_of}(B,C) \rightarrow \textbf{subclass\_of}(A,C)$

Proof. Let A and B be such that $\textbf{subclass\_of}(A,B)$ and $\textbf{subclass\_of}(B,C)$. Also let x and t be such that x is a member of A at t. By $\textbf{subclass\_of}(A,B)$, $\textbf{subclass\_of}(B,C)$, and the definition of **subclass_of** (**Definition 1**) it holds that x is a member of C at t. Therefore $\textbf{subclass\_of}(A,C)$. □

Also, membership at a time *does not* presuppose that existence spans that time:

Axiom 3. $\neg\forall xt(\textbf{member\_of}(x,A,t) \rightarrow \textbf{exists\_at}(x,t))$

For example, *George Washington* is a member of a class we define under the general term 'former president' at the present time, t. George Washington does not exist at t (he passed away in 1799). Therefore not all members that satisfy a definition of a class at t exists at t.

Note here that for humans we equate existence to being alive. Existence conditions for other classes may differ. When $\textbf{member\_of}(x,A,t)$ and $\neg\textbf{exists\_at}(x,t)$ both hold for a certain x, A, and t, it means that x satisfies the definition of A at some time t, and exists at some time which is not t. In the case of the class *Former President*, informally we assume its definition to be such that its members are those individuals that at some time in the past served as the President of the United States. Given this, George Washington is a member of this class for all time, whether alive or not, once he satisfies the definition.

A notable feature of some classes is that of having at least one member at a time at which the member exists in the world, thus satisfying the unary relation **Instantiated**:

Definition 2. $\textbf{Instantiated}(\mathrm{A}) =_{\mathrm{def}} \exists xt(\textbf{member\_of}(x,A,t) \wedge \textbf{exists\_at}(x,t))$

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

Definition 3. $\textbf{Empty}(A) =_{\mathrm{def}} \neg\exists xt(\textbf{member\_of}(x,A,t))$

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

Definition 4. $\textbf{Members\_Exist}(A) =_{\mathrm{def}} \forall xt(\textbf{member\_of}(x,A,t) \rightarrow \textbf{exists\_at}(x,t))$

For example, if the class *President* is defined not to include dead ex-presidents as being presidents, then **Members_Exist** holds for that class. As another example, if the class *Human Race* is defined to have members as current members any people that have passed away, it would not satisfy **Members_Exist**. In both examples, where humans are being classified, we equate existing with being alive.

Note that if a class satisfies **Empty**, then the class also trivially satisfies both **Rigid** and **Members_Exist**. This does not cause any limitations because, as I describe in **Chapter** 6, these predicates are applied to the notion of *type* and the BFO Standard for evaluating classes.

5.6 Integrating Rigidity

5.6.1 Introduction

In Chapter 3, we informally described OntoClean in natural language, which is an approach taken by Guarino and Welty in a recent publication (Guarino and Welty, 2009). In this chapter, we analyze Rigidity as it is given in the formal system chosen by Guarino and Welty, and carefully inspect its underlying intuitions. We then redefine the core ideas of Rigidity within our formal system, which we will later use to help define BFO's theory of types.

Rigidity is a notion of OntoClean 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. Categorizing properties on this basis is valuable because it helps a modeler make clear their commitment (i.e., assumptions) for a given conceptualization (i.e., perspective). Note also that OntoClean allows for multiple perspectives, thereby does not impress upon a modeler that a property must have one certain ontological nature.

For the first iteration of Guarino and Welty's work (*ThGW₁*)[1] Rigidity is presented in S5 modal logic with the Barcan Formula (Hughes and Cresswell, 1996). (The Barcan Formula states the relationships between modalities and quantifiers.) With S5 it is assumed that every world is accessible from every other world, and for the domain of every world there are both possible and actual objects. Objects are treated as in Section 5.5, those entities belonging to the intended domain. The domain of quantification consists of the union of the domains of all possible worlds (i.e., fixed domain semantics), as opposed to the domain ranging over the current world of evaluation, the actual world (Welty and Guarino, 2001, p.56)(Andersen and Menzel, 2004, p.120).

Because the domain of quantification includes all objects, an existence predicate was introduced in a later revision (*ThGW₂*)[2] to represent actual existence (Welty and Andersen, 2005, p.109),(Andersen and Menzel, 2004, p.122)(Carrara, 2004, p.132). Therefore x exists at t in a world iff 'x' denotes an object that is actual in that world at t.

Also because there is a fixed domain, times across worlds are the same. Given this, for the informal definitions that follow we assume that OntoClean's notion of time is the same as that given in Section 5.5. Given that time is the

[1] ThGW_1 stands for 'Guarino and Welty's first theory'.
[2] ThGW_2 stands for 'Guarino and Welty's second theory'.

same across worlds, an object can be actual in one world and possible in another at the same time.

Welty and Andersen (2005, p.108) (*ThWA,* and assumed for *ThGW*$_2$) [3] define subsumption between properties such that property ϕ subsumes property ψ iff, in all worlds, x has property ϕ at time t only if x has property ψ at t. Welty and Guarino (2001, p.56) treat the definiendum of the metaproperty definitions as first-order schemata in order to avoid second-order semantics; in our formal system classes are reified, therefore classes are first-order objects of the formal system.

Welty and Andersen (2005, p.109) admit there are many forms of Rigid, the usage of which depends on the type of ontology being constructed. They also suggest that although modal logic is used for the formalisms of Rigid in the OntoClean literature, reformulations that do not require modal logic can also be useful. In what follows we inspect what it means for a property to be Rigid, Non-Rigid, and Anti-Rigid in the context of the modal formal system used by Guarino and Welty, and then redefine these notions in our formal system using classes as the categorical unit, as part of our integration.

5.7 Rigid

Recall from section 3.5.2 that a property ϕ is Rigid iff ϕ is essential to all of those particulars to which ϕ is applied. Within *ThGW*$_1$, a property ϕ is Rigid only if, in all worlds, if x has the property ϕ at a time, then in all worlds, if x exists at a time then x has the property ϕ at that same time. The underling intuition, however, is that each object that has a Rigid property has that property at all times at which the object exists. We formalize this under classes as follows:

Definition 5.
$$\mathbf{Rigid}(A) =_{\text{def}} \forall x(\exists t(\mathbf{member\_of}(x,A,t)) \rightarrow \forall t_1(\mathbf{exists\_at}(x,t_1) \rightarrow \mathbf{member\_of}(x,A,t_1)))$$

For example, *Person* is a Rigid class iff all members of the class are people at

[3]ThWA stands for 'Welty and Andersen's first theory'.

all times at which they exist.

As an amendment to the previous formulation, within the later formualation, *ThGW$_2$*, a property ϕ is Rigid only if, in all worlds, if x has the property ϕ at a time, then x exists at that time. We have captured this intuition separately from Rigid, under the **Members_Exist** predicate.

Because under *ThGW$_1$* unexemplifiable properties are trivially Rigid, within *ThGW$_2$* the domain is constrained (as suggested by Andersen and Menzel (2004, p.121) and Carrara (2004, p.132)) to properties for which it is possible in every world that the property has an instance at some time (Welty and Andersen, 2005, p.108). We have separately defined this notion, also, under the **Instantiated** predicate.

5.8 Non-Rigid

Recall from section 3.5.2 that a property is Non-Rigid iff there is some particular for which that property is not essential. In terms of the modal semantics given by Guarino and Welty, a property ϕ is Non-Rigid iff, in some world, some x has property ϕ at a time and in some other world x does not have property ϕ at a time. Formally, Non-Rigid was originally presented as the negation of the core notion of Rigid, which we apply for our class formulation as well:

Definition 6. $\textbf{Non-Rigid}(A) =_{\text{def}} \neg\textbf{Rigid}(A)$

From which it follows:

Theorem 3. $\textbf{Non-Rigid}(A) \leftrightarrow \exists x(\exists t(\textbf{member\_of}(x,A,t)) \wedge \exists t_1(\textbf{exists\_at}(x,t_1) \wedge \neg\textbf{member\_of}(x,A,t_1)))$

Proof. Follows from the definition of Rigid (**Definition 5**) and Non-Rigid (**Definition 6**). □

For example, *Student* is a Non-Rigid class iff a member of the class exists at a time at which he or she is not a student. Finally, every class must be Rigid or Non-Rigid:

Theorem 4. $\textbf{Rigid}(A) \oplus \textbf{Non-Rigid}(A)$

Proof. By **Definition 6**, we have the implication that if A is Non-Rigid then A is not Rigid. In disjunctive form, we have that A is not Non-Rigid or not Rigid (A cannot be both Rigid and Non-Rigid.) To introduce an exclusive OR, we must establish that A cannot be neither Rigid nor Non-Rigid. With the disjunctive form just given, by **Definition 6** we can replace not Non-Rigid with Rigid and not Rigid with Non-Rigid, therefore we have that A is Rigid or Non-Rigid (A must be at least one of Rigid or Non-Rigid). Ergo, we have that A is exclusively Rigid or Non-Rigid.

□

5.9 Anti-Rigid

Anti-Rigid is a further restriction on the meaning of Non-Rigid. Recall that a property is Anti-Rigid iff it is not essential to all particulars with that property. In terms of the modal semantics given by Guarino and Welty, a property is Anti-Rigid iff, in all worlds, if x has property ϕ at a time, then there is some world in which x does not have property ϕ at some time. It follows from the definitions of Rigid and Anti-Rigid that if ϕ is Rigid and ψ is Anti-Rigid, then it is not the case that ψ subsumes ϕ. The corresponding proof requires the formula of $ThGW_2$ that all properties are exemplifiable.

Welty and Andersen (2005, p.108) argue that modal logic is necessary to define the meaning of Anti-Rigid. To support their argument they suggest there are Anti-Rigid properties, for example 'being a hospital patient', which a person can have at all times at which he exists, with the possibility that he could have existed without having that property.

Our theory is not concerned with possible futures, or "what could have been", but rather what has been or currently is. Therefore Anti-Rigid is irrelevant to our theory. Given this, the partition of classes as (non-modally) Rigid or Non-Rigid is the extent of our integration of Rigidity with BFO, and

Non-Rigid becomes the designation that plays a prominent role in our theory, not Anti-Rigid. This issue is non-limiting, as we explain in Part II, **Method**.

5.10 Discussion

Note here that by eliminating Anti-Rigid we cannot constrain the subsumption relation in the same manner as was intended by OntoClean. In the modal system, it can be proven that an Anti-Rigid property cannot subsume a Rigid property. In our system, we cannot show that the subclass relation does not hold between two classes if one is Non-Rigid and the other is Rigid. Firstly, we can show that there is a Rigid class that is a subclass of a Non-Rigid class:

Theorem 5. $\exists AB(\textbf{Rigid}(A) \wedge \textbf{Non-Rigid}(B) \wedge \textbf{subclass\_of}(A,B))$

Proof. Consider a class *Human*, that is Rigid. Also consider a class *B*, defined by the disjunction of *Human* and another class, *Reactant*, that is Non-Rigid. Class *B* is Non-Rigid because there is some member of *B* (i.e., some non-human reactant) that exists at a time it is not a reactant. By definition of subclass, *Human* is a subclass of *B*. Therefore a Rigid class is a subclass of a Non-Rigid class. □

Note that it is only contingent that a class defined by the disjunction of a Rigid class and a Non-Rigid class is Non-Rigid. For example, a class defined by the disjunction of *Human* and *Student* is Rigid under the assumption that all students are human.

Secondly, we can show that there is a Non-Rigid class that is a subclass of a Rigid class:

Theorem 6. $\exists AB(\textbf{Rigid}(A) \wedge \textbf{Non-Rigid}(B) \wedge \textbf{subclass\_of}(B,A))$

Proof. Consider a class *Human*, that is Rigid, a class *Student* that is Non-Rigid, and that every member of student is a member of Human. By definition of subclass, *Student* is a subclass of *Human*. Therefore a Non-Rigid class is a subclass of a Rigid class. □

Due to this contingent relationship, assignments of Rigid and Non-Rigid cannot be immediately applied to inform a modeler when the **subclass_of** relation cannot hold between classes, as was the case for assignments that included Anti-Rigid. The Rigid/Non-Rigid distinction, however, is useful within the scope of BFO's Theory of Types, as we shall see in the next chapter, **Chapter 6**. Although assignments of Rigid and Non-Rigid cannot inform when the **subclass_of** relation does not hold, careful inspection of classes that satisfy **Non-Rigid** does. When a class A satisfies **Non-Rigid**, then by definition *some* member x of A at some time t is not a member of A at some other time t_1. If the modeler were to make explicit that x is a member of another class B at t_1, then it would be true that A is not a subclass of B (regardless of whether or not B satisfies **Rigid**).

Chapter 6

BFO Theory of Types

6.1 Formal Theory

The objects of BFO's domain are what we described in Chapter 2 as particulars. Therefore the two most basic objects evaluated for BFO are, disjointly, *particular* and *class*. We do not assume *particular* and *type* because it is for the evaluation of all classes, and the detection of those classes that are not types, that we propose our system.

Given the commitment to particulars, **exists_at**(x,t) is true iff x has real-world physical existence at t. This means x is observable at some level of granularity and/or causal by some scientifically-based measure at t, and can somehow be validated by a community of scientists. By this token, abstract objects like numbers are not particulars.[1] Particulars only exist in the past or at present, and cannot exist as "future" objects. (We admit however that it is not intuitive to discuss regions of space, time, and space-time in this manner, an issue we visited in Section 3.4.)

Types are classes which satisfy additional criteria. We introduce a unary

[1]Note however that BFO is not a closed world artifact. Thus if a purported particular is not observable by science, this does not necessary imply that by BFO that the entity does not exist. Furthermore, the proposals that we make refer strictly to BFO 1.1, since BFO evolves only very incrementally, but the bulk of the ideas will be valid even for the later versions of BFO.

relation, **Type**, which holds for classes that are BFO types. A criterion each class that is a type must satisfy is that it is instantiated by some particular at some time, in the actual world (Smith, 2003, p. 6). Therefore each type satisfies the unary relation **Instantiated**:

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

A class that satisfies **Instantiated**, but has as a member an object which is a not a particular, satisfies **Partial**:

Definition 7. $\textbf{Partial}(A) =_{\text{def}} \textbf{Instantiated}(A) \wedge$
$\exists xt(\textbf{member\_of}(x,A,t) \wedge \neg\textbf{exists\_at}(x,t))$

Another criterion for every class that is a type is that every member of the class at some time is such that it exists at that time. Therefore each type also satisfies the unary relation **Members_Exist**:

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

Theorem 7. $\textbf{Type}(A) \rightarrow \neg\textbf{Partial}(A)$

Proof. As a reductio proof, let us assume that there is some A such that $\textbf{Type}(A)$ and $\textbf{Partial}(A)$. By $\textbf{Partial}(A)$ and the definition of **Partial** (**Definition** 7), there is some x and t be such that $\textbf{member\_of}(x,A,t)$ and $\neg\textbf{exist\_at}(x,t)$. By **Axiom 5** and $\textbf{Type}(A)$ it follows that $\textbf{Members\_Exist}(A)$. By the definition of $\textbf{Members\_Exist}(A)$ and $\textbf{member\_of}(x, A,t)$, it follows that $\textbf{exists\_at}(x,t)$, a contradiction. □

In the BFO literature, types are however not characterized as what we define as **Rigid**, primarily due to controversy surrounding the existence conditions of particulars that fall under classes which correspond to stages of (human) development, e.g., *Embryo*, *Fetus*, *Neonate*, *Infant*, *Child*, and *Adult*. It is unsettled in the OBO Foundry community whether the identity of some x persists across the times it is a member of these classes. For example it is not clear if an embryo that develops into a fetus can be identified as being the "same thing" as the embryo, or is something new altogether. *Are the changes that an embryo undergoes* *during the span of development between being a embryo and a fetus sufficient to cause a change in identity?* If the answer is "yes", then *Embryo* and *Fetus* are Rigid, otherwise they are Non-Rigid. If the latter is correct, because by some opinions these classes are types, the question then becomes whether some types are in fact Non-Rigid.

It might be suggested that since these classes, which are based on developmental stages are in fact roles, are Non-Rigid they might be categorized as Role types. However they are not "optional" in the way that roles are. Note the following schema: everything that is a human embryo is a human, everything that is a human embryo was a fetus, and everything that is an adult was an embryo.

Because these class are not types in BFO (in both the current version 1.1, in 2011, and the upcoming 2.0) and they are not optional in the way that roles are, I will exclude the modeling and evaluation of these developmental stage based classes from the domain.[2]

With this exclusion in place, Rigidity is in fact fundamental to the relationship between a particular and its *type*, which is the the classification unit for BFO compliant ontologies. This has not been formulated to this point, for the reasons given, which we think will assist in the process of modeling BFO-compliant ontologies.

Under the assumptions given, each type satisfies the **Rigid** relation:

Axiom 6. $\mathbf{Type}(A) \rightarrow \mathbf{Rigid}(A)$

From which it follows that classes which satisfy **Empty** are not BFO types:

Theorem 8. $\mathbf{Type}(A) \rightarrow \neg\mathbf{Empty}(A)$

Proof. Let A be a class that is a type. Because each type is instantiated (**Axiom 4**), and an instantiated class is not empty (**Definition 2**, **Definition 3**), it holds that A is not empty. □

[2]BFO developers want these classes to be considered as types, but the relevant portion of BFO has not been officially documented, yet, and will not be included in BFO 2.0 (Barry Smith, Personal Communication).

It is worth noting that if a class satisfies **Empty**, it is vacuously true that the class satisfies **Members_Exist**. Because all types satisfy **Instantiated**, this is of no interesting consequence, since our framework is centered on type candidacy.

Also, our definition of **Empty** is such that there is no member at some time, which trivially entails there is no member at some time that exists at that time. What is important is that classes that satisfy **Empty** or this entailment are not relevant to BFO. This includes those classes for which, at some time in the future, members are anticipated to exist. This remains a hotly debated topic. Dumontier and Hoehndorf (2010) among others argue that there are certain classes that we can fully describe the features of that, although they have never had members, are valuable in an ontology of the biomedical sciences. Take for example *Chemical Compound Not Yet Synthesized*, which has as members "potential" pharmaceutical drugs, which are modeled in computer simulations but have never been synthesized. The originators of BFO argue that these kinds of classes are not types because the intended members are not such that they exist. B. Smith (Personal Communication) suggests that they are descriptions or plans, and the latter are generically dependent continuants rather than classes.

To further this point, if "future" types are admitted into an ontology, then those which are fictional, imaginary, abstract, etc. are given equal (ontological) status. The root of this debate is unfortunately a perpetual one between realists and conceptualists and/or epistemologists on what role in scientific research an ontology is supposed to play.

Given that types (under the restriction given) are Rigid, it follows also that no Non-Rigid class is a type:

Theorem 9. $\textbf{Non-Rigid}(A) \rightarrow \neg\textbf{Type}(A)$

Proof. Let A be a class which is Non-Rigid. By **Definition 6** it is not Rigid, so by **Axiom 6** it is not a type. □

Non-Rigid is very useful as a modeling construct because it often picks out classes that are often dependent continuant types and whose members are independent continuant particulars. For modeling BFO ontologies, we can now use **Rigid** and **Non-Rigid** as a replacement for OntoClean's modal definitions of Rigidity. In what follows we present the rest of BFO's theory of types, which also serve as criteria that BFO types must meet.

The fundamental relation between a particular and a type is **instance_of**. **instance_of**(x,A,t) is true iff 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 8. **instance_of**$(x,A,t) =_{\text{def}}$ **member_of**$(x,A,t) \wedge$ **Type**(A)

This definition is important, because it means that classes that are types are class that *only have particulars as members*, which therefore exist at all times they are members:

Theorem 10. **Type**(A) $\wedge$ **member_of**(x,A,t) $\rightarrow$ **exists_at**(x,t)

Proof. Let x, A, and t be such that **Type**(A) and **instance_of**(x,A,t). By the definition of **instance_of** (**Definition 8**), it follows that **member_of**(x,A,t). By **Axiom 5** and **Type**(A) it follows that **Members_Exist**(A). By the definition of **Members_Exist** (**Definition 4**) and **member_of**(x,A,t) it follows that **exists_at**(x,t).

We therefore easily show that every x that instantiates a type at t exists at t:

Theorem 11. **instance_of**$(x,A,t) \rightarrow$ **exists_at**(x,t)

Proof. Let x, A, and t be such that x is an instance of A at t. By the definition of **instance_of** (**Definition 8**), x is a member of A at t and A satisfies **Type**. By **Axiom 5** A satisfies **Members_Exist**. By the definition of **Members_Exist** it follows that each member exists when a member. □

Also, it follows that every type has an instance:

Theorem 12. **Type**$(A) \rightarrow \exists xt($**instance_of**$(x,A,t))$

Proof. Let A be such that **Type**(A). From **Axiom 4** it holds that there is some x and t such that **member_of**(x,A,t). From the definition of **instance_of** (**Definition 8**), **Type**(A), and **member_of**(x,A,t) it follows that **instance_of**(x,A,t). □

It also follows that if a member of a class does not instantiate it, that class is not a type:

Theorem 13. $\exists xt($**member_of**$(x,A,t) \wedge \neg$**instance_of**$(x,A,t)) \rightarrow \neg$**Type**$(A)$

Proof. Let x, A, and t be such that **member_of**(x,A,t) and $\neg$**instance_of**(x,A,t). From $\neg$**instance_of**(x,A,t) and the definition of **instance_of** (**Definition 8**), it follows that $\neg$**member_of**(x,A,t) or $\neg$**Type**(A). Because we assume **member_of**(x,A,t), the latter disjunct, $\neg$**Type**(A), holds. □

While there is no restriction on which objects can be members of a class, only particulars are instances of a type:

Axiom 7. **Type**$(A) \wedge$ **Type**$(B) \rightarrow \forall t(\neg$**instance_of**$(A,B,t))$

BFO considers **instance_of**, not **member_of**, the most basic relation for constructing ontologies. Particulars that instantiate some type are in some sense a part of a whole; types exist in their corresponding particulars (Smith, 2003, p.6) and so there is a dependency between them. (This may be more controversial for types that are artifactual, however.) The existence conditions of types is however out of the current scope.

With the **member_of**(x,A,t) relation, however, x is presupposed to be a particular, but there is no commitment about the nature of A, and certainly not that it characterizes what is essential to x. It is simply a classification, which is why we apply **member_of** relation as a primitive for defining Rigid and Non-Rigid, and as a modeling construct for evaluating candidate types.

is_a is a relation between types and is the basic or "backbone" BFO relation for scientific classification, i.e., building taxonomies. The definition of a type serves as a more general definition for its subtypes. Hence every particular x that instantiates a type A at a time t also instantiates a supertype B at the same time t:[3]

Definition 9. $\textbf{is\_a}(A,B) =_{\text{def}} \forall xt(\textbf{instance\_of}(x,A,t) \rightarrow \textbf{instance\_of}(x,B,t))$

is_a is a relation between types:

Theorem 14. $\textbf{is\_a}(A,B) \rightarrow \textbf{Type}(A) \wedge \textbf{Type}(B)$

Proof. Let A and B be such that $\textbf{is\_a}(A,B)$. From **Theorem 12** and the definition of **is_a** (**Definition 9**) it follows there is some x and t such that $\textbf{instance\_of}(x,A,t)$ and $\textbf{instance\_of}(x,B,t)$. From the definition of **instance_of** (**Definition 8**) it follows that $\textbf{Type}(A)$ and $\textbf{Type}(B)$.

□

Two types are identical if and only if one is a subtype of the other, and vice versa:[4]

Axiom 8. $A{=}B \leftrightarrow (\textbf{is\_a}(A,B) \wedge \textbf{is\_a}(B,A))$

Given the definition of **is_a**, two types are identical if and only if every time at which a particular is an instance of one type, it is a member of the other, and vice versa.

is_a is therefore provably anti-symmetric. **is_a** is also provably reflexive and transitive, as proven for **Theorem 1** and **Theorem 2**, where types and instances must be substituted for classes and members. Because by definition $\textbf{instance\_of}(x,A,t)$ entails $\textbf{member\_of}(x,A,t)$, we can easily show that if A is a subtype of B, then A is a subclass of B:

Theorem 15. $\textbf{is\_a}(A,B) \rightarrow \textbf{subclass\_of}(A,B)$

[3]We use 'time' to refer to the time objects of BFO, temporal regions, which includes temporal instants.

[4]It is not clear from the BFO literature that the '$\rightarrow$' holds; however, from personal communication Barry Smith indicates where it is not true (in the OBO Foundry), it is a mistake.

Proof. Let A and B be such that **is_a**(A,B). Also let x and t be such that x is a member of A at t. By **Theorem 14**, A is a type, and by the definition of **instance_of** (**Definition 8**), x is an instance of A at t. Applying the definition of **is_a** (**Definition 9**) to **is_a**(A,B), x is an instance of B at t. By **Definition 8**, x is a member of B at t. Therefore **subclass_of**(A,B). □

Our non-modal definitions of **Rigid** and **Non-Rigid** are useful for domain modeling by constraining use of the BFO classification relation, **is_a**:

Theorem 16. $\textbf{is\_a}(A,B) \rightarrow \textbf{Rigid}(A) \wedge \textbf{Rigid}(B)$

Proof. Follows from **Theorem 14** and **Axiom 6**. □

Theorem 17. $\textbf{Non-Rigid}(A) \rightarrow \forall B(\neg\textbf{is\_a}(A,B) \wedge \neg\textbf{is\_a}(B,A))$

Proof. Follows from **Theorem 6** and **Theorem 14**. □

As noted, **Instantiated**, **Exists**, and **Empty** also restrict the **is_a** relation. In BFO, it is assumed that only types subsume types because BFO does not acknowledge any other kind of category. Therefore no Non-Rigid class is a supertype of a Rigid class, and no Rigid class is a supertype of a Non-Rigid class. The latter implication is unique to BFO; in OntoClean it is only that Non-Rigid cannot subsume Rigid, there is no constraint that Non-Rigid cannot be subsumed by Rigid. We henceforth treat these formulations as constraints that a class within a candidate BFO domain ontology must satisfy in order to be a type.

Every type is a subtype of BFO's root type, *Entity*:

Axiom 9. $\textbf{Type}(A) \rightarrow \textbf{is\_a}(A,\textit{Entity})$

and every class rooted under (i.e., a subtype of) *Entity* is a type:

Theorem 18. $\textbf{is\_a}(A,\textit{Entity}) \rightarrow \textbf{Type}(A)$

Proof. Follows from **Theorem 14**. □

From **Theorem 18** it also follows that **Type**(*Entity*).

Every subtype or supertype is a subtype of *Entity*:

Theorem 19. $\textbf{is\_a}(A,B) \rightarrow \textbf{is\_a}(A,\mathit{Entity}) \wedge \textbf{is\_a}(B,\mathit{Entity})$

Proof. Follows from **Theorem 14** and **Axiom 9**. □

Under BFO for any particular x that exists at a time t, x instantiates some type A at t:

Axiom 10. $\textbf{exists\_at}(x,t) \rightarrow \exists A(\textbf{instance\_of}(x,A,t))$

therefore if a particular instantiates any type at a time, it instantiates *Entity* at that time:

Theorem 20. $\exists A(\textbf{instance\_of}(x,A,t)) \rightarrow \textbf{instance\_of}(x,\mathit{Entity},t)$

Proof. Follows from the definition of **instance_of** (**Definition 8**), **Axiom 9**, and the definition of **is_a** (**Definition 9**). □

Given these last two formalisms, we can show:

Theorem 21. $\textbf{member\_of}(x,A,t) \wedge \textbf{exists\_at}(x,t) \rightarrow$
$\exists B(\textbf{member\_of}(x,B,t) \wedge \textbf{Rigid}(B))$

Proof. By **Axiom 10**, under BFO's theory, every object that exists at some time is an instance of some type at that time (hence is a particular). By **Axiom 6**, every type is a Rigid class. Therefore, every member of a Non-Rigid class that is a particular is a member of some Rigid class. □

Similarly, and tying in our OntoClean reformulation of Rigid and restricting members to particulars by virtue of **exists_at**, it follows that:

Theorem 22. $(\textbf{Non-Rigid}(A) \wedge \textbf{Members\_Exist}(A)) \rightarrow$
$\exists B(\textbf{subclass\_of}(A,B) \wedge \textbf{Rigid}(B))$

Proof. Let x, A, and t be such that **member_of**(x,A,t), and assume that **Non-Rigid**(A) (which has no bearing) and **Members_Exist**(A). By the definition of **Members_Exist** (**Definition** 5), it follows that **exists_at**(x,t). By **Axiom 10**, follows **instance_of**(x,B,t) for some indefinite B. It follows from the definition of **instance_of** (**Definition 8**) that **member_of**(x,B,t) holds. Therefore **subclass_of**(A,B). From the definition of **instance_of** and **instance_of**(x,B,t) it follows that **Type**(B). By **Axiom 6**, it follows that **Rigid**(B). □

Furthermore, we can show that if a class is a subclass of *Entity* (therefore if it is a subclass of any BFO type), it satisfies **Members_Exist**:

Theorem 23. **subclass_of**(A,*Entity*) $\rightarrow$ **Members_Exist**(A)

Proof. Let x, A, and t be such that x is a member of A at t, and A is a subclass of *Entity*. By the definition of **subclass_of** (**Definition 1**) it follows that x is a member of *Entity* at t. Since *Entity* is a type (**Theorem 18**), *Entity* satisfies **Members_Exist** (**Axiom 5**). Therefore, x exists at t; ergo, A satisfies **Members_Exist**. □

As discussed in Section 3.4.2, the root type of the BFO upper ontology is *Entity*; *Continuant* and *Occurrent* are its subtypes. For occurrents, if they instantiate a type at some time, they instantiate that type for all time:

Axiom 11. $\exists t$(**instance_of**(x,*Occurrent*,t)) $\rightarrow$
$\forall t_1$(**instance_of**(x,*Occurrent*,t_1))

It follows that all occurrents exist for all time:

Theorem 24. $\exists t$(**instance_of**(x,*Occurrent*,t)) $\rightarrow \forall t_1$(**exists_at**(x,t_1))

Proof. Let x and t be such that x is an instance of *Occurrent* at a time t. Therefore by **Axiom 11** x is an instance of *Occurrent* for all time. Since for all times at which something is an instance of a type, it exists at those times (**Theorem 11**), x exists for all time. □

Also, for a subtype A of *Occurrent*, its instances are instances of A for all time:

Theorem 25. $$\textbf{is\_a}(A, \mathit{Occurrent}) \rightarrow \forall x(\exists t(\textbf{instance\_of}(x{,}A{,}t)) \rightarrow \forall t(\textbf{instance\_of}(x{,}A{,}t)))$$

Proof. Let A be such that it is a subtype of *Occurrent*. Let x and t be such that x is an instance of A at t. By the definition of **is_a** (**Definition 9**), it follows that x is an instance of *Occurrent* at t. By **Axiom 11**, it follows that x is an instance of A for all time. □

Terms used to represent types within our formal system are simply labels chosen for their mnemonic usefulness. In practice, unique numeric identifiers are used for domain types (The Gene Ontology Consortium, 2008; Ruttenberg, 2009), leaving what they denote to be defined by formalisms and natural language parses.

Besides **is_a**, we define the relationship between a type and an immediate supertype (i.e., there being no other type "between" a type and its immediate supertype in the **is_a** hierarchy in which the types are included), with the **immediate_is_a** relation, as below:

Definition 10. $$\textbf{immediate\_is\_a}(A{,}B) =_{\text{def}} \textbf{is\_a}(A{,}B) \wedge A{\neq}B \wedge \forall C(\textbf{is\_a}(A{,}C) \wedge \textbf{is\_a}(C{,}B) \rightarrow A{=}C \oplus C{=}B)$$

immediate_is_a is irreflexive, intransitive, and asymmetric, and **is_a** is its transitive closure. From this definition it follows that:

Theorem 26. $$\textbf{immediate\_is\_a}(A{,}B) \rightarrow \exists xt(\textbf{instance\_of}(x{,}B{,}t) \wedge \neg\textbf{instance\_of}(x{,}A{,}t))$$

Proof. Given the identity of **is_a** (**Axiom 8**), if B is a subtype of A, then A and B are identical. But B cannot be a subtype of A, because by definition of **immediate_is_a** (**Definition 10**), $A{\neq}B$. B is not a subtype of A only if there is some particular x that instantiates B that does not instantiate A.

□

BFO's theory of types commits to the notion that two types are disjoint (i.e., have no instances in common) unless one is a subtype of the other (henceforth the 'Disjointness Principle'):

Disjointness Principle

Axiom 12. $\exists xt(\textbf{instance\_of}(x,A,t) \wedge \textbf{instance\_of}(x,B,t)) \rightarrow \textbf{is\_a}(A,B) \vee \textbf{is\_a}(B,A)$

Candidates that violate these principles are not types. Given **Axiom 4** and **Axiom 12**, we derive a theorem for the disjointness of types:

Theorem 27. $\textbf{is\_a}(A,B) \wedge \textbf{is\_a}(A,C) \rightarrow \textbf{is\_a}(B,C) \vee \textbf{is\_a}(C,B)$

Proof. Let A, B, and C be such that $\textbf{is\_a}(A,B)$ and $\textbf{is\_a}(A,C)$. From **Axiom 4** every type A has an instance, so there is some particular x that is an instance of both B and C. By the Disjointness Principle (**Axiom 12**), $\textbf{is\_a}(B,C)$ or $\textbf{is\_a}(C,B)$. □

It also follows that if there is an instance of a type A, and A is a subclass of B and C, if B and C are not related by **is_a**, then either B or C is not a type:

Theorem 28. $\exists xt(\textbf{instance\_of}(x,A,t)) \wedge \textbf{subclass\_of}(A,B) \wedge \textbf{subclass\_of}(A,C) \rightarrow \neg\textbf{is\_a}(B,C) \vee \neg\textbf{is\_a}(C,B) \rightarrow \neg\textbf{Type}(B) \vee \neg\textbf{Type}(C)$

Proof. If B and C are types, then by the definitions of **subclass** and **instance_of**, $\textbf{is\_a}(A,B)$ and $\textbf{is\_a}(A,C)$. Then by the disjointness of types (**Theorem 27**), it follows that $\textbf{is\_a}(B,C)$ or $\textbf{is\_a}(B,C)$. If we assume that neither hold, it follows that B or C is not a type. □

BFO was also developed under the assumption that no type has more than one direct supertype, henceforth the 'Single Inheritance Principle', which follows from the Disjointness Principle:

Single Inheritance Principle

Theorem 29. $\textbf{immediate\_is\_a}(A,B) \rightarrow (\textbf{immediate\_is\_a}(A,C) \leftrightarrow B{=}C)$

Proof. Let A, B, and C be such that $\textbf{immediate\_is\_a}(A,B)$ and $\textbf{immediate\_is\_a}(A,C)$. Therefore by the definition of **immediate_is_a** (**Definition** 10), $\textbf{is\_a}(A,B)$, $\textbf{is\_a}(A,C)$, $A{\neq}B$, and $A{\neq}C$. By the disjointness of types (**Theorem** 27), $\textbf{is\_a}(B,C)$ or $\textbf{is\_a}(C,B)$. By **Definition** 10 for $\textbf{is\_a}(A,B)$ and the disjunct $\textbf{is\_a}(B,C)$, it holds that $A{=}B$ or $B{=}C$. Because $A{\neq}B$, it holds that $B{=}C$. By **Definition** 10 for $\textbf{is\_a}(A,C)$ and the disjunct $\textbf{is\_a}(C,B)$, it holds that $A{=}C$ or $C{=}B$. Because $A{\neq}C$, it holds that $C{=}B$. Therefore $C{=}B$. □

From the definition of **immediate_is_a** and the *Disjointness Principle*, it follows that two types which have an instance and a direct parent in common are identical:

Theorem 30. $(\textbf{instance\_of}(x,A,t) \wedge \textbf{instance\_of}(x,B,t) \wedge \textbf{immediate\_is\_a}(A,C) \wedge \textbf{immediate\_is\_a}(B,C)) \rightarrow A{=}B$

Proof. Let x, t, A, B, and C be such that $\textbf{instance\_of}(x,A,t)$, $\textbf{instance\_of}(x,B,t)$, $\textbf{immediate\_is\_a}(A,C)$, and $\textbf{immediate\_is\_a}(B,C)$. By the definition of **immediate_is_a** (**Definition** 10) it then holds that $\textbf{isa}(A,C)$, $\textbf{isa}(B,C)$, $A{\neq}C$, and $B{\neq}C$. By this definition it also holds that if $\textbf{is\_a}(A,B)$ and $\textbf{is\_a}(B,C)$ then $A{=}B$ or $B{=}C$. Similarly, if $\textbf{is\_a}(B,A)$ and $\textbf{is\_a}(A,C)$ then $B{=}A$ or $A{=}C$. By the Disjointness Principle (**Axiom** 12), $\textbf{instance\_of}(x,A,t)$, and $\textbf{instance\_of}(x,B,t)$, it follows that $\textbf{is\_a}(A,B)$ or $\textbf{is\_a}(B,A)$. Because $\textbf{isa}(A,C)$ and $\textbf{is\_a}(B,C)$ hold, and $\textbf{is\_a}(A,B)$ or $\textbf{is\_a}(B,A)$ holds, $A{=}B$ or $B{=}C$ or $A{=}C$ follows. Due to $A{\neq}C$ and and $B{\neq}C$, $A{=}B$ follows. □

We can also show that:

Theorem 31. $(\exists A(\mathbf{is\_a}(A,B) \land \mathbf{is\_a}(A,C)) \land \exists D(\mathbf{immediate\_is\_a}(B,D) \land \mathbf{immediate\_is\_a}(C,D))) \rightarrow B{=}C$

Proof. Let A, B, C and D be such that $\mathbf{is\_a}(A,B)$, $\mathbf{is\_a}(A,C)$, $\mathbf{immediate\_is\_a}(B,D)$, and $\mathbf{immediate\_is\_a}(C,D)$. From **Axiom 4** every type A has an instance, so there is some particular x that is an instance of both B and C. By **Theorem 30**, $B{=}C$. □

Figure 6.1 illustrates a portion of the top seven levels of the BFO upper ontology type (or **is_a**) hierarchy. We propose that the subtyping relation between upper ontology types is **immediate_is_a**, which is a restricted version of **is_a**. We make this assumption given that the types of BFO's upper ontology fall within a finite domain. If BFO is changed such that for upper ontology types A and B a new upper ontology type C is placed between them in the BFO hierarchy, then it is a different ontology. Axioms corresponding to those illustrated are listed in Appendix D.
We define a relation for disjoint classes, $\mathbf{disjoint\_from_{class}}$:

Definition 11. $\mathbf{disjoint\_from_{class}}(A,B) =_{\text{def}} \forall xt(\mathbf{member\_of}(x,A,t) \rightarrow \neg\mathbf{member\_of}(x,B,t))$

We also define a relation for disjoint types. $\mathbf{disjoint\_from}(A,B)$ holds iff A and B do not share any instances at any time:

Definition 12. $\mathbf{disjoint\_from_{type}}(A,B) =_{\text{def}} \forall xt(\mathbf{instance\_of}(x,A,t) \rightarrow \neg\mathbf{instance\_of}(x,B,t))$

Theorem 32. $\mathbf{disjoint\_from_{type}}(A,B) \rightarrow \mathbf{Type}(A) \land \mathbf{Type}(B)$

Proof. By the definition of **instance_of** (**Definition 8**). □

A class that has as members members of disjoint types satisfies the unary predicate **Heterogeneous**:

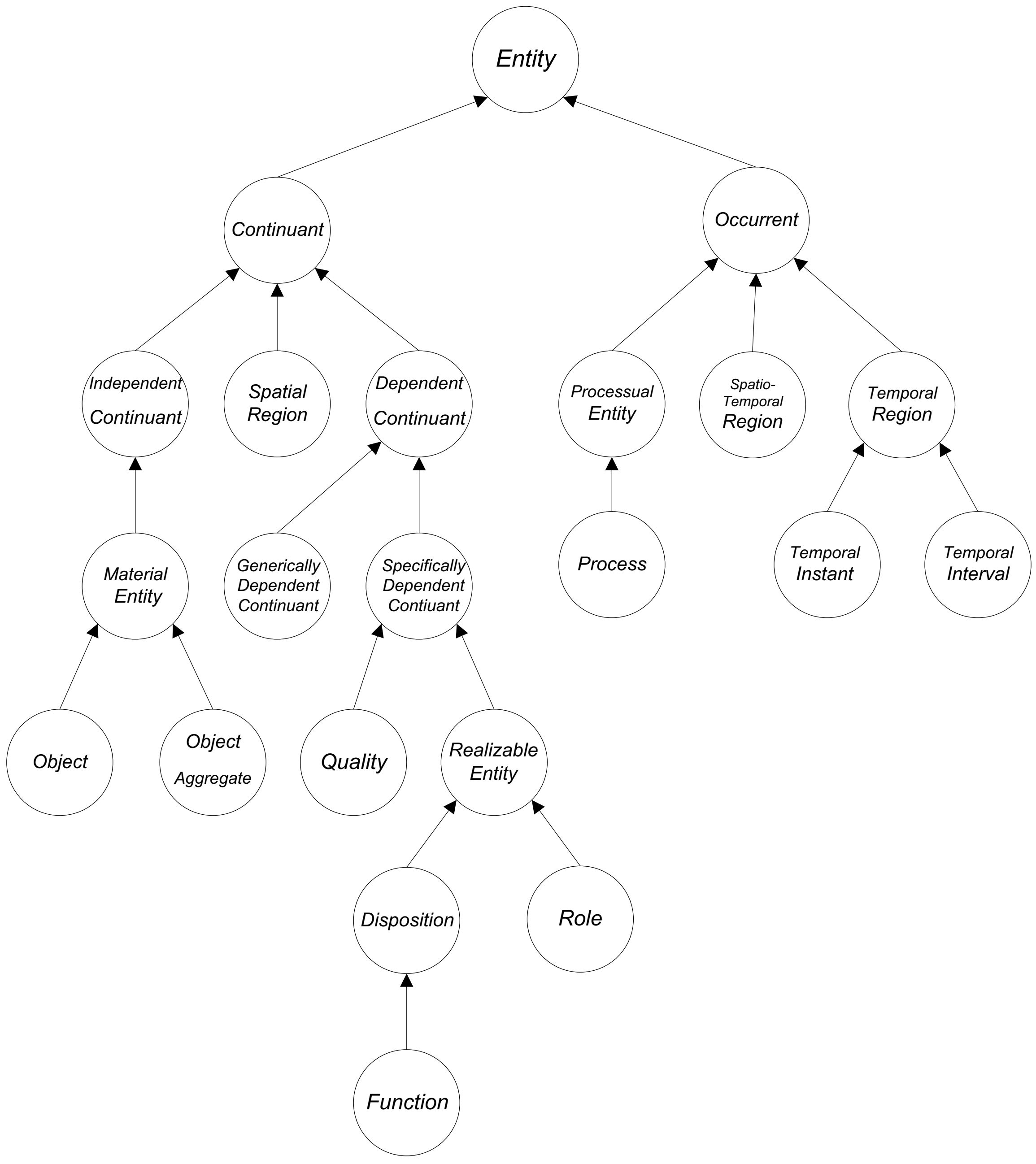


Figure 6.1: BFO Continuant Type **immediate_is_a** Hierarchy (Partial View)

Definition 13. $\textbf{Heterogeneous}(A) =_{\text{def}} \exists xBCt(\textbf{member\_of}(x,A,t) \wedge \textbf{member\_of}(x,B,t) \wedge \textbf{member\_of}(x,C,t) \wedge \textbf{disjoint\_from}_{\textbf{type}}(B,C))$

Based on this definition we can show that a heterogeneous class is an instantiated one:

Theorem 33. $\textbf{Heterogeneous}(A) \rightarrow \textbf{Instantiated}(A)$

Proof. Let A be such that **Heterogeneous**(A). x is an indefinite member of A at t and of two indefinite classes B and C at t, which are disjoint types, as entailed by **Theorem 32**. Every type satisfies **Members_Exist** (**Axiom 5**), which means that every member of B or C at some time exists at that time. Therefore, x exists at a time it is a member of A, and so A satisfies **Instantiated** (**Definition 2**). □

It follows that disjointness among types implies that the types do not share any members, due to the fact that all members of a class that is type are its instances:

Theorem 34. $\textbf{disjoint\_from}_{\textbf{type}}(A,B) \rightarrow \textbf{disjoint\_from}_{\textbf{class}}(A,B)$

Proof. Let A and B be such that $\textbf{disjoint\_from}_{\textbf{type}}(A,B)$, and also let x and t be such that **member_of**(x,A,t). By **Theorem 32**, it follows that**Type**(A) and **Type**(B) hold. By the definition of **instance_of** (**Definition 8**), it follows that **instance_of**(x,A,t). From the definition of $\textbf{disjoint\_from}_{\textbf{type}}$ (**Definition 12**) and **instance_of**(x,A,t), it follows that ¬**instance**(x,B,t). From the definition of **instance_of**, if **instance_of**(x,B,t) does not hold, then either **Type**(B) and **member_of**(x,B,t) (or both) do not hold. Because the former holds, it follows that ¬**member**(x,B,t). □

$\textbf{disjoint\_from}_{\textbf{class}}$ is non-reflexive (i.e., not irreflexive and not reflexive):

Theorem 35. $\neg\forall A(\neg\textbf{disjoint\_from}_{\textbf{class}}(A,A))$

Proof. As a reductio proof, we assume that there is no A such that **disjoint_from**(A,A). Because there is some (and only one) class which satisfies **Empty** (**Definition 3**), which is disjoint from itself because it has no members, there is a contradiction. ☐

Theorem 36. $\neg\forall A(\textbf{disjoint\_from}_{\textbf{class}}(A,A))$

Proof. As a reductio proof, let A be such that $\textbf{disjoint\_from}_{\textbf{class}}(A,A)$. It follows that $\textbf{disjoint\_from}_{\textbf{class}}(\mathit{Entity},\mathit{Entity})$. Since every type satisfies **Instantiated**, there is some x such that $\textbf{member\_of}(x,\mathit{Entity},t)$. By our assumption, this entails $\neg\textbf{member\_of}(x,\mathit{Entity},t)$, a contradiction. ☐

$\textbf{disjoint\_from}_{\textbf{type}}$ however is irreflexive:

Theorem 37. $\forall A(\neg\textbf{disjoint\_from}_{\textbf{type}}(A,A))$

Proof. As a reductio proof, let some A be such that $\textbf{disjoint\_from}_{\textbf{type}}(A,A)$. A is a type, and because every type satisfies **Instantiated**, there is some x and t such that $\textbf{instance\_of}(x,A,t)$. By our assumption it follows that $\neg\textbf{instance\_of}(x,A,t)$, a contradiction. ☐

$\textbf{disjoint\_from}_{\text{class}}$ is symmetric:

Theorem 38. $\textbf{disjoint\_from}_{\textbf{class}}(A,B) \rightarrow \textbf{disjoint\_from}_{\textbf{class}}(B,A)$

Proof. Let A and B be such that $\textbf{disjoint\_from}_{\textbf{class}}(A,B)$. From the definition of $\textbf{disjoint\_from}_{\textbf{class}}$ (**Definition 11**) it follows that every member of B at t is not a member of A at t. Therefore $\textbf{disjoint\_from}_{\textbf{class}}(B,A)$. ☐

$\textbf{disjoint\_from}_{\text{type}}$ is also symmetric:

Theorem 39. $\textbf{disjoint\_from}_{\textbf{type}}(A,B) \rightarrow \textbf{disjoint\_from}_{\textbf{type}}(B,A)$

Proof. Let A and B be such that $\textbf{disjoint\_from}_{\textbf{type}}(A,B)$. From the definition of $\textbf{disjoint\_from}_{\textbf{type}}$ (**Definition 12**) it follows that every member of B at t is not a member of A at t. Therefore $\textbf{disjoint\_from}_{\textbf{type}}(B,A)$. ☐

If a class A is a subclass of two disjoint classes, B and C, then A is empty:

Theorem 40. $\textbf{subclass\_of}(A,B) \wedge \textbf{subclass\_of}(A,C) \wedge \textbf{disjoint\_from}_{\textbf{class}}(B,C) \rightarrow \textbf{Empty}(A)$

Proof. Let A, B, and C be such that **subclass_of**(A,B), **subclass_of**(A,C), and $\textbf{disjoint\_from}_{\textbf{class}}(B,C)$. It follows that there is no x such that it is a member of B and C at some t. From this it follows that there is no x such that it is a member of A at some t. Therefore by definition A satisfies **Empty** (**Definition 3**). □

If it holds that $\textbf{disjoint\_from}_{\textbf{type}}(A,B)$, we infer that all subtypes of A are disjoint from B:

Theorem 41. $\textbf{disjoint\_from}_{\textbf{type}}(A,B) \wedge \textbf{is\_a}(C,A) \rightarrow \textbf{disjoint\_from}_{\textbf{type}}(C,B)$

Proof. Let A, B, and C be such that $\textbf{disjoint\_from}_{\textbf{type}}(A,B)$ and **is_a**(C,A). By **is_a**(C,A) every instance of C is also an instance of A (**Definition 9**), and by $\textbf{disjoint\_from}_{\textbf{type}}(A,B)$ no instance of A is an instance of B (**Definition 12**). Therefore no instance of C is an instance of B, hence $\textbf{disjoint\_from}_{\textbf{type}}(C,B)$. □

and from **Definition 12** and **Theorem 41** we further derive that two subtypes of two respective disjoint types are also disjoint:

Theorem 42. $\textbf{disjoint\_from}_{\textbf{type}}(A,B) \wedge \textbf{is\_a}(C,A) \wedge \textbf{is\_a}(D,B) \rightarrow \textbf{disjoint\_from}_{\textbf{type}}(C,D)$

Proof. By application of **Theorem 42** for $\textbf{disjoint\_from}_{\textbf{type}}(A,B)$ and **is_a**(C,A), $\textbf{disjoint\_from}_{\textbf{type}}(C,B)$ holds. By application of **Theorem 42** for $\textbf{disjoint\_from}_{\textbf{type}}(C,B)$ and **is_a**(D,B), $\textbf{disjoint\_from}_{\textbf{type}}(C,D)$ holds. □

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

Theorem 43. $\textbf{immediate\_is\_a}(A,C) \wedge \textbf{immediate\_is\_a}(B,C) \wedge A{\neq}B \rightarrow \textbf{disjoint\_from}_{\textbf{type}}(A,B)$

Proof. Let A, B, and C be such that **immediate_is_a**(A,C), **immediate_is_a**(B,C), and $A \neq B$. By the definition of **immediate_is_a**, it holds that **is_a**(A,C), **is_a**(B,C), $A \neq C$ and $B \neq C$. As a reductio proof, let's also assume that ¬**disjoint_from$_{\mathbf{type}}$**(A,B) holds, which implies there is some particular x that is a member of A and a member of B at some time t. Due to the Disjointness Principle (**Axiom 12**), **is_a**(A,B) or **is_a**(B,A). If **is_a**(A,B) holds, then by the definition of **immediate_is_a** (**Definition 10**), $A=B$ or $B=C$ holds. Because neither holds, it does not hold that **is_a**(A,B). If **is_a**(B,A) holds, then by **Definition 10**, $B=A$ or $A=C$ hold. Because neither hold, it does not hold that **is_a**(B,A). Since neither **is_a**(A,B) nor **is_a**(B,A) hold, there is a contradiction. Therefore it holds that **disjoint_from$_{\mathbf{type}}$**(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 **is_a**, are disjoint types:

Theorem 44. (**is_a**(A,B) ∨ **is_a**(B,A)) ⊕ **disjoint_from$_{\mathbf{type}}$**(A,B)

Proof. From the Disjointness Principle (**Axiom 12**) we have that $\forall xt$(¬**instance_of**(x,A,t) ∨ ¬**instance_of**(x,B,t)) ∨ **is_a**(A,B) ∨ **is_a**(B,A) which we can transform into $\forall xt$(**instance_of**(x,A,t) → ¬**instance_of**(x,B,t)) ∨ **is_a**(A,B) ∨ **is_a**(B,A). From the definition of **disjoint_from$_{\mathbf{type}}$** (**Definition 12**), it follows that
disjoint_from$_{\mathbf{type}}$(A,B) ∨ (**is_a**(A,B) ∨ **is_a**(B,A)). Because disjoint types cannot share instances, and vice versa, the outer disjunction is exclusive. □

By applying **Theorem 43** we can prove that sibling BFO upper ontology types are disjoint. For instance, *Continuant* and *Occurrent* are disjoint types, as are *IndependentContinuant* and *DependentContinuant*. (We list the disjointness theorems for sibling types of BFO in Appendix D.2 (**A59**-**A70**). We can also prove that non-sibling upper ontology types not related by **is_a** are disjoint, by applying **Theorem 41** and **Theorem 39**.
Finally, **disjoint_from$_{\mathbf{type}}$** is non-transitive (i.e., not transitive or intransitive):

Theorem 45. $\neg\forall ABC(\mathbf{disjoint\_from_{type}}(A,B) \wedge \mathbf{disjoint\_from_{type}}(B,C) \rightarrow \mathbf{disjoint\_from_{type}}(A,C))$

Proof. As a reductio proof, we assume the negation of our theorem. Therefore if *Continuant* is disjoint from *Occurrent*, and *Occurrent* is disjoint from *Continuant*, then *Occurrent* is disjoint from *Occurrent*. Because *Continuant* is disjoint from *Occurrent*, and due to the symmetry of $\mathbf{disjoint\_from_{type}}$ (**Theorem 26**) the inverse also holds, *Occurrent* is disjoint from *Occurrent*. Since every type has an instance (**Theorem 12**), it is not the case that *Occurrent* is disjoint from *Occurrent*, a contradiction. □

Theorem 46. $\neg\forall ABC(\mathbf{disjoint\_from_{type}}(A,B) \wedge \mathbf{disjoint\_from_{type}}(B,C) \rightarrow \neg\mathbf{disjoint\_from_{type}}(A,C))$

Proof. As a reductio proof, we assume the negation of our theorem. Therefore if *IndependentContinuant* is disjoint from *DependentContinuant* and *DependentContinuant* is disjoint from *Occurrent*, then *IndependentContinuant* is not disjoint from *Occurrent*. Both conjuncts hold, therefore *IndependentContinuant* is not disjoint from *Occurrent*, which is false. □

Ontologies that do not observe the principles of disjointness and single inheritence are often considered ill-formed (Smith, 2003), and are even described as “tangled” (Welty and Guarino, 2001). Rector (2003) recommends a similar approach to enable modularization. He recommends that an ontology specified in a Description Logic should avoid multiple inheritance in its asserted hierarchy.

BFO’s upper ontology was designed with these principles in mind. An upper ontology like BFO provides basic, or upper ontology types with the intent that the types of a domain ontology are subtypes of them. It is also the design principles of BFO which are reflected in its axioms that must also be applied in order for the rooting’s benefit to be apparent to the ontology modeler.

Given this important role that BFO plays in shaping ontologies, there is a potential and also a need to evaluate the correctness of ontologies that are proposed to be domain ontologies using BFO. This role can be played by evaluating candidate domain types in a manner that determines their rooting under BFO.[5] Further, formalisms of candidate types can be checked for consistency with BFO axioms. (Smith and Ceusters, 2010, p.143) suggests that a decision procedure that determines which classes are types would be valuable to the community, but that this determination is an ongoing process. Our method for evaluating and standardizing candidate types is a procedure which we assert in that process.

By rooting domain types in those of BFO, what particulars the types are asserted to refer becomes. It is a necessary task to be undertaken for any domain ontology, in order for BFO to serve as a facilitator of interoperability, one that is ideally undertaken during the design of the domain ontology, but one that can also occur afterwards. We discuss this process in detail in Chapter 8.

6.2 Relation Ontology

After the development of the BFO **is_a** hierarchy, it was determined that additional relations were needed, as evidenced by the **exists_at** relation. BFO was later complemented with a core set of primitive relations, formalized within the Relation Ontology (RO) (Smith et al., 2005).(RO is integrated into the next version of BFO, 2.0, but we consider version 1.1 stable and "frozen" for our research.)

BFO is influenced by Aristotle's work in many respects, including the division of particulars into substances and accidents, which BFO refers to as independent continuants and dependent continuants, respectively. An example of an independent continuant is a specific chunk of wood; an example of a dependent continuant is the texture of that chunk of wood. The texture is given

[5]By 'rooting' under BFO we mean that each candidate type is subtyped under a leaf BFO type, i.e., a type that has no additional subtypes in BFO.

the status of a particular in BFO, but conditionally so. The texture of the wood only exists when the chunk of wood exists. The texture exists insofar as the chunk does; in this respect, the texture "depends on" it. We hereby formalize the **depends_on** relation, **depends_on**(x,y,t), which means that particular x depends on particular y at a time t (we axiomatize this notion below). (We also refer to y as the bearer of x.) This relation commits x to being a dependent continuant particular and y to being an independent continuant particular:

Axiom 13. **depends_on**(x,y,t) $\rightarrow$
instance_of(x,*DependentContinuant*,t) $\wedge$
instance_of(y,*IndependentContinuant*,t)

The dependence relation between a dependent and independent continuant only holds at a time at which they both exist:

Theorem 47. **depends_on**(x,y,t) $\rightarrow$ **exists_at**(x,t) $\wedge$ **exists_at**(y,t)

Proof. By **Axiom 13**, the definition of **instance_of** (**Definition 8**), **Axiom 5**, and the definition of **Members_Exist** (**Definition 4**). □

This entails that if x or y does not exist at time t, then **depends_on**(x,y,t) does not hold at that time.

As mentioned in our example, a dependent continuant particular only exists when the independent continuant particular it depends on exists. Therefore, if a dependent continuant particular x depends on an independent continuant particular y at a time t, then the existence of x at time t_1 implies the existence of y at the same time, t_1:

Axiom 14. $\exists t$(**depends_on**(x,y,t)) $\rightarrow \forall t_1$(**exists_at**(x,t_1) $\rightarrow$
exists_at(y,t_1))

(Note that in BFO, the existence of dependent continuants does not depend on other dependent continuants.) Relative to our example, if our chunk of wood ceases to exist, so does its texture. Dependent continuants cannot migrate from one bearer to another, therefore:

Axiom 15. $\exists t t_1(\textbf{depends\_on}(x,y,t) \wedge \textbf{depends\_on}(x,z,t_1)) \rightarrow y{=}z$

Bittner and Donnelly (2005, p.3) refer to this as the *single-immediate-successor* (logical) property. (This relation would be functional in the set-theoretic sense if specified as a binary, atemporal relation.)

This kind of dependence we refer to as *specific* dependence. Another kind of dependence is *generic* dependence. It is a relationship that holds between a dependent continuant particular and an independent continuant type (instead of an independent continuant particular). Generic dependence is such that if one bearer of the dependent continuant ceases to exist, then the dependent continuant can survive only if there are other bearers. For example, a specific PDF document is generically dependent on some instance of the type *Storage Medium* since there has to be at least one storage medium on which the PDF document is stored. If the PDF document is erased, for example from a hard drive, it survives only if it is stored on other media, for example a backup hard drive or USB key. (One might rightfully suggest that the PDF document is also generically dependent on some file saved on the storage medium.) **generically_depends_on**(x,A,t) means that particular x is generically dependent on type A at time t. This relation commits x to being a generically dependent continuant particular and A to being an independent continuant type:

Axiom 16. $\textbf{generically\_depends\_on}(x,A,t) \rightarrow$
$\textbf{instance\_of}(x, \mathit{GenericallyDependentContinuant},t) \wedge$
$\textbf{is\_a}(A, \mathit{IndependentContinuant})$

There are several kinds of specifically dependent continuants. There are qualities and also realizable entities. A quality is exhibited at a time if it inheres in an independent continuant that exists at that time. Although the biconcave disc shape of a red blood cell is dependent on its cell, the shape is more specifically a quality of it. Realizable entities include primarily roles and functions.

Roles are externally-grounded realizable entities, and "exist because the bearer is in some special physical, social, or institutional set of circumstances" (Arp and Smith, 2008), in which the bearer does not have to be. They are not such that if they cease to exist, then the physical make-up of their bearers is thereby changed. An example of a role is that of a student, which is the role of a person when she is enrolled and participating in courses. Another example of a role is that of reporter gene, which is a role of the green florescent protein gene (GFP) when it is fused to a promoter of the gene of interest in a genetically engineered model (Phillips, 2001).

Functions, in BFO's use of the term, which is in a specific teleological sense, are realizable entities that are internally grounded. A disposition is "a realizable entity which if it ceases to exist, then its bearer is physically changed, and whose realization occurs when this bearer is in some special physical circumstances, in virtue of the bearer's physical make-up" (Arp and Smith, 2008).

A function exists in virtue of the bearer's make-up, and this physical make-up is something the bearer possesses because it came into being, either through evolution (in the case of natural biological entities) or through intentional design (in the case of artifacts) in order to realize processes of a certain kind. An example of a function is to pump blood, which is a function of any human heart. Another example is to withstand protein denaturing conditions, which is a function of taq polymerase in polymerase chain reaction (Saiki et al., 1988). Although there are separate RO relations for the relationships of qualities, roles, and functions with independent continuants, there no additional formal distinctions between them. Given that, for the purposes of our method we use only the **depends_on** relation, which holds for all specifically dependent continuants.

The RO primitive relations also include **part_of**, **located_at**, and **participates_in**. **part_of**(x,y,t) means that particular x is a part of particular y at time t. **part_of** is transitive, anti-symmetric, and reflexive. As with **instance_of**, the **part_of** relation with respect to time is different for occurrents:

Axiom 17. $$\mathbf{is\_a}(A, \mathit{Occurrent}) \leftrightarrow \forall x(\exists t(\mathbf{part\_of}(x,A,t)) \rightarrow \forall t(\mathbf{part\_of}(x,A,t)))$$

Relation	arg1	arg2	arg3
depends_on	*Dependent Continuant*	*Independent Continuant*	*Temporal Region*
part_of	*Particular*	*Particular*	*Temporal Region*
participates_in	*Continuant*	*Occurrent*	*Temporal Region*
located_in	*Continuant*	*Spatial Region*	*Temporal Region*

Figure 6.2: Instance-Level Relations of Relation Ontology

For any x that fully exists at some temporal instance t, any part of x at t also exists at t:

Axiom 18. $\textbf{exists\_at}(x,t) \rightarrow \forall y(\textbf{part\_of}(y,x,t) \rightarrow \textbf{exists\_at}(y,t))$

located_in(x,y,t) means that x is located in particular y at t. Here x is committed to being a continuant particular and y is committed to being a spatial region particular:

Axiom 19.
$$\textbf{located\_in}(x,y,t) \rightarrow$$
$$\textbf{instance\_of}(x,\mathit{Continuant},t) \wedge$$
$$\textbf{instance\_of}(y,\mathit{SpatialRegion},t)$$

participates_in(x,y,t) is a primitive relation between a continuant x, a process y, and a time t at which x participates in some way in the y (Smith et al., 2005, p. 10):

Axiom 20.
$$\textbf{participates\_in}(x,y,t) \rightarrow$$
$$\textbf{instance\_of}(x,\mathit{Continuant},t) \wedge$$
$$\textbf{instance\_of}(y,\mathit{Process},t)$$

For example, a cell participates in a process of cell transport. Figure 6.2 displays the RO instance-level relations discussed and the types of their arguments. Discussions of other RO instance-level relations are provided in (Mungall, 2007).

Besides generic dependence, we also define other type-level relations to formalize relationship patterns between particulars of two types. Recall that the **depends_on** relation represents the relationship between a dependent continuant and an independent continuant at a time. At the type level, for every instance of A, **Depends_On**(A,B) means that there is some instance of B where the former instance depends on the latter instance:

Definition 14. $\textbf{Depends\_On}(A,B) =_{\text{def}}$

$$\forall xt(\textbf{instance\_of}(x,A,t) \rightarrow \exists y(\textbf{instance\_of}(y,B,t) \wedge \textbf{depends\_on}(x,y,t)))$$

For example, every instance of supplying cellular energy depends on (or more specifically, is a function of) some mitochondrion. We can formalized this as **Depends_On**$(\mathit{SupplyingCellularEnergy},\mathit{Mitochondrion})$.

The **part_of** relation represents the parthood relationship between particulars, for instance a particular appendix, *appendix03*, is a part of a human body, *body022*, at some time, formalized: $\exists t(\textbf{part\_of}(\mathit{appendix03},\mathit{body022},t))$ where **instance_of**$(\mathit{appendix03},\mathit{Appendix})$ and **instance_of**$(\mathit{body022},\mathit{Body})$. Another type-level relation is that for parthood. For every instance of type A **Part_Of**(A,B) means that there is some instance of type B where the former instances is a part of the latter instance (Smith and Rosse, 2004, p.445):

As another type level relation for parthood, for every instance of type B **Has_Part**(B,A) means that there is some instance of type A where the former instance has the latter instance as a part of it (Smith and Rosse, 2004, p.445).

Definition 16. $\textbf{Has\_Part}(B,A) =_{\text{def}}$

$$\forall yt(\textbf{instance\_of}(y,B,t) \rightarrow \exists z(\textbf{instance\_of}(z,A,t) \wedge \textbf{part\_of}(z,y,t)))$$

Note that these two type-level relations for parthood are not converses. The relational distinction provided in the previous two definitions is useful when considering cases where only one of these relations holds. Given the previous definition, we can represent the relationship between types *Human* and *Heart* as **Has_Part**$(\mathit{Human},\mathit{Heart})$. Notice that **Part_Of**$(\mathit{Heart},\mathit{Human})$ does not hold since there are non-humans that have hearts (Smith and Rosse, 2004, p.445). Discussions of other RO type-level relations are provided in (Mungall, 2007).

Type-level relations like **Depends_On**, **Part_Of**, and **Has_Part** are different from the type-level relation **is_a** in that the formalization of **is_a** concerns the generalization of types, while the type-level parthood relations describe the relationship of a particular of one type with some particular of another type. It is also the case that these type-level relations are mutually exclusive with **is_a**. For example if we have the following formula, **Part_Of**(*Prostate*,*Male_Genital_System*), which means that every prostate is a part of a male genital system, then it is not the case that **is_a**(*Prostate*,*Male_Genital_System*). Conversely, given the mutual exclusivity, if for some types A and B the **is_a** relation holds, then none of the type-level RO relations hold. Mutual exclusivity among the type-level RO relations is enforced through type restrictions.

6.3 Applying BFO Distinctions to Properties

As discussed in Chapter 5, there are some predicates$_{\text{lin}}$, like 'has mass' that make reference to entities that BFO deems particulars, that is dependent continuants. Consider another example, a specific pile of sugar on a table, which we refer to as *MySugar*. Consider also two properties *Sugar* and *Hydrophilic*. We assume the property *Sugar* has as instances[6] those entities that are "edible crystalline carbohydrates [...] characterized by a sweet flavor." (Campbell and Reece, 2007, p.70), and *Hydrophilic* those that "[have] an affinity for water [and] can transiently bound with water through hydrogen bonding" (Campbell and Reece, 2007, p.51). Oftentimes in ontologies such natural language definitions are not given and are treated as implicit for the general term provided. By applying the property-centric approach we assert that *MySugar* is an instance of, or has the property, *Sugar* and *Hydrophilic*. The latter is a property that any amount

[6]We remind the reader that the sense of 'instance' and 'property' is given in Chapter 1 and is based on Guarion and Welty. Therefore *Sugar* and *Hydrophillic* are shortcuts for *being sugar* and *being hydrophillic*. 'instance' is used differently in BFO, and 'property' is used different in philosophy at large.

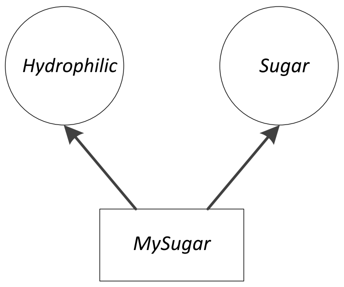


Figure 6.3: Initial Approach for Relationships of *MySugar*, *Sugar*, and *Hydrophilic*

of sugar has.

Given that we have “unified” properties and types with classes, and that classes under evaluation are treated as candidate types, we therefore impose that *Sugar* and *Hydrophilic* are denoting types, and the generalized classification relation is **instance_of**. We are therefore left with an ontology portion of the following formulas, which represent the asserted relationships between these three entities (also illustrated in Figure 6.3):

- **instance_of**(*MySugar*,*Sugar*)
- **instance_of**(*MySugar*,*Hydrophilic*)

Because *Sugar* and *Hydrophilic* are herein proposed as “candidate” BFO types, they must be evaluated as such. From the BFO perspective, the assignment that the particular *MySugar* is an instance of *Sugar* is correct because the class *Sugar* defines what *MySugar* is essentially. *MySugar* exists as a sugar: any time in which *MySugar* exists, it is an instance of *Sugar*.

The relationship between *MySugar* and *Hydrophilic* is however different. According to BFO there is a specific attribute of *MySugar*, its affinity for water, that is itself a particular, a dependent continuant. Henceforth, we denote this particular *HydrophilicityOfMySugar*.

However as *Hydrophilic* is defined its instances includes amounts of sugar, which are material entities, not dependent continuants like *HydrophilicityOfMy-*

Sugar. For *Hydrophilicity* to classify dependent continuants, we would need to reclarify our assumption for its definition from "having an affinity for water" to "an affinity for water". From the latter information definition it follows that *HydrophilicityOfMySugar* is an instance of *Hydrophilic*.

Strawson (1959, p.168), whose explication of sortals inspired OntoClean, illustrates these same distinctions. When someone speaks of characterizing an object, says of something what it is, and attributes something to something else, these are described as characteristic tie, universal tie, and attributive tie, respectively. Applied to our current example, the relationship between *MySugar* and *Hydrophilic* is a characteristic tie, that between *MySugar* and *Sugar* is a universal tie, and that between *HydrophilicityOfMySugar* and *MySugar* is an attributive tie.

These relationships correspond to what we will define shortly as **exemplifies** (Smith, 2005, p. 168), and also the **instance_of** and **depends** relations, respectively. A crucial distinction for modeling a domain is that **instance_of** is used for classification and **exemplifies** is used for characterization.

exemplifies is not included in RO, due to its not being a primitive relation, but we use it here to denote the oft-ignored relation between an independent continuant particular and a dependent continuant type of which the independent continuant bears some instance. We define **exemplifies** as follows:

Definition 17.
$$\begin{aligned}\textbf{exemplifies}(x,A,t) =_{\text{def}} \\ &\exists y(\textbf{depends\_on}(y,x,t) \wedge \\ &\quad \textbf{instance\_of}(y,A,t)) \wedge \\ &\quad \textbf{is\_a}(A,\mathit{SpecificallyDependentContinuant})\end{aligned}$$

exemplifies(x,A,t) means that independent continuant particular x exemplifies the dependent continuant type A at time t. It is defined such that there is some dependent continuant particular y which depends on x at time t at which it instantiates A, a dependent continuant type.

With this relation we represent the relationship between *MySugar* and *Hy-*

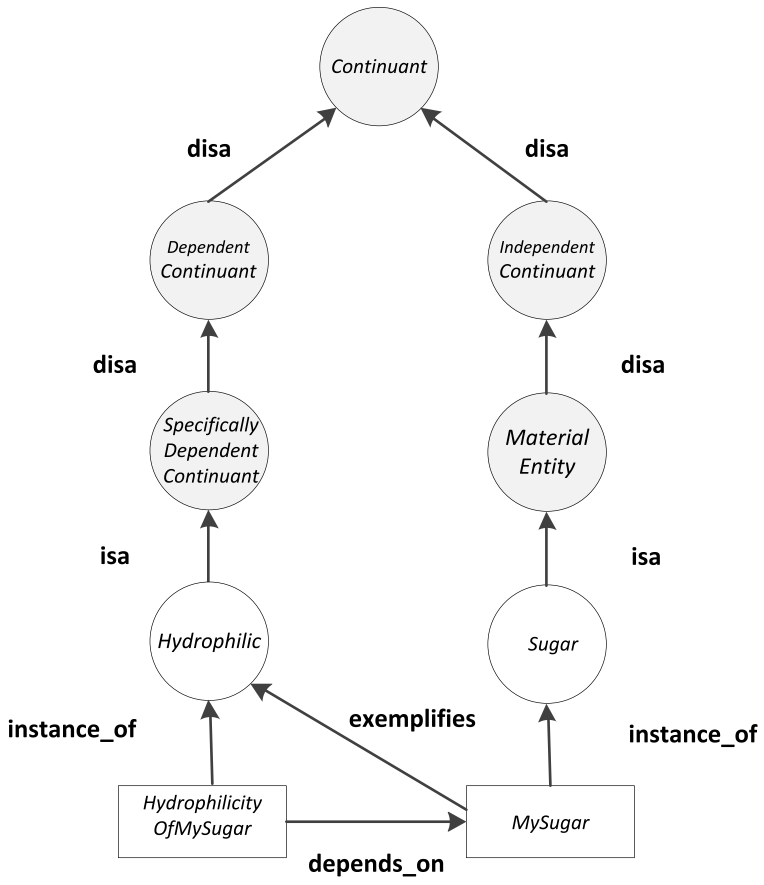


Figure 6.4: BFO Relationships of *MySugar*, *Sugar*, and *Hydrophilic*

drophilic directly in a formula (below), as originally proposed. We formalize the three mentioned relationships of *MySugar* and the classes *Sugar* and *Hydrophilic*, as follows:

- $\exists t$(**instance_of**(*MySugar*, *Sugar*,t))
- $\exists t$(**instance_of**(*HydrophilicityOfMySugar*,*Hydrophilic*,t)
- $\exists t$(**depends_on**(*HydrophilicityOfMySugar*, *MySugar*,t))
- $\exists t$(**exemplifies**(*MySugar*,*Hydrophilic*,t))

The remaining relationships holding for the entities discussed are those that require a rooting in the BFO upper ontology, which is as follows:

- **is_a**(*Hydrophilic*, *SpecificallyDependentContinuant*)
- **is_a**(*Sugar*, *MaterialEntity*)

We illustrate each of these relationships in Figure 6.4. In this figure, each directed arc represents a relationship and each circle node represents a type and each square node represents a particular. The darkened nodes represent BFO upper ontology types.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 7

Integrating Unity and Identity with BFO

7.1 Integrating Unity with BFO

7.1.1 Introduction

The matter of Unity set forth by Guarino and Welty concerns, for the objects with a given property, whether there is a relation that holds only between the parts of the object and nothing else. In what follows we discuss the perspectives of several figures in the field of metaphysics on this topic, and then provide an analysis of the OntoClean notions of Unity and Unity criteria.

According to Bunge (1979), *an aggregate* is a collection of things not held together by "bonds", and therefore lacking in integrity or unity. Bunge states that the parts of a concrete aggregate (as opposed to a conceptual one) are not coupled, linked, connected, or bonded, such as a random sample of a biological population.

In contrast, for a *whole*, its parts are interrelated rather than loose. By his account, in order to distinguish between objects that are wholes and those that are aggregates, we must distinguish what between their parts is a "mere relation" (e.g., older than, bigger than, travelling towards), and a "connection" (e.g., exerting pressure). Two things are connected if at least one of them acts upon the other, i.e., if it in some way modifies the latter's history. Under this criterion, a whole is permitted to have spatially separated parts. Because of this causal connection, the history of a whole is not merely the union of the history of its parts, whereas the history of an aggregate is such a union.

In an opposing position Schlick (1965) argues that there is no ontological difference between aggregates and wholes, and that they merely reflect two modes of representation of the same objects. Simons (1987, p. 324) however suggests that a theory which considers the aggregate of two arbitrary objects as one object in all cases, is to be rejected as weak. Simons (1987, p. 324) states that the difference between purely formal ontological aggregates and wholes can be understood by comparing their existence conditions. Aggregates exist just in case the constituent parts exist. In contrast, for a whole to exist a further constitutive condition, or *unifying condition*, among the constituent parts must be fulfilled. We discuss Guarino and Welty's account of this condition, explicated as a relation held among certain parts of a whole, in what follows.

7.1.2 Reformulating OntoClean's Notion of Unity for the Formal Theory of Classes

OntoClean's notion of Unity is heavily influenced by Simons (1987). By Welty and Guarino's account, Unity is a metaproperty that a modeler assigns to properties to help distinguish, for each instance of the property, it's parts from the rest of the world. If a property has Unity, this means there is some *unifying relation* that binds together certain parts of each instance of that property such that the parts compose the whole object. When a modeler tries to identify a unifying relation that applies to each object of a property, she is trying to answer: *What are the parts such that they form a whole*? Guarino and Welty (2000b, p.3) cite Simons when offering an explanation of what it means for an object to be a whole, which is as follows:[1]

> Every member of some division of the object stands in a *certain relation* to every other member, and no member bears this relation to anything other than members of the division. (Simons, 1987, p. 327)

Simons emphasizes that this certain relation holds only among parts of *a certain division*. These parts form a whole system. For example, in a human's skeletal system there is such a division of its parts; the parts (i.e., bones) form a path that is connected by joints. Therefore the certain relation is *connected by joints*.

Simons also emphasizes that this certain relation does not hold among arbitrary parts of the whole. There are arbitrary divisions of a human's skeletal system such that the parts of the division are not related by *connected by joints*. For example, in each human finger a distal phalanx bone and a intermediate (i.e., middle) phalange bone are connected by a distal interphalangeal joint. If there is a division where the sum of the distal phalange and distal interphalangeal joint is one part of that division and the intermediate phalange is another part, then that summed part is not in the *connected by joints* relation with all other parts, specifically not the intermediate phalange part.

Guarino and Welty (2000b, p. 3) apply Simons' (1987) theory for a *closed system* to their theory of Unity. An object x is closed under a relation r, or simply *r-closed*, iff, if y is a part of x, and if y is in the r relation to z, then z is a part of x.[2] Although not cited, Guarino and Welty also apply Simons' theory of a *connected system* (Simons, 1987) to their theory of Unity. An object x is connected under a relation r, or *r-connected* iff, if y and z are a part of x, then

[1]Note that 'member of' is used here to provide in set-theoretic terms the relationship between a part of an object and the object, which is different from our standard usage as the relationship between a particular and a class.

[2]Relations introduced from other works that are not or not yet included in our formal system are introduced in italics and their respective axioms are introduced in plain English.

y and z are related by r.

Guarino and Welty apply these notions to what it is to be an integrated whole (although they do not give it formally for Unity):

> An object x is a (contingent) **r**-integrated whole if there exists some division of x such that it is a closed system. **r** will be called a base unifying relation for x (Guarino and Welty, 2000b, p.3).

(The use of 'contingent' here means "at some time" and is with respect to one snapshot in time only.) In more recent work Guarino and Welty (2001, p.10) say that x is "whole under r" and the definition above regarding "some division" is omitted altogether. In both (Guarino and Welty, 2000b) and (Guarino and Welty, 2001), they provide a definition for what it means for an object to be whole under a unifying relation, ω.

If x is *whole under* ω at t, then if y is a part of x at t and z is a part of x at t then y and z are in the ω relation at t; furthermore, if y is a part of x at t, and y and z are in the ω relation at t, then z is a part of x at t.

Therefore a unifying relation ω is reflexive, symmetric, and transitive, i.e., an equivalence relation.[3] The notion of "some division" is omitted in this definition, however it is crucial to the theory because it implies that ω does not hold between just any parts.

They define that x is an *intrinsic whole* under ω if for all times x exists it is *whole under* t. Applying this notion to properties, property ϕ is *unified under* ω iff for each instance of ϕ it is an *intrinsic whole* under ω. Kaplan (2001) proves that if ϕ is *unified under* ω, then the instances of ϕ are non-overlapping wholes, i.e., they do not partially overlap with other entities with the same property.

Guarino and Welty define three categories of properties based on these notions: *Unity*, *Non-Unity*, and *Anti-Unity*. A property has *Unity* if there is a relation it is unified under, a property has *Non-Unity* if there is no one relation it is unified under, and finally, a property has *Anti-Unity* if there is no instance

[3]Clearly these are properties of binary relation; as discussed previously, we assume that the relation is between two entities at some time represented in the third argument.

of the property that is an intrinsic whole under some relation.

Guarino and Welty (2000b)'s theory of Unity includes a purported non-triviality stipulation, that it is not the case that there is a universal unifying relation such that every object is an intrinsic whole under it. Kaplan (2001) shows that this axiom does not accomplish its intent; it rules out a universal unifying relation, but also allows for an infinite number of other unifying relations that are trivially true.

We reformulate the notion that a property has Unity, i.e., that it has a unifying relation, under our theory of classes. We provide a definition schema for introducing the meta-predicate **Unified_Under**:

Definition Schema 1. $\textbf{Unified\_Under}(A,\omega,\mathbf{p}) =_{\text{def}}$

$$\begin{aligned}
&\forall x(\exists t(\textbf{member\_of}(x,A,t)) \rightarrow\\
&\quad \forall t(\textbf{exists\_at}(x,t) \rightarrow\\
&\quad\quad \forall y(\mathbf{p}(y,x,t) \rightarrow\\
&\quad\quad\quad \forall z(\mathbf{p}(z,x,t) \leftrightarrow\\
&\quad\quad\quad\quad \omega(z,y,t))))) \land\\
&\forall wvt(\mathbf{p}(w,v,t) \rightarrow \textbf{part\_of}(w,v,t)) \land\\
&\neg\forall wvt(\textbf{part\_of}(w,v,t) \rightarrow \mathbf{p}(w,v,t))
\end{aligned}$$

We elucidate why the lattermost conjunct holds in **Section 7.3**.

As in any schemata, the constants which are applied, i.e., the constants that take the place of A, ω, and $\mathbf{p}$ replace occurrences of A, ω, and $\mathbf{p}$ of the wffs in the definiens (i.e., right hand side) of the definition schema. As a definition schema, A serves as a meta-variable that represents any particular class and ω and $\mathbf{p}$ serve as meta-variables that represent any particular relations.

Unifying relations have a transitive, symmetric, and reflexive nature; for a class A that is unified under ω with respect to parthood subrelation $\mathbf{p}$, it follows that:

Metatheorem 1. $\textbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\forall x(\exists t(\textbf{member\_of}(x,A,t)) \rightarrow$$
$$\forall t(\textbf{exists\_at}(x,t) \rightarrow$$
$$\forall y(\mathbf{p}(y,x,t) \rightarrow$$
$$\forall zw(\omega(y,z,t) \wedge \omega(z,w,t) \rightarrow \omega(y,w,t)))))$$

Proof. Let A, ω, and $\mathbf{p}$ be such that $\textbf{Unified\_Under}(A,\omega,\mathbf{p})$. Let w, x, y, z, t, and A be such that $\textbf{member\_of}(x,A,t)$, $\textbf{exists\_at}(x,t)$, $\mathbf{p}(y,x,t)$, $\omega(y,z,t)$, and $\omega(z,w,t)$. By $\mathbf{p}(y,x,t)$ and $\omega(y,z,t)$ and our definition schema for **Unified_Under (Metatheorem 1)** it follows that $\mathbf{p}(z,x,t)$. Because $\omega(z,w,t)$, and also $\mathbf{p}(z,x,t)$, via the definition schema for **Unified_Under**, it follows that $\mathbf{p}(w,x,t)$. Due to $\mathbf{p}(w,x,t)$ and $\mathbf{p}(y,x,t)$ and the definition schema for **Unified_Under**, it follows that $\omega(y,w,t)$. □

Metatheorem 2. $\textbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\forall x(\exists t(\textbf{member\_of}(x,A,t)) \rightarrow$$
$$\forall t(\textbf{exists\_at}(x,t) \rightarrow$$
$$\forall y(\mathbf{p}(y,x,t) \rightarrow$$
$$\forall zw(\omega(y,z,t) \rightarrow \omega(z,y,t)))))$$

Proof. Let A, ω, and $\mathbf{p}$ be such that $\textbf{Unified\_Under}(A,\omega,\mathbf{p})$. Let x, y, z, t, and A be such that $\textbf{member\_of}(x,A,t)$, $\textbf{exists\_at}(x,t)$, $\mathbf{p}(y,x,t)$, and $\omega(y,z,t)$. By $\mathbf{p}(y,x,t)$ and $\omega(y,z,t)$ and our definition schema for **Unified_Under (Metatheorem 1)** it follows that $\mathbf{p}(z,x,t)$. Due to $\mathbf{p}(z,x,t)$ and $\mathbf{p}(y,x,t)$ and the definition schema for **Unified_Under**, it follows that $\omega(z,y,t)$. □

Metatheorem 3. $\textbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\forall x(\exists t(\textbf{member\_of}(x,A,t)) \rightarrow$$
$$\forall t(\textbf{exists\_at}(x,t) \rightarrow$$
$$\forall y(\mathbf{p}(y,x,t) \rightarrow \omega(y,y,t))$$

Proof. Let A, ω, and $\mathbf{p}$ be such that $\textbf{Unified\_Under}(A,\omega,\mathbf{p})$. Let x, y, z, t, and A be such that $\textbf{member\_of}(x,A,t)$, $\textbf{exists\_at}(x,t)$, and $\mathbf{p}(y,x,t)$. Due to

$\mathbf{p}(z,x,t)$, simple repetition (i.e., $\mathbf{p}(y,x,t)$), and the definition schema for **Unified_Under**, it follows that $\omega(y,y,t)$. □

Clearly then, in all cases a unifying relation has properties similar to that of an equivalence relation.

As mentioned, a unifying relation only holds among parts of a *certain division* and not arbitrary parts. As given, we define this formulation by applying a proper subrelation **p** of the **part_of** relation (of **Chapter 6.2, p. 129**) instead of the generalized parthood relation applied by Guarino and Welty, which corresponds to **part_of**. Here **p** is a relation based on a restricted notion of parthood; however, it is not always clear how best to formalize this relation, therefore specifying **Unified_Under**(A,ω) can serve as a shortcut.

As mentioned unifying relation are reflexive, transitive, and symmetric. Furthermore, as shown by Kaplan (2001) for Unity, for our reinterpretation: for each object of a class, if all and only its parts are unified under some relation, then it does not partially overlap with other members of that class:

Metatheorem 4. $\textbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\begin{aligned}&\forall xyt(\textbf{member\_of}(x,A,t) \wedge \textbf{member\_of}(y,A,t) \rightarrow\\ &\quad \textbf{exists\_at}(x,t) \wedge \textbf{exists\_at}(y,t) \rightarrow\\ &\quad \exists z(\mathbf{p}(z,x,t) \wedge \mathbf{p}(z,y,t)) \rightarrow\\ &\quad \forall w(\mathbf{p}(w,x,t) \leftrightarrow \mathbf{p}(w,y,t))))\end{aligned}$$

Proof. Let A, z, w, x, y, t, ω, and p be such that **Unified_Under**(A,ω,p), **member_of**(x,A,t), **member_of**(y,A,t), **exists_at**(x,t), **exists_at**(y,t), $\mathbf{p}(z,x,t)$, and $\mathbf{p}(z,y,t)$. By the definition schema of **Unified_Under**, parts of a certain division of x and y have the ω relation with all other of those parts. Because $\mathbf{p}(z,x,t)$ and an additional assumption $\mathbf{p}(w,x,t)$, then $\omega(w,z,t)$. And again, from $\mathbf{p}(z,y,t)$ and $\omega(w,z,t)$, which was just derived, it follows that $\mathbf{p}(w,y,t)$.

These steps are repeatable to derive the other consequent, $\mathbf{p}(z,x,t)$, assuming $\mathbf{p}(z,y,t)$. Because $\mathbf{p}(w,y,t)$ and the additional assumption $\mathbf{p}(z,y,t)$,

$\omega(w,z,t)$. And again, from $\mathbf{p}(w,x,t)$ and from $\omega(w,z,t)$, which was just derived, it follows that $\mathbf{p}(z,x,t)$. It therefore follows that if a class A satisfies **Unified_Under** for a relation ω and $\mathbf{p}$, then if any of its members share parts of a certain division at a time they exist, they share all parts of that division; or in other words, the members of A do not partially overlap.

$\square$

In our effort to apply OntoClean's theory of Unity and our reformulation to BFO, we must consider what if any relations serve as a unifying relation for certain classes, i.e., a relation ω and $\mathbf{p}$ for a class A such that $\mathbf{Unified\_Under}(A,\omega,p)$. To begin this discussion we consider a relation that captures the notion of physical connectedness, which is formalized by the Region Connection Calculus 8 (RCC8) relation *connects with* (Randell et al., 1992). In RCC8, *connects with* is a primitive, binary relation, it is symmetric and reflexive, and it is formalized on the basis of point-set topology. x *connects with* y means that x and y share a common point.[4] Applying this relation to real-world entities, the hand is connected to the arm, and the arm is connected to the torso. In RCC8 the *part of* relation is defined by the *connects with* relation: if x is a part of y then if z is connected to x then z is connected to y. Another RCC8 relation, *disconnected with*, is simply defined as the negation of *connects with*.

BFO/RO does not include a relation that corresponds to this notion. It does however include a primitive relation that corresponds to a stricter notion, that of external connectedness: **adjacent_to** (Smith et al., 2005). This relation corresponds to the RCC8 relation x is *externally connected to* y, which is a defined relation, and means x is connected with y and x and y do not overlap (i.e., do not have any parts in common). In BFO/RO, $\mathbf{adjacent\_to}(x,y,t)$ implies that x and y do not overlap at t, but it is given as a primitive relation, since the more generalized connection relation is not formalized.

[4]If introduced to BFO/RO, the representing predicate would include a third argument for time indexing, since the BFO/RO relations hold at some instant or region of time.

We introduce a relation that corresponds to RCC8's *connected with*, **connected_with**(x,y,t). For a potential unifying relation, we introduce a relation that we define as the transitive closure of **connected_with**, $\mathbf{connected\_with_{tr}}$($x$,$y$,$t$). So for example, this relation holds between a hand and a torso at some time.

We consider whether *Ball* is unified under the $\mathbf{connected\_with_{tr}}$ relation; that there are certain parts of a ball that are connected via a chain of connections, and only those parts are connected in this manner. We do not suggest that all parts of a ball hold in this relation; in fact, this is prohibited by the definition schema for **Unified_Under** since **p** is a proper subrelation of **part_of**. This is reflected in the domain also since for the class *Ball* the internals of some balls may contain loose, disconnected pieces of the ball that are parts and not connected. Even though all parts of a solid ball are connected in this manner (i.e., in the $\mathbf{connected\_with_{tr}}$ with every other part), this does not hold for all balls.

For this generalized class *Ball* and the proposed unifying relation $\mathbf{connected\_with_{tr}}$, a proper subrelation of **part_of** must be given to constraint the parts of the division that is unified under $\mathbf{connected\_with_{tr}}$ or a more restricted notion of $\mathbf{connected\_with_{tr}}$ that holds among just the parts that maintain the round shape of a ball must be given. By itself $\mathbf{connected\_with_{tr}}$ does not sufficiently restrict the parts. Therefore for an appropriate unifying relation and part of subrelation for balls, we suggest that there is some relation that holds just between parts that form the boundary of the ball.

If we also consider a class *Human Skeletal System*, it is not simply unified under the relation $\mathbf{connected\_with_{tr}}$ either. For example, the Achilles' tendon (calcaneal tendon) connects the plantaris, gastrocnemius (calf) and soleus muscles to the calcaneus (heel) bone, but these three muscles are not part of the human skeletal system.[5] Again, a more specific relation and part of relation is required, here to unify the class *Human Skeletal System*.

[5]We acknowledge there is a system called the musculoskeletal system which includes muscles, as well as bones, cartilage, tendons, ligaments, joints and other connective tissues. For our example we only refer to the skeletal system to outline connected entities that are not considered part of that system.

In the event that the parts of two balls share some boundary parts at some time (e.g., they melt together in the sun), then the conjoined objects are no longer balls (balls are spherical, and can roll in any direction when placed on a flat surface) or *Ball* is not unified under the restricted notion of **connected_with$_{\mathbf{tr}}$** we suggested. This latter point holds true if balls that are conjoined are, individually, members of *Ball*. Based on these conclusions, **connected_with$_{\mathbf{tr}}$** serves as the basis for other defined relations that capture the notion how particulars are whole. As given, it is not sufficient to capture the notion of a unifying relation for any class of particulars.

Beyond **connected_with$_{\mathbf{tr}}$**, we must consider if any other relations can serve as unifying relations among parts of objects which are particulars in BFO's domain. We turn our attention to particulars that are aggregates (instances of BFO's *Object Aggregate*), particulars composed of spatially-separated particulars.

Take for example the relation *has the same parents as* which is proposed as a relation the property *Aggregate of Siblings* is unified under (Gangemi et al., 2001, p. 6).[6] We first note that the description 'has the same parents as' may be interpreted as 'at some time has the same parents as' or 'born of the same parents'. In the former description the existence of parents (at some arbitrary time) must be assumed, therefore the relation under consideration here is better and more accurately given as *born of the same parents.* In this case the parts of the division are members of the aggregate. We discuss this in more detail in section on object aggregates.

There are many other similarly defined unifying relations for aggregates, for example the unifying relation *having the same boss* and *being located in the same designated spatio-temporal region.*, the former holding for parts of an aggregate of co-workers, and the latter holding for parts of an aggregate of

[6]This passage assumes what in everyday language are considered members of a collection are formally parts of the collection. This is assumed for BFO/RO (Barry Smith, personal communication), however the mereology of entities which are aggregates of people is not formally given for BFO/RO. Nevertheless, as is demonstrated, a rejection of the purported unifying relation in question does not require a rejection of this assumption.

audience members of some performance, speech, etc.

There are other purported unifying relations of classes of aggregates which suffer from a problem in which they only hold at the level of particulars (where the *intrinsic whole under* relation applies) and do not at the level of class (where the *unified under* relation applies). For example, if the organization *PETA* is an intrinsic whole under the relation *pays dues to PETA*, there is not a more generalized relation like *pays due to an organization* that applies to all social organizations, because, clearly, some person part of one organization is a person part of a different organization at some time (therefore it is not transitive). The same can be said, for an aggregate *a* and a relation *being a member part of a*. Furthermore, this relation and others of its kind are self-referring to the aggregate in question, and in this case it is trivial.

Ultimately then, as reinforced by our examples, the utility of the definition schema for *Unity* is that we apply it for a specific class and relation, which may or may not hold for purported subclasses. If it does not hold, then the purported subclass is therefore identified as not a subclass. This notion does not cover the stronger notion of Non-Unity introduced by Guarino and Welty, but has immediate utility in that it covers the notion of not having a specific kind of Unity, i.e., relative to an identified unifying relation. Anti-Unity primarily holds for classes whose members are considered amounts of matter, which we address in **Section 7.3.4**.

7.2 Integrating Identity with BFO

7.2.1 Introduction

Identity is a relation that every object has to itself and to nothing else. Following Frege (1950), we view a criterion of identity as a way to determine when the identity relation holds, or informally, to recognize an object as the same again. It is very difficult to discuss the identity of a class of objects without presupposing what the objects *are* based on an assumed class definition, therefore identity criteria are better expressed as "identifying criteria" (Brand, 1977, p. 1). Lowe (1989b, p. 12) advises that this is permissible, because it is unavoidable in so many cases, that an identity criterion make reference to the class of objects the criterion of identity is being given for. However, says Lowe, it must not presuppose the criterion of identity for the class of objects whose identity criterion is being given. For example, a criterion for the identity of events should not be *having the same causes and effects* if causes and effects are themselves events (Lowe, 1989b, p. 12). Similarly a criterion of identity for sets should not be *having the same subsets*. We refer to this notion, henceforth, as the *Principle of Identity Criteria Non-Circularity*.

7.2.2 Redefining OntoClean's Notion of Identity for the Formal Theory of Classes

Identity criteria that are both necessary and sufficient include *occupying the same spatio-temporal region* for material entities or processes, and *having the same members* for sets. Outside these examples, identity criteria that are both necessary and sufficient are rare. To address this issue, Guarino and Welty define necessary and sufficient criteria of identity separately. Another issue they address is identity with respect to time; identity can be defined with respect to one time (synchronic) or defined across times (diachronic). Therefore Guarino and Welty provide time arguments that allow for either kind of identity criterion.

In many cases, analysis of identity can be limited to detecting the features that are just necessary for keeping the identity of a given entity, based on what can be described as essential properties. It is on these properties that Guarino and Welty base necessary criteria of identity. According to Guarino and Welty, a necessary criterion of identity θ of a property ϕ is defined such that for x and y that are instances of ϕ at t and t_1, respectively, and exist at t and t_1, respectively, if x and y are the same object, then they are the same under θ (i.e., $\theta(x,y,t,t_1)$ holds).[7] Where θ stands for 'having the same genotype' $\theta(x,y,t,t_1)$ is

[7]We again remind the reader that the sense of 'instance' and 'property' is given in Chapter

read x at t and y at t_1 have the same genotype. For Guarino and Welty (2001, p.5) 'same under' captures the intuition that, based on the identity criterion θ, there is some characteristic feature that is unique to the entity to which the criterion is applied.

In considering what an identity criterion is ontologically about, for such a criterion to be applied, there must be some procedure where x and y are evaluated. More formally, **confirms**(P,x,y,t,t_1) means that for x at t and y at t_1, procedure type P confirms x and y are the same thing. We discuss why the predicate **confirms** applies to a type, P, rather than a particular instance of P, shortly. We define this proposed notion under our theory of classes, and define a predicate **Necessary-IP**(A,P), which means that a class A has a *necessary identity procedur*e P:

Definition 18. $\textbf{Necessary-IP}(A,P) =_{\text{def}} \forall xytt_1((\textbf{member\_of}(x,A,t) \land \textbf{exists\_at}(x,t) \land \textbf{member\_of}(y,A,t_1) \land \textbf{exists\_at}(y,t_1)) \rightarrow (x{=}y \rightarrow \textbf{confirms}(P,x,y,t,t_1)))$

We note that the nature of **confirms**(P,x,y,t,t_1) is such that there is no specific instance of P for confirming if x and y are the same, and that the predicate relies on past instances of P. If **confirms**(P,x,y,t,t_1) holds then there is at least one entity whose identity has been confirmed in the past.[8] More formally, if **Necessary-IP**(A,P) and **confirms**(P,x,y,t,t_1), then there is some w that is a member of A and exists at a time t_2, some v that is a member of A and exists at a time t_3 and **confirmed**(P,p,w,v,t_3,t_4), which means that w and v were confirmed as the same by a procedure instance p of P:[9]

Axiom 21. $(\textbf{Necessary-IP}(A,P) \land \exists xytt_1(\textbf{confirms}(P,x,y,t,t_1))) \rightarrow \exists pwvt_2t_3(\textbf{member\_of}(w,A,t_2) \land \textbf{exists\_at}(w,t_2) \land$

1 and is based on Guarion and Welty. Therefore *Sugar* and *Hydrophillic* are shortcuts for *being sugar* and *being hydrophillic.* 'instance' is used differently in BFO, and 'property' is used differently in philosophy at large.

[8]We also observe that for P to be a legitimate identity procedure, there are many occurrences, i.e., instances of P where the identity of an entity has been confirmed in the past.

[9]As previously discussed, we quantify processes based on what are processes that have already occurred, excluding what are in the present considered future processes.

$\mathbf{member\_of}(v,A,t_3) \land \mathbf{exists\_at}(v,t_3)) \land$

$\mathbf{confirmed}(P,p,w,v,t_2,t_3))$

Axiom 22. $\mathbf{confirmed}(P,p,w,v,t_2,t_3) \rightarrow \exists t(\mathbf{instance\_of}(p,P,t))$

For an instance *p* of the procedure class *P*, there is some part of the procedure where a result of *w* is derived, and some part of the procedure where a result of *v* is derived, and finally, there is an end part of the procedure where these results are compared to determine whether or not *w* and *v* are the same thing. Note, the procedure *p* need not occupy contiguous spatio-temporal regions.

We provide a formalization of identity procedures that more concisely represent the aforementioned procedure parts and their results. This requires additional predicates, **matches** and **result_of_procedure**. The latter predicate, **result_of_procedure**(p_1,w,t_2), is a function that maps to some result of procedure p_1 that applies to the entity w at t_2 (but need not span t_2). For example, p_1 may be a process that has as a result the fingerprint pattern of a person. To evaluate identity for an entity, another result must be acquired; hence, a second procedure, p_2, is applied to an entity v at t_3, (**result_of_procedure**(p_2,v,t_3)). Therefore **matches**(**result_of_procedure**(p_1,w,t_2), **result_of_procedure**(p_2,v,t_3)) if the result of the first procedure "matches" the result of the second procedure (e.g., if two fingerprint patterns match). What 'matches' means here depends entirely upon the identity procedure type, and it also depends upon the identity of other things, since, as we discuss shortly, there is a recursive nature to identity procedures. Given these formulations we have the following axiom:

Axiom 23. $\mathbf{confirmed}(P,p,w,v,t_2,t_3) \rightarrow$

$\exists p_1 p_2(\mathbf{matches}(\mathbf{result\_of\_procedure}(p_1,w,t_2), \mathbf{result\_of\_procedure}(p_2,v,t_3)) \land$

$\mathbf{part\_of}(p_1,p) \land \mathbf{part\_of}(p_2,p))$

The practical use of **matches** is that when applied to two results, if p_1 and p_2 are parts of an instance of P that is a necessary identity procedure, if false, then x and y are not identical. That said, because **confirms**(P,x,y,t,$t1$) captures the notion of an identity procedure categorization applicable to every member of a class, we take the **confirms** predicate to be our primary notion for formalizing the relation between identity procedure types and classes of particulars to which the identity procedures apply.

With respect to our example, one such procedure is *DNA_Profiling*, which is a subtype of BFO's *Process*. When **Necessary-IP**(*Person*,*DNA_Profiling*) holds, if x exists and is a member of *Person* at t and y exists and is a member of *Person* at t_1, if x and y are identical, then **confirms**(*DNA_profiling*,x,y,t,t_1) holds. Therefore if **Necessary-IP**(*Person*,*DNA_Profiling*) holds, by our formulation of **confirmed** (**Axiom 20**), it is also true that an instance of *DNA_Profiling* has in the past served to confirm necessary identity for an instance of the class *Person*.

DNA profiling requires the object being evaluated have a genotype, and since this is what we consider an essential property for people, it is accurate to presume that the existence of a person at some time entails the existence of their genotype at the same time. Note that although *DNA profiling* is a procedure class that confirms the necessary identity of people, it is clearly not sufficient for confirming identity, due to the existence of genetically identical twins.

The notion of a necessary identity procedure is perhaps more intuitive to think of in terms of the contrapositive of the nested implication, x=y → **confirms**(B,x,y,t,t_1), of **Definition 18**. If two objects are not confirmed as identical by procedure B, they do not have the same essential properties, therefore they certainly cannot be identical. In the context of a modeler thinking about what a necessary identity procedure of a class is, it is helpful, in order to identify essential properties, for her to answer the question: *What feature must change or no longer exist for a member of the class, at some time t, to no longer be the same thing at a time after t?*

According to Guarino and Welty, a sufficient identity criterion θ of a property ϕ is defined such that for x and y that are instances of ϕ at t and t_1, respectively, and exist at t and t_1, respectively, if $\theta(x,y,t,t_1)$ holds, then x and y are identical.

We reconsider criteria again, in this case for sufficient identity, and put forth a notion of sufficient identity procedures under our theory of classes. **Sufficient-IP**(A,B) means that a class A has a *sufficient identity procedure* P:

Definition 19. $$\textbf{Sufficient-IP}(A,P) =_{\text{def}} \forall xytt_1((\textbf{member\_of}(x,A,t) \wedge \textbf{exists\_at}(x,t) \wedge \textbf{member\_of}(y,A,t_1) \wedge \textbf{exists\_at}(y,t_1)) \rightarrow (\textbf{confirms}(P,x,y,t,t_1) \rightarrow x{=}y))$$

Take for example, a sufficient identity procedure for the class *Person*, *Fingerprinting*.[10] When **Sufficient-IP**(*Person*,*Fingerprinting*) holds, if x exists and is a member of *Person* at t and y exists and is a member of *Person* at t_1, if **confirms**(*Fingerprinting*,x,y,t,t_1) holds, that is, if one or two instances of the type *Fingerprinting* confirms x at t and y at t_1 are the same person, then x and y are identical.

The procedure type *Fingerprinting* is defined under the following natural language parse: 'a procedure in which a fingerprint pattern *that exists* is analyzed, the results of which are comparable to confirm identity'. The procedure requires a fingerprint pattern that represents an actual fingerprint's pattern. Because fingerprints can be removed, it is not possible to compute and compare fingerprint patterns between arbitrary people *at any time*; therefore, the assumption that the fingerprint pattern being evaluated during the procedure in question exists is needed as a basis for the class definition of *Fingerprinting*, in order for it to be a legitimate sufficient identity procedure.

This clarification of sufficient identity procedures brings attention to an important point about what we consider necessary and sufficient procedures for identity. In each case, the procedure involves the identity of functions which map from the objects of the class in question (**Axiom 23, p. 96**). Lowe (1989a, p. 20) noted that identity criteria often make use of the notion of identity itself, and can only do so informatively by alluding to the identity of things of another class. With respect to identity procedures, for the necessary identity proce-

[10]The exact precision for unique identification by fingerprinting is debated.

dure class *DNA profiling* of the class *Person*, it is dependent on the identity of genotypes, which must account for genetic variations over time that are due to mutations. By this token, the corresponding identity criterion is non-primitive and can be reduced to identity of functions mapped from individual people to their genotype, i.e., 'genotype of'.

Applying Guarino and Welty's formulation, if $\theta(x,y,t,t_1)$ holds where θ is 'having the same genotype', this implies that the genotype of x at t is identical to the genotype of y at t_1. If a person x at t and a person y at t_1 are identical under the sufficient identity criterion *having the same fingerprint pattern*, there is some fingerprint pattern of x at t and fingerprint pattern of y at t_1 which are identical. Nevertheless, by defining identity procedures for necessary and sufficient identity, instead of criteria, these issues are dealt with simply, and by the designated identity procedure type that is a subtype of *Process*.

We designate necessary and sufficient identity procedures to be two kinds of identity procedures (**IP**):

Definition 20. $\textbf{IP}(A,P) =_{\text{def}} \textbf{Necessary-IP}(A,P) \lor \textbf{Sufficient-IP}(A,P)$

We also formally designate a necessary and sufficient identity procedure (**N&S-IP**) as a conjunction of both of our two kinds of identity procedure:

Definition 21. $\textbf{N\&S-IP}(A,P) =_{\text{def}} \textbf{Necessary-IP}(A,P) \land \textbf{Sufficient-IP}(A,P)$

The necessary and sufficient identity procedure for the duration of time of a process is *time measurement procedure*. Here, the measurements, i.e., the values that results from measurement of time, of x and of y, are identical according to some specific scale.

Guarino and Welty (2000a) discuss how Non-Rigid properties seem to only "carry" (i.e., inherit) identity criteria, for example *being a student* inherits its identity from *being a person* which "supplies it". An identity criterion proposed to be "supplied" (i.e., not inherited) by *being a student*, for example *having the same registration number*, is only held within certain durations of the student's existence. Given this limitation, Guarino and Welty decide that identity criteria that are not held by Rigid properties are not of interest to their theory. Therefore they exclude these "local" identity criteria, like *having the same registration number*. This approach is in line with the previously defined notion that Non-Rigid classes are not types of BFO. We provide this informal part of their theory formally and with respect to classes and Identity procedures:

Axiom 24. $\mathbf{IP}(A,P) \rightarrow \exists B(\mathbf{Rigid}(B) \wedge \mathbf{IP}(B,P) \wedge \mathbf{subclass\_of}(A,B))$

Note that by this formulation A and B may be identical. It follows immediately that, for an identity procedure type P of a Non-Rigid class A, there is some Rigid class B with that identity procedure class that is a superclass of A:

Theorem 48. $(\mathbf{Non\text{-}Rigid}(A) \wedge \mathbf{IP}(A,P)) \rightarrow \exists B(\mathbf{Rigid}(B) \wedge \mathbf{IP}(B,P) \wedge A{\neq}B \wedge \mathbf{subclass\_of}(A,B))$

Proof. Follows trivially from **Axiom 24**. $A{\neq}$B is due to **Theorem 4**, which states that no class is both Rigid and Non-Rigid. □

Guarino and Welty define that a property ϕ "supplies" an identity criterion iff ϕ is Rigid, has the identity criterion, and does not have a parent an ancestor with that identity criterion.

We consider their definition for a notion of supplying an identity procedure, and provide it in terms of classes. If A supplies an identity procedure P, that means all other classes with identity procedure P are subclasses:

Definition 22. $\mathbf{supplies\text{-}IP}(A,P) =_{\text{def}} \mathbf{IP}(A,P) \wedge \mathbf{Rigid}(A) \wedge (\forall B(\mathbf{IP}(B,P) \rightarrow \mathbf{subclass\_of}(B,A))$

Theorem 49. $\mathbf{supplies\text{-}IP}(A,P) \wedge \mathbf{IP}(B,P) \wedge \mathbf{subclass\_of}(A,B) \rightarrow A{=}B$

Proof. Let A, B, and P be such that $\mathbf{supplies\text{-}IP}(A,P)$, $\mathbf{IP}(B,P)$, and $\mathbf{subclass\_of}(A,B)$. From the definition of **supplies-IP** (**Definition 22**) and $\mathbf{IP}(B,P)$,

subclass_of(B,A). By **subclass_of**(B,A), **subclass_of**(A,B), and the definition of **subclass_of** (**Definition 1**), $A{=}B$. $\square$

Axiom 25. $\mathbf{IP}(A,P) \land \neg\mathbf{supplies\text{-}IP}(A,P) \to$
$\exists B(\mathrm{A}{\neq}B \land \mathbf{subclass\_of}(A,B) \land \mathbf{supplies\text{-}IP}(B,P))$

Theorem 50. $\mathbf{IP}(\mathit{Entity},P) \to \mathbf{supplies\text{-}IP}(\mathit{Entity},P)$

Proof. Let P be such that **IP**(*Entity*,P), and let B be such that **IP**(B,P). By definition of **is_a**, **is_a**(B,B), therefore by Axiom 7 it follows that **is_a**(B,*Entity*). Therefore by definition (**Definition 22**) it follows that **supplies-IP**(*Entity*,P). $\square$

If a class has an identity procedure, there is some class that supplies it:

Theorem 51. $\exists A(\mathbf{IP}(A,P)) \to \exists B(\mathbf{supplies\text{-}IP}(B,P))$

Proof. Let A and P be such that **IP**(A,P). A supplies P or A does not supply P. If A does not supply P, by **Axiom 25** a superclass of A supplies P. Therefore some class supplies P (which may or may not be identical to A). $\square$

For example, *Primate* supplies the identity procedure *fingerprinting*, which is inherited by classes such as *Human* and *Gorilla*. If a Non-Rigid property has an identity criterion θ then it is subsumed by a Rigid property that supplies it. We also provide this in terms of classes and identity procedures:

Theorem 52. $(\mathbf{Non\text{-}Rigid}(A) \land \mathbf{IP}(A,P)) \to$
$\exists B(\mathbf{supplies\text{-}IP}(B,P) \land \mathbf{subclass\_of}(A,B))$

Proof. Follows trivially from **Theorem 51** and **Definition 22**.
It follows that classes inherit necessary identity procedures:

Theorem 53. $(\mathbf{Necessary\text{-}IP}(A,P) \land \mathbf{subclass\_of}(B,A)) \to$
$\mathbf{Necessary\text{-}IP}(B,P)$

Proof. Let A, B, and P be such that **Necessary-IP**(A,P) and **subclass_of**(B,A). Also let x, y, and t be such that **exists_at**(x,t), **member_of**(x,B,t), **exists_at**(y,t_1), **member_of**(y,B,t_1), and $x{=}y$. By **subclass_of**(B,A) and the definition of **subclass_of** (**D1**), it holds that **member_of**(x,A,t) and **member_of**(y,A,t_1). It follows from **Necessary-IP**(A,P), **Definition 18**, **exists_at**(x,t), **member_of**(x,A,t), **exists_at**(y,t_1), **member_of**(y,A,t_1), and $x{=}y$ that **confirms**(P,x,y,t,t_1). Therefore it holds that **Necessary-IP**(B,P). □

It also follows that classes inherit sufficient identity procedures:

Theorem 54. **Sufficient-IP**$(A,P) \wedge$ **subclass_of**$(B,A) \rightarrow$ **Sufficient-IP**(B,P)

Proof. Let A, B, and P be such that **Sufficient-IP**(A,P) and **subclass_of**(B,A). Also let x, y, and t be such that **exists_at**(x,t), **member_of**(x,B,t), **exists_at**(y,t_1), **member_of**(y,B,t_1), and **confirms**(P,x,y,t,t_1) . By **subclass_of**(B,A) and the definition of **subclass_of** (**D1**), it holds that **member_of**(x,A,t) and **member_of**(y,A,t_1). It follows from **Sufficient-IP**(A,P), **Definition 19**, **exists_at**(x,t), **member_of**(x,A,t), **exists_at**(y,t_1), **member_of**(y,A,t_1), and **confirms**(P,x,y,t,t_1) that $x{=}y$. Therefore it holds that **Sufficient-IP**(B,P). □

Because every type is a class, identity procedures are also inherited by types:

Theorem 55. **Sufficient-IP**$(A,P) \wedge$ **is_a**$(B,A) \rightarrow$ **Sufficient-IP**(B,P)

Proof. Follows from **Theorem 15** and **Theorem 54**. □

Theorem 56. **Necessary-IP**$(A,P) \wedge$ **is_a**$(B,A) \rightarrow$ **Necessary-IP**(B,P)

Proof. Follows from **Theorem 15** and **Theorem 53**. □

Guarino and Welty discuss the notion that if two identity criteria are incompatible, then a property cannot have both. Examples are given but the notion is not formalized.

In what follows, **Compatible-IP**(P,Q) means that some class has identity conditions P and Q, and **Incompatible-IP**(P,Q) means the negation, that no class has identity conditions P and Q:

Definition 23. $\textbf{Compatible-IP}(P,Q) =_{\text{def}} \exists A(\textbf{IP}(A,P) \wedge \textbf{IP}(A,Q))$

Definition 24. $\textbf{Incompatible-IP}(P,Q) =_{\text{def}} \neg\textbf{Compatible-IP}(P,Q)$

Theorem 57. $\textbf{Incompatible-IP}(P,Q) \leftrightarrow \forall A \neg(\textbf{IP}(A,P) \wedge \textbf{IP}(A,Q))$

Proof. Follows trivially from the definition of **Compatible-IC** (**Definition 23**) and **Incompatible-IC** (**Definition 24**). □

If ω and θ are incompatible identity conditions that are held by A and B, respectively, and A and B are types, then they are disjoint types:

Theorem 58.
$$\begin{aligned}&(\textbf{Necessary-IP}(A,P) \wedge \textbf{Necessary-IP}(B,Q) \wedge \\ &\quad \textbf{Incompatible-IP}(P,Q)) \rightarrow \\ &\quad\quad (\textbf{Type}(A) \wedge \textbf{Type}(B)) \rightarrow \\ &\quad\quad\quad \textbf{disjoint\_from}_{\textbf{type}}(A,B)\end{aligned}$$

Proof. By the definition of **Incompatible-IP** (**Definition 24**), **Incompatible-IP**(P,Q), and **Necessary-IP**(A,P) it follows that $\neg$**Necessary-IP**(A,Q). By **Theorem 53**, $\neg$**Necessary-IP**(A,Q), and **Necessary-IP**(B,Q), it follows that $\neg$**subclass_of**(A,B).

By the definition of **Incompatible-IP** (**Definition 24**), **Incompatible-IP**(P,Q), and **Necessary-IP**(B,Q) it follows that $\neg$**Necessary-IP**(B,P). By **Theorem 53**, $\neg$**Necessary-IP**(B,P), and **Necessary-IP**(A,P) it follows that $\neg$**subclass_of**(B,A). If A and B are types, it follows from the definition of $\textbf{disjoint\_from}_{\textbf{type}}$ (**Definition 12**) that $\textbf{disjoint\_from}_{\textbf{type}}(A,B)$. □

We can also give a similar proof for a theorem where **Sufficient-IP**(A,θ) and **Sufficient-IP**(B,ω) hold and **Theorem 54** is applied.

It is stated in the OntoClean literature that incompatible ICs are disjoint, but what incompatible means is not formalized, but it is given here. We formalize it here, and note that the utility of the **Incompatible-IC**(θ,ω) is that if a class is purported to have both Identity procedures, then there is a mistake in classification or a mistake in assignments of Identity procedures to classes. Ultimately it is a useful tool for modeling and isolating modeling mistakes.

We also formalize an identity procedure based precisely on mereological extensionality (*ME*) in the context of our discussion on object aggregates, in **Section 7.3.3**.

7.3 Unity and Identity of Material Entities

7.3.1 Introduction

BFO aims to represent reality on the basis of the best current scientific understanding. Although as it is described under OntoClean's theory, identity criteria are partially epistemic notions, we think that they can prove useful for ontological modeling for domain ontologies rooted under BFO. In what follows we inspect BFO upper ontology types given our formalisms for Unity and Identity.

7.3.2 Object

The natural language definition for the BFO type *Object* is as follows (Spear, 2007, p. 48):

> A material entity that is spatially extended, maximally self-connected and self-contained (the parts of a substance are not separated from each other by spatial gaps) and possesses an internal unity. The identity of object entities is independent of that of other entities and can be maintained through time. **Examples**: an organism, a heart, a chair, a lung, an apple.

It is included in the definition that an object is self-connected and not separated by spatial gaps. Consider a person at t and that person with a fingernail detached at t_1. The entity consisting of the sum of the person and the fingernail at t_1 is not the same entity as that mentioned at t.

By the BFO definition above, objects are self-connected, are not merely the sum of their parts, and thus can survive the gain and loss of *some* parts. In our example the nail at t_1 is no longer a part of the person. Given this, we conclude that the identity criterion of *Object* is not that of *ME* (as given in **Axiom 29**), i.e., identity that is based precisely on its parts.

Consider also a piece of gold. *Is it the same piece of gold in liquid form?* This depends on what we consider to be the entity under consideration: Is it the substance (i.e., what) the piece of gold is made of or is it the piece of gold itself? BFO does not allow spatial coincidence among distinct particulars; therefore, the object under consideration must be the latter. Hence, BFO's answer to the question is "no" in the case that the shape, or more specifically, the molecular arrangement of the piece of gold, is essential to what it is. Therefore for a class with a piece of gold as one of its members, its identity procedure is not based on *ME* either.

We also consider how the type *Object* fits with our analysis of Unity. Under BFO's theory, instances of *Object* are described as having "internal unity", which we believe is a relation closed under boundary parts, as discussed in the previous section. Further we cannot assume that objects are unified under a more generalized relation like $\mathbf{connected\_with_{tr}}$, due to the fact that it does not transitively hold for just the parts of each particular that is an instance of *Object*. For example, there are fused particulars, such as conjoined twins, where, even though $\mathbf{connected\_with_{tr}}$ holds between any two parts of the fused totality, still the fused totality is not considered to be a single unified particular, but a fusion of two particulars. Another view to reinforce this same point is that, as given in **Theorem 4**, if a class is unified under a relation, then the specifically unified parts of a member of the class do not partially overlap with other members.

7.3.3 Object Aggregate

The natural language definition for the BFO type *ObjectAggregate* is as follows (Spear, 2007, p. 48):

> A material entity that is a mereological sum of separate object entities and possesses non-connected boundaries. **Examples**: a heap of stones, a group of commuters on the subway, a collection of random bacteria, a flock of geese, the patients in a hospital.

Every object aggregate is composed of at least two separate objects:

Axiom 26.

$$\begin{aligned}&\textbf{instance\_of}(x, \mathit{ObjectAggregate}, t) \rightarrow\\ &\quad\exists yz(\textbf{part\_of}(y,x,t) \wedge \textbf{part\_of}(z,x,t) \wedge\\ &\quad\quad\textbf{instance\_of}(y, \mathit{Object}, t) \wedge\\ &\quad\quad\textbf{instance\_of}(z, \mathit{Object}, t) \wedge y \neq z)\end{aligned}$$

There is no one identity procedure that applies to *ObjectAggregate*, but various identity procedures may be identified by further inspecting different kinds of object aggregates. Also, *ObjectAggregate* satisfies **non-Unity** because there is some instance that shares parts with another instance (e.g., Barack Obama is a part of the collection of people present for his the State of the Union Address on January 25, 2011, and at the same time a part of the collection of people intending to run for president in the 2012 Presidental Election). In what follows we inspect subclasses of *Object Aggregate* which we refer to as *Collection*, *Collective*, and *Organization*.

A *collection* is a mere grouping of spatially separated particulars,[11] where the particulars are object entities (in the BFO sense) and serve as the parts, although not the only parts, of the collection. A collection is composed of the

[11] We remind the reader that we introduce and define our usage of the term 'collection' here, regardless of how the term is used elsewhere.

sum of these object entity parts. Example collections include the utensils in your kitchen, Christmas gifts under a tree, or a collection of stones.

A collection cannot survive the gain and loss of its object entity parts. What we refer to in our examples, respectively, are precisely and at one time the aggregation of every utensil in your kitchen, the aggregation of every gift under the tree, and the aggregation every stone in the collection. Therefore a collection of stones cannot survive the loss of an entire stone (e.g., an entire stone is pulverized), but can survive various subtle changes in the stones in the collection (e.g., erosion).

Does this mean that a collection can survive changes in its parts? In some literature, including (Guizzardi, 2008, p.185), the member/collection relationship is defined separate from the 'part of' relation, but it also can be considered a restricted subrelation of it. It is defined as a relation between what we consider an object entity and an object aggregate, and it is intransitive, irreflexive, and asymmetric. A fiat part of a stone (i.e., a part that does not have distinct boundaries)[12] in a collection of stones is not a member of the collection of stones.

This sort of contextually defined 'part of' relation is not given for BFO, and the **part_of** relation is used for the composition of both objects and object aggregates. Given that the **part_of** relation is transitive, a fiat part of a stone in a collection of stones is a part of the collection.

It is worthy to note that some argue that the relationship between a stone and a collection of stones is not 'part of' at all. Under this view the relationship is more specialized, as described above for the member/collection relation, that addresses the nature of the whole. There is also a tendency to associate the 'part of' relation with physical connectedness, and under such an account the relation does not apply to the composition of aggregates. Ultimately, for those who argue this position, the reading of 'a piece of a stone is part of a stone collection' goes against common sense knowledge representation, and conceptually-speaking,

[12]A fiat part object part is part of a object but is not demarcated by any physical discontinuities, e.g., upper and lower lobes of the left lung (Spear, 2007, p. 51).

this much is true.

We propose a relation for BFO/RO, **member_of_aggregate**, to represent the member/collection relation, where **member_of_aggregate** is a subrelation of **part_of**, and it is not the case that every part is a member part of the aggregate:

Axiom 27. $\textbf{member\_of\_aggregate}(x,y,t) \rightarrow \textbf{part\_of}(x,y,t)$

Axiom 28. $\neg\forall xyt(\textbf{part\_of}(x,y,t) \rightarrow \textbf{member\_of\_aggregate}(x,y,t))$

Given this relation, it is not the case that a fiat part of a stone is in the **member_of_aggregate** relation with a collection of stones. Further, if a stone is a member of a collection, the stone is also more basically a part of the collection.[13]

Given this newly introduced relation, we can now formalize the (necessary and sufficient) identity procedure of a collection based on its extensionality under this division; a collection is the same over time iff it has the same *member parts*:

Axiom 29.
$$\begin{aligned} &\textbf{N\&S-IP}(A,\ ME) \leftrightarrow \\ &\quad \forall xyt((\textbf{member\_of}(x,A,t) \wedge \\ &\qquad \textbf{member\_of}(y,A,t)) \rightarrow \\ &\qquad\quad (x{=}y \leftrightarrow \forall z(\textbf{member\_of\_aggregate}(z,x,t) \leftrightarrow \\ &\qquad\qquad \textbf{member\_of\_aggregate}(z,y,t)))) \end{aligned}$$

If a class whose members are aggregates is defined in such a way that there is a case where all of the member parts of one of its members does not uniquely identify the member of the class, then *ME* is not an identity procedure for that class. Aggregates compose a material entity type in BFO, and defining a subclass as having the identity procedure of *ME* imposes that all subclass have this identity procedure; otherwise there is a modeling mistake.

The motivation behind BFO's position of primarily using **part_of** is for maintaining transitivity of parthood across levels of granularity. BFO/RO's

[13]This conclusion is based on email conversations with Barry Smith.

part_of relation based on classical mereology, therefore **part_of** is always transitive, and variations of 'part of' that are not transitive are other distinct relations.

So for example, if you "zoom out" what is an object aggregate at the microscopic level may be an object at the human eye level; in either case, what composes the particular is a part of it. Nevertheless, introduction of a grain-specific 'part of' relation, as we have done with **member_of_aggregate**(z,x,t), maintains this transitivity, via its super-relation **part_of**. *Collection* has an identity procedure, and it is that of *ME*, as we defined the notion in **Axiom 29**.

A *collective* differs from a collection in that the parts of a collective play a role—the same role—that contributes to what the aggregate is as a whole.[14] Collectives are like collections in that they have "atomic parts"; however, these parts have a special status, because they have roles. Collectives also differ from collections because *certain* parts of a collective are not fixed. For example, my stamp collection is a collective because stamps of this collective may be traded away or newly introduced, and the collective remains the same. Therefore the identity procedure for *Collective* class is not based on *ME*.

An *organization* is a kind of collective, and is an aggregate of humans, where the humans that compose the aggregate may be different humans at different times. The humans are not just parts but also, in the social sense, members, in virtue of the fact that they play roles. For example in a corporation there is a generalized employee role all members have and more specialized roles such as chairman. An organization differs from a collective in that in the former the roles differ among its parts. There may be collectives of humans, however, for example a soccer fan mob. Take also for example an organization composed of musicians, e.g., a musical band. The identity of the band may remain the same while its band members change, e.g., the Beatles after replacing drummer Pete

[14]This use of the term 'collective' differs from that introduced by Rector et al. (2006), where they describe a theory of emergent characteristics that apply to the aggregate and not any of part.

Best with Ringo Starr. As another example, a professional basketball team is the same team after a trade. *Organization* has an identity procedure, but it is not that of ME. Ultimately, *ME* is not an identity procedure for the type *ObjectAggregate*.

7.3.4 Amounts of Matter

That which in philosophical literature ((Zimmerman, 1995) (Barnett, 2004)) is referred to as *an amount of matter* does not clearly fall under BFO's *Object* or *ObjectAggregate*. An amount of matter is an object (in the formal sense) that usually falls under terms that, in everyday language, take singular verbs, cannot occur with numerals (unless elliptical for some measurement), and take determiners like 'some', 'little', and 'much', as opposed to 'every', 'few', and 'many' (Zimmerman, 1995). These terms include 'gold', 'sugar', and 'water'. Note that although 'time' and 'freedom' also fall under this linguistic characterization, they do not occupy spatial regions and require a different category, and subsequently a different treatment then the one being developed here. The determiners 'every', 'few', and 'many' are normally reserved for nouns that denote individuated, countable objects. Amounts of matter do not denote anything that is essentially a lump, cube, bit, piece, portion, or fragment (Zimmerman, 1995).

Barnett (2004) argues that some of what are considered *portions* are not simply the sum of their parts, which amounts of matter are. For example for a portion (or piece) of tofu, if it is chopped into several additional pieces, the tofu survives while the piece of tofu no longer exists. Given this, a detached part is no longer part of an object; an amount of matter can be scattered about. However, Barnett additionally argues that some amounts of matter are not merely just aggregates. Some certain kinds of stuff can gain or lose subportions; these are structured stuff. This work however falls out of the current scope.

BFO's position on this topic is that two different entities cannot occupy the same space at the same time; therefore, what is referred to as the portion of tofu and what referred to as the tofu "stuff" might be considered identical (Grenon, 2003a, p. 12). However the tofu "stuff" has different existence conditions, for example, if it is split into one hundred pieces it still exists while the portion of tofu no longer exists. Because the class is cross-granular in nature, having features of both an *Object* and *Object Aggregate*, and such the class cause problems for BFO. More specifically, the tofu stuff is a whole object, in the sense that all the parts are physically connected by a chain of connections, while at the same time it is an aggregate, and one in which its parts need not be connected to be a part. For BFO an ontology represents a certain level of granularity, based on what is referred to as perspectivalism (**Section 3.4**), and by this approach the types *Object* and *Object Aggregate* are disjoint (**Theorem 44, p. 69**). Clearly, these assumptions lead to an inconsistency given our prior conclusion that the portion of tofu and the tofu stuff are identical. Given this, something that demonstrates the existence conditions of tofu stuff is not a particular in BFO's domain.

7.4 Discussion

The notion of Rigidity can help a modeler identify classes that are essential and non-essential (with respect to time) to their members' existence. The notion of necessary identity procedures can help a modeler identify precisely what features of the members of Rigid classes make them members of those classes, which can be applied to compare any two members in a procedure that determines that they are distinct or the same. Therefore, necessary identity procedures provide a facility that supplements our formal theory of Rigidity, ergo types, and vice versa.

The notion of a sufficient identity procedure is more clearly epistemic in nature and less relevant to an upper ontology with the ontological position such as BFO. Fingerprints can be removed and social security numbers can be changed, all while the entities they corresponded to can continue to exist.

Nevertheless, sufficient identity procedures can be useful for modeling, because they are inherited, and for ontologies that are linked to databases, they can help construct primary keys.

The notion of a unifying relation that applies to all members of a class is based on how it is that the members of the class are wholes (i.e., how they are composed), at any time they exist. By its formulation, clearly, a unifying relation for a subclass of *Object* can serve as a basis for a sufficient (but not a necessary) identity procedure for the same class.

We must also take into account BFO's perspective on Identity and Unity with respect to its types, *Object* and *Object Aggregate*, as well as more specific types of aggregates. As mentioned, what is considered an amount of matter is, according to BFO, cross-granular and leads to an inconsistent ontology, since *Object* and *ObjectAggregate* are disjoint. Furthermore, BFO is clear on its position for certain Identity criteria; what is a person that is alive at t and what remains of the person after passing away at a time after t are identical, but are not identical with what remains after the body is decomposed in such a way that its parts form an object aggregate rather than an object. As another example, what is a statue at t and what are the remains of the statue after being shattered are not identical.

In **Axiom 29** provides a identity based *mereological extensionality* (**ME**), which object aggregates may or may not have. Among aggregates we have those that are identified purely by their parts and also those that are not, and are identified based on some emergent properties. This is a useful principle to determine what sort of aggregate a class is.

Part III

Method

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 8

Evaluating Candidate Types

8.1 Introduction

Our aim in this dissertation is to improve domain ontologies that are intended to be consistent with BFO. We address this challenge in two parts. The first part is provided in **Part II**, where we integrated the formal principles of OntoClean with those of BFO. We address the second part in this chapter, by applying the integration to the practice of constructing ontologies. This includes consistent rooting in upper ontology types and correct use of RO relations, particularly if an RO relation holds when the **is_a** relation is incorrectly asserted.

8.2 Violations of the Disjointness Principle

Isolating violations of the Disjointness Principle helps reveal where some candidates are not types. These violations follow the pattern:

$$\mathbf{is\_a}(A,B) \wedge \mathbf{is\_a}(A,C)$$

where it does not hold that:

$$\textbf{is\_a}(B,C) \vee \textbf{is\_a}(C,B)$$

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

1. **is_a**(B,C) or **is_a**(C,B) holds.

2. **is_a**(A,B) or **is_a**(A,C) is removed, including the choice that **is_a** 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 it 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 has as members instances of disjoint upper ontology types (satisfying **Heterogeneous**).

8.3 Applying Type Criteria

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 8.1)[1] that assists a modeler in evaluating whether a candidate is a type according to criteria provided in Chapter 6 (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, which uses the decision tree to classify each in turn. A candidate that satisfies any combination of **Empty**, **Partial**, or ¬**Members_Exist** satisfies ¬**Type** and requires further inspection.

[1]Redundant subtrees for Question 2 choices a, b, or c are combined. Variables that represent term input by the modeler appear in square brackets.

In our decision tree in Figure 8.1, each non-terminal node in the tree represents a question that is asked of the modeler. Each connecting "branch" represents a potential answer given by the modeler in response to the question it extends from. Each terminal node represents an assertion that is made as a result of the series of answers leading to the node.

In Figure 8.1, the descriptions of the answer choices for **Question 2** correspond to more commonly modeled types under *IC*, *DC*, and *Occurrent*, namely *MaterialEntity*, *SpecificallyDependentContinuant*, and *Process*.[2] The other major types of BFO, as illustrated in Figure 26, are *SpatialRegion*, *TemporalRegion*, and *SpatioTemporalRegion*, and are based on the Newtonian space-time container theory. We exclude these types from our evaluation work, because their instances are simply not the sort of objects scientists reference directly in real-world settings. This is evidenced by the fact that there are no subtypes for these in the OBO Foundry's Ontology for Biomedical Investigations (see http://purl.obolibrary.org/obo/obi.owl). There are certain other types, (e.g., *GenericallyDependentContinuant*) that will appear in an expanded version of the tree, in future work.

The goal of the procedure is to have a modeler perform an analysis of the metaphysical nature of what it is she aims to classify. To address this normally complicated task, many of the questions asked of the modeler are answerable by "yes" or "no". To supplement this approach, some questions include the term the modeler has labeled the candidate under evaluation.

Before the first question is asked, the modeler is prompted for the name of the class he wishes to add to his ontology. The first question asked for a candidate A being evaluated is:

Question 1. What is a specific example of something that is a prototypical member of the class A?

Using the answer x given by the modeler, the following question is asked:

[2]This is reflected in the Gene Ontology's division of classes into *Cellular Component*, *Molecular Function*, and *Biological Process*.

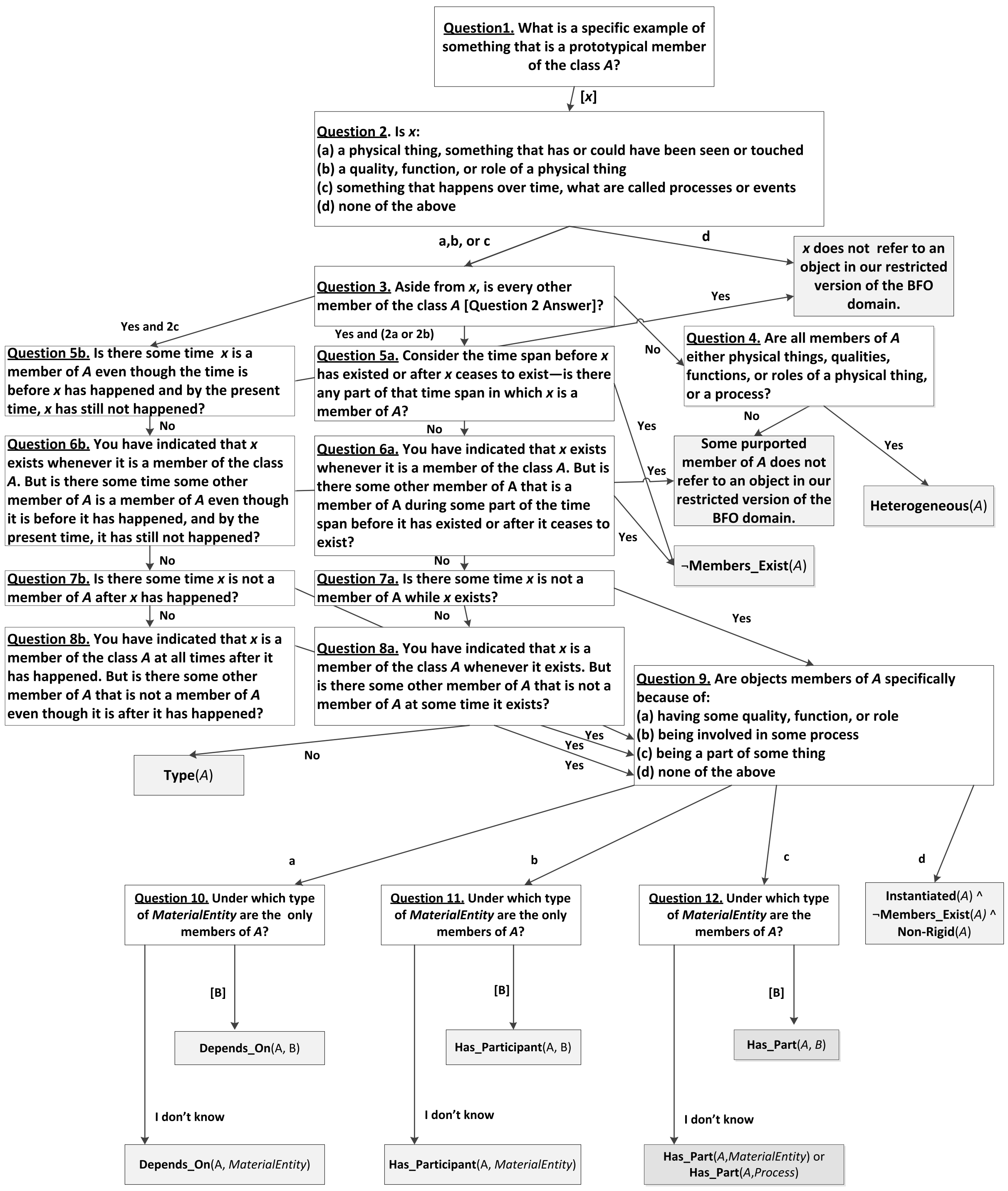

Question1. What is a specific example of something that is a prototypical member of the class A?
[x]
Question 2. Is x:
(a) a physical thing, something that has or could have been seen or touched
(b) a quality, function, or role of a physical thing
(c) something that happens over time, what are called processes or events
(d) none of the above
a,b, or c
d
x does not refer to an object in our restricted version of the BFO domain.
Question 3. Aside from x, is every other member of the class A [Question 2 Answer]?
Yes
Yes and 2c
Yes and (2a or 2b)
No
Question 4. Are all members of A either physical things, qualities, functions, or roles of a physical thing, or a process?
Question 5b. Is there some time x is a member of A even though the time is before x has happened and by the present time, x has still not happened?
Question 5a. Consider the time span before x has existed or after x ceases to exist—is there any part of that time span in which x is a member of A?
No
No
Yes
No
Yes
Some purported member of A does not refer to an object in our restricted version of the BFO domain.
Heterogeneous(A)
Question 6b. You have indicated that x exists whenever it is a member of the class A. But is there some time some other member of A is a member of A even though it is before it has happened, and by the present time, it has still not happened?
Question 6a. You have indicated that x exists whenever it is a member of the class A. But is there some other member of A that is a member of A during some part of the time span before it has existed or after it ceases to exist?
Yes
Yes
¬Members_Exist(A)
No
No
Question 7b. Is there some time x is not a member of A after x has happened?
Question 7a. Is there some time x is not a member of A while x exists?
No
No
Yes
Question 8b. You have indicated that x is a member of the class A at all times after it has happened. But is there some other member of A that is not a member of A even though it is after it has happened?
Question 8a. You have indicated that x is a member of the class A whenever it exists. But is there some other member of A that is not a member of A at some time it exists?
Question 9. Are objects members of A specifically because of:
(a) having some quality, function, or role
(b) being involved in some process
(c) being a part of some thing
(d) none of the above
Yes
Yes
Yes
No
Type(A)
a
b
c
d
Question 10. Under which type of MaterialEntity are the only members of A?
Question 11. Under which type of MaterialEntity are the only members of A?
Question 12. Under which type of MaterialEntity are the members of A?
Instantiated(A) ^ ¬Members_Exist(A) ^ Non-Rigid(A)
[B]
[B]
[B]
Depends_On(A, B)
Has_Participant(A, B)
Has_Part(A, B)
I don't know
I don't know
I don't know
Depends_On(A, MaterialEntity)
Has_Participant(A, MaterialEntity)
Has_Part(A,MaterialEntity) or Has_Part(A,Process)

Figure 8.1: Decision Tree for Standardizing a Candidate Type

Question 2. Is x:

(a) a physical thing, something that can be seen or touched

(b) a quality, function, or role of a physical thing

(c) something that happens over time, like processes or events

(d) none of the above

If the modeler answers "d" (i.e., none of the above) to **Question 2** then the candidate includes at least one purported member that does not fall within our restricted version of BFO's domain. In this case the following error message is given "x does not refer to an object in our restricted version of the BFO domain" and the procedure terminates.

Note that even if the modeler answers a, b, or c, it is not necessarily the case that x exists, at any time, according to BFO's theory. The reason is that the sentence may be interpreted by the modeler as assuming the notion of existence. As an analogy, if a modeler is asked "Is Bigfoot a vertebrate?" then the modeler is more likely to answer "no" then if the modeler is asked "Is Bigfoot a vertebrate, an invertebrate, or neither". Even though "neither" is given as an option in the latter sentence, the modeler may assume that the question is implicitly "If Bigfoot were real, is Bigfoot a vertebrate, an invertebrate, or neither". Given this, a potentially useful question reordering could be that a question about the existence of the example come before questions about its categorization. However, the current approach attempts to ground the questions first in notions that are more accessible to the modeler. First, the modeler is asked for an example for the class she is modeling (**Question 1**), and second the modeler is asked to choose a corresponding description, each of which describes one of the BFO types *MaterialEntity*, *SpecificiallyDependentContinuant*, and *Process* (**Question 2**). Because existence means something different based on the type of particular being considered, we maintain the current ordering. Furthermore, the issue of assuming existence in **Question 2** is dealt with directly in subsequent questions.

If the modeler answers a, b, or c, then the next question asked is:

Question 3. Aside from x, is every other member of A [Question 2 answer]?

If the modeler answers "no", then the next question asked is:

Question 4. Are members of A either physical things, or else qualities, or else functions, or else roles of a physical thing, or else processes?

If the modeler answers "no", then the class includes some entities that do not exist under BFO's theory (although may include those rare types excluded), therefore the candidate is not a type and the procedure provides the error message "Some purported member of A does not refer to an object in our restricted version of the BFO domain". Following the error message the procedure terminates.

If the modeler answers "yes" then the candidate satisfies **Empty**. The procedure provides the error message "the candidate includes as members objects that are instances of types that have no instances in common", and terminates. In either case if **Question 4** is reached, $\neg\mathbf{Type}(A)$ holds, and re-conceptualization of A is required in order for it to be introduced by our method into the ontology.

If the answer given for Question 3 was "yes", then the next question asked of the modeler is different depending on how the modeler answered Question 2. If the modeler answers (a) or (b) the question is fit to material entities or specifically dependent continuants, but if the modeler answered (c) the question is fit to processes:

Question 5a. Consider the time span before x has existed or after x ceases to exist– is there any part of that time span in which x is a member of A?

Question 5b. Is there some time x is a member of A even though the time is before x has happened, and by the present time, x has still not happened?

Here the example given by the modeler is part of the question to determine if the candidate satisfies **Instantiated**, which is true if the answer is "no". Note though, it is not necessary false that the candidate satisfies **Instantiated** if the modeler answers “no”, specifically because the question is about one particular. Also, the question for material entities, **5a**, helps determine if the candidate satisfies **Members_Exist**, because if the answer given is “yes”, then it follows that $\neg$**Members_Exist**(A). The question for processes, **5b**, helps determine if what x represents is in BFO’s domain. This is due to the fact that a purported process exists for all time if it is indeed a process in BFO’s domain (**Theorem 24**). If the modeler answers “no” to either **Question 5a** or **Question 5b**, a follow-up question is required, which relates the respective question to all members. Again the question is tailored to the kind of particular the example is, and also in this case the other members of the class, as identified through the answers given for **Question 2** and **Question 3**:

Question 6a. You have indicated that x exists whenever it is a member of the class A. But is there some other member of A that is a member of A during some part of the time span before it has existed or after it ceases to exist?

Question 6b. You have indicated that x exists whenever it is a member of the class A. But is there some time some other member of A is a member of A even though it is before x has happened and by the present time, x has still not happened?

If the modeler answers “yes” to **Question 6a** it holds that $\neg$**Members_Exist**(A). If the modeler answers “yes” to **Question 6b** then one or more purported member of A does not refer to an object in our restricted version of the BFO domain. If “no” is the answer given in either case, the next question asked following **Question 6a** and **Question 6b**, respectively, is:

Question 7a. Is there some time x is not a member of A while x exists?

Question 7b. Is there some time x is not a member of A after x has happened?

If the modeler answers "yes" to either **Question 7a** or **Question 7b** then the candidate satisfies **Non-Rigid**. If the modeler answers "no" then the next question is asked following **Question 7a** and **Question 7b**, respectively:

Question 8a. You have indicated that x is a member of the class A whenever it exists. But is there some other member of A that is not a member of A at some time it exists?

Question 8b. You have indicated that x is a member of the class A at all times after it has happened. But is there some other member of A that is not a member of A even though it is after it has happened?

If the modeler answers "yes" to **Question 8b or 8c**, it is also the case that A satisfies the definition of **Non-Rigid**. If the modeler answers "no" then A satisfies **Type**(A). If Question 9 is reached, then A satisfies **Non-Rigid**, and additional questions are required to make the candidate BFO compliant. Question 9 is:

Question 9. Are objects members of A specifically because of:

(a) having some quality, function, or role

(b) being involved in some process

(c) being a part of some thing

(d) none of the above

Answer "a" confirms that the modeler's class definition of A implicitly references a specifically dependent continuant of some type. A could still be a type if re-conceived as a subtype of *SpecificallyDependentContinuant*. A follow-up question is asked:[3]

Question 10. Under which type of *MaterialEntity* are the only members of A?

[3]In the implementation of the decision tree introduced in the next chapter, **Chapter 9**, **Question 10**, **11**, and **12** correspond to the presentation of the ontology's class hierarchy where the modeler is asked to pick the most specific class that all members of the class are also members.

This question is asked to determine the classification of the members of A (as it was originally conceived) under *MaterialEntity*. If the modeler selects a type, B, from the existing ontology, then it is asserted that **Depends_On**(A,B). If the modeler selects "I don't know", then it is more generally asserted that **Depends_On**(A,*MaterialEntity*).

Answer "b" confirms that the modeler's class definition implicitly considers the participants of some indefinite process to be its members. For example, a modeler may conceive a class *Fertilization* such that its members are gametes that participate in some fertilization process. As with Question 10, a follow-up question is asked:

Question 11. Under which type of *MaterialEntity* are the only members of A?

This question is asked to determine of which type of *MaterialEntity*, the objects being classified are instances. If the modeler selects a type, B, from the existing ontology, then it is asserted that **Has_Participant**(A,B). If the modeler selects "I don't know", then it is more generally asserted that **Has_Participant**(A,*MaterialEntity*).

Answer "c" confirms that the modeler's class definition implicitly considers the parts of some indefinite particular to be the members. For example, a modeler may conceive a class *Endocrine System* such that a member is a particular pituitary gland that is part of some endocrine system. A follow-up question is asked:

Question 12. Under which type of *MaterialEntity* are the members of A?

This question is asked to determine under what type of *MaterialEntity*, the objects being classified, are a part. If the modeler selects a type, B, from the existing ontology, then it is asserted that **Has_Part**(A,B). If the modeler selects 'none", then it is more generally asserted that **Has_Part**(A,*MaterialEntity*).

In each case of Questions 10, 11, and 12, the objects being classified are re-classified under a type that has a type-level relation with another type whose definition was implicit in the original (although implicit) class definition. Those candidates conceived as satisfying any combination of **Empty**, **Partial**, ¬**Members_Exist**, or **Non-Rigid** require additional inspection and definition modification if the modeler aims for her ontology that includes such classes to be consistent with BFO.

If a candidate satisfies **Instantiated**, **Members_Exist**, and **Rigid** then we consider the candidate BFO-compliant and satisfies **Type**. If a candidate satisfies **Instantiated**, **Members_Exist**, and **Non-Rigid**, our procedure poses questions to help re-conceive and modify the class such that it is BFO-compliant. The table in Figure 8.2 and 8.3 illustrate six uses cases that follow our decision procedure for standardizing types.

	Candidate 1: *Compound*	Candidate 2: *Reactant*	Candidate 3: *Pineal_Gland*
Question	**Assumed Definition**: ``A substance consisting of two or more different elements combined in a fixed ratio."	**Assumed Definition**: ``The electron donor in a redox reaction."	**Assumed Definition**: ``A small gland on the dorsal surface of the vertebrate forebrain that secretes the hormone melatonin."
1	"sodium chloride in this container"	"sodium chloride is this container"	"an organ inside my body"
2	a	a	a
3	yes	yes	yes
4	-	-	-
5	no	no	no
6	no	no	no
7	no	yes	no
8	no	-	no
9	-	a	-
10	-	Compound	-
11	-	-	-
12	-	-	-
Result	**is_a**(*Compound* , *MaterialEntity*)	**Depends_On**(*Reactant* , *Compound*)	**is_a**(*PinealGland* , *MaterialEntity*)

Figure 8.2: Responses to Decision Procedure Following Decision Tree

	Candidate 4: *Gamete*	Candidate: 5 *Myocardial_Infarction*	Candidate 6: *Full_Eye_Transplant*
Question	**Assumed Definition**: ```A haploid reproductive cell, such as an egg or sperm. Gametes unite during sexual reproduction to produce a diploid zygote.''	**Assumed Definition:** "Death of a section of heart muscle when its blood supply is cut off, usually by a blood clot in a coronary artery narrowed by atherosclerosis" (merriam-webster.com)	**Assumed Definition:** ``Transplant of an entire eye.''
1	"A sperm cell''	"experienced by an individual with high blood pressure, also locus identified by a pathologist"	"a procedure that may be possible in the future"
2	a	c	c
3	yes	no	yes
4	-	yes	-
5	no	-	yes
6	no	-	-
7	no	-	-
8	no	-	-
9	-	-	-
10	-	-	-
11	-	-	-
12	-	-	-
Result	**is_a**(*Gamete* ,*MaterialEntity*)	**Heterogeneous**(*Myocardial_Infarction*)	member of *Full_Eye_Transplant* outside BFO domain

Figure 8.3: Responses to Decision Procedure Following Decision Tree

Chapter 9

Implementation

9.1 Introduction

Our decision-tree algorithm (introduced in **Section 8.3**) for adding classes to an ontology that should be compliant with BFO—enforces that a modeler take into consideration, for each subclassing assertion, the models of the ontology based on the assertion. The decision tree algorithm is implemented in the form of an interactive Wizard Plugin for Protégé 4.1 (Knublauch et al., 2004). We chose the Wizard-style plugin format because it lends itself to the decision-tree question answering. Within a software installation wizard, a user is allowed to configure the software installation with certain parameters. Similarly, our Wizard plugin allows a user to formally model a class a certain way. Additionally, a software installation wizard may not complete its execution successfully due to requirements of the software installation not being met (e.g., not enough disk space). Similarly, our Wizard Plugin will not allow a class conceived and formalized in a way not compliant with BFO to be added to an ontology.

The wizard follows the decision-tree algorithm, and assumes that a modeler is starting to build an ontology from scratch, introducing one class at a time. As mentioned, the basis for the questions of the decision-tree is the formal integration of BFOs theory of types and OntoClean's notion Rigidity. In the approach we use natural language questions to determine from the modeler if **Instantiated**, **Members_Exist**, **Rigid** or **Non-Rigid** hold. This approach is limited by the fact that we attempt to use everyday language in the questions, and BFO's theory has its own informal and formal vocabulary which overlaps with everyday language. To help address this limitation, for questions that are presented to the modeler in the software, in some cases we present hints, in the form of elaborations and examples, aimed to help a modeler understand BFO's usage of terms. Further, in the event where the system does not add a class to the ontology, based on how it is conceived by the modeler, in all cases the system presents an explanation as to why the class was not added to the ontology.

Violations of disjointness axioms of an ontology loaded into Protégé are made apparent only after a classifier (i.e., DL reasoner) is run on the ontology, therefore there may be inconsistencies in the ontology without a modeler's knowledge. To mitigate this problem, Rector et al. (2004) encourage users to execute a classifier early and often. However, in typically modeling situations, the classifier may not be run until after several modeling mistakes have been made. Our implementation, based on our decision tree algorithm, enforces that an ontology remains consistent with respect to the disjointness axioms of upper ontology types, by asking certain questions, and applying the respective answers to restricting where a class is rooted. There is no current plugin for Protégé that accomplishes this for a modeler with respect to an upper ontology.

9.2 Demonstration

In this section we provide a demonstration of the BFO-Rigidity Decision Tree Wizard Protégé plugin, which implements the decision tree presented in **Chapter 8**. A demonstration is given for the introduction of what is found to be a Rigid class into an ontology (as illustrated in **Appendix E**), and another demonstration is given of what is found to be a Non-Rigid class into an ontology (as illustrated in **Appendix F - H**).

9.2.1 Rigid Example

The first screen of the Wizard asks the modeler for the name of the class they are adding to the ontology (**Figure E.1**). The bottom of the screen provides useful tips that helps the modeler choose the class name. The tips suggest that the modeler avoid mass nouns (e.g., glass, sugar), which are cross-granular in nature, and recommend that the modeler choose a term that takes a determiner like 'every', 'few' and 'many' and avoid terms that take determines like 'some', 'little' and 'much'. In this case the modeler enters 'Person'.

In the next screen the Wizard asks the modeler to enter a prototypical example member of the class *Person* (**Figure E.2**). Again the screen provides tips, which help the modeler understand that only particulars are members of classes and provide various examples of particulars. There is also a tip that the modeler may reuse previously given examples. In this case, the modeler enters 'Tom'.

In the next screen the Wizard asks the modeler to categorize Tom, and for provides choices based on descriptions that correspond to the BFO types *MaterialEntity*, *SpecificallyDependentContinuant*, and *Process* (**Figure E.3**). To help the modeler with what it means for something matching one of these descriptions to fall under BFO's domain, tips are given that describes what kinds of particulars are included. Also given are specific examples of things that are excluded. In this case the user selects the first choice, which is the choice corresponding to *MaterialEntity*.

In the next screen the Wizard asks a question to determine if all members of *Person* are instances of the BFO type of which Tom is an instance (**Figure E.4**). In this case the user selects 'Yes'. In the next screen, the Wizard asks a question to determine if Tom is a member only at times he exists (**Figure E.5**). The same sort of question is asked in the next screen, but with respect to all members of the class (**Figure E.6**). For both screens the question is tailored to the notion of existence for the corresponding BFO type chosen for Tom and all members in the 'Categorize Example' screen (**Figure E.3**) and the

'Homogeneity' screen (**Figure E.4**), respectively. In both screens the modeler answers 'no'.

In the next screen the Wizard asks a question to determine if Tom is not a member of *Person* at a time he exists (**Figure E.7**). Here examples are given to assist the modeler with the decision. In the next screen the Wizard asks a question to determine if any other member of *Person* exists at a time it is not a member of the class (**Figure E.8**). In both screens the modeler answers 'no'.

In the next screen the Wizard presents the modeler with the current class hierarchy, and asks him to choose the some specific class that all member of the class *Person* is also a member (**Figure E.9**). In this case the modeler chooses *MaterialEntity*. In the next screen the Wizard indicates the assertion that it makes on behalf of the modeler, which in this case is that *Person* is a subtype of *MaterialEntity* (Figure **E.10**). The modeler is given an opportunity to add additional classes to the ontology, and in this case selects 'Yes', and proceeds to add the next class he wants to introduce into the ontology through repetition of the system.

9.2.2 Non-Rigid Example

For the first and second screen the modeler enters 'Reactant' and 'the compound on the table' as the class name and the example member, respectively (**Figure F.1, F.2**). In the next four screens (**Figure F.3 - F.6**) the modeler answers the questions as he did when introducing the *Person* class. In the very next screen the modeler answers 'Yes', which confirms that the example does exist at a time it is not a member of *Reactant* (**Figure F.7**). This implies that the class satisfies **Non-Rigid**, which leads to a different set of menus then if the class satisfied **Rigid**, as was the case in the previous example.

In the next screen the Wizard asks the modeler if objects are members of *Reactant* because of having some quality, function, or role, being involved in some process, being a part of something, or none (**Figure F.8**). Various tips are given to help the modeler decide. In this case the modeler makes the first choice. In the next screen the Wizard asks the modeler to choose the most specific class that all member of *Reactant* are members of (**Figure F.9**). In this case the modeler chooses *MaterialEntity*.

The next screen requires that the Wizard provide the modeler with information about BFO's commitment before asking a modeling question (**Figure F.10**). This is given under 'Wizard Interpretations'. Based on our decision tree, the Wizard uses the class name given for the class being modeled to create a new class under the type hierarchy of *SpecificallyDependentContinuant*. The question the Wizard asks helps it determine if things that are reactants belong to a more specific class or classes. In this case the modeler decides that there is a more specific class or are more specific classes of particulars that are reactants. As described, making this choice means that the modeler will be guided through the process of adding the respective additional class(es), and for this the Wizard repeats the process of adding a class.

For the first and second screen the modeler enters 'Compound' and 'the compound on the table' as the class name and the example member, respectively (**Figure G.1, G.2**). In the next seven screens (**Figure G.3 - G.9**) the modeler answers the questions as he did when introducing the *Person* class (**Figure E.3 - E.9**), which, among other assertions, confirms that *Compound* satisfies **Rigid**. Note that there is one difference in that **Figure G.3** includes a Yes/No question to ask the modeler whether or not the class's members satisfy the description of *MaterialEntity*. This is asked instead of the usual multiple choice question because the modeler has already confirmed that all members of *Reactant* are material entities (**Figure F.4**). Therefore if members of *Compound* are not material entities, it would be a modeling mistake.

The next screen, **Figure G.10**, is very similar to **Figure F.10**, with the exception that *Compound* is now the class that the *SpecificDependentContinuant* type *Reactant* is restricted to, instead of the more general type *MaterialEntity*. Again, the modeler decides that there is a more specific class or are more specific classes of particulars that are reactants. Again, as described, making this choice means that the modeler will be guided through the process of adding the respective additional class(es), and for this the Wizard repeats the process of adding a class.

For the first and second screen the modeler enters ‘Element’ and ‘the element on the table’ as the class name and the example member, respectively (**Figure H.1, H.2**). In the next seven screens (**Figure H.3 - H.9**) the modeler answers the questions as he did it when introducing the *Compound* class (**Figure G.3 - G.9**) and the *Person* class (**Figure E.3 - E.9)**, which among other assertions confirms that *Element* satisfies **Rigid**. The next screen, **Figure H.10**, is very similar to **Figure G.10**, with the exception that the disjunction *Compound* $\vee$ *Element* is now the class that the *SpecificDependentContinuant* type *Reactant* is restricted to, instead of more specifically *Compound.* In this screen the modeler decides this is a sufficient restriction for the modeling of the class *Reactant.*

In the next screen the Wizard display the axioms to be asserted, which is that *Reactant* is a subtype of *SpecificallyDependentContinuant*, and the formal equivalent that every reactant depends on some compound or element, and is dependent only on compounds and reactants (**Figure H.11**). **Figure H.12** provides the axioms as they appear in Protégé.

Part IV

Discussion and Future Directions

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 10

Introduction

The decision tree presented in **Section 8.3** is based on the axioms and theorems which result from our integration of OntoClean with BFO. The decision tree helps a modeler determine whether or not **Instantiated**, **Members_Exist**, **Rigid** or **Non-Rigid** hold for a given class. When **Instantiated**, **Members_Exist**, and **Non-Rigid** hold for a class, the tree helps determine what implicit relations hold between members of the class and members of some other classes that are BFO types.

The OntoClean metaproperties Rigidity, Unity, and Identity are based on an analysis of classes called *sortals*; the entities that are members of various sortals are also instances of BFO's type *Object.* Given this, the integration work squarely falls within the scope of modeling objects. However as given in **Chapter 8**, we apply the notion of Non-Rigid to formalize classes in such a way that they are compliant with BFO, which clearly requires and enables the modeling of instances of *Specifically Dependent Continuant* (e.g., qualities, functions, roles) in connection with objects. It also requires and enables the modeling of processes in connection with objects, and furthermore, it enables the modeling of the parthood relation for objects and processes. Therefore within this scope our decision tree does help model classes that are not sortals, however; given the current state of the BFO literature, there are certain limitations to doing so.

There are some modeling assumptions for a class that are not compliant with BFO (a class satisfying **Members_Exist**, **Instantiated**, and **Non-Rigid**), but that our decision tree attempts to help augment in such a way that the class can be reconceived to be compliant with BFO. There are other modeling assumption for a class that the decision tree cannot help reconceive easily or at all (a class satisfying ¬**Members_Exist**, ¬**Instantiated**, and/or **Heterogeneous**). When these assumptions are detected through answers to decision tree questions, informative error messages are given to help the modeler address how the class might be reconceived to be compliant with BFO. Finally, there are some assumptions for a class that the decision tree cannot help reconceive due to certain limitations of BFO (a class satisfying **Members_Exist**, **Instantiated**, and **Non-Rigid** and the implicit relationship being modeled is not clearly one of the BFO primitive relations **depends_on**, **participates_in**, and **part_of**). In what follows we discuss in detail these limitations and, where appropriate, propose them as areas for future work. In what follows we discuss in detail these limitations of the integration work, which in some respects reflect the current state of development of BFO.

Chapter 11

Restricted Domain Modeling

As discussed, we restrict our decision tree to the modeling of material entities, specifically dependent continuants (SDCs), and processes. The tree does not assist with the modeling of *Region* classes, for the reasons given in **Section 8.3, p. 116**, and also does not assist with the modeling of controversial classes based on stages of development (e.g., fetus, child), for the reasons given in **Section 6.1, p. 52**. It also excludes modeling of classes which are more specifically subtypes of the differentia of *SpecificiallyDependentContinuant*: *Function*, *Role*, and *Quality*. For the body of work that covers realizable entities (includes functions and roles, but excludes qualities), Arp and Smith (2008, p. 3) discuss that roles are optional in comparison to the other subclasses of specifically dependent continuant. Specifically, there are certain questions mentioned that, if a modeler should answer them, will help them designate if a class is a *Role* type or a *Function* type. If the question "Is this realizable such that its typical manifestations are based on physical structures?" is answered "yes", then the particular is an instance of *Function* type. If the question "Is this realizable such that its typical manifestation is a reflection of surrounding circumstances, especially involving social ascription, which are optional?" is answered "yes", then the particular is an instance of a *Role* type. Unfortunately though, this part of BFO's theory, and that involving qualities, is given entirely in English without axioms. Our approach for this dissertation is to provide a formal integration of BFO with OntoClean, and apply those formal notions to constructing a decision tree that serves as a basis for a software interface for constructing ontologies that are compliant with BFO. Since the work on realizables, and the distinction between realizables and qualities is not yet formal, we exclude these types, the differentia of *SpecificiallyDependentContinuant,* from our modeling approach also.

For the same reason and also for the reason that it is not relevant to modeling sortals or modeling of BFO types in connection with modeling sortals, identity procedures for realizables, qualities, generically dependent continuants, and processes are not part of the work either. Realizables can exist while not being realized, and under what sort of conditions a realizable exists while not being realized is not stated or axiomatized yet. If and once this work is completed, future work involves adapting the integration axioms, the decision tree, and software to these further developments in BFO.

Chapter 12

Non-Rigidity and Canonicity

As demonstrated in **Section 8.3**, we are able to use the notion of Non-Rigid to assist a modeler in identifying implicit type-level relations. In the case of parthood, there are some weaknesses in the way that the current type-level relations are formalized. For example, using the decision tree, a modeler may be led to assert **Has_Part**(*Human Body*,*Appendix*), while it is not the case that every human body has an appendix as a part of it, since an appendix can be removed from a body during surgery. The issue can be resolved by representing what we consider a prototype of what we aim to represent, by a *canonical ontology.* As opposed to constructing an ontology based on how an organism in the world is anatomically structured, a canonical ontology of anatomy is restricted to organisms that follow a model of anatomy (Neuhaus and Smith, 2008, p. 15). The current case is human canonical anatomy, which is the anatomical makeup of the typical human, or in other words, the model of anatomy explained in medical textbooks.

To address this issue it is not sufficient to express that the type-level relation holds merely between canonical objects where **part_of** holds. For example, it is not the case that every canonical pituitary gland is a part of some canonical endocrine system, because a canonical pituitary gland may be part of an endocrine system without a pineal gland, for example, it having been removed from a cadaver. We must also constrain the type-level relation for parthood to pairs of objects which are both part of a larger canonical "system". In the examples covered thus far, this system would be the canonical human body.

Neuhaus and Smith (2008, p. 15) address this problem by defining a relation, $\mathbf{I}_{\mathrm{u}}(x,t)$, which, to address our examples, means that x is a human body that is in conformity with anatomy u at t. By $\mathbf{I}_{\mathrm{u}}(x,t)$, x is within the domain of bodies that have a canonical anatomy. If x is in conformity with u at t then the the formulas in the canonical ontology at t are true. This includes that if every member *Hand* is a part of some member of *Arm* within the canonical domain, for these pairs of hands and arms,[1] there is some body x of which they are both of a part. This is also formulated for the **Has_Part** relation: if every member *Arm* has as a part some member of *Hand* within the canonical domain, for these pairs of hands and arms, there is some body x of which they are both of a part.

Note however that this work is not a part of BFO 1.1, and since this version of BFO is one of the foundations of our integration, this work falls out of the current scope. For the current work we must assume that for type-level parthood relations canonical domains are assumed.

[1]We assume this specific relationship is functional.

Chapter 13

Status of Classes that are Not Types

13.1 Introduction

In what follows we discuss issues of modeling under the constraints of the Disjointness and Single Inheritance principles, modeling (Non-Rigid) classes that are not types, and modeling Rigid class that are not types.

13.2 Constraints of the Disjointness and Single Inheritance Principles

As was previously discussed the original BFO literature said that ontologies should consist only of types, excluding classes that are not types (Spear, 2007, p. 21). Furthermore, ontologies must abide by the Disjointness Principle and the Single Inheritance Principle (Spear, 2007, p. 121). These approaches were therefore naturally assumed for the OBO Foundry ontologies once BFO was adopted as the project's upper ontology. In recent efforts of the OBO Foundry community, it has become clear that these constraints are often difficult to enforce in practical modeling. For example, in the Ontology for Biomedical Investigations (OBI),[1] *Hybridization Oven* is a subclass of *Incubator* and *Container*, where neither is given as a subclass of the other.

In recent work, Smith and Ceusters (2010) advocate the principle of asserted single inheritance, citing Alan Rector's work on the topic:

> Each reference ontology module should be built as an *asserted mono-hierarchy*, a hierarchy in which each term has at most one parent. (Rector, 2003)

The notion of asserted versus inferred hierarchy is relevant within the context of Description Logics, where an asserted class hierarchy is constructed purely on the set of axioms of an ontology, and an inferred hierarchy is obtained by the application of a classifier (i.e., a classification reasoner) on the set of axioms of an ontology. A classifier determines, from the necessary and sufficient conditions of each class definition, what all its subclasses and superclasses are. Note that if A is asserted as a subclass of B, it is trivial that A is inferred as a subclass of B.

I evaluate this principle in the scope of our example from OBI, that *Hybridization Oven* is asserted as a subclass of both *Incubator* and *Container*. Together, the two corresponding asserted axioms violate the aforementioned principle of asserted single inheritance. *Incubator* is an inferred subclass of *Device*, and *Container* is fully defined (i.e., the necessary and sufficient conditions are given) such that it is a subclass of *Device* and has the function of *contain function*.[2] Therefore, only the condition that *Hybridization Oven* has the function of *contain function* would need to be given to infer that *Hybridization Oven* is a subclass of *Container*. If the definition of *Hybridization Oven* were changed in this manner, i.e., that it has the function *contain function* instead of being a subclass of *Container*, the violation of the principle of asserted mono-hierarchy,

[1] http://purl.obolibrary.org/obo/obi

[2] In BFO the notion of *a function* is stricter than what are considered functions in everyday language. As mentioned in **Section 6.2**, artifacts have function through intentional design and exist in virtue of their physical makeup.

condoned by Rector, would be alleviated.

Even with these changes, *Hybridization Oven* still has two superclasses (one asserted, the other inferred), where one is not a subclass of the other, which is a violation of the principle of disjointness. By adopting the principle of asserted single inheritance for OBO, much is conceded in the way of ontology, for modeling practicality. Firstly, if A is a subclass of B, by inference instead of by an assertion, it still holds that A is a subclass of B. The objects that satisfy A are objects that satisfy B, i.e., the semantics do not differ. Adopting this weaker principle for the **is_a** hierarchy and disregarding the principle of disjointness allows for multiple inheritance, plain and simple. Another issue within the discussion of (Smith and Ceusters, 2010) is that the **is_a** relation is discussed as the only relation for classification, therefore the dichotomy of types and classes that are not types is not addressed.

I believe that the principle of asserted mono-hierarchy misses the point, however, because it focuses on the manner in which an ontology is specified and not on the *the classification units* applied to the objects of the ontology. There is a way to preserve the original principles of BFO, in part by maintaining the classification units, *class* and *type*, and instead adopting another, related principle that Rector (2003) sets forth. The principle is that in an ontology each class has no more than one *primitive* parent (i.e., immediate superclass). This principle and the principle of asserted mono-hierarchies are actually very similar, and would be one and the same under the assumption that classes can only be asserted as subclasses of those classes that are primitive. Alas, no such assumption is made nor is the assertion of subclasses of defined classes prevented in any domain modeling tools, such as Protege.

The notion of a primitive is connected to that of types. Types are extremely difficult to define completely in a formal language (Rector et al., 2004, p.13). Given this, types are usually primitive classes of an ontology, with the exception of types that are fully defined via a covering axiom.[3] Note that, however, not

[3]In the OWL version of BFO (http://www.ifomis.org/bfo/), *Entity* is defined by the disjunction of *Continuant* and *Occurrent*.

all primitive classes are types, simply by the fact that some classes may be left incompletely specified by a modeler, for any number of potential reasons.

From our example, *Hybridization Oven* is consistent with this principle because *Container* is a defined class, therefore *Hybridization Oven* only has one primitive superclass, *Incubator*. By this approach, then, *Container* is not a type because it is a defined class, and not a defined class due to a covering axiom. In adapting this principle, the modeler must have a way of annotating those classes which are primitive temporarily and will be defined at a later time, for excluding them for the evaluation of whether or not the ontology in question is consistent with the one primitive parent principle. Looking more closely at the class *Incubator*, we suggest that it could be one such class and not a type, because its primitive definition is primarily based on a function.

Given this clarification, the main issue is then whether or not a BFO-compliant ontology may include classes which are not types. If not, then *Container* would be excluded from the ontology, as it is fully defined. In general, there are certain classes that are not types but satisfy **Instantiated** and **Members_Exist** (e.g., *Container*) that may be appropriate in a domain ontology and there are other classes that do not satisfy either **Instantiated** or **Members_Exist** (e.g., *Unicorn*) that are certainly not appropriate in a domain ontology. We continue to exclude both kinds of classes from our ontology evaluation method, since our method remains centered on the importance of developing a well-founded type hierarchy and compliance with BFO. We do concede, however, in some cases, that the former kind may have utility and recommend that they be added later in development of the ontology, for application-specific purposes and with clear annotation that designates them as such. The type hierarchy, hence, provides a foundation for adding such classes which in the context of DLs are "defined".

13.3 Modeling Classes that are not Types

In the event that there is a potential change in policy such that “defined” classes (as discussed in the previous section **Section 9.4**) are deemed BFO-compliant, we herein explore the impact this has on our method of evaluation, i.e., the decision tree, and how it might be changed. For some branches of the decision tree (Question 9 answer “a” and “b”), the introduced candidate must be “re-conceived” in order to reach type status. The designation of a class as *Non-Rigid* allows the decision tree to facilitate this reconceptualization. For example if the modeler introduces the candidate *Student*, given the expected answers to the decision tree questions for such a class, at the end of the decision tree an asserted axiom is **Depends_On**(*Student*,*Person*). By our approach the label applied for the class of students that is originally conceived by the modeler is applied to a different class, that which BFO considers a subtype of *Role*. The benefit of our approach is that the modeler’s intent is applied to help her reconceive the class such that it is compliant with BFO. The difficulty with this approach is that (a) the modeler is required to change their thinking for the class they introduced and (b) the class the modeler originally had in mind is not a part of the ontology following application of the decision tree.

The difficulty of (a) cannot be avoided, it can only be explained, because ultimately the user must comprehend why the Wizard has asserted certain axioms on their behalf, and, ergo, come to comprehend the categories of BFO. With the assumed change in policy, for (b), alternatively the class *Student* would be kept in the ontology and defined by its relationship with a class in the *Role* hierarchy (here assumed *Student-Role*) and its classification under a BFO type (here assumed *Person*) **subclass_of**(*Student*,*Person*) $\wedge$ **Depends_On**(*Student-Role*,*Person*) $\wedge$ **Has_Dependent**(*Student*,*Student-Role*).[4] Clearly then, the change in policy allows the modeler to still introduce the class she had in mind, in the case of Non-Rigid classes like these.

[4]This is formalized similar to how **Has_Part** is in comparison to **Part_Of**. **Has_Dependent**(A,B) iff for every instance x of A at t there is some instance y of B at t such that y depends on x at t.

13.4 Rigid Classes That are Not Types

There are some classes which satisfy **Instantiated**, **Members_Exist**, and **Rigid** that the originators of BFO would not consider types. Take for example the class *Made in China* or the class *Made in 2011*, which have as members material artifacts. Modeling of these classes also leads to multiple inheritance. Although clearly BFO is applied as the upper ontology of biomedical ontologies, as a representation of reality the BFO theory needs to address these cases.

13.5 Non-Rigid Processes

Although nothing in its axiomatization prevents a wider scope, we have applied **Non-Rigid** to classes whose members are material entities, given its purpose for modeling sortals and their relationships with other entities. Therefore our decision tree does not address the issue of modeling Non-Rigid *Process* classes. An example of a Non-Rigid process is a symptom of feeling nauseous. The notion that the process of feeling nauseous is a symptom is based purely on it being reported in the context of a medical health care situation. Prior to being reported, the process is not a symptom.

During the testing of the decision tree Wizard, it was determined that such a process is Non-Rigid. We don't have a way to formulate the relationship of the underlying process with another entity based on the Non-Rigid designation, as we did for specifically dependent continuants. We suggest that for future work the BFO originators address this matter, and the decision tree can be adapted accordingly.

Chapter 14

Exploiting Heterogeneity to Identify Other Type-Level Relations

We define a predicate **Heterogeneous** which is satisfied by any class that has as members instances of at least two types that are disjoint. Examples include a class *PDF* that has as members print-outs of pdf files and pdf files themselves (which are informational artifacts that fall under *GenericallyDependentContinuant*), and a class *Myocardial Infarction* which has as members heart attack events and collections of dead cardiac cells. As is demonstrated for Non-Rigid classes, there is a potential opportunity to identify type-level relations that hold between the members of classes that are heterogeneous. Alas, neither the Relation Ontology nor the Information Artifact Ontology are mature enough to provide such relations, at least for the example cases given.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 15

Experimental Work on Connecting Identity to Information Artifact Ontology Classes

In **Section 7.2.2**, we provide a detailed formalization of identity procedures, such that there are two parts of the procedure which have a result and where the results are matched against one another. In what follows we explore how connect this formalization to an existing ontology that formalizes these notions, but ultimately leave the completion of such a task for future work.

To answer this question For this exploration we turn to the Information Artifact Ontology (IAO), a domain level ontology that uses BFO as an upper ontology.[1] This ontology includes a subtype of *Process*, *Planned Process*, which is defined such that an instance is "a processual entity that realizes a plan which is the concretization of a plan specification". Our notion of necessary identity

[1]http://purl.obolibrary.org/obo/iao.owl

procedure fits this definition of *Planned Process* therefore we could consider that *Identity Procedure* is a subtype of *Planned Process.* An editor note explains that the notion of a plan is not vetted, indicating "this class is included to make clear how the plan specification, the plan, and the planned process relate. OBI will however only subclass and work under the 'plan specification', and 'planned process' class, as we want to avoid to get deep into discussions of 'intent' etc".

In the formal OWL definition plan specification is restricted to those that have as a part some objective specification. By the natural language definition given in OBI an objective specification is:[2]

> A directive information entity that describes an intended process endpoint. When part of a plan specification, the concretization is realized in a planned process in which the bearer tries to effect the world so that the process endpoint is achieved.

Examples are given as "purpose of a study, support of hypothesis, or discovery"and an editor note indicates that an instance of this type "Answers the question, why did you do this experiment?". This class may be close to what would be a subclass of a class that whose definition corresponds to the notion of identity criterion put forth by Guarino and Welty. For example, the description for the purported necessary identity criterion *having the same genotype*, "having the same genotype", serves to partially describe the object specification class "to determine if two evaluants have the same genotype".

Furthermore, IAO includes a relation (i.e., an OWL property) that is a subrelation of **has_participant**, **has_specified_output**, which is defined as:

> A relation between a planned process and a continuant participating in that process. The presence of the continuant at the end of the process is explicitly specified in the objective specification which the process realizes the concretization of.

In the case of an objective specification class we describe as "to determine

[2] "Note that the status indicates "pending final vetting".

if two evaluants have the same genotype" of a planned process P, given **confirmed**(P,p,w,v,t,t_1), one such continuant in this relation is what OBI considers an information content entity, which is about the identity of w at t, another such continuant is another information content entity that is about the identity of v at t_1, and finally an information content entity that is a boolean value that is the output of a test comparing those information content entities about w and v.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 16

External Dependence

Previously we mentioned that we do not model subtypes and subrelations of *SDC* and **depends_on**, respectively, due to lack of axioms that differentiate them in BFO. For this same reason we do not integrate external dependence with BFO, which heavily relies on a theory of roles.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 17

Implementation Advances

When a candidate type does not follow the criteria to be a type nor is reconceived to be type following confirmation of the candidate as Non-Rigid, the software simply terminates with an explanation as to why the candidate is not a type. There however is value in storing decision-tree question answers, in cases where the answers to questions change upon further analysis by the modeler, for updated answers which confirm the candidate as a type. To provide for the flexibility of allowing a modeler to revisit the modeling of such a candidate, future work on the implementation will store the question answers to non-compliant classes in annotation properties, and will add as a subclass of a class designated for those candidate not yet types, *Candidate*.

Other future implementation advances include incorporating the integration of Unity and Identity into the decision tree and Wizard implementation. We envision for this work that for each class that is rooted in the class hierarchy, the wizard queries the identity procedures and unifying relations for the corresponding superclasses. If the modeler confirms that they hold for the candidate type, the Wizard asks the user to add any additional identity procedures or unifying relations that apply to the newly introduced candidate type.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 18

Conclusions

The BFO-Rigidity Decision Tree Wizard offers an analysis that an experienced ontologist normally performs intuitively when adding classes to an ontology. The Wizard therefore makes this exercise both explicit and directed. While testing the user interface of the Wizard it became apparent that the Wizard forces a modeler to think about aspects of the class are modeling that he may not have considered when first introducing the class. We think this is one of the values of the Wizard; a class will not be added to the ontology until the modeler undergoes the necessary thinking process needed for the class to be considered an addition to a BFO-compliant ontology.

One of the benefits of isolating classes which satisfy **Heterogeneous** is that it identifies and disallows candidate types into a domain ontology which are inherently upper level. For example if a modeler introduces a class *Thing* which has as members what BFO considers *continuants* and *occurrents*, it is an upper level class and is redundant with *Entity*. Another benefit is to isolate and eliminate candidate types that conflate the classification under both *Continuant* and *Occurrent*, as was done in the LinKBase system (Simon and Smith, 2004, p. 6) (Simon et al., 2006, p. 227)

After a class name A is entered, the first question (**Question 1**) asks: "What is a specific example of something that is a prototypical member of the class A?"

Once **Question 1** is answered the modeler has entered both a class name A and an example member a of the class, therefore $\exists t(\textbf{member\_of}(a,A,t))$. **Question 2** attempts to determine if a is an instance of the type (a) *MaterialEntity*, (b) *SpecificallyDependentContinuant*, or (c) *Process*. **Question 3** then determines if all members of A are instances of the same type selected in **Question 2**. So for instance, if (a) is selected for **Question 2**, this corresponds to the assertion $\exists t(\textbf{member\_of}(a,\textit{MaterialEntity},t))$, and if **Question 3** is answered "yes", then this corresponds to the assertion $\exists t(\textbf{member\_of}(x,A,t)) \rightarrow \exists t_1(\textbf{member\_of}(x,\textit{MaterialEntity},t))$. If **Question 2** is answered (d), then, based on our restriction of modeling to material entities, specifically dependent continuants, and processes, it is not the case that what 'a' represents falls within BFO's domain.[1]

Axiom 1 represents that an entity x in BFO's domain exists at some time. What cannot be represented, due to the contradictory nature of the assertion, but is assumed under BFO theory is that if something does not fall within what x ranges over, it does not exist at any time. Therefore the choice (d) here, although unintutively, corresponds to the assertion that $\neg\exists t(\textbf{exists\_at}(a,t))$ due to the fact that 'a' does not refer to an object in our restricted version of the BFO domain. This however leads to an inconsistent ontology because by application of **Axiom 1** $\neg\exists t(\textbf{exists\_at}(a,t)) \wedge \exists t(\textbf{exists\_at}(a,t))$, a contradiction. In terms of revising the ontology such that it is consistent, at least one of those conjuncts must be removed. Since the latter conjunct is based on an axiom of our system, we would normally consider that it is the former conjunct that should be removed, $\neg\exists t(\textbf{exists\_at}(a,t))$. Unfortunately, removal of this conjunct is not the case either, based on the answer (d) given by the modeler to **Question 2**. Ultimately, to what 'a' refers does not fall within BFO's domain, and the inconsistency is resolved by removing the conjunctive assertion and a as a purported particular. Since there may be other objects the modelers consider members of A, it is not necessarily the case that A satisfies **Empty**, but it is

[1]As discussed in **Section 8.3**, we restrict our domain of particulars by excluding uncommonly represented types.

the case if every member follows the modeler's assumptions for a.

Question 5 and **Question 6** determine whether or not the example and then all members of the class exist at all times they are members, respectively.[2] For **Question 5** the answers and corresponding asserted formulas are:

(yes): $\exists t(\mathbf{member\_of}(a,A,t) \wedge \neg\mathbf{exists\_at}(a,t))$
(no): $\mathbf{member\_of}(a,A,t) \rightarrow \mathbf{exists\_at}(a,t)$

Based on a "no" assertion for **Question 5**, and the answer to **Question 1**, which corresponds to $\exists t(\mathbf{member\_of}(a,A,t)$, it follows that $\exists t(\mathbf{member\_of}(a,A,t) \wedge \mathbf{exists\_at}(a,t))$. From this formula and **Definition 2** it follows that $\mathbf{Instantiated}(A)$. From the "yes" assertion for **Question 5a** it follows that $\neg\mathbf{Members\_Exists}(A)$ (based on **Definition 4**), and from that it follows that $\neg\mathbf{Type}(A)$ (based on **Axiom 5**). Note that for the version of **Question 5** for processes, **5b**, if "yes" were to hold, this would lead to an inconsistent ontology. This is due to the fact that processes exist for all time. For **Question 6** the answers and corresponding asserted formulas are:

(yes): $\exists xt(\mathbf{member\_of}(x,A,t) \wedge \neg\mathbf{exists\_at}(x,t))$
(no): $\mathbf{member\_of}(x,A,t) \rightarrow \mathbf{exists\_at}(x,t)$

From the "yes" assertion for **Question 6a** also it follows that $\neg\mathbf{Members\_Exists}(A)$ (based on **Definition 4**), and from that it follows that $\neg\mathbf{Type}(A)$ (based on **Axiom 5**). As with **Question 5,** for the version of **Question 6** for processes, **6b**, if "yes" were to hold, this would lead to an inconsistent ontology. If a "no" assertion is made for both **Question 5** or **Question 6**, additional decision tree questions are posed.

Question 7 and **Question 8** determine whether or not the example and then all members of the class are members at all time they exist. For **Question 7** the answers and corresponding assertions asserted formulas are:

[2]**Question** 5, 6, 7, and 8 have alternate versions: (a) phrases the question to address the existence of material entities or specifically dependent continuants, and (b) phrases the question to address the existence of processes. The version asked of the modeler is based on previous answers. For simplicity, in this chapter we refer to the overall question instead of the specific version.

(yes): $\exists t(\neg \textbf{member\_of}(a,A,t) \wedge \textbf{exists\_at}(a,t))$

(no): $\textbf{member\_of}(a,A,t) \rightarrow \forall t_1(\textbf{exists\_at}(a,t_1) \rightarrow \textbf{member\_of}(a,A,t_1))$

For **Question 8** the answer and corresponding asserted formulas are:

(yes): $\exists xt(\neg \textbf{member\_of}(x,A,t) \wedge \textbf{exists\_at}(x,t))$

(no): $\textbf{member\_of}(x,A,t) \rightarrow \forall t_1(\textbf{exists\_at}(x,t_1) \rightarrow \textbf{member\_of}(x,A,t_1))$

Based on "yes" assertions for **Question 7** and **Question 8**, and the assertion of **Question 1** ($\exists t(\textbf{member\_of}(a,A,t))$), it follows that **Rigid**(A) (based on **Definition 5**). Based on a "no" assertion for either **Question 7** or **Question 8**, it follows that **Non-Rigid**(A) (based on **Definition 5** and **Definition 6**, but also given immediately in **Theorem 21**). Although we don't formalize it in our system due to possible exceptions,[3] for the sake of the decision tree we assume that if A satisfies **Instantiated**, **Members_Exist**, and **Rigid** it also satisfies **Type**. For the candidate A, the answer "no" to **Question** 5, 6, 7 or 8 results in the inference that $\neg$**Type**(A), which can be shown using **Axiom 5** and **Axiom 6**.

Ultimately, the answers to questions of the decision tree correspond to formulas that can be asserted in our formal system, and the inferences based on these formulas are implicit made within the decision tree structure. The rest of the decision tree starting at **Question 9** addresses how to model the candidate based on it satisfying **Non-Rigid** (and also satisfying **Instantiated** and **Members_Exist**).

The notion that a class is exclusively **Rigid** or **Non-Rigid** (**Theorem 4**) is a very useful modeling device; if a modeler cannot make the assignment of a candidate type as Rigid or Non-Rigid, this indicates that he has not yet considered this aspect of the class he is introducing. It is this aspect that is vital for modeling a BFO-compliant domain ontology, because a Non-Rigid class does not belong in the **is_a** hierarchy (**Theorem 17**). The example given by

[3]Please see **Section 13.4**, which discuss "false positives", i.e., classes which satisfy **Instantiated**, **Members_Exist**, and **Rigid** but may not be types under BFO's theory.

the modeler and the other members of the class, at a time that they exist, are members of some other class that is Rigid (**Theorem 21**) which, under BFO's theory, better represents what they are. Further, there are additional entities that these members are in some relation with that should be represented (as explicated in **Section 6.3**). Specifically there is some class the Non-Rigid class is a subclass of that is Rigid (**Theorem 21**), which is one criterion for being a type. **Question 9** helps determine the better representation and the additional entities for the members of the introduced class.

The answer for **Question 9** helps determine if the the additional entity to represent is (a) a quality, function, or role, (b) a process (c) a part of something else under the same BFO types. Although there are not necessarily additional inferences based on these answers, type-level relationships are formalized on behalf of the modeler once they help identify what the type of the additional entity is.

Ultimately, the decision tree makes the above assertions implicitly and performs these inferences on behalf of the user, instead of having the user create these axioms directly and in their system. This is useful because generally these assertions are not considered within the scope of a domain level ontology.

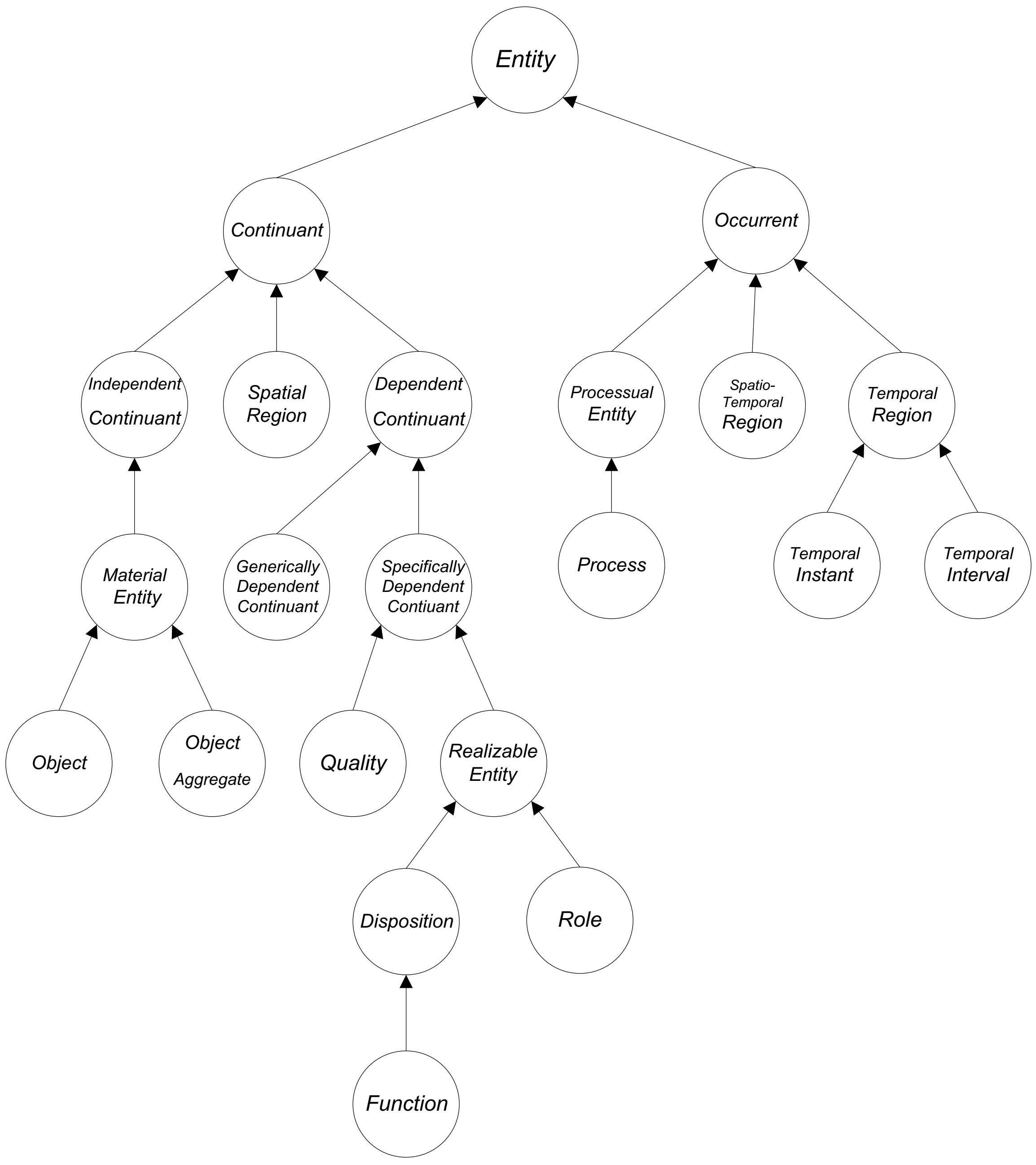
One of the major challenges of integrating BFO with OntoClean is the lack of clarity of the formal theories of both BFO and OntoClean. For instance, the document titled the BFO manual on the BFO website (http://www.ifomis.org/bfo) explicates all the types of BFO, but it does not include any axioms. There are many other papers by the originator of BFO, Barry Smith, that do include axioms, but it is not clear if these papers are considered a part of BFO or not. To address this challenge I communicated with Barry Smith directly, while at the same time have made the effort to to take into account only what is considered a part of BFO version 1.1's theory, and not what is considered BFO 2.0, which is yet to be released at the time of this writing.

On the OntoClean side, the formulations of Rigidity have changed significantly through the various papers published by Guarino and Welty, and in more recent publications (Guarino and Welty, 2004) the axioms have been left out altogether. This is an approach taken to address certain flaws that are difficult to deal with formally and at the same time an attempt to make the OntoClean theory more accessible to laypeople. With our work we were able to maintain that balance, in terms of informal and formal explication of our integrated theory, while also providing an implementation based on a decision tree that can serve as a training tool. As previously mentioned, the questions asked in the decision tree have a close mapping to the questions an experienced ontologist asks themselves when trying to create a BFO-compliant ontology.

In performing this work we have discovered that a reformulated notion of Rigidity can play a key role in domain modeling under BFO's upper ontology. Under BFO's theory, using classes as the only categorical unit is limiting, hence the introduction of types. All types are Rigid, and a candidate type being assigned by a modeler as Non-Rigid reveals that addition work is required by the modeler for their ontology to be BFO-compliant.

Appendix A

BFO Type immediate_is_a Hierarchy (Partial View)


Entity
Continuant
Occurrent
Independent Continuant
Spatial Region
Dependent Continuant
Processual Entity
Spatio-Temporal Region
Temporal Region
Material Entity
Generically Dependent Continuant
Specifically Dependent Contiuant
Process
Temporal Instant
Temporal Interval
Object
Object Aggregate
Quality
Realizable Entity
Disposition
Role
Function

Appendix B

Instance-Level Relations of Relation Ontology

Relation	arg1	arg2	arg3
depends_on	*Dependent Continuant*	*Independent Continuant*	*Temporal Region*
part_of	*Particular*	*Particular*	*Temporal Region*
participates_in	*Continuant*	*Occurrent*	*Temporal Region*
located_in	*Continuant*	*Spatial Region*	*Temporal Region*

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix C

Formulas

C.1 In Order of Appearance

Axiom 1. $\exists t(\textbf{exists\_at}(x,t))$ [p. 41]

Definition 1. $\textbf{subclass\_of}(A,B) =_{\text{def}} \forall xt(\textbf{member\_of}(x,A,t) \rightarrow \textbf{member\_of}(x,B,t))$ [p. 42]

Axiom 2. $A{=}B \leftrightarrow (\textbf{subclass\_of}(A,B) \wedge \textbf{subclass\_of}(B,A))$ [p. 42]

Theorem 1. $\textbf{subclass\_of}(A,A)$ [p. 43]

Theorem 2. $\textbf{subclass\_of}(A,B) \wedge \textbf{subclass\_of}(B,C) \rightarrow \textbf{subclass\_of}(A,C)$ [p. 43]

Axiom 3. $\neg\forall xt(\textbf{member\_of}(x,A,t) \rightarrow \textbf{exists\_at}(x,t))$ [p. 43]

Definition 2. $\textbf{Instantiated}(\mathrm{A}) =_{\text{def}} \exists xt(\textbf{member\_of}(x,A,t) \wedge \textbf{exists\_at}(x,t))$ [p. 44]

Definition 3. $\textbf{Empty}(A) =_{\text{def}} \neg\exists xt(\textbf{member\_of}(x,A,t))$ [p. 44]

Definition 4. $\textbf{Members\_Exist}(A) =_{\text{def}} \forall xt(\textbf{member\_of}(x,A,t) \rightarrow \textbf{exists\_at}(x,t))$ [p. 44]

Definition 5. $\mathbf{Rigid}(A) =_{\mathrm{def}} \forall x(\exists t(\mathbf{member\_of}(x,A,t)) \rightarrow \forall t_1(\mathbf{exists\_at}(x,t_1) \rightarrow \mathbf{member\_of}(x,A,t_1)))$ [p. 46]

Definition 6. $\mathbf{Non\text{-}Rigid}(A) =_{\mathrm{def}} \neg\mathbf{Rigid}(A)$ [p. 47]

Theorem 3. $\mathbf{Non\text{-}Rigid}(A) \leftrightarrow \exists x(\exists t(\mathbf{member\_of}(x,A,t)) \wedge \exists t_1(\mathbf{exists\_at}(x,t_1) \wedge \neg\mathbf{member\_of}(x,A,t_1)))$ [p. 47]

Theorem 4. $\mathbf{Rigid}(A) \oplus \mathbf{Non\text{-}Rigid}(A)$ [p. 48]

Theorem 5. $\exists AB(\mathbf{Rigid}(A) \wedge \mathbf{Non\text{-}Rigid}(B) \wedge \mathbf{subclass\_of}(A,B))$

Theorem 6. $\exists AB(\mathbf{Rigid}(A) \wedge \mathbf{Non\text{-}Rigid}(B) \wedge \mathbf{subclass\_of}(B,A))$

Axiom 4. $\mathbf{Type}(A) \rightarrow \mathbf{Instantiated}(A)$ [p. 52]

Definition 7. $\mathbf{Partial}(A) =_{\mathrm{def}} \mathbf{Instantiated}(A) \wedge \exists xt(\mathbf{member\_of}(x,A,t) \wedge \neg\mathbf{exists\_at}(x,t))$ [p. 52]

Axiom 5. $\mathbf{Type}(A) \rightarrow \mathbf{Members\_Exist}(A)$ [p. 52]

Theorem 7. $\mathbf{Type}(A) \rightarrow \neg\mathbf{Partial}(A)$ [p. 52]

Axiom 6. $\mathbf{Type}(A) \rightarrow \mathbf{Rigid}(A)$ [p. 53]

Theorem 8. $\mathbf{Type}(A) \rightarrow \neg\mathbf{Empty}(A)$ [p. 53]

Theorem 9. $\mathbf{Non\text{-}Rigid}(A) \rightarrow \neg\mathbf{Type}(A)$ [p. 59]

Definition 8. $\mathbf{instance\_of}(x,A,t) =_{\mathrm{def}} \mathbf{member\_of}(x,A,t) \wedge \mathbf{Type}(A)$ [p. 55]

Theorem 10. $\mathbf{Type}(\mathrm{A}) \wedge \mathbf{member\_of}(\mathrm{x},A,t) \rightarrow \mathbf{exists\_at}(x,t)$ [p. 55]

Theorem 11. $\mathbf{instance\_of}(x,A,t) \rightarrow \mathbf{exists\_at}(x,t)$ [p. 55]

Theorem 12. $\mathbf{Type}(A) \rightarrow \exists xt(\mathbf{instance\_of}(x,A,t))$ [p. 55]

Theorem 13. $\exists xt(\mathbf{member\_of}(x,A,t) \wedge \neg\mathbf{instance\_of}(x,A,t)) \rightarrow \neg\mathbf{Type}(A)$ [p. 56]

Axiom 7. $\mathbf{Type}(A) \wedge \mathbf{Type}(B) \rightarrow \forall t(\neg\mathbf{instance\_of}(A,B,t))$ [p. 106]

Definition 9. $\textbf{is\_a}(A,B) =_{\text{def}} \forall xt(\textbf{instance\_of}(x,A,t) \rightarrow \textbf{instance\_of}(x,B,t))$ [p. 57]

Theorem 14. $\textbf{is\_a}(A,B) \rightarrow \textbf{Type}(A) \wedge \textbf{Type}(B)$ [p. 57]

Axiom 8. $A{=}B \leftrightarrow (\textbf{is\_a}(A,B) \wedge \textbf{is\_a}(B,A))$ [p. 57]

Theorem 15. $\textbf{is\_a}(A,B) \rightarrow \textbf{subclass\_of}(A,B)$ [p. 57]

Theorem 16. $\textbf{is\_a}(A,B) \rightarrow \textbf{Rigid}(A) \wedge \textbf{Rigid}(B)$ [p. 58]

Theorem 17. $\textbf{Non-Rigid}(A) \rightarrow \forall B(\neg\textbf{is\_a}(A,B) \wedge \neg\textbf{is\_a}(B,A))$ [p. 58]

Axiom 9. $\textbf{Type}(A) \rightarrow \textbf{is\_a}(A,\mathit{Entity})$ [p. 58]

Theorem 18. $\textbf{is\_a}(A,\mathit{Entity}) \rightarrow \textbf{Type}(A)$ [p. 58]

Theorem 19. $\textbf{is\_a}(A,B) \rightarrow \textbf{is\_a}(A,\mathit{Entity}) \wedge \textbf{is\_a}(B,\mathit{Entity})$ [p. 59]

Axiom 10. $\textbf{exists\_at}(x,t) \rightarrow \exists A(\textbf{instance\_of}(x,A,t))$ [p. 59]

Theorem 20. $\exists A(\textbf{instance\_of}(x,A,t)) \rightarrow \textbf{instance\_of}(x,\mathit{Entity},t)$ [p. 59]

Theorem 21. $\textbf{member\_of}(x,A,t) \wedge \textbf{exists\_at}(x,t) \rightarrow \exists B(\textbf{member\_of}(x,B,t) \wedge \textbf{Rigid}(B))$ [p. 59]

Theorem 22. $(\textbf{Non-Rigid}(A) \wedge \textbf{Members\_Exist}(A)) \rightarrow \exists B(\textbf{subclass\_of}(A,B) \wedge \textbf{Rigid}(B))$ [p. 164]

Theorem 23. $\textbf{subclass\_of}(A,\mathit{Entity}) \rightarrow \textbf{Members\_Exist}(A)$ [p. 60]

Axiom 11. $\exists t(\textbf{instance\_of}(x,\mathit{Occurrent},t)) \rightarrow \forall t_1(\textbf{instance\_of}(x,\mathit{Occurrent},t_1))$ [p. 60]

Theorem 24. $\exists t(\textbf{instance\_of}(x,\mathit{Occurrent},t)) \rightarrow \forall t_1(\textbf{exists\_at}(x,t_1))$ [p. 60]

Theorem 25. $\textbf{is\_a}(A,\mathit{Occurrent}) \rightarrow \forall x(\exists t(\textbf{instance\_of}(x,A,t)) \rightarrow \forall t(\textbf{instance\_of}(x,A,t)))$ [p. 74]

Definition 10. $\textbf{immediate\_is\_a}(A,B) =_{\text{def}} \textbf{is\_a}(A,B) \wedge A{\neq}B \wedge \forall C(\textbf{is\_a}(A,C) \wedge \textbf{is\_a}(C,B) \rightarrow A{=}C \oplus C{=}B)$ [p. 61]

Theorem 26. $\mathbf{immediate\_is\_a}(A,B) \rightarrow \exists xt(\mathbf{instance\_of}(x,B,t) \wedge \neg\mathbf{instance\_of}(x,A,t))$ [p. 61]

Axiom 12. $\exists xt(\mathbf{instance\_of}(x,A,t) \wedge \mathbf{instance\_of}(x,B,t)) \rightarrow \mathbf{is\_a}(A,B) \vee \mathbf{is\_a}(B,A)$ [p. 62]

Theorem 27. $\mathbf{is\_a}(A,B) \wedge \mathbf{is\_a}(A,C) \rightarrow \mathbf{is\_a}(B,C) \vee \mathbf{is\_a}(C,B)$ [p. 62]

Theorem 28. $\exists xt(\mathbf{instance\_of}(x,A,t)) \wedge \mathbf{subclass\_of}(A,B) \wedge \mathbf{subclass\_of}(A,C) \rightarrow \neg\mathbf{is\_a}(B,C) \vee \neg\mathbf{is\_a}(C,B) \rightarrow \neg\mathbf{Type}(B) \vee \neg\mathbf{Type}(C)$ [p. 106]

Theorem 29. $\mathbf{immediate\_is\_a}(A,B) \rightarrow (\mathbf{immediate\_is\_a}(A,C) \leftrightarrow B{=}C)$ [p. 63]

Theorem 30. $(\mathbf{instance\_of}(x,A,t) \wedge \mathbf{instance\_of}(x,B,t) \wedge \mathbf{immediate\_is\_a}(A,C) \wedge \mathbf{immediate\_is\_a}(B,C)) \rightarrow A{=}B$ [p. 63]

Theorem 31. $(\exists A(\mathbf{is\_a}(A,B) \wedge \mathbf{is\_a}(A,C)) \wedge \exists D(\mathbf{immediate\_is\_a}(B,D) \wedge \mathbf{immediate\_is\_a}(C,D))) \rightarrow B{=}C$ [p. 64]

Definition 11. $\mathbf{disjoint\_from}_{\mathbf{class}}(A,B) =_{\text{def}} \forall xt(\mathbf{member\_of}(x,A,t) \rightarrow \neg\mathbf{member\_of}(x,B,t))$ [p. 64]

Definition 12. $\mathbf{disjoint\_from}_{\mathbf{type}}(A,B) =_{\text{def}} \forall xt(\mathbf{instance\_of}(x,A,t) \rightarrow \neg\mathbf{instance\_of}(x,B,t))$ [p. 64]

Theorem 32. $\mathbf{disjoint\_from}_{\mathbf{type}}(A,B) \rightarrow \mathbf{Type}(A) \wedge \mathbf{Type}(B)$ [p. 64]

Definition 13. $\mathbf{Heterogeneous}(A) =_{\text{def}} \exists xBCt(\mathbf{member\_of}(x,A,t) \wedge \mathbf{member\_of}(x,B,t) \wedge \mathbf{member\_of}(x,C,t) \wedge \mathbf{disjoint\_from}_{\mathbf{type}}(B,C))$ [p. 66]

Theorem 33. $\mathbf{Heterogeneous}(A) \rightarrow \mathbf{Instantiated}(A)$ [p. 66]

Theorem 34. $\mathbf{disjoint\_from}_{\mathbf{type}}(A,B) \rightarrow \mathbf{disjoint\_from}_{\mathbf{class}}(A,B)$ [p. 66]

Theorem 35. $\neg\forall A(\neg\mathbf{disjoint\_from}_{\mathbf{class}}(A,A))$ [p. 66]

Theorem 36. $\neg\forall A(\mathbf{disjoint\_from}_{\mathbf{class}}(A,A))$ [p. 67]

Theorem 37. $\forall A(\neg\mathbf{disjoint\_from}_{\mathbf{type}}(A,A))$ [p. 67]

Theorem 38. $\mathbf{disjoint\_from}_{\mathbf{class}}(A,B) \rightarrow \mathbf{disjoint\_from}_{\mathbf{class}}(B,A)$ [p. 67]

Theorem 39. $\mathbf{disjoint\_from}_{\mathbf{type}}(A,B) \rightarrow \mathbf{disjoint\_from}_{\mathbf{type}}(B,A)$ [p. 67]

Theorem 40. $\mathbf{subclass\_of}(A,B) \wedge \mathbf{subclass\_of}(A,C) \wedge$
$\mathbf{disjoint\_from}_{\mathbf{class}}(B,C) \rightarrow \mathbf{Empty}(A)$ [p. 68]

Theorem 41. $\mathbf{disjoint\_from}_{\mathbf{type}}(A,B) \wedge \mathbf{is\_a}(C,A) \rightarrow$
$\mathbf{disjoint\_from}_{\mathbf{type}}(C,B)$ [p. 68]

Theorem 42. $\mathbf{disjoint\_from}_{\mathbf{type}}(A,B) \wedge \mathbf{is\_a}(C,A) \wedge \mathbf{is\_a}(D,B) \rightarrow$
$\mathbf{disjoint\_from}_{\mathbf{type}}(C,D)$ [p. 68]

Theorem 43. $\mathbf{immediate\_is\_a}(A,C) \wedge \mathbf{immediate\_is\_a}(B,C) \wedge A\neq B \rightarrow$
$\mathbf{disjoint\_from}_{\mathbf{type}}(A,B)$ [p. 68]

Theorem 44. $(\mathbf{is\_a}(A,B) \vee \mathbf{is\_a}(B,A)) \oplus \mathbf{disjoint\_from}_{\mathbf{type}}(A,B)$ [p. 69]

Theorem 45. $\neg\forall ABC(\mathbf{disjoint\_from}_{\mathbf{type}}(A,B) \wedge$
$\mathbf{disjoint\_from}_{\mathbf{type}}(B,C) \rightarrow$
$\mathbf{disjoint\_from}_{\mathbf{type}}(A,C))$ [p. 70]

Theorem 46. $\neg\forall ABC(\mathbf{disjoint\_from}_{\mathbf{type}}(A,B) \wedge$
$\mathbf{disjoint\_from}_{\mathbf{type}}(B,C) \rightarrow$
$\neg\mathbf{disjoint\_from}_{\mathbf{type}}(A,C))$ [p. 70]

Axiom 13. $\mathbf{depends\_on}(x,y,t) \rightarrow$
$\mathbf{instance\_of}(x,\mathit{DependentContinuant},t) \wedge$
$\mathbf{instance\_of}(y,\mathit{IndependentContinuant},t)$ [p. 72]

Theorem 47. $\mathbf{depends\_on}(x,y,t) \rightarrow \mathbf{exists\_at}(x,t) \wedge \mathbf{exists\_at}(y,t)$ [p. 72]

Axiom 14. $\exists t(\textbf{depends\_on}(x,y,t)) \rightarrow \forall t_1(\textbf{exists\_at}(x,t_1) \rightarrow \textbf{exists\_at}(y,t_1))$ [p. 72]

Axiom 15. $\exists t t_1(\textbf{depends\_on}(x,y,t) \wedge \textbf{depends\_on}(x,z,t_1)) \rightarrow y{=}z$ [p. 73]

Axiom 16. $\textbf{generically\_depends\_on}(x,A,t) \rightarrow \textbf{instance\_of}(x,\mathit{GenericallyDependentContinuant},t) \wedge \textbf{is\_a}(A,\mathit{IndependentContinuant})$ [p. 73]

Axiom 17. $\textbf{is\_a}(A,\mathit{Occurrent}) \leftrightarrow \forall x(\exists t(\textbf{part\_of}(x,A,t)) \rightarrow \forall t(\textbf{part\_of}(x,A,t)))$ [p. 74]

Axiom 18. $\textbf{exists\_at}(x,t) \rightarrow \forall y(\textbf{part\_of}(y,x,t) \rightarrow \textbf{exists\_at}(y,t))$ [p. 75]

Axiom 19. $\textbf{located\_in}(x,y,t) \rightarrow \textbf{instance\_of}(x,\mathit{Continuant},t) \wedge \textbf{instance\_of}(y,\mathit{SpatialRegion},t)$ [p. 75]

Axiom 20. $\textbf{participates\_in}(x,y,t) \rightarrow \textbf{instance\_of}(x,\mathit{Continuant},t) \wedge \textbf{instance\_of}(y,\mathit{Process},t)$ [p. 75]

Definition 14. $\textbf{Depends\_On}(A,B) =_{\text{def}} \forall xt(\textbf{instance\_of}(x,A,t) \rightarrow \exists y(\textbf{instance\_of}(y,B,t) \wedge \textbf{depends\_on}(x,y,t)))$ [p. 76]

Definition 16. $\textbf{Has\_Part}(B,A) =_{\text{def}} \forall yt(\textbf{instance\_of}(y,B,t) \rightarrow \exists z(\textbf{instance\_of}(z,A,t) \wedge \textbf{part\_of}(z,y,t)))$ [p. 76]

Definition 17. $\textbf{exemplifies}(x,A,t) =_{\text{def}} \exists y(\textbf{depends\_on}(y,x,t) \wedge \textbf{instance\_of}(y,A,t)) \wedge \textbf{is\_a}(A,\mathit{SpecificallyDependentContinuant})$ [p. 79]

Definition Schema 1. $\mathbf{Unified\_Under}(A,\omega,\mathbf{p}) =_{\text{def}}$

$$\begin{aligned}&\forall x(\exists t(\mathbf{member\_of}(x,A,t)) \rightarrow\\&\quad\forall t(\mathbf{exists\_at}(x,t) \rightarrow\\&\qquad\forall y(\mathbf{p}(y,x,t) \rightarrow\\&\qquad\quad\forall z(\mathbf{p}(z,x,t) \leftrightarrow\\&\qquad\qquad\omega(z,y,t))))) \wedge\\&\forall wvt(\mathbf{p}(w,v,t) \rightarrow \mathbf{part\_of}(w,v,t)) \wedge\\&\neg\forall wvt(\mathbf{part\_of}(w,v,t) \rightarrow \mathbf{p}(w,v,t))\end{aligned}$$ [p. 87]

Metatheorem 1. $\mathbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\begin{aligned}&\forall x(\exists t(\mathbf{member\_of}(x,A,t)) \rightarrow\\&\quad\forall t(\mathbf{exists\_at}(x,t) \rightarrow\\&\qquad\forall y(\mathbf{p}(y,x,t) \rightarrow\\&\qquad\quad\forall zw(\omega(y,z,t) \wedge \omega(z,w,t) \rightarrow \omega(y,w,t)))))\end{aligned}$$ [p. 88]

Metatheorem 2. $\mathbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\begin{aligned}&\forall x(\exists t(\mathbf{member\_of}(x,A,t)) \rightarrow\\&\quad\forall t(\mathbf{exists\_at}(x,t) \rightarrow\\&\qquad\forall y(\mathbf{p}(y,x,t) \rightarrow\\&\qquad\quad\forall zw(\omega(y,z,t) \rightarrow \omega(z,y,t)))))\end{aligned}$$ [p. 88]

Metatheorem 3. $\mathbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\begin{aligned}&\forall x(\exists t(\mathbf{member\_of}(x,A,t)) \rightarrow\\&\quad\forall t(\mathbf{exists\_at}(x,t) \rightarrow\\&\qquad\forall y(\mathbf{p}(y,x,t) \rightarrow \omega(y,y,t))\end{aligned}$$ [p. 88]

Metatheorem 4. $\mathbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\begin{aligned}&\forall xyt(\mathbf{member\_of}(x,A,t) \wedge \mathbf{member\_of}(y,A,t) \rightarrow\\&\quad\mathbf{exists\_at}(x,t) \wedge \mathbf{exists\_at}(y,t) \rightarrow\\&\quad\exists z(\mathbf{p}(z,x,t) \wedge \mathbf{p}(z,y,t)) \rightarrow\\&\quad\forall w(\mathbf{p}(w,x,t) \leftrightarrow \mathbf{p}(w,y,t))))\end{aligned}$$ [p. 89]

Definition 18. $\mathbf{Necessary\text{-}IP}(A,P) =_{\text{def}} \forall xytt_1((\mathbf{member\_of}(x,A,t) \wedge$

$\textbf{exists\_at}(x,t) \land \textbf{member\_of}(y,A,t_1) \land \textbf{exists\_at}(y,t_1)) \rightarrow$
$(x{=}y \rightarrow \textbf{confirms}(P,x,y,t,t_1)))$ [p. 95]

Axiom 21. $(\textbf{Necessary-IP}(A,P) \land \exists xytt_1(\textbf{confirms}(P,x,y,t,t_1))) \rightarrow$
$\exists pwvt_2t_3(\textbf{member\_of}(w,A,t_2) \land \textbf{exists\_at}(w,t_2) \land$
$\textbf{member\_of}(v,A,t_3) \land \textbf{exists\_at}(v,t_3)) \land$
$\textbf{confirmed}(P,p,w,v,t_2,t_3))$ [p. 95]

Axiom 22. $\textbf{confirmed}(P,p,w,v,t_2,t_3) \rightarrow \exists t(\textbf{instance\_of}(p,P,t))$ [p. 96]

Axiom 23. $\textbf{confirmed}(P,p,w,v,t_2,t_3) \rightarrow$
$\exists p_1p_2(\textbf{matches}(\textbf{result\_of\_procedure}(p_1,w,t_2), \textbf{result\_of\_procedure}(p_2,v,t_3)) \land$
$\textbf{part\_of}(p_1,p) \land \textbf{part\_of}(p_2,p))$ [p. 96]

Definition 19. $\textbf{Sufficient-IP}(A,P) =_{\text{def}} \forall xytt_1((\textbf{member\_of}(x,A,t) \land$
$\textbf{exists\_at}(x,t) \land \textbf{member\_of}(y,A,t_1) \land \textbf{exists\_at}(y,t_1)) \rightarrow$
[p. 98] $(\textbf{confirms}(P,x,y,t,t_1) \rightarrow x{=}y))$

Definition 20. $\textbf{IP}(A,P) =_{\text{def}} \textbf{Necessary-IP}(A,P) \lor \textbf{Sufficient-IP}(A,P)$ [p. 99]

Definition 21. $\textbf{N\&S-IP}(A,P) =_{\text{def}} \textbf{Necessary-IP}(A,P) \land \textbf{Sufficient-IP}(A,P)$ [p. 99]

Axiom 24. $\textbf{IP}(A,P) \rightarrow \exists B(\textbf{Rigid}(B) \land \textbf{IP}(B,P) \land$
$\textbf{subclass\_of}(A,B))$ [p. 100]

Theorem 48. $(\textbf{Non-Rigid}(A) \land \textbf{IP}(A,P)) \rightarrow \exists B(\textbf{Rigid}(B) \land \textbf{IP}(B,P) \land$
$A{\neq}B \land \textbf{subclass\_of}(A,B))$ [p. 100]

Definition 22. $\textbf{supplies-IP}(A,P) =_{\text{def}} \textbf{IP}(A,P) \land \textbf{Rigid}(A) \land$
$(\forall B(\textbf{IP}(B,P) \rightarrow \textbf{subclass\_of}(B,A))$ [p. 100]

Theorem 49. $\textbf{supplies-IP}(A,P) \land \textbf{IP}(B,P) \land \textbf{subclass\_of}(A,B) \rightarrow A{=}B$ [p. 100]

Axiom 25. $\textbf{IP}(A,P) \land \neg\textbf{supplies-IP}(A,P) \rightarrow$
$\exists B(\mathrm{A}{\neq}B \land \textbf{subclass\_of}(A,B) \land \textbf{supplies-IP}(B,P))$ [p. 101]

Theorem 50. $\textbf{IP}(\mathit{Entity},P) \rightarrow \textbf{supplies-IP}(\mathit{Entity},P)$ [p. 101]

Theorem 51. $\exists A(\mathbf{IP}(A,P)) \rightarrow \exists B(\mathbf{supplies\text{-}IP}(B,P))$ [p. 101]

Theorem 52. $(\mathbf{Non\text{-}Rigid}(A) \wedge \mathbf{IP}(A,P)) \rightarrow$
$\exists B(\mathbf{supplies\text{-}IP}(B,P) \wedge \mathbf{subclass\_of}(A,B))$ [p. 101]

Theorem 53. $(\mathbf{Necessary\text{-}IP}(A,P) \wedge \mathbf{subclass\_of}(B,A)) \rightarrow$
$\mathbf{Necessary\text{-}IP}(B,P)$ [p. 101]

Theorem 54. $\mathbf{Sufficient\text{-}IP}(A,P) \wedge \mathbf{subclass\_of}(B,A) \rightarrow$
$\mathbf{Sufficient\text{-}IP}(B,P)$ [p. 102]

Theorem 55. $\mathbf{Sufficient\text{-}IP}(A,P) \wedge \mathbf{is\_a}(B,A) \rightarrow \mathbf{Sufficient\text{-}IP}(B,P)$ [p. 102]

Theorem 56. $\mathbf{Necessary\text{-}IP}(A,P) \wedge \mathbf{is\_a}(B,A) \rightarrow \mathbf{Necessary\text{-}IP}(B,P)$ [p. 102]

Definition 23. $\mathbf{Compatible\text{-}IP}(P,Q) =_{\mathrm{def}} \exists A(\mathbf{IP}(A,P) \wedge \mathbf{IP}(A,Q))$ [p. 103]

Definition 24. $\mathbf{Incompatible\text{-}IP}(P,Q) =_{\mathrm{def}} \neg\mathbf{Compatible\text{-}IP}(P,Q)$ [p. 103]

Theorem 57. $\mathbf{Incompatible\text{-}IP}(P,Q) \leftrightarrow \forall A\neg(\mathbf{IP}(A,P) \wedge \mathbf{IP}(A,Q))$ [p. 103]

Theorem 58. $(\mathbf{Necessary\text{-}IP}(A,P) \wedge \mathbf{Necessary\text{-}IP}(B,Q) \wedge$
$\mathbf{Incompatible\text{-}IP}(P,Q)) \rightarrow$
$(\mathbf{Type}(A) \wedge \mathbf{Type}(B)) \rightarrow$
$\mathbf{disjoint\_from}_{\mathbf{type}}(A,B)$ [p. 106]

Axiom 26. $\mathbf{instance\_of}(x,\mathit{ObjectAggregate},t) \rightarrow$
$\exists yz(\mathbf{part\_of}(y,x,t) \wedge \mathbf{part\_of}(z,x,t) \wedge$
$\mathbf{instance\_of}(y,\mathit{Object},t) \wedge$
$\mathbf{instance\_of}(z,\mathit{Object},t) \wedge y{\neq}z)$ [p. 106]

Axiom 27. $\mathbf{member\_of\_aggregate}(x,y,t) \rightarrow \mathbf{part\_of}(x,y,t)$ [p. 108]

Axiom 28. $\neg\forall xyt(\mathbf{part\_of}(x,y,t) \rightarrow \mathbf{member\_of\_aggregate}(x,y,t))$ [p. 108]

Axiom 29. $\mathbf{N\&S\text{-}IP}(A,\ \mathit{ME}) \leftrightarrow$
$\forall xyt((\mathbf{member\_of}(x,A,t) \wedge$
$\mathbf{member\_of}(y,A,t)) \rightarrow$
$(x{=}y \leftrightarrow \forall z(\mathbf{member\_of\_aggregate}(z,x,t) \leftrightarrow$
$\mathbf{member\_of\_aggregate}(z,y,t))))$ [p. 108]

C.2 In Order of Type

C.2.1 Definitions

Definition 1. $\mathbf{subclass\_of}(A,B) =_{\text{def}} \forall xt(\mathbf{member\_of}(x,A,t) \rightarrow \mathbf{member\_of}(x,B,t))$ [p. 42]

Definition 2. $\mathbf{Instantiated}(\mathrm{A}) =_{\text{def}} \exists xt(\mathbf{member\_of}(x,A,t) \wedge \mathbf{exists\_at}(x,t))$ [p. 44]

Definition 3. $\mathbf{Empty}(A) =_{\text{def}} \neg\exists xt(\mathbf{member\_of}(x,A,t))$ [p. 44]

Definition 4. $\mathbf{Members\_Exist}(A) =_{\text{def}} \forall xt(\mathbf{member\_of}(x,A,t) \rightarrow \mathbf{exists\_at}(x,t))$ [p. 44]

Definition 5. $\mathbf{Rigid}(A) =_{\text{def}} \forall x(\exists t(\mathbf{member\_of}(x,A,t)) \rightarrow \forall t_1(\mathbf{exists\_at}(x,t_1) \rightarrow \mathbf{member\_of}(x,A,t_1)))$ [p. 46]

Definition 6. $\mathbf{Non\text{-}Rigid}(A) =_{\text{def}} \neg\mathbf{Rigid}(A)$ [p. 47]

Definition 7. $\mathbf{Partial}(A) =_{\text{def}} \mathbf{Instantiated}(A) \wedge \exists xt(\mathbf{member\_of}(x,A,t) \wedge \neg\mathbf{exists\_at}(x,t))$ [p. 52]

Definition 8. $\mathbf{instance\_of}(x,A,t) =_{\text{def}} \mathbf{member\_of}(x,A,t) \wedge \mathbf{Type}(A)$ [p. 55]

Definition 9. $\mathbf{is\_a}(A,B) =_{\text{def}} \forall xt(\mathbf{instance\_of}(x,A,t) \rightarrow \mathbf{instance\_of}(x,B,t))$ [p. 57]

Definition 10. $\mathbf{immediate\_is\_a}(A,B) =_{\text{def}} \mathbf{is\_a}(A,B) \wedge A{\neq}B \wedge \forall C(\mathbf{is\_a}(A,C) \wedge \mathbf{is\_a}(C,B) \rightarrow A{=}C \oplus C{=}B)$ [p. 61]

Definition 11. $\mathbf{disjoint\_from_{class}}(A,B) =_{\text{def}} \forall xt(\mathbf{member\_of}(x,A,t) \rightarrow \neg\mathbf{member\_of}(x,B,t))$ [p. 64]

Definition 12. $\mathbf{disjoint\_from_{type}}(A,B) =_{\text{def}} \forall xt(\mathbf{instance\_of}(x,A,t) \rightarrow \neg\mathbf{instance\_of}(x,B,t))$ [p. 64]

Definition 13. $\mathbf{Heterogeneous}(A) =_{\text{def}} \exists xBCt(\mathbf{member\_of}(x,A,t) \wedge \mathbf{member\_of}(x,B,t) \wedge \mathbf{member\_of}(x,C,t) \wedge \mathbf{disjoint\_from_{type}}(B,C))$ [p. 66]

Definition 14. $\textbf{Depends\_On}(A,B) =_{\text{def}}$
$\forall xt(\textbf{instance\_of}(x,A,t) \rightarrow$
$\exists y(\textbf{instance\_of}(y,B,t) \wedge$
$\textbf{depends\_on}(x,y,t)))$ [p. 76]

Definition 16. $\textbf{Has\_Part}(B,A) =_{\text{def}}$
$\forall yt(\textbf{instance\_of}(y,B,t) \rightarrow$
$\exists z(\textbf{instance\_of}(z,A,t) \wedge \textbf{part\_of}(z,y,t)))$ [p. 76]

Definition 17. $\textbf{exemplifies}(x,A,t) =_{\text{def}}$
$\exists y(\textbf{depends\_on}(y,x,t) \wedge$
$\textbf{instance\_of}(y,A,t)) \wedge$
$\textbf{is\_a}(A,\mathit{SpecificallyDependentContinuant})$ [p. 79]

Definition 18. $\textbf{Necessary-IP}(A,P) =_{\text{def}} \forall xytt_1((\textbf{member\_of}(x,A,t) \wedge$
$\textbf{exists\_at}(x,t) \wedge \textbf{member\_of}(y,A,t_1) \wedge \textbf{exists\_at}(y,t_1)) \rightarrow$
$(x{=}y \rightarrow \textbf{confirms}(P,x,y,t,t_1)))$ [p. 95]

Definition 19. $\textbf{Sufficient-IP}(A,P) =_{\text{def}} \forall xytt_1((\textbf{member\_of}(x,A,t) \wedge$
$\textbf{exists\_at}(x,t) \wedge \textbf{member\_of}(y,A,t_1) \wedge \textbf{exists\_at}(y,t_1)) \rightarrow$
[p. 98] $(\textbf{confirms}(P,x,y,t,t_1) \rightarrow x{=}y))$

Definition 20. $\textbf{IP}(A,P) =_{\text{def}} \textbf{Necessary-IP}(A,P) \vee \textbf{Sufficient-IP}(A,P)$ [p. 99]

Definition 21. $\textbf{N\&S-IP}(A,P) =_{\text{def}} \textbf{Necessary-IP}(A,P) \wedge \textbf{Sufficient-IP}(A,P)$ [p. 99]

Definition 22. $\textbf{supplies-IP}(A,P) =_{\text{def}} \textbf{IP}(A,P) \wedge \textbf{Rigid}(A) \wedge$
$(\forall B(\textbf{IP}(B,P) \rightarrow \textbf{subclass\_of}(B,A))$ [p. 100]

Definition 23. $\textbf{Compatible-IP}(P,Q) =_{\text{def}} \exists A(\textbf{IP}(A,P) \wedge \textbf{IP}(A,Q))$ [p. 103]

Definition 24. $\textbf{Incompatible-IP}(P,Q) =_{\text{def}} \neg\textbf{Compatible-IP}(P,Q)$ [p. 103]

C.2.2 Definition Schema

Definition Schema 1. $\mathbf{Unified\_Under}(A,\omega,\mathbf{p}) =_{\mathrm{def}}$

$$\begin{aligned}
&\forall x(\exists t(\mathbf{member\_of}(x,A,t)) \rightarrow \\
&\quad \forall t(\mathbf{exists\_at}(x,t) \rightarrow \\
&\quad\quad \forall y(\mathbf{p}(y,x,t) \rightarrow \\
&\quad\quad\quad \forall z(\mathbf{p}(z,x,t) \leftrightarrow \\
&\quad\quad\quad\quad \omega(z,y,t))))) \wedge \\
&\forall wvt(\mathbf{p}(w,v,t) \rightarrow \mathbf{part\_of}(w,v,t)) \wedge \\
&\neg\forall wvt(\mathbf{part\_of}(w,v,t) \rightarrow \mathbf{p}(w,v,t))
\end{aligned}$$

[p. 87]

C.2.3 Axioms

Axiom 1. $\exists t(\textbf{exists\_at}(x,t))$ [p. 41]

Axiom 2. $A{=}B \leftrightarrow (\textbf{subclass\_of}(A,B) \wedge \textbf{subclass\_of}(B,A))$ [p. 42]

Axiom 3. $\neg\forall xt(\textbf{member\_of}(x,A,t) \rightarrow \textbf{exists\_at}(x,t))$ [p. 43]

Axiom 4. $\textbf{Type}(A) \rightarrow \textbf{Instantiated}(A)$ [p. 52]

Axiom 5. $\textbf{Type}(A) \rightarrow \textbf{Members\_Exist}(A)$ [p. 52]

Axiom 6. $\textbf{Type}(A) \rightarrow \textbf{Rigid}(A)$ [p. 53]

Axiom 7. $\textbf{Type}(A) \wedge \textbf{Type}(B) \rightarrow \forall t(\neg\textbf{instance\_of}(A,B,t))$ [p. 106]

Axiom 8. $A{=}B \leftrightarrow (\textbf{is\_a}(A,B) \wedge \textbf{is\_a}(B,A))$ [p. 57]

Axiom 9. $\textbf{Type}(A) \rightarrow \textbf{is\_a}(A,\mathit{Entity})$ [p. 58]

Axiom 10. $\textbf{exists\_at}(x,t) \rightarrow \exists A(\textbf{instance\_of}(x,A,t))$ [p. 59]

Axiom 11. $\exists t(\textbf{instance\_of}(x,\mathit{Occurrent},t)) \rightarrow$
$\forall t_1(\textbf{instance\_of}(x,\mathit{Occurrent},t_1))$ [p. 60]

Axiom 12. $\exists xt(\textbf{instance\_of}(x,A,t) \wedge \textbf{instance\_of}(x,B,t)) \rightarrow$
$\textbf{is\_a}(A,B) \vee \textbf{is\_a}(B,A)$ [p. 62]

Axiom 13. $\textbf{depends\_on}(x,y,t) \rightarrow$
$\textbf{instance\_of}(x,\mathit{DependentContinuant},t) \wedge$
$\textbf{instance\_of}(y,\mathit{IndependentContinuant},t)$ [p. 72]

Axiom 14. $\exists t(\textbf{depends\_on}(x,y,t)) \rightarrow \forall t_1(\textbf{exists\_at}(x,t_1) \rightarrow$
$\textbf{exists\_at}(y,t_1))$ [p. 72]

Axiom 15. $\exists tt_1(\textbf{depends\_on}(x,y,t) \wedge \textbf{depends\_on}(x,z,t_1)) \rightarrow y{=}z$ [p. 73]

Axiom 16. $\textbf{generically\_depends\_on}(x,A,t) \rightarrow$
$\textbf{instance\_of}(x,\mathit{GenericallyDependentContinuant},t) \wedge$
$\textbf{is\_a}(A,\mathit{IndependentContinuant})$ [p. 73]

Axiom 17. $\textbf{is\_a}(A,\mathit{Occurrent}) \leftrightarrow \forall x(\exists t(\textbf{part\_of}(x,A,t)) \rightarrow \forall t(\textbf{part\_of}(x,A,t)))$ [p. 74]

Axiom 18. $\textbf{exists\_at}(x,t) \rightarrow \forall y(\textbf{part\_of}(y,x,t) \rightarrow \textbf{exists\_at}(y,t))$ [p. 75]

Axiom 19. $\textbf{located\_in}(x,y,t) \rightarrow \textbf{instance\_of}(x,\mathit{Continuant},t) \wedge \textbf{instance\_of}(y,\mathit{SpatialRegion},t)$ [p. 75]

Axiom 20. $\textbf{participates\_in}(x,y,t) \rightarrow \textbf{instance\_of}(x,\mathit{Continuant},t) \wedge \textbf{instance\_of}(y,\mathit{Process},t)$ [p. 75]

Axiom 21. $(\textbf{Necessary-IP}(A,P) \wedge \exists xytt_1(\textbf{confirms}(P,x,y,t,t_1))) \rightarrow \exists pwvt_2t_3(\textbf{member\_of}(w,A,t_2) \wedge \textbf{exists\_at}(w,t_2) \wedge \textbf{member\_of}(v,A,t_3) \wedge \textbf{exists\_at}(v,t_3)) \wedge \textbf{confirmed}(P,p,w,v,t_2,t_3))$ [p. 95]

Axiom 22. $\textbf{confirmed}(P,p,w,v,t_2,t_3) \rightarrow \exists t(\textbf{instance\_of}(p,P,t))$ [p. 96]

Axiom 23. $\textbf{confirmed}(P,p,w,v,t_2,t_3) \rightarrow \exists p_1p_2(\textbf{matches}(\textbf{result\_of\_procedure}(p_1,w,t_2), \textbf{result\_of\_procedure}(p_2,v,t_3)) \wedge \textbf{part\_of}(p_1,p) \wedge \textbf{part\_of}(p_2,p))$ [p. 96]

Axiom 24. $\textbf{IP}(A,P) \rightarrow \exists B(\textbf{Rigid}(B) \wedge \textbf{IP}(B,P) \wedge \textbf{subclass\_of}(A,B))$ [p. 100]

Axiom 25. $\textbf{IP}(A,P) \wedge \neg\textbf{supplies-IP}(A,P) \rightarrow \exists B(\mathrm{A}{\neq}B \wedge \textbf{subclass\_of}(A,B) \wedge \textbf{supplies-IP}(B,P))$ [p. 101]

Axiom 26. $\textbf{instance\_of}(x,\mathit{ObjectAggregate},t) \rightarrow \exists yz(\textbf{part\_of}(y,x,t) \wedge \textbf{part\_of}(z,x,t) \wedge \textbf{instance\_of}(y,\mathit{Object},t) \wedge \textbf{instance\_of}(z,\mathit{Object},t) \wedge y{\neq}z)$ [p. 106]

Axiom 27. $\textbf{member\_of\_aggregate}(x,y,t) \rightarrow \textbf{part\_of}(x,y,t)$ [p. 108]

Axiom 28. $\neg\forall xyt(\mathbf{part\_of}(x{,}y{,}t) \rightarrow \mathbf{member\_of\_aggregate}(x{,}y{,}t))$ [p. 108]

Axiom 29.
$$\begin{aligned}&\mathbf{N\&S\text{-}IP}(A,\ \mathit{ME}) \leftrightarrow\\ &\quad\forall xyt((\mathbf{member\_of}(x{,}A{,}t) \wedge\\ &\qquad\mathbf{member\_of}(y{,}A{,}t)) \rightarrow\\ &\qquad(x{=}y \leftrightarrow \forall z(\mathbf{member\_of\_aggregate}(z{,}x{,}t) \leftrightarrow\\ &\qquad\qquad\mathbf{member\_of\_aggregate}(z{,}y{,}t))))\end{aligned}$$
[p. 108]

C.2.4 Theorems

Theorem 1. $\textbf{subclass\_of}(A,A)$ [p. 43]

Theorem 2. $\textbf{subclass\_of}(A,B) \wedge \textbf{subclass\_of}(B,C) \rightarrow \textbf{subclass\_of}(A,C)$ [p. 43]

Theorem 3. $\textbf{Non-Rigid}(A) \leftrightarrow \exists x(\exists t(\textbf{member\_of}(x,A,t)) \wedge \exists t_1(\textbf{exists\_at}(x,t_1) \wedge \neg\textbf{member\_of}(x,A,t_1)))$ [p. 47]

Theorem 4. $\textbf{Rigid}(A) \oplus \textbf{Non-Rigid}(A)$ [p. 48]

Theorem 5. $\exists AB(\textbf{Rigid}(A) \wedge \textbf{Non-Rigid}(B) \wedge \textbf{subclass\_of}(A,B))$

Theorem 6. $\exists AB(\textbf{Rigid}(A) \wedge \textbf{Non-Rigid}(B) \wedge \textbf{subclass\_of}(B,A))$

Theorem 7. $\textbf{Type}(A) \rightarrow \neg\textbf{Partial}(A)$ [p. 52]

Theorem 8. $\textbf{Type}(A) \rightarrow \neg\textbf{Empty}(A)$ [p. 53]

Theorem 9. $\textbf{Non-Rigid}(A) \rightarrow \neg\textbf{Type}(A)$ [p. 59]

Theorem 10. $\textbf{Type}(\mathrm{A}) \wedge \textbf{member\_of}(\mathrm{x},A,t) \rightarrow \textbf{exists\_at}(x,t)$ [p. 55]

Theorem 11. $\textbf{instance\_of}(x,A,t) \rightarrow \textbf{exists\_at}(x,t)$ [p. 55]

Theorem 12. $\textbf{Type}(A) \rightarrow \exists xt(\textbf{instance\_of}(x,A,t))$ [p. 55]

Theorem 13. $\exists xt(\textbf{member\_of}(x,A,t) \wedge \neg\textbf{instance\_of}(x,A,t)) \rightarrow \neg\textbf{Type}(A)$ [p. 56]

Theorem 14. $\textbf{is\_a}(A,B) \rightarrow \textbf{Type}(A) \wedge \textbf{Type}(B)$ [p. 57]

Theorem 15. $\textbf{is\_a}(A,B) \rightarrow \textbf{subclass\_of}(A,B)$ [p. 57]

Theorem 16. $\textbf{is\_a}(A,B) \rightarrow \textbf{Rigid}(A) \wedge \textbf{Rigid}(B)$ [p. 58]

Theorem 17. $\textbf{Non-Rigid}(A) \rightarrow \forall B(\neg\textbf{is\_a}(A,B) \wedge \neg\textbf{is\_a}(B,A))$ [p. 58]

Theorem 18. $\textbf{is\_a}(A,\mathit{Entity}) \rightarrow \textbf{Type}(A)$ [p. 58]

Theorem 19. $\textbf{is\_a}(A,B) \rightarrow \textbf{is\_a}(A,\mathit{Entity}) \wedge \textbf{is\_a}(B,\mathit{Entity})$ [p. 59]

Theorem 20. $\exists A(\textbf{instance\_of}(x,A,t)) \rightarrow \textbf{instance\_of}(x,\mathit{Entity},t)$ [p. 59]

Theorem 21. $\textbf{member\_of}(x,A,t) \wedge \textbf{exists\_at}(x,t) \rightarrow$
$\exists B(\textbf{member\_of}(x,B,t) \wedge \textbf{Rigid}(B))$ [p. 59]

Theorem 22. $(\textbf{Non-Rigid}(A) \wedge \textbf{Members\_Exist}(A)) \rightarrow$
$\exists B(\textbf{subclass\_of}(A,B) \wedge \textbf{Rigid}(B))$ [p. 164]

Theorem 23. $\textbf{subclass\_of}(A,\mathit{Entity}) \rightarrow \textbf{Members\_Exist}(A)$ [p. 60]

Theorem 24. $\exists t(\textbf{instance\_of}(x,\mathit{Occurrent},t)) \rightarrow \forall t_1(\textbf{exists\_at}(x,t_1))$ [p. 60]

Theorem 25. $\textbf{is\_a}(A,\mathit{Occurrent}) \rightarrow \forall x(\exists t(\textbf{instance\_of}(x,A,t)) \rightarrow$
$\forall t(\textbf{instance\_of}(x,A,t)))$ [p. 74]

Theorem 26. $\textbf{immediate\_is\_a}(A,B) \rightarrow \exists xt(\textbf{instance\_of}(x,B,t) \wedge$
$\neg\textbf{instance\_of}(x,A,t))$ [p. 61]

Theorem 27. $\textbf{is\_a}(A,B) \wedge \textbf{is\_a}(A,C) \rightarrow \textbf{is\_a}(B,C) \vee \textbf{is\_a}(C,B)$ [p. 62]

Theorem 28. $\exists xt(\textbf{instance\_of}(x,A,t)) \wedge$
$\textbf{subclass\_of}(A,B) \wedge \textbf{subclass\_of}(A,C) \rightarrow$
$\neg\textbf{is\_a}(B,C) \vee \neg\textbf{is\_a}(C,B) \rightarrow$
$\neg\textbf{Type}(B) \vee \neg\textbf{Type}(C)$ [p. 106]

Theorem 29. $\textbf{immediate\_is\_a}(A,B) \rightarrow (\textbf{immediate\_is\_a}(A,C) \leftrightarrow$
$B{=}C)$ [p. 63]

Theorem 30. $(\textbf{instance\_of}(x,A,t) \wedge \textbf{instance\_of}(x,B,t) \wedge$
$\textbf{immediate\_is\_a}(A,C) \wedge \textbf{immediate\_is\_a}(B,C)) \rightarrow$
$A{=}B$ [p. 63]

Theorem 31. $(\exists A(\textbf{is\_a}(A,B) \wedge \textbf{is\_a}(A,C)) \wedge$
$\exists D(\textbf{immediate\_is\_a}(B,D) \wedge$
$\textbf{immediate\_is\_a}(C,D))) \rightarrow B{=}C$ [p. 64]

Theorem 32. $\textbf{disjoint\_from}_{\textbf{type}}(A,B) \rightarrow \textbf{Type}(A) \wedge \textbf{Type}(B)$ [p. 64]

Theorem 33. $\textbf{Heterogeneous}(A) \rightarrow \textbf{Instantiated}(A)$ [p. 66]

Theorem 34. $\textbf{disjoint\_from}_{\textbf{type}}(A,B) \rightarrow \textbf{disjoint\_from}_{\textbf{class}}(A,B)$ [p. 66]

Theorem 35. $\neg\forall A(\neg\textbf{disjoint\_from}_{\textbf{class}}(A,A))$ [p. 66]

Theorem 36. $\neg\forall A(\textbf{disjoint\_from}_{\textbf{class}}(A,A))$ [p. 67]

Theorem 37. $\forall A(\neg\textbf{disjoint\_from}_{\textbf{type}}(A,A))$ [p. 67]

Theorem 38. $\textbf{disjoint\_from}_{\textbf{class}}(A,B) \rightarrow \textbf{disjoint\_from}_{\textbf{class}}(B,A)$ [p. 67]

Theorem 39. $\textbf{disjoint\_from}_{\textbf{type}}(A,B) \rightarrow \textbf{disjoint\_from}_{\textbf{type}}(B,A)$ [p. 67]

Theorem 40. $\textbf{subclass\_of}(A,B) \wedge \textbf{subclass\_of}(A,C) \wedge \textbf{disjoint\_from}_{\textbf{class}}(B,C) \rightarrow \textbf{Empty}(A)$ [p. 68]

Theorem 41. $\textbf{disjoint\_from}_{\textbf{type}}(A,B) \wedge \textbf{is\_a}(C,A) \rightarrow \textbf{disjoint\_from}_{\textbf{type}}(C,B)$ [p. 68]

Theorem 42. $\textbf{disjoint\_from}_{\textbf{type}}(A,B) \wedge \textbf{is\_a}(C,A) \wedge \textbf{is\_a}(D,B) \rightarrow \textbf{disjoint\_from}_{\textbf{type}}(C,D)$ [p. 68]

Theorem 43. $\textbf{immediate\_is\_a}(A,C) \wedge \textbf{immediate\_is\_a}(B,C) \wedge A{\neq}B \rightarrow \textbf{disjoint\_from}_{\textbf{type}}(A,B)$ [p. 68]

Theorem 44. $(\textbf{is\_a}(A,B) \vee \textbf{is\_a}(B,A)) \oplus \textbf{disjoint\_from}_{\textbf{type}}(A,B)$ [p. 69]

Theorem 45. $\neg\forall ABC(\textbf{disjoint\_from}_{\textbf{type}}(A,B) \wedge \textbf{disjoint\_from}_{\textbf{type}}(B,C) \rightarrow \textbf{disjoint\_from}_{\textbf{type}}(A,C))$ [p. 70]

Theorem 46. $\neg\forall ABC(\textbf{disjoint\_from}_{\textbf{type}}(A,B) \wedge \textbf{disjoint\_from}_{\textbf{type}}(B,C) \rightarrow \neg\textbf{disjoint\_from}_{\textbf{type}}(A,C))$ [p. 70]

Theorem 47. $\textbf{depends\_on}(x,y,t) \rightarrow \textbf{exists\_at}(x,t) \wedge \textbf{exists\_at}(y,t)$ [p. 72]

Theorem 48. $(\textbf{Non-Rigid}(A) \wedge \textbf{IP}(A,P)) \rightarrow \exists B(\textbf{Rigid}(B) \wedge \textbf{IP}(B,P) \wedge A{\neq}B \wedge \textbf{subclass\_of}(A,B))$ [p. 100]

Theorem 49. $\textbf{supplies-IP}(A{,}P) \wedge \textbf{IP}(B{,}P) \wedge \textbf{subclass\_of}(A{,}B) \rightarrow A{=}B$ [p. 100]

Theorem 50. $\textbf{IP}(\mathit{Entity}{,}P) \rightarrow \textbf{supplies-IP}(\mathit{Entity}{,}P)$ [p. 101]

Theorem 51. $\exists A(\textbf{IP}(A{,}P)) \rightarrow \exists B(\textbf{supplies-IP}(B{,}P))$ [p. 101]

Theorem 52. $(\textbf{Non-Rigid}(A) \wedge \textbf{IP}(A{,}P)) \rightarrow$
$\exists B(\textbf{supplies-IP}(B{,}P) \wedge \textbf{subclass\_of}(A{,}B))$ [p. 101]

Theorem 53. $(\textbf{Necessary-IP}(A{,}P) \wedge \textbf{subclass\_of}(B{,}A)) \rightarrow$
$\textbf{Necessary-IP}(B{,}P)$ [p. 101]

Theorem 54. $\textbf{Sufficient-IP}(A{,}P) \wedge \textbf{subclass\_of}(B{,}A) \rightarrow$
$\textbf{Sufficient-IP}(B{,}P)$ [p. 102]

Theorem 55. $\textbf{Sufficient-IP}(A{,}P) \wedge \textbf{is\_a}(B{,}A) \rightarrow \textbf{Sufficient-IP}(B{,}P)$ [p. 102]

Theorem 56. $\textbf{Necessary-IP}(A{,}P) \wedge \textbf{is\_a}(B{,}A) \rightarrow \textbf{Necessary-IP}(B{,}P)$ [p. 102]

Theorem 57. $\textbf{Incompatible-IP}(P{,}Q) \leftrightarrow \forall A \neg(\textbf{IP}(A{,}P) \wedge \textbf{IP}(A{,}Q))$ [p. 103]

Theorem 58. $(\textbf{Necessary-IP}(A{,}P) \wedge \textbf{Necessary-IP}(B{,}Q) \wedge$
$\textbf{Incompatible-IP}(P{,}Q)) \rightarrow$
$(\textbf{Type}(A) \wedge \textbf{Type}(B)) \rightarrow$
$\textbf{disjoint\_from}_{\textbf{type}}(A{,}B)$ [p. 106]

C.2.5 Metatheorems

Metatheorem 1. $\textbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\forall x(\exists t(\textbf{member\_of}(x,A,t)) \rightarrow$$
$$\forall t(\textbf{exists\_at}(x,t) \rightarrow$$
$$\forall y(\mathbf{p}(y,x,t) \rightarrow$$
$$\forall zw(\omega(y,z,t) \wedge \omega(z,w,t) \rightarrow \omega(y,w,t)))))$$ [p. 88]

Metatheorem 2. $\textbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\forall x(\exists t(\textbf{member\_of}(x,A,t)) \rightarrow$$
$$\forall t(\textbf{exists\_at}(x,t) \rightarrow$$
$$\forall y(\mathbf{p}(y,x,t) \rightarrow$$
$$\forall zw(\omega(y,z,t) \rightarrow \omega(z,y,t)))))$$ [p. 88]

Metatheorem 3. $\textbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\forall x(\exists t(\textbf{member\_of}(x,A,t)) \rightarrow$$
$$\forall t(\textbf{exists\_at}(x,t) \rightarrow$$
$$\forall y(\mathbf{p}(y,x,t) \rightarrow \omega(y,y,t))$$ [p. 88]

Metatheorem 4. $\textbf{Unified\_Under}(A,\omega,\mathbf{p}) \rightarrow$

$$\forall xyt(\textbf{member\_of}(x,A,t) \wedge \textbf{member\_of}(y,A,t) \rightarrow$$
$$\textbf{exists\_at}(x,t) \wedge \textbf{exists\_at}(y,t) \rightarrow$$
$$\exists z(\mathbf{p}(z,x,t) \wedge \mathbf{p}(z,y,t)) \rightarrow$$
$$\forall w(\mathbf{p}(w,x,t) \leftrightarrow \mathbf{p}(w,y,t))))$$ [p. 89]

Appendix D

BFO Upper Ontology Formulas

D.1 Taxonomic Axioms

Axiom 30. **immediate_is_a**(*Continuant*,*Entity*)

Axiom 31. **immediate_is_a**(*Occurrent*,*Entity*)

Axiom 32. **immediate_is_a**(*IndependentContinuant*,*Continuant*)

Axiom 33. **immediate_is_a**(*DependentContinuant*,*Continuant*)

Axiom 34. **immediate_is_a**(*SpatialRegion*,*Continuant*)

Axiom 35. **immediate_is_a**(*MaterialEntity*,*IndependentContinuant*)

Axiom 36. **immediate_is_a**(*Object*,*MaterialEntity*)

Axiom 37. **immediate_is_a**(*ObjectAggregate*,*MaterialEntity*)

Axiom 38. **immediate_is_a**(*GenericallyDependentContinuant*,*DependentContinuant*)

Axiom 39. **immediate_is_a**(*SpecificallyDependentContinuant*,*DependentContinuant*)

Axiom 40. **immediate_is_a**(*Quality*,*SpecificallyDependentContinuant*)

Axiom 41. **immediate_is_a**(*RealizableEntity*,*SpecificallyDependentContinuant*)

Axiom 42. **immediate_is_a**(*Role*,*RealizableEntity*)

Axiom 43. **immediate_is_a**(*Disposition*,*RealizableEntity*)

Axiom 44. **immediate_is_a**(*Capability*,*RealizableEntity*)

Axiom 45. **immediate_is_a**(*Function*,*RealizableEntity*)

Axiom 46. **immediate_is_a**(*ProcessualEntity*,*Occurrent*)

Axiom 47. **immediate_is_a**(*SpatioTemporalRegion*,*Occurrent*)

Axiom 48. **immediate_is_a**(*TemporalRegion*,*Occurrent*)

Axiom 49. **immediate_is_a**(*Process*,*ProcessualEntity*)

Axiom 50. **immediate_is_a**(*TemporalInstant*,*TemporalRegion*)

Axiom 51. **immediate_is_a**(*TemporalInterval*,*TemporalRegion*)

D.2 Disjointness Theorems

Theorem 59. $\textbf{disjoint\_from}(\mathit{Continuant}, \mathit{Occurrent})$

Theorem 60. $\textbf{disjoint\_from}(\mathit{IndependentContinuant}, \mathit{DependentContinuant})$

Theorem 61. $\textbf{disjoint\_from}(\mathit{IndependentContinuant}, \mathit{SpatialRegion})$

Theorem 62. $\textbf{disjoint\_from}(\mathit{SpatialRegion}, \mathit{DependentContinuant})$

Theorem 63. $\textbf{disjoint\_from}(\mathit{Object}, \mathit{ObjectAggregate})$

Theorem 64. $\textbf{disjoint\_from}(\mathit{GenericallyDependentContinuant}, \mathit{SpecificallyDependentContinuant})$

Theorem 65. $\textbf{disjoint\_from}(\mathit{Quality}, \mathit{RealizableEntity})$

Theorem 66. $\textbf{disjoint\_from}(\mathit{Disposition}, \mathit{Role})$

Theorem 67. $\textbf{disjoint\_from}(\mathit{ProcessualEntity}, \mathit{SpatioTemporalRegion})$

Theorem 68. $\textbf{disjoint\_from}(\mathit{ProcessualEntity}, \mathit{TemporalRegion})$

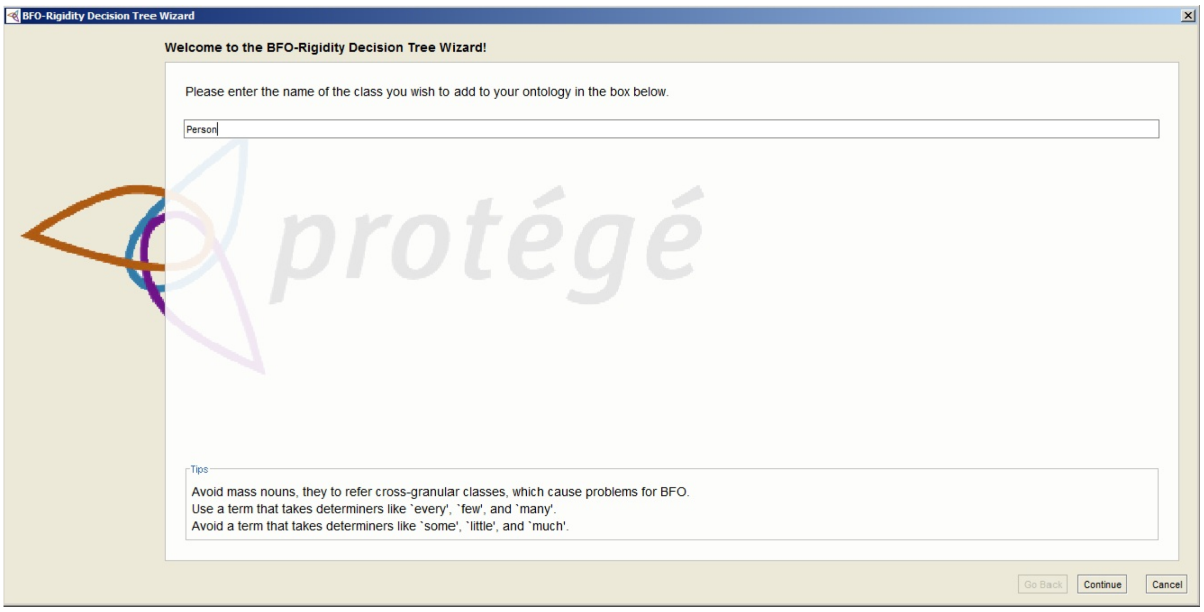
Theorem 69. $\textbf{disjoint\_from}(\mathit{SpatioTemporalRegion}, \mathit{TemporalRegion})$

Theorem 70. $\textbf{disjoint\_from}(\mathit{TemporalInstant}, \mathit{TemporalInterval})$

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix E

BFO-Rigidity Decision Tree Wizard Menu Choice Screenshots for Person



Welcome to the BFO-Rigidity Decision Tree Wizard!

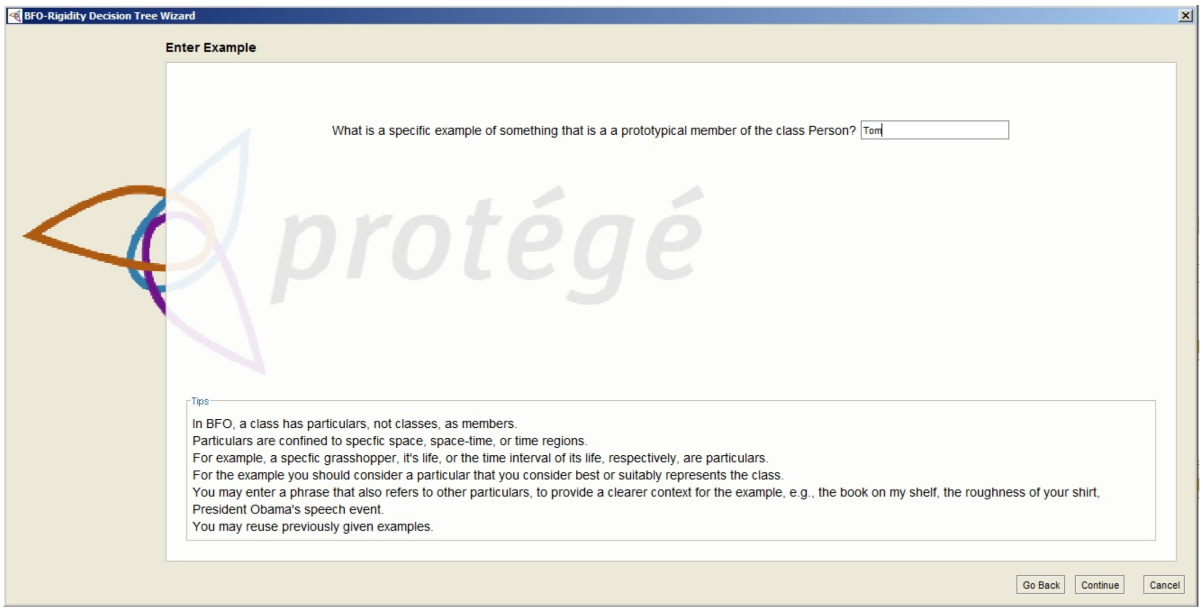
Please enter the name of the class you wish to add to your ontology in the box below.

Person

Tips

Avoid mass nouns, they to refer cross-granular classes, which cause problems for BFO.
Use a term that takes determiners like `every', `few', and `many'.
Avoid a term that takes determiners like `some', `little', and `much'.

Figure E.1: Enter Class Screen for Person



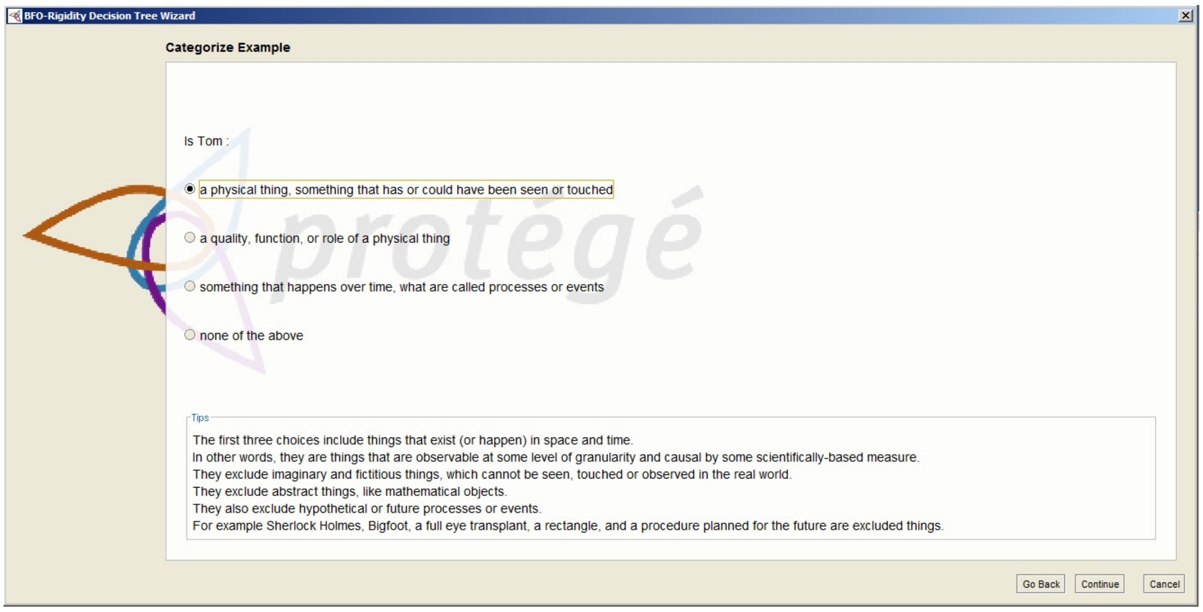
Enter Example

What is a specific example of something that is a a prototypical member of the class Person? Tom

Tips

In BFO, a class has particulars, not classes, as members.
Particulars are confined to specfic space, space-time, or time regions.
For example, a specfic grasshopper, it's life, or the time interval of its life, respectively, are particulars.
For the example you should consider a particular that you consider best or suitably represents the class.
You may enter a phrase that also refers to other particulars, to provide a clearer context for the example, e.g., the book on my shelf, the roughness of your shirt, President Obama's speech event.
You may reuse previously given examples.

Figure E.2: Enter Example Screen For Person



Categorize Example

Is Tom :

[✓ a physical thing, something that has or could have been seen or touched]

a quality, function, or role of a physical thing

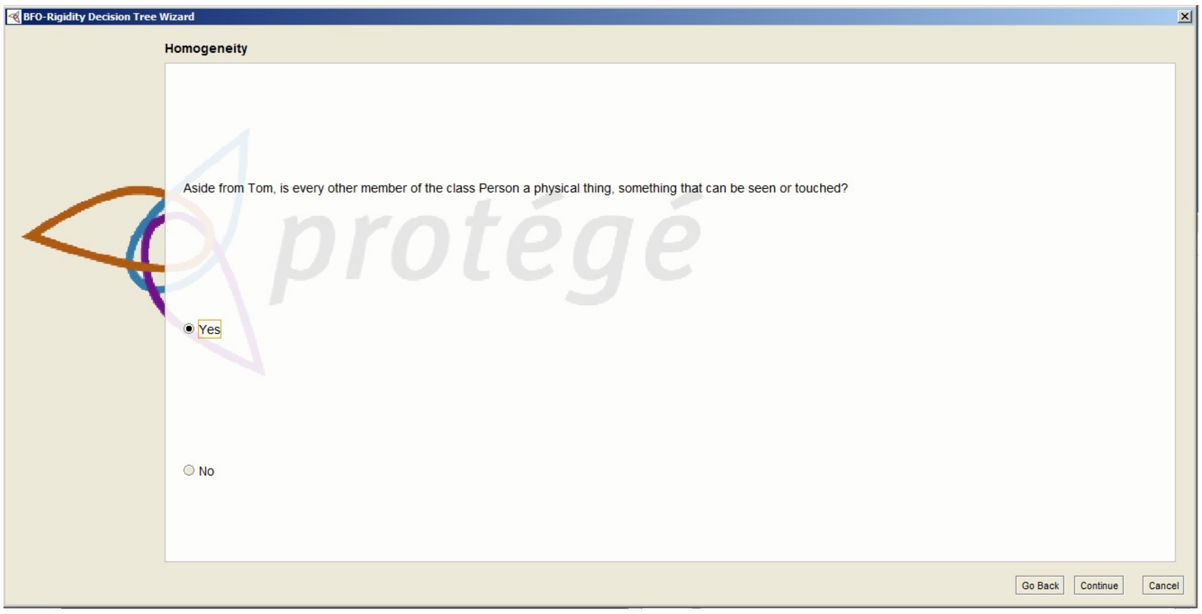
something that happens over time, what are called processes or events

none of the above

Tips

The first three choices include things that exist (or happen) in space and time.
In other words, they are things that are observable at some level of granularity and causal by some scientifically-based measure.
They exclude imaginary and fictitious things, which cannot be seen, touched or observed in the real world.
They exclude abstract things, like mathematical objects.
They also exclude hypothetical or future processes or events.
For example Sherlock Holmes, Bigfoot, a full eye transplant, a rectangle, and a procedure planned for the future are excluded things.

Figure E.3: Categorize Example Screen for Person



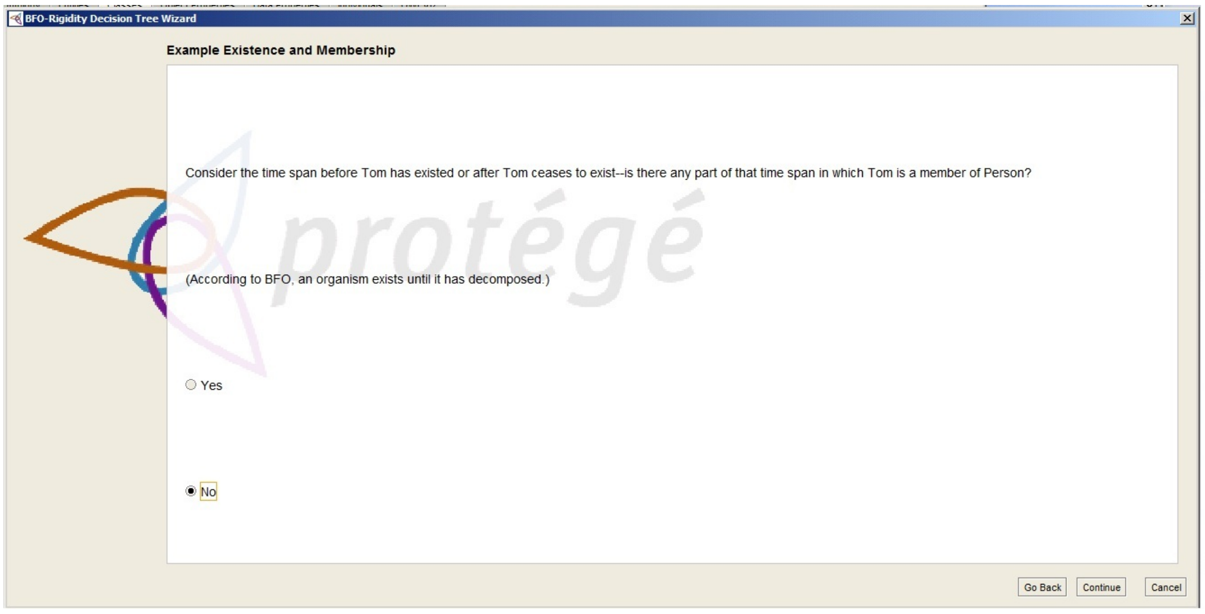
Homogeneity

Aside from Tom, is every other member of the class Person a physical thing, something that can be seen or touched?

[✓ Yes]

No

Figure E.4: Homogeneity Screen for Person



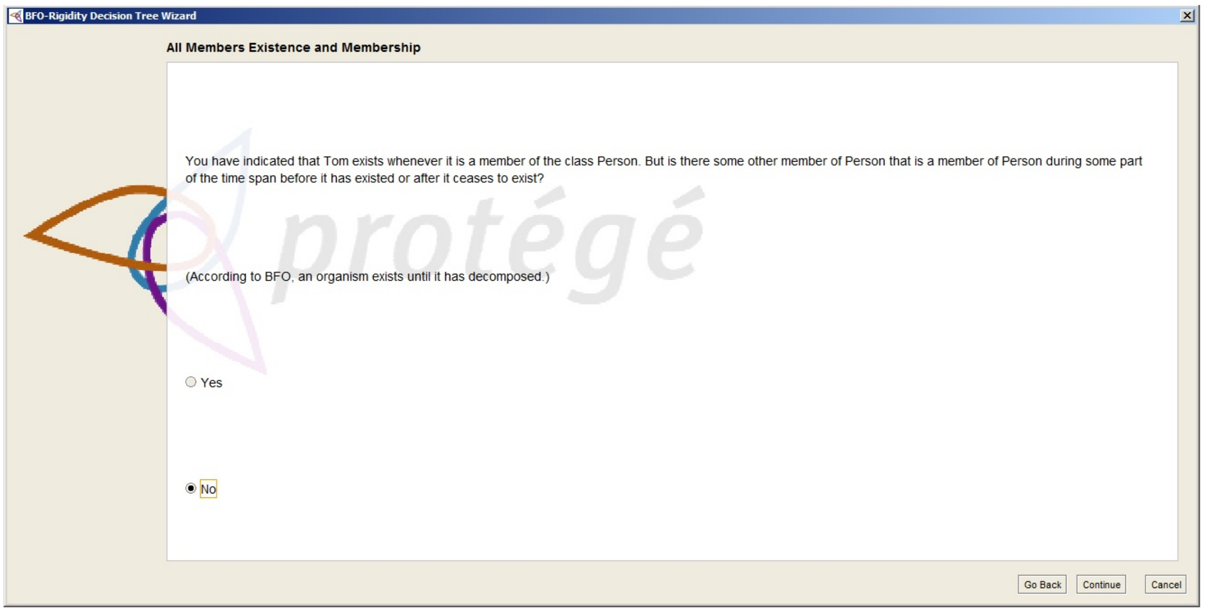
Example Existence and Membership

Consider the time span before Tom has existed or after Tom ceases to exist--is there any part of that time span in which Tom is a member of Person?

(According to BFO, an organism exists until it has decomposed.)

Yes [✓ No]

Figure E.5: Example Existence and Membership Screen For Person



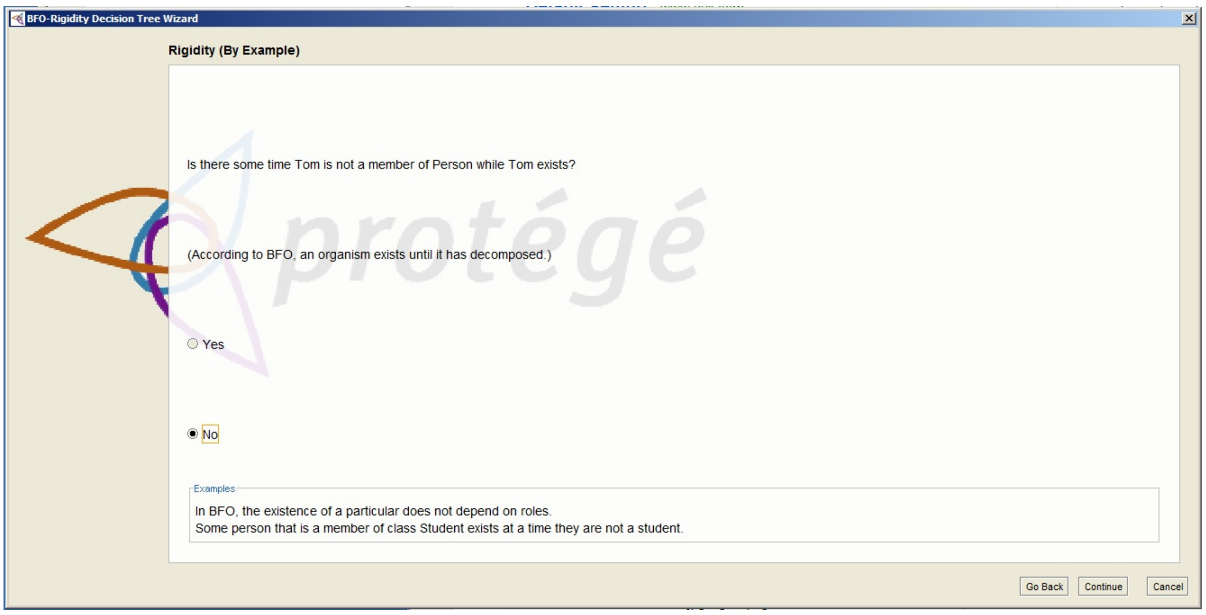
All Members Existence and Membership

You have indicated that Tom exists whenever it is a member of the class Person. But is there some other member of Person that is a member of Person during some part of the time span before it has existed or after it ceases to exist?

(According to BFO, an organism exists until it has decomposed.)

Yes [✓ No]

Figure E.6: All Members Existence and Membership Screen For Person



BFO-Rigidity Decision Tree Wizard

Rigidity (By Example)

Is there some time Tom is not a member of Person while Tom exists?

(According to BFO, an organism exists until it has decomposed.)

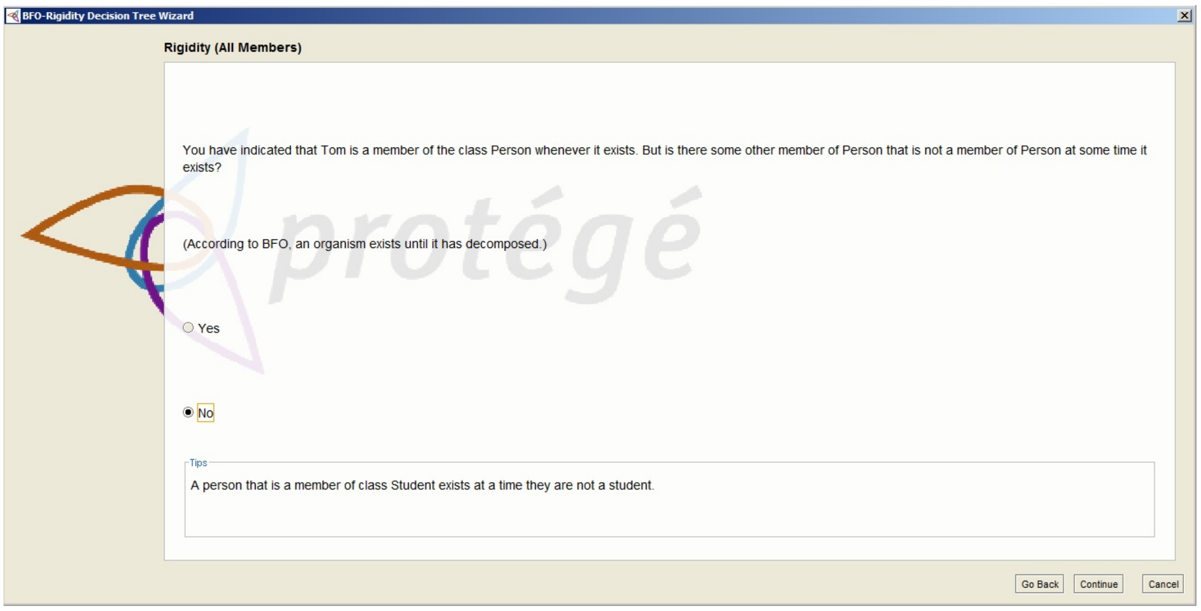
○ Yes

● No

Examples

In BFO, the existence of a particular does not depend on roles.
Some person that is a member of class Student exists at a time they are not a student.

Figure E.7: Rigidity (By Example) Screen For Person



BFO-Rigidity Decision Tree Wizard

Rigidity (All Members)

You have indicated that Tom is a member of the class Person whenever it exists. But is there some other member of Person that is not a member of Person at some time it exists?

(According to BFO, an organism exists until it has decomposed.)

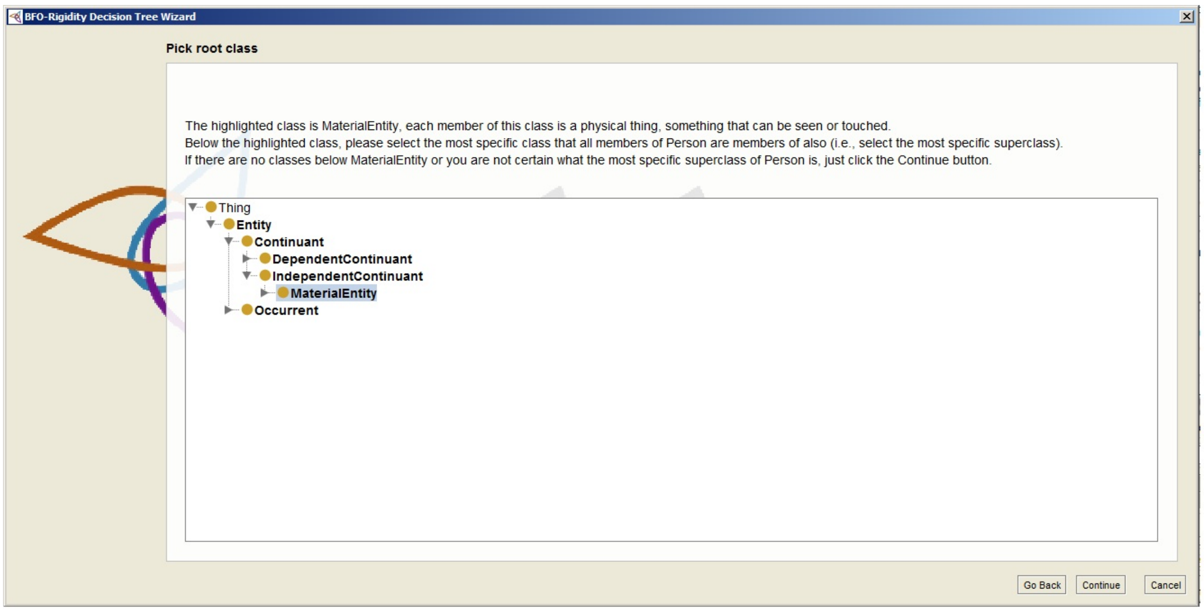
○ Yes

● No

Tips

A person that is a member of class Student exists at a time they are not a student.

Figure E.8: Rigidity (All Members) Screen For Person

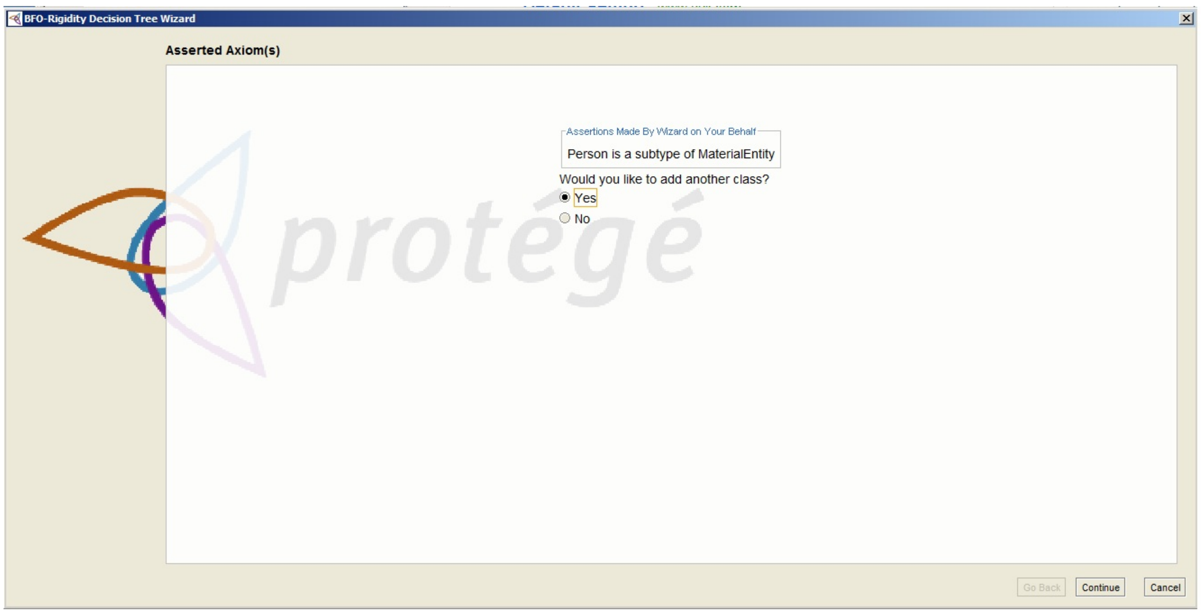


Pick root class

The highlighted class is MaterialEntity, each member of this class is a physical thing, something that can be seen or touched.
Below the highlighted class, please select the most specific class that all members of Person are members of also (i.e., select the most specific superclass).
If there are no classes below MaterialEntity or you are not certain what the most specific superclass of Person is, just click the Continue button.

```
▼ ● Thing
  ▼ ● Entity
    ▼ ● Continuant
      ▶ ● DependentContinuant
      ▼ ● IndependentContinuant
        ▶ ● MaterialEntity
    ▶ ● Occurrent
```

Figure E.9: Pick Root Class Screen for Person



Asserted Axiom(s)

Assertions Made By Wizard on Your Behalf

Person is a subtype of MaterialEntity

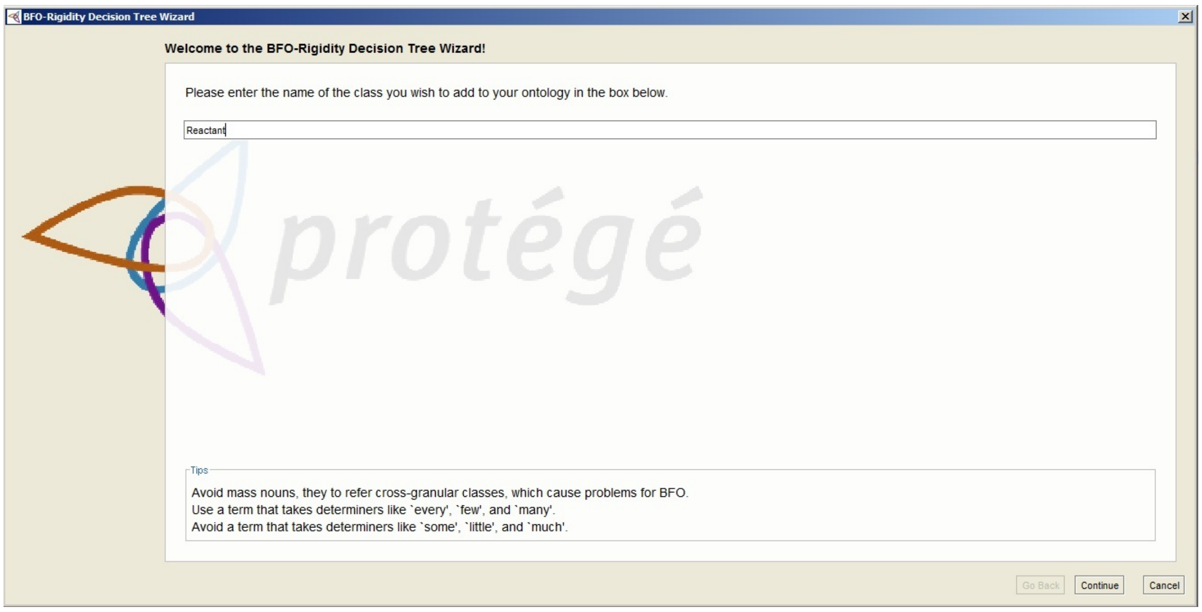
Would you like to add another class?

[✓ Yes] No

Figure E.10: Asserted Axiom(s) Screen for Person

Appendix F

BFO-Rigidity Decision Tree Wizard Menu Choice Screenshots for Reactant (Prior to Restrictions)



Welcome to the BFO-Rigidity Decision Tree Wizard!

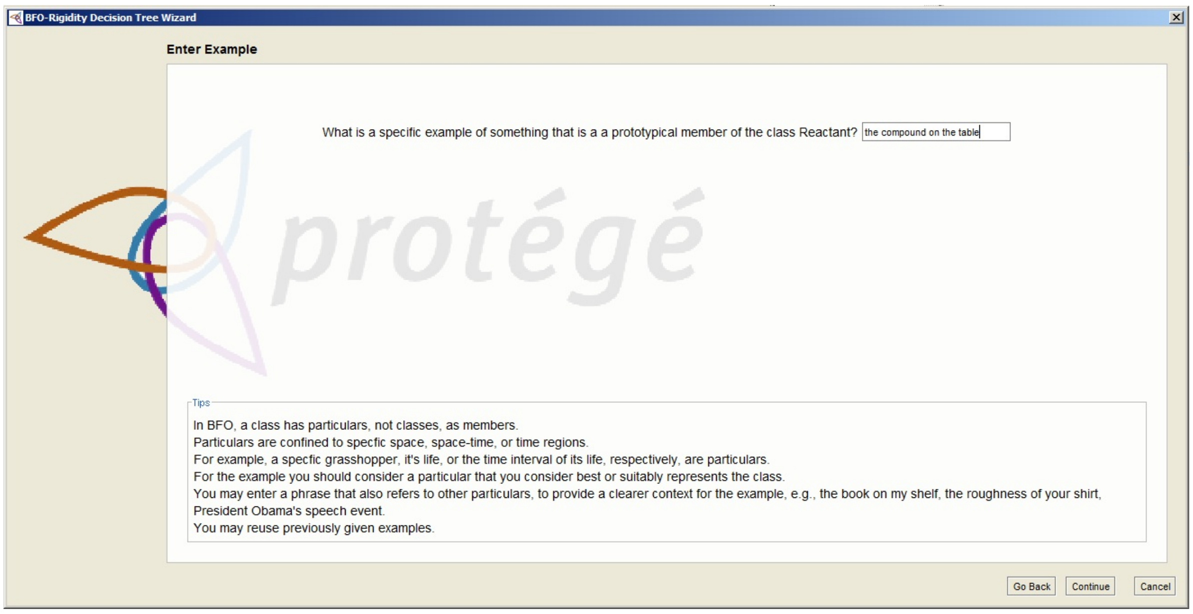
Please enter the name of the class you wish to add to your ontology in the box below.

Reactant

Tips

Avoid mass nouns, they to refer cross-granular classes, which cause problems for BFO.
Use a term that takes determiners like `every', `few', and `many'.
Avoid a term that takes determiners like `some', `little', and `much'.

Figure F.1: Enter Class Screen for Reactant



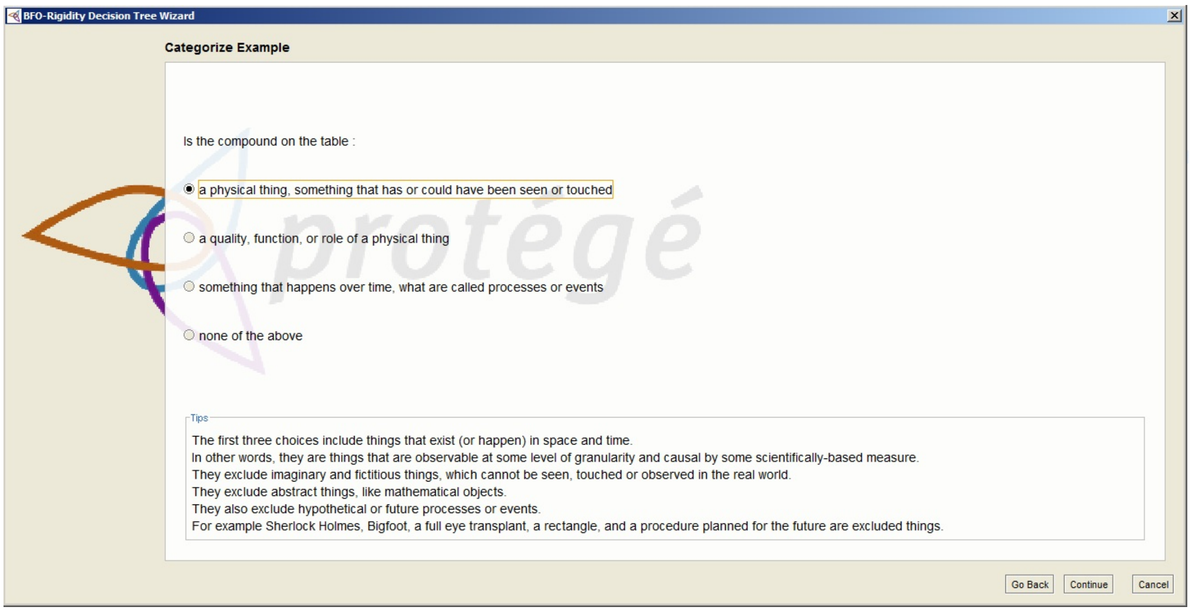
Enter Example

What is a specific example of something that is a a prototypical member of the class Reactant? the compound on the table

Tips

In BFO, a class has particulars, not classes, as members.
Particulars are confined to specfic space, space-time, or time regions.
For example, a specfic grasshopper, it's life, or the time interval of its life, respectively, are particulars.
For the example you should consider a particular that you consider best or suitably represents the class.
You may enter a phrase that also refers to other particulars, to provide a clearer context for the example, e.g., the book on my shelf, the roughness of your shirt, President Obama's speech event.
You may reuse previously given examples.

Figure F.2: Enter Example Screen for Reactant



Categorize Example

Is the compound on the table :

● a physical thing, something that has or could have been seen or touched

○ a quality, function, or role of a physical thing

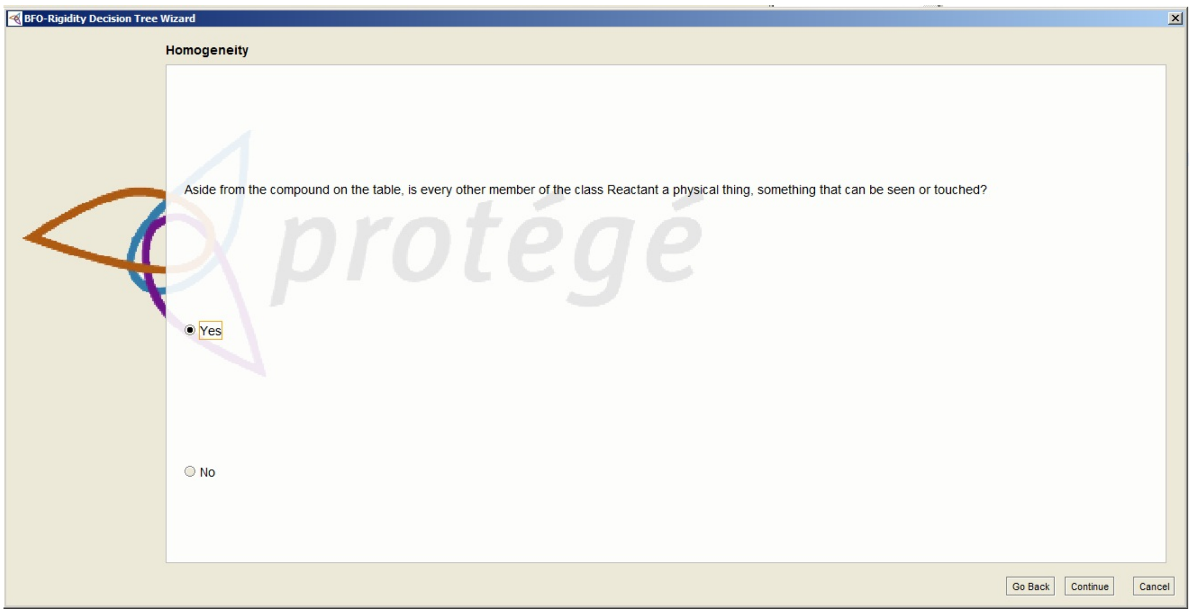
○ something that happens over time, what are called processes or events

○ none of the above

Tips

The first three choices include things that exist (or happen) in space and time.
In other words, they are things that are observable at some level of granularity and causal by some scientifically-based measure.
They exclude imaginary and fictitious things, which cannot be seen, touched or observed in the real world.
They exclude abstract things, like mathematical objects.
They also exclude hypothetical or future processes or events.
For example Sherlock Holmes, Bigfoot, a full eye transplant, a rectangle, and a procedure planned for the future are excluded things.

Figure F.3: Categorize Example for Reactant



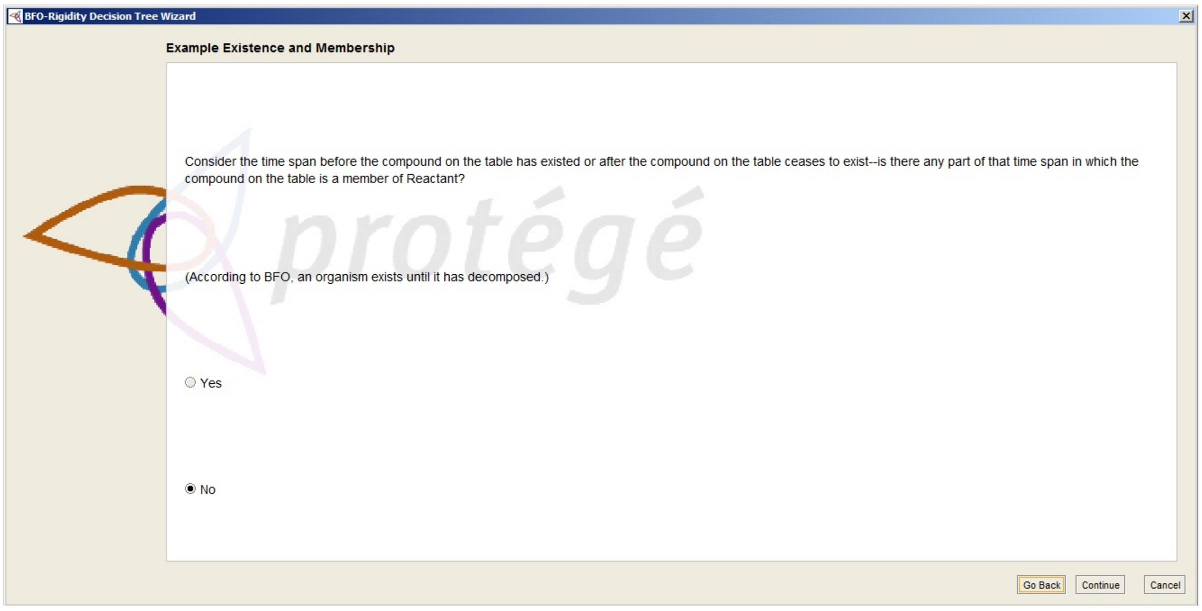
Homogeneity

Aside from the compound on the table, is every other member of the class Reactant a physical thing, something that can be seen or touched?

● Yes

○ No

Figure F.4: Homogeneity Screen for Reactant



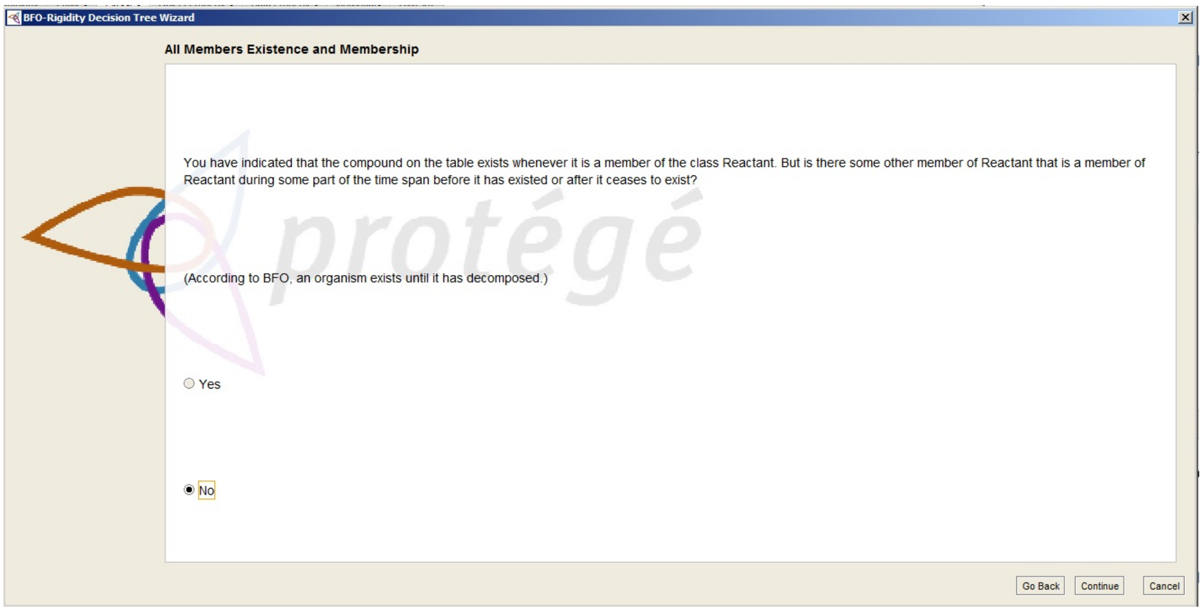
Example Existence and Membership

Consider the time span before the compound on the table has existed or after the compound on the table ceases to exist--is there any part of that time span in which the compound on the table is a member of Reactant?

(According to BFO, an organism exists until it has decomposed.)

Yes [✓ No]

Figure F.5: Example Existence and Membership Screen For Reactant



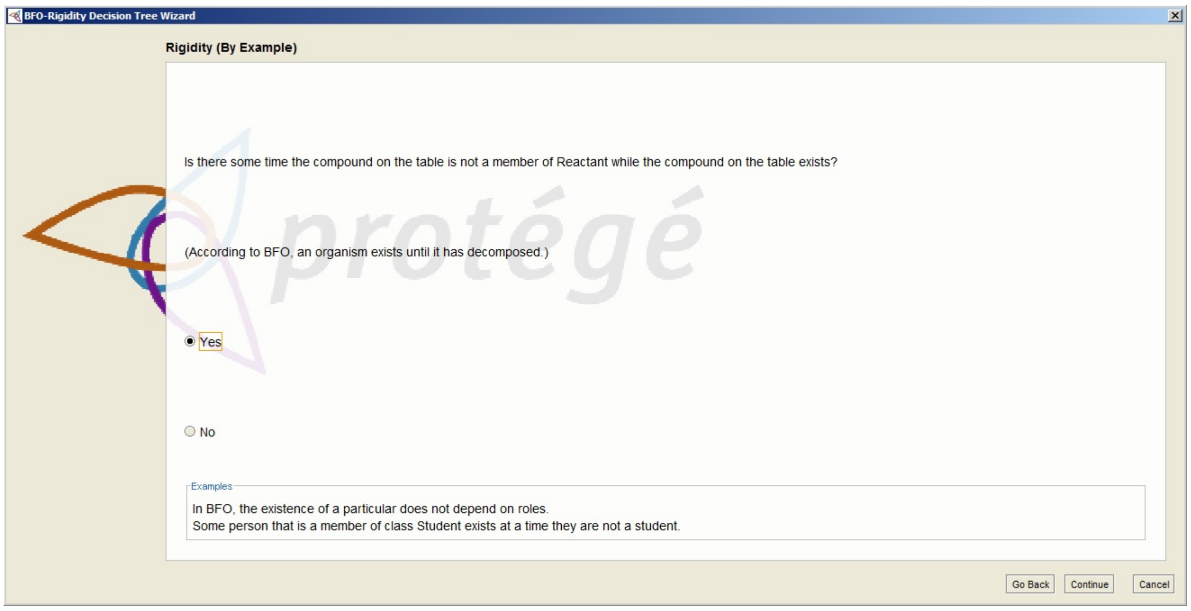
All Members Existence and Membership

You have indicated that the compound on the table exists whenever it is a member of the class Reactant. But is there some other member of Reactant that is a member of Reactant during some part of the time span before it has existed or after it ceases to exist?

(According to BFO, an organism exists until it has decomposed.)

Yes [✓ No]

Figure F.6: All Members Existence and Membership Screen for Reactant



Rigidity (By Example)

Is there some time the compound on the table is not a member of Reactant while the compound on the table exists?

(According to BFO, an organism exists until it has decomposed.)

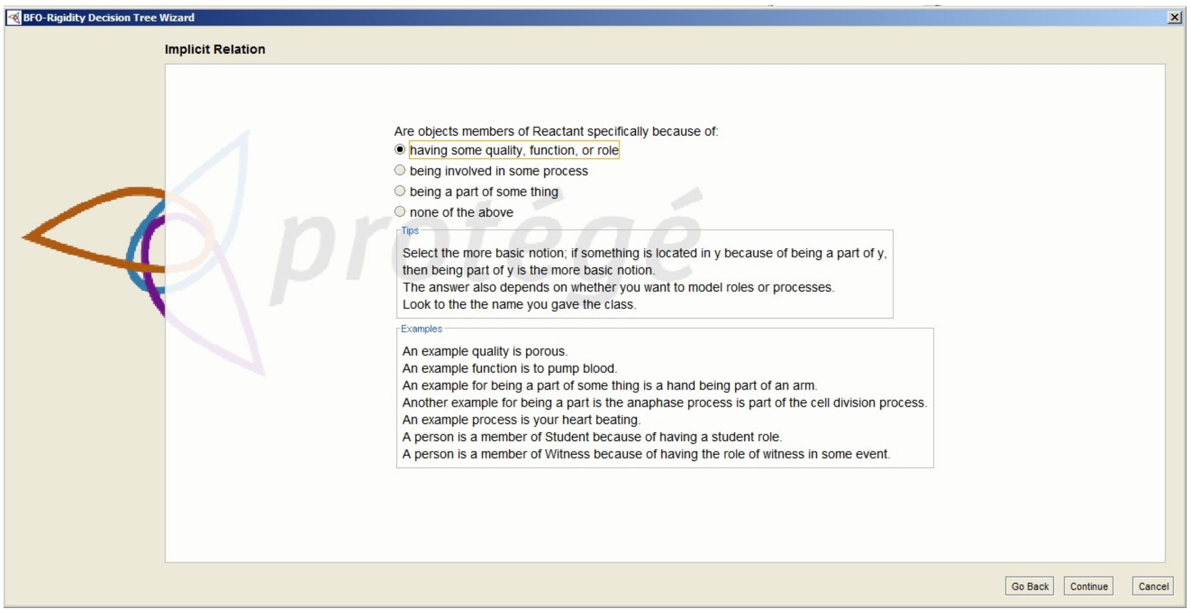
[✓ Yes]

No

Examples

In BFO, the existence of a particular does not depend on roles.
Some person that is a member of class Student exists at a time they are not a student.

Figure F.7: Rigidity (By Example) for Reactant



Implicit Relation

Are objects members of Reactant specifically because of:

[✓ having some quality, function, or role]
being involved in some process
being a part of some thing
none of the above

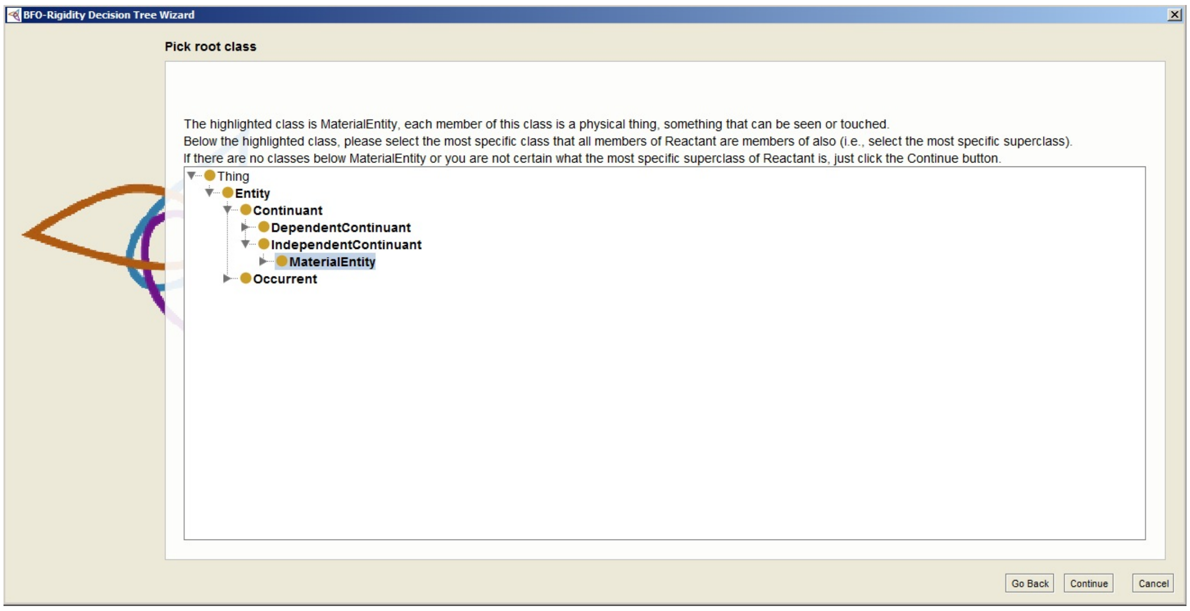
Tips

Select the more basic notion; if something is located in y because of being a part of y, then being part of y is the more basic notion.
The answer also depends on whether you want to model roles or processes.
Look to the the name you gave the class.

Examples

An example quality is porous.
An example function is to pump blood.
An example for being a part of some thing is a hand being part of an arm.
Another example for being a part is the anaphase process is part of the cell division process.
An example process is your heart beating.
A person is a member of Student because of having a student role.
A person is a member of Witness because of having the role of witness in some event.

Figure F.8: Implicit Relation for Reactant

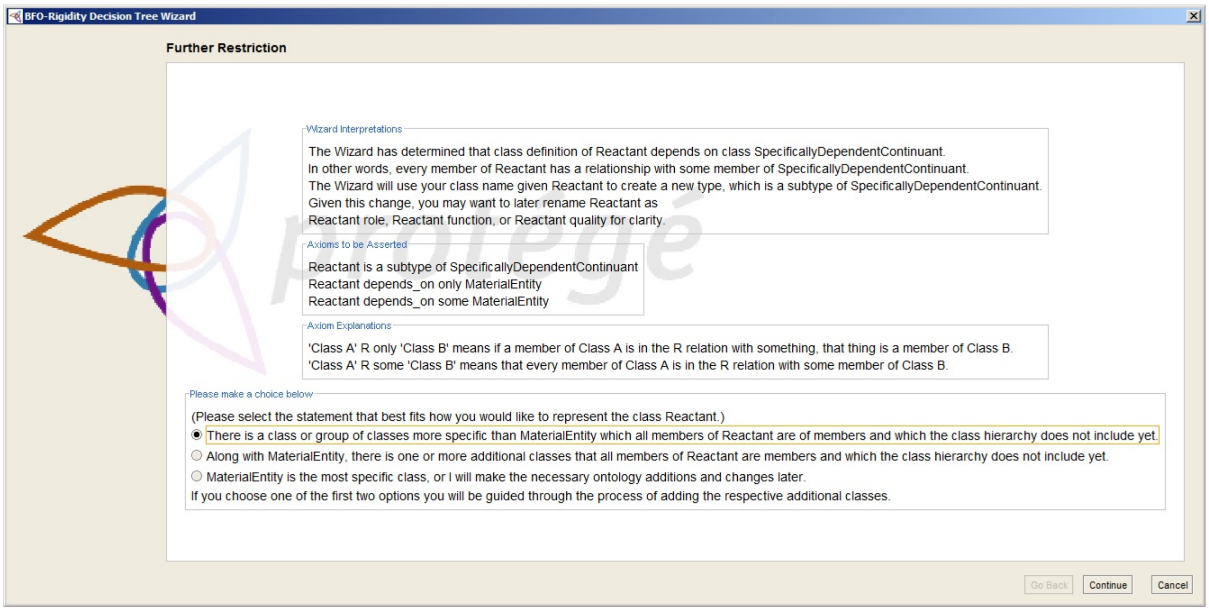


Pick root class

The highlighted class is MaterialEntity, each member of this class is a physical thing, something that can be seen or touched.
Below the highlighted class, please select the most specific class that all members of Reactant are members of also (i.e., select the most specific superclass).
If there are no classes below MaterialEntity or you are not certain what the most specific superclass of Reactant is, just click the Continue button.

- Thing
  - Entity
    - Continuant
      - DependentContinuant
      - IndependentContinuant
        - MaterialEntity
    - Occurrent

Figure F.9: Pick Root Screen for Reactant



Further Restriction

Wizard Interpretations

The Wizard has determined that class definition of Reactant depends on class SpecificallyDependentContinuant.
In other words, every member of Reactant has a relationship with some member of SpecificallyDependentContinuant.
The Wizard will use your class name given Reactant to create a new type, which is a subtype of SpecificallyDependentContinuant.
Given this change, you may want to later rename Reactant as
Reactant role, Reactant function, or Reactant quality for clarity.

Axioms to be Asserted

Reactant is a subtype of SpecificallyDependentContinuant
Reactant depends_on only MaterialEntity
Reactant depends_on some MaterialEntity

Axiom Explanations

'Class A' R only 'Class B' means if a member of Class A is in the R relation with something, that thing is a member of Class B.
'Class A' R some 'Class B' means that every member of Class A is in the R relation with some member of Class B.

Please make a choice below

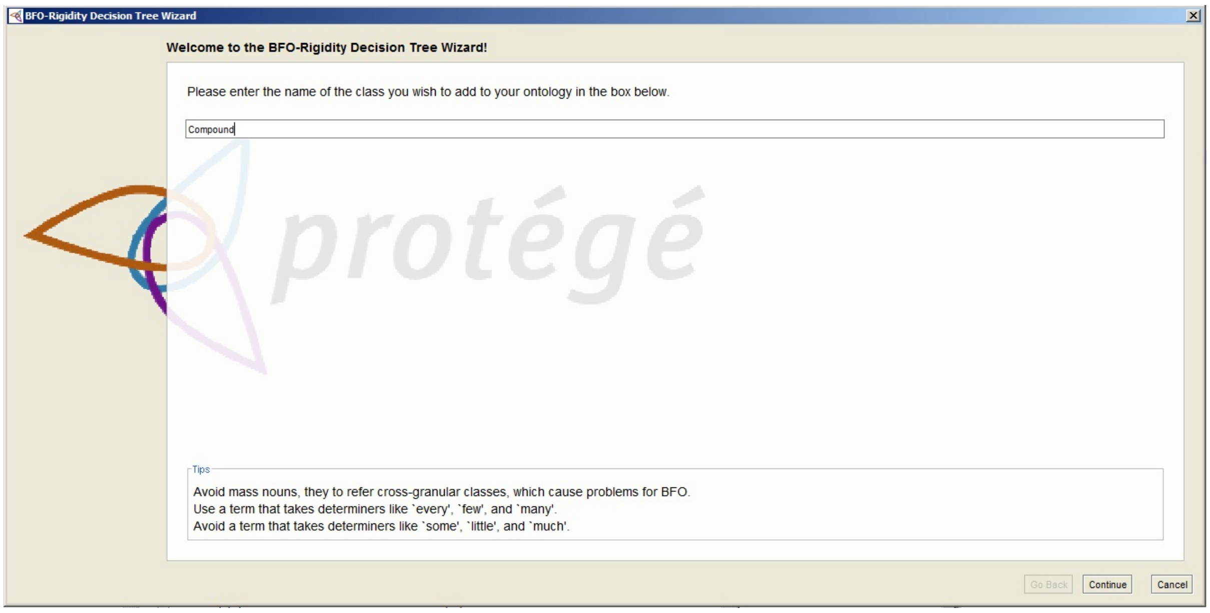
(Please select the statement that best fits how you would like to represent the class Reactant.)

[✓ There is a class or group of classes more specific than MaterialEntity which all members of Reactant are of members and which the class hierarchy does not include yet.]
Along with MaterialEntity, there is one or more additional classes that all members of Reactant are members and which the class hierarchy does not include yet.
MaterialEntity is the most specific class, or I will make the necessary ontology additions and changes later.
If you choose one of the first two options you will be guided through the process of adding the respective additional classes.

Figure F.10: Further Restriction Screen (1) for Reactant

Appendix G

BFO-Rigidity Decision Tree Wizard Menu Choice Screenshots for Restricting Reactant to Compound



Welcome to the BFO-Rigidity Decision Tree Wizard!

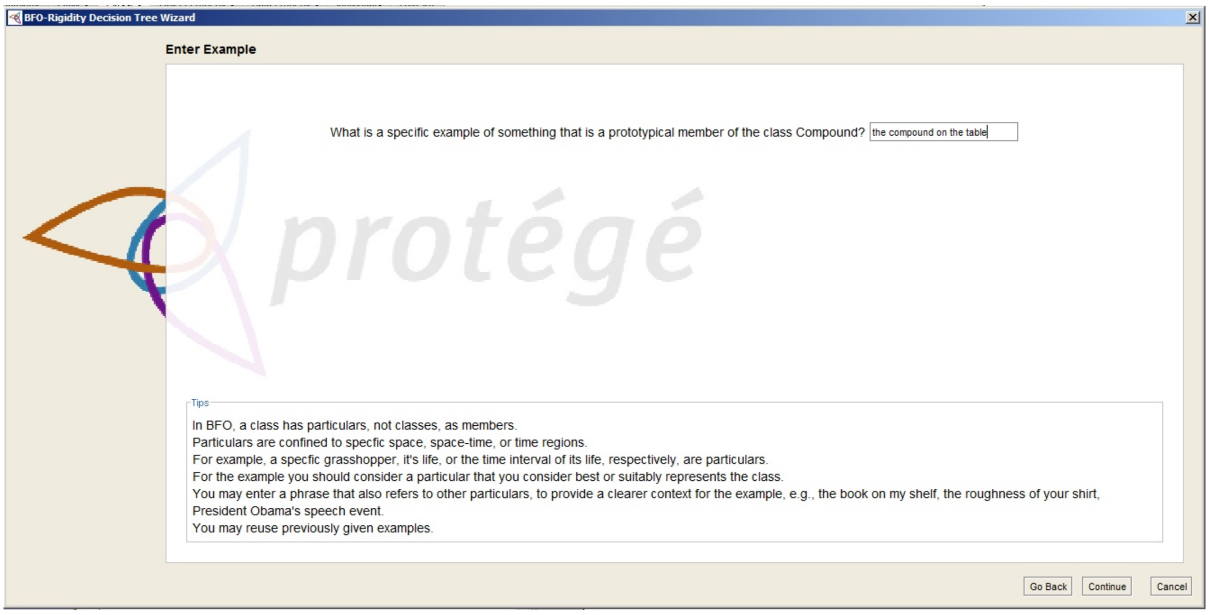
Please enter the name of the class you wish to add to your ontology in the box below.

Compound

Tips

Avoid mass nouns, they to refer cross-granular classes, which cause problems for BFO.
Use a term that takes determiners like `every', `few', and `many'.
Avoid a term that takes determiners like `some', `little', and `much'.

Figure G.1: Enter Class Screen for Compound



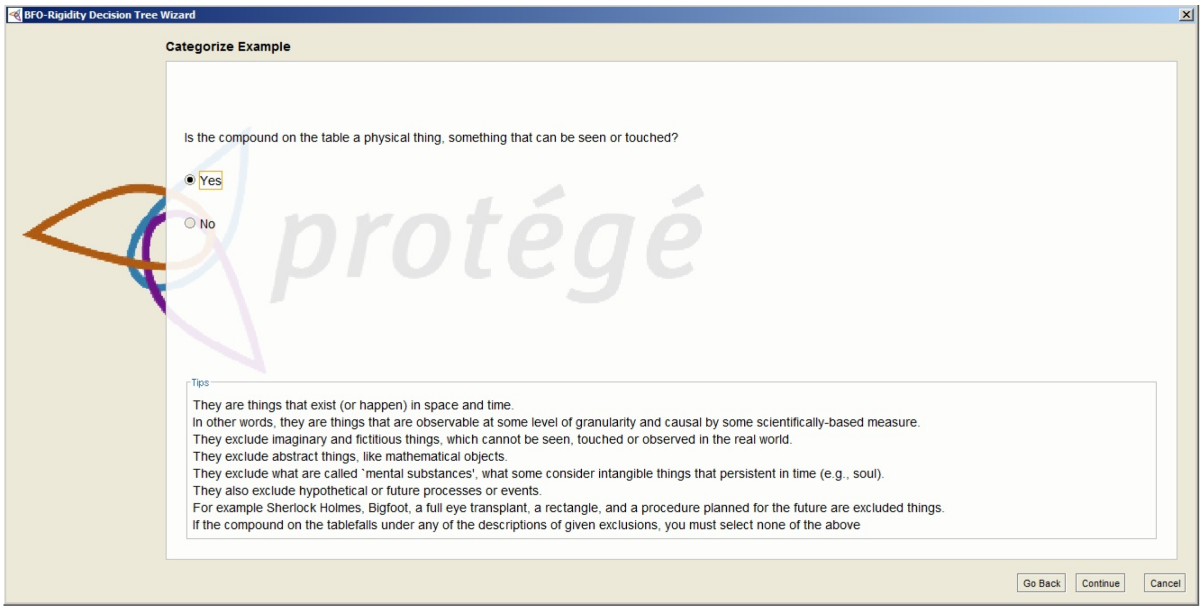
Enter Example

What is a specific example of something that is a prototypical member of the class Compound? the compound on the table

Tips

In BFO, a class has particulars, not classes, as members.
Particulars are confined to specfic space, space-time, or time regions.
For example, a specfic grasshopper, it's life, or the time interval of its life, respectively, are particulars.
For the example you should consider a particular that you consider best or suitably represents the class.
You may enter a phrase that also refers to other particulars, to provide a clearer context for the example, e.g., the book on my shelf, the roughness of your shirt, President Obama's speech event.
You may reuse previously given examples.

Figure G.2: Enter Example Screen for Compound



Categorize Example

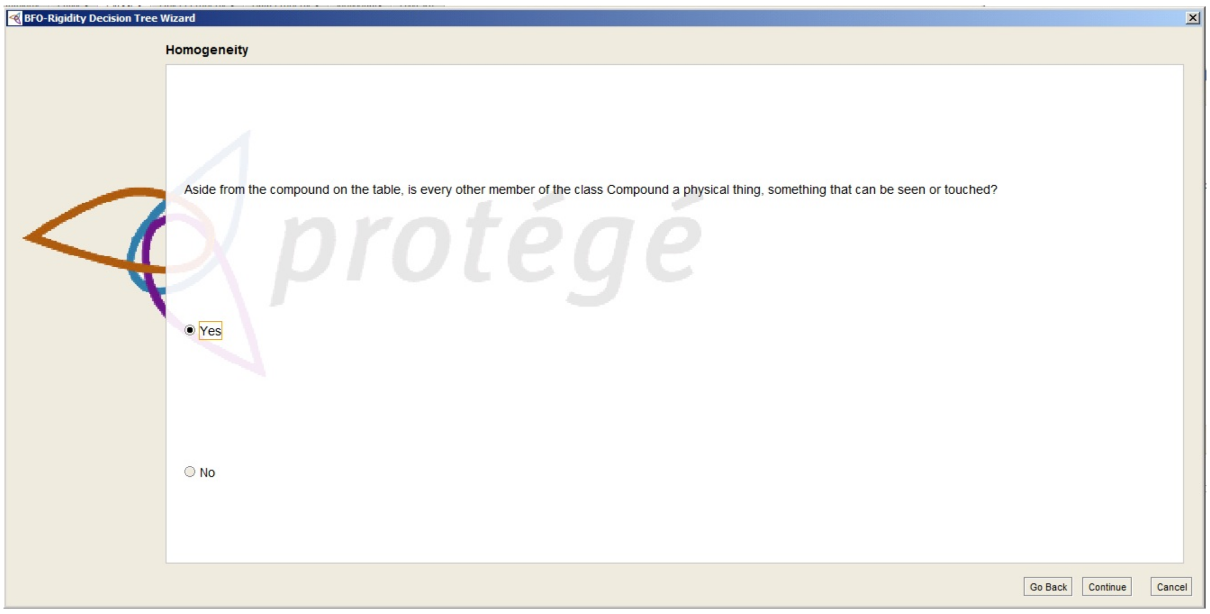
Is the compound on the table a physical thing, something that can be seen or touched?

[✓ Yes] No

Tips

They are things that exist (or happen) in space and time.
In other words, they are things that are observable at some level of granularity and causal by some scientifically-based measure.
They exclude imaginary and fictitious things, which cannot be seen, touched or observed in the real world.
They exclude abstract things, like mathematical objects.
They exclude what are called `mental substances', what some consider intangible things that persistent in time (e.g., soul).
They also exclude hypothetical or future processes or events.
For example Sherlock Holmes, Bigfoot, a full eye transplant, a rectangle, and a procedure planned for the future are excluded things.
If the compound on the tablefalls under any of the descriptions of given exclusions, you must select none of the above

Figure G.3: Categorize Example for Compound

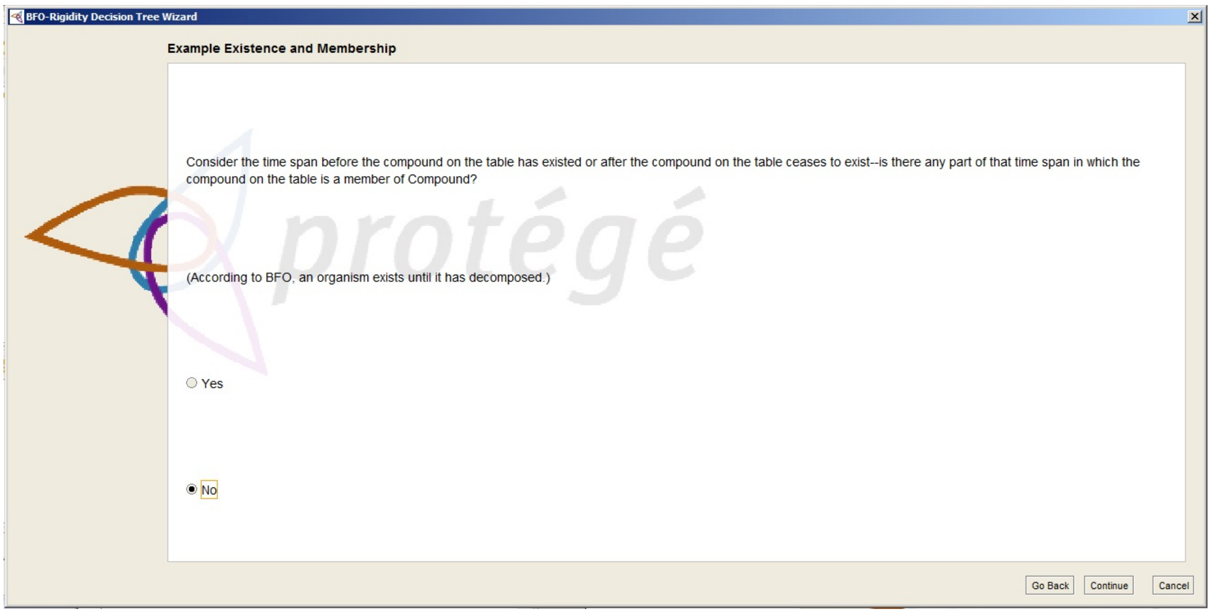


Homogeneity

Aside from the compound on the table, is every other member of the class Compound a physical thing, something that can be seen or touched?

[✓ Yes] No

Figure G.4: Homogeneity Screen for Compound



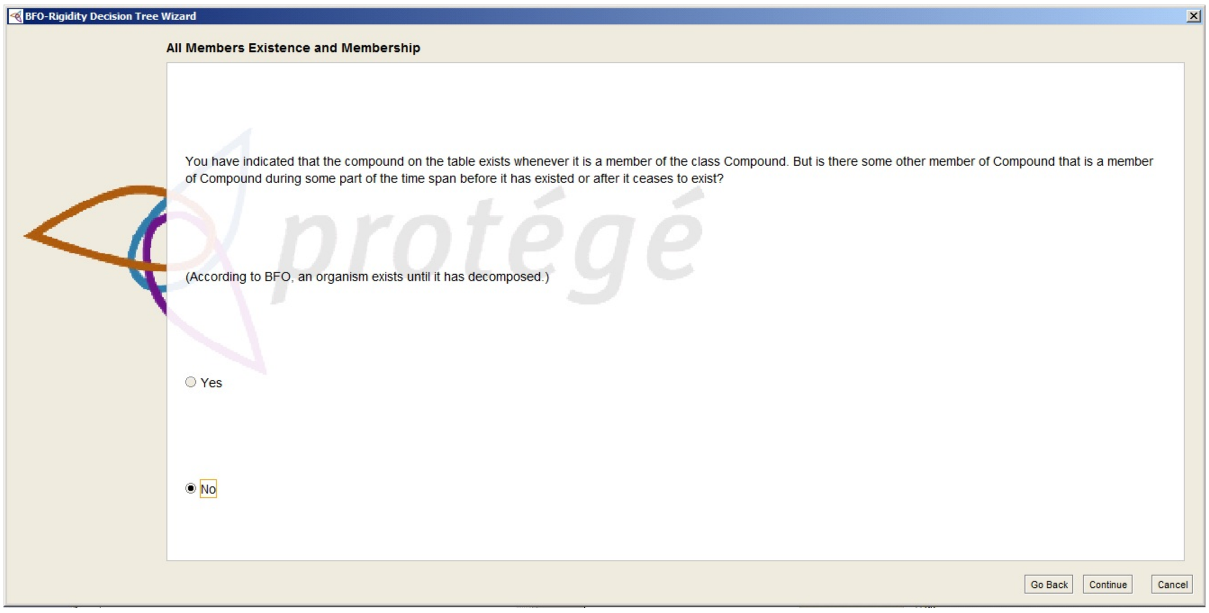
Example Existence and Membership

Consider the time span before the compound on the table has existed or after the compound on the table ceases to exist--is there any part of that time span in which the compound on the table is a member of Compound?

(According to BFO, an organism exists until it has decomposed.)

Yes [✓ No]

Figure G.5: Example Existence and Membership for Compound



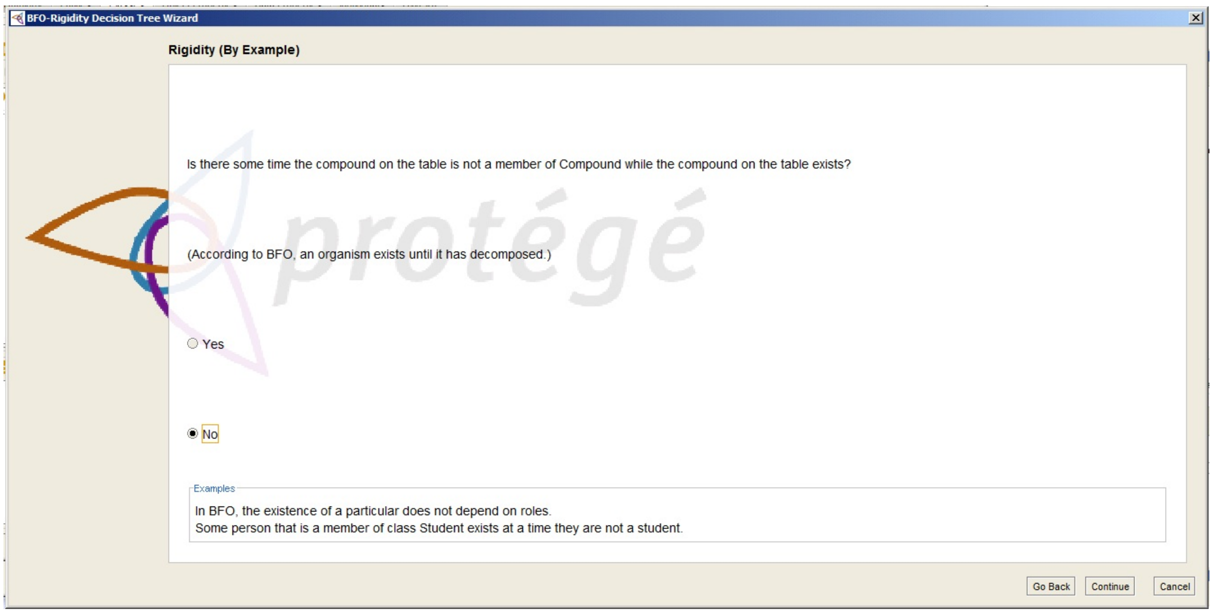
All Members Existence and Membership

You have indicated that the compound on the table exists whenever it is a member of the class Compound. But is there some other member of Compound that is a member of Compound during some part of the time span before it has existed or after it ceases to exist?

(According to BFO, an organism exists until it has decomposed.)

Yes [✓ No]

Figure G.6: All Members Existence and Membership for Compound



Rigidity (By Example)

Is there some time the compound on the table is not a member of Compound while the compound on the table exists?

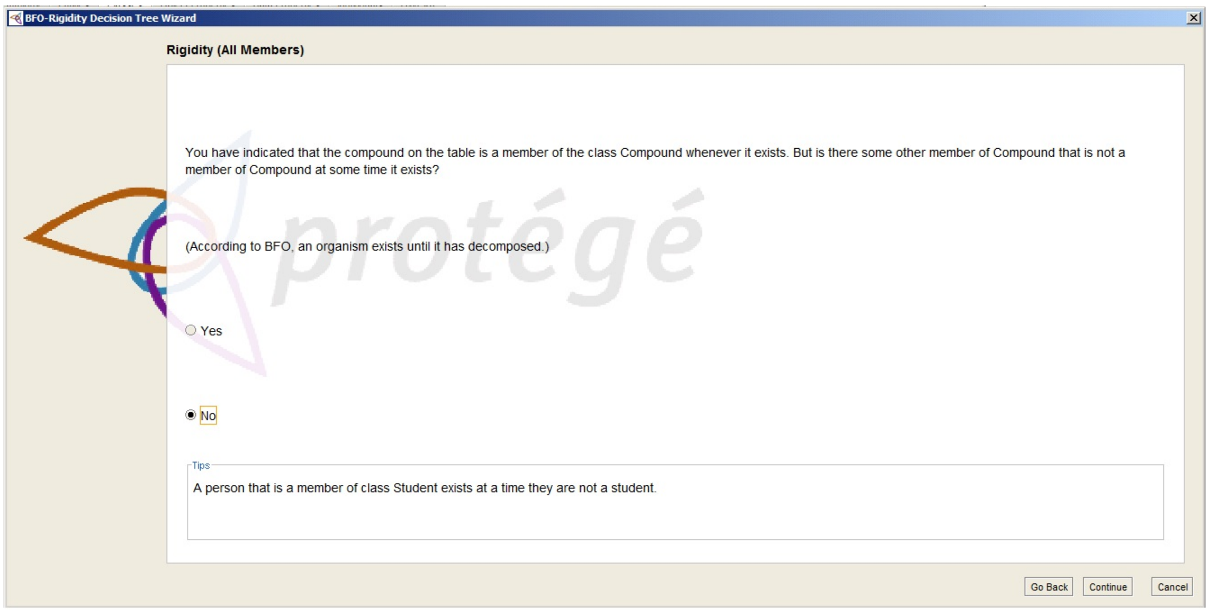
(According to BFO, an organism exists until it has decomposed.)

Yes [✓ No]

Examples

In BFO, the existence of a particular does not depend on roles.
Some person that is a member of class Student exists at a time they are not a student.

Figure G.7: Rigidity (By Example) for Compound



Rigidity (All Members)

You have indicated that the compound on the table is a member of the class Compound whenever it exists. But is there some other member of Compound that is not a member of Compound at some time it exists?

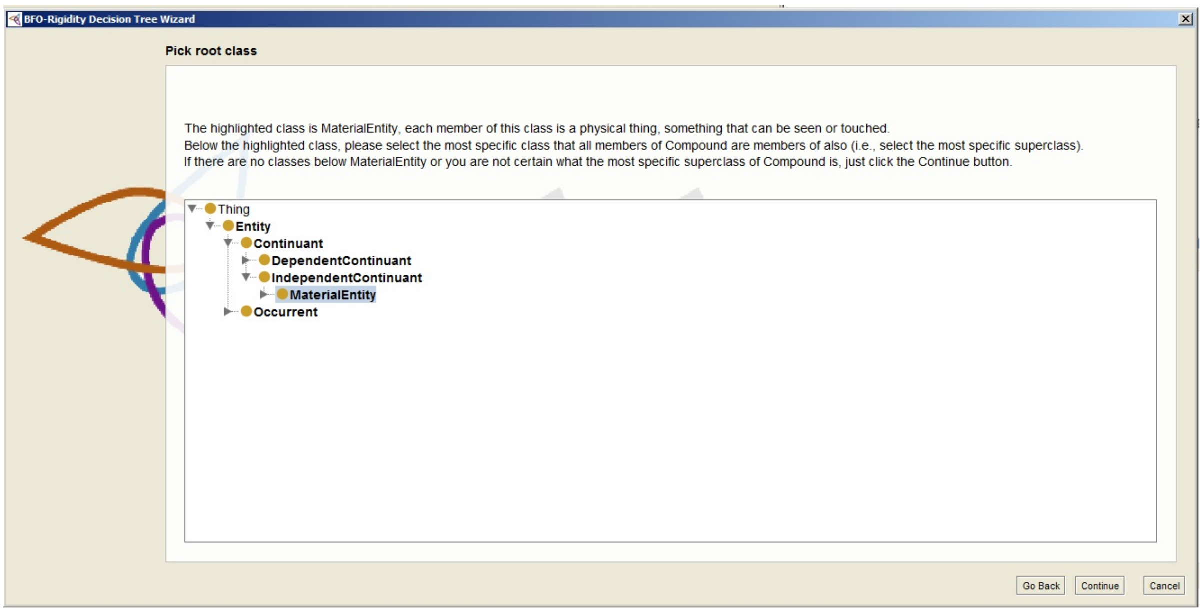
(According to BFO, an organism exists until it has decomposed.)

Yes [✓ No]

Tips

A person that is a member of class Student exists at a time they are not a student.

Figure G.8: Rigidity (All Members) for Compound

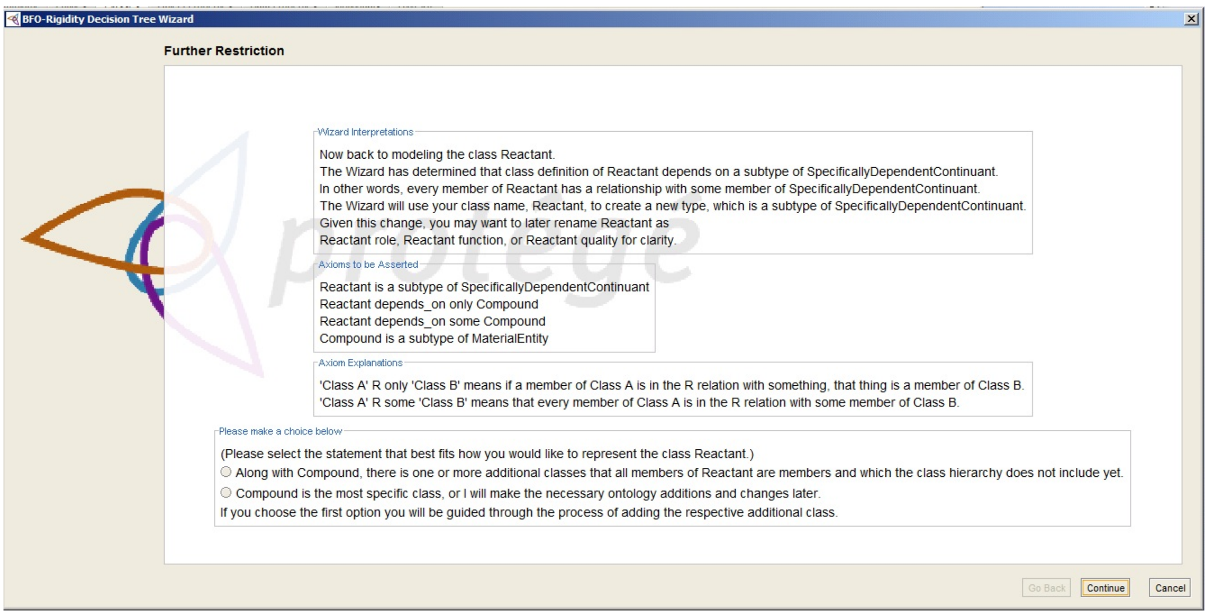


Pick root class

The highlighted class is MaterialEntity, each member of this class is a physical thing, something that can be seen or touched.
Below the highlighted class, please select the most specific class that all members of Compound are members of also (i.e., select the most specific superclass).
If there are no classes below MaterialEntity or you are not certain what the most specific superclass of Compound is, just click the Continue button.

- Thing
  - Entity
    - Continuant
      - DependentContinuant
      - IndependentContinuant
        - MaterialEntity
    - Occurrent

Figure G.9: Pick Root Class Screen for Compound



Further Restriction

Wizard Interpretations

Now back to modeling the class Reactant.
The Wizard has determined that class definition of Reactant depends on a subtype of SpecificallyDependentContinuant.
In other words, every member of Reactant has a relationship with some member of SpecificallyDependentContinuant.
The Wizard will use your class name, Reactant, to create a new type, which is a subtype of SpecificallyDependentContinuant.
Given this change, you may want to later rename Reactant as
Reactant role, Reactant function, or Reactant quality for clarity.

Axioms to be Asserted

Reactant is a subtype of SpecificallyDependentContinuant
Reactant depends_on only Compound
Reactant depends_on some Compound
Compound is a subtype of MaterialEntity

Axiom Explanations

'Class A' R only 'Class B' means if a member of Class A is in the R relation with something, that thing is a member of Class B.
'Class A' R some 'Class B' means that every member of Class A is in the R relation with some member of Class B.

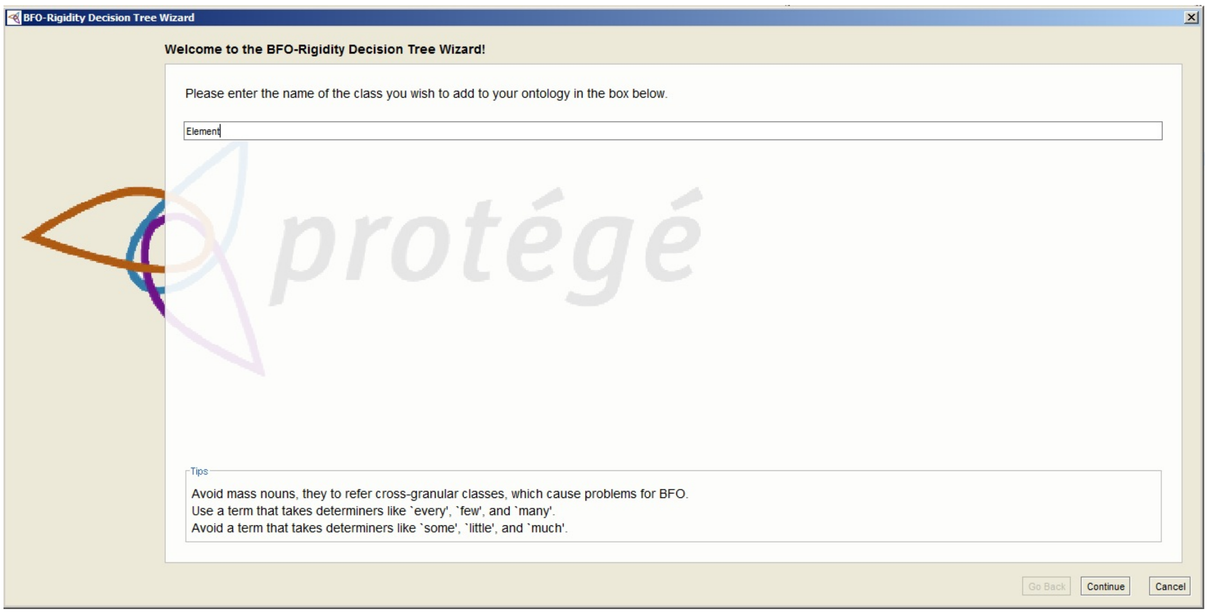
Please make a choice below

(Please select the statement that best fits how you would like to represent the class Reactant.)
○ Along with Compound, there is one or more additional classes that all members of Reactant are members and which the class hierarchy does not include yet.
○ Compound is the most specific class, or I will make the necessary ontology additions and changes later.
If you choose the first option you will be guided through the process of adding the respective additional class.

Figure G.10: Further Restriction Screen (2) for Reactant

Appendix H

BFO-Rigidity Decision Tree Wizard Menu Choice Screenshots for for Restricting Reactant to Element



Welcome to the BFO-Rigidity Decision Tree Wizard!

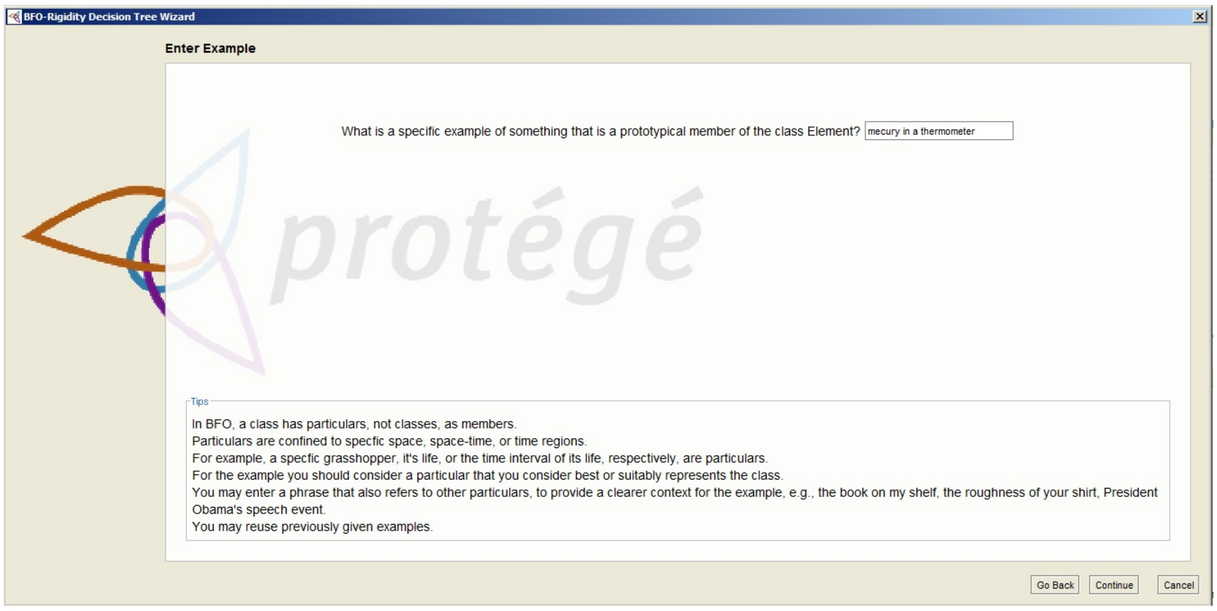
Please enter the name of the class you wish to add to your ontology in the box below.

Element

Tips

Avoid mass nouns, they to refer cross-granular classes, which cause problems for BFO.
Use a term that takes determiners like `every', `few', and `many'.
Avoid a term that takes determiners like `some', `little', and `much'.

Figure H.1: Enter Class Screen for Element



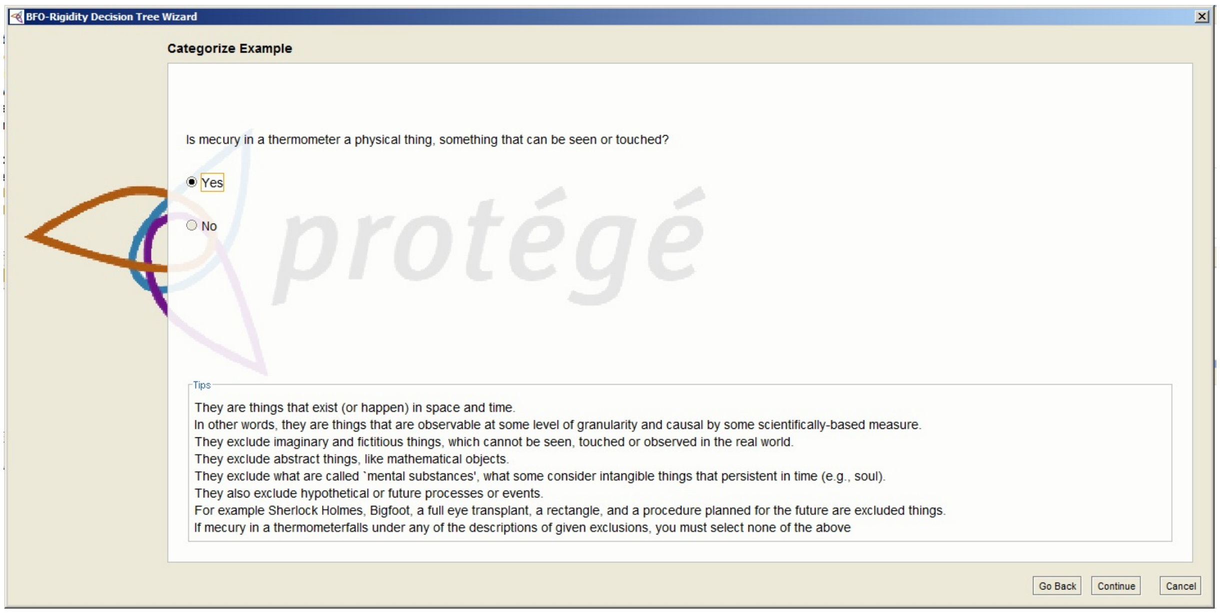
Enter Example

What is a specific example of something that is a prototypical member of the class Element? mecury in a thermometer

Tips

In BFO, a class has particulars, not classes, as members.
Particulars are confined to specfic space, space-time, or time regions.
For example, a specfic grasshopper, it's life, or the time interval of its life, respectively, are particulars.
For the example you should consider a particular that you consider best or suitably represents the class.
You may enter a phrase that also refers to other particulars, to provide a clearer context for the example, e.g., the book on my shelf, the roughness of your shirt, President Obama's speech event.
You may reuse previously given examples.

Figure H.2: Enter Example Screen for Element


BFO-Rigidity Decision Tree Wizard

Categorize Example

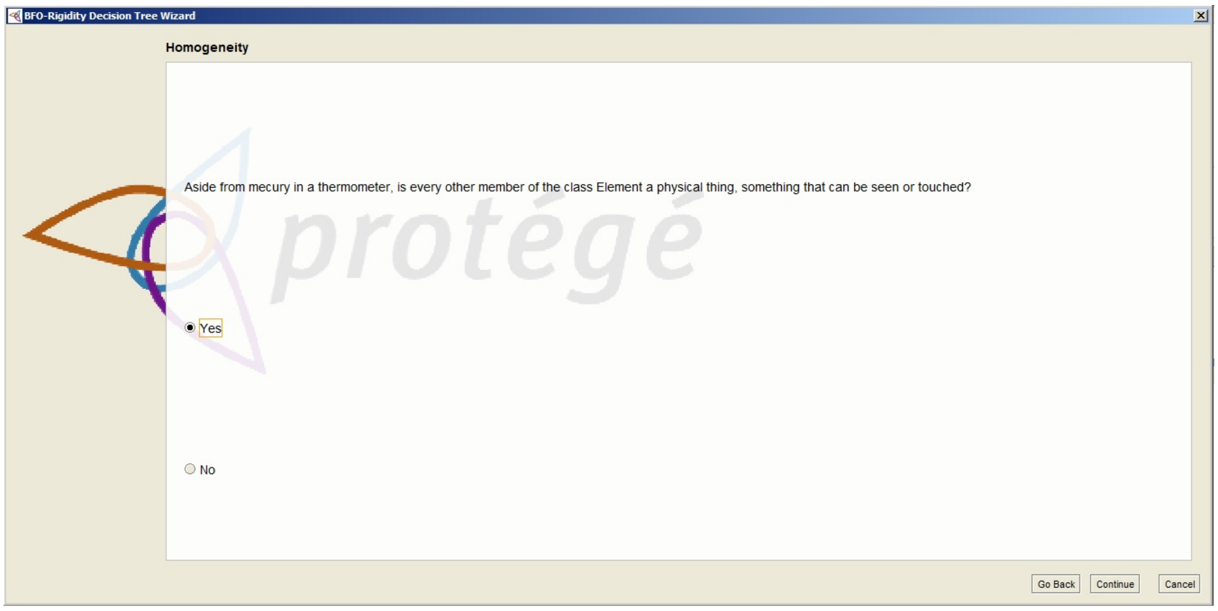
Is mecury in a thermometer a physical thing, something that can be seen or touched?

[✓ Yes] No

Tips

They are things that exist (or happen) in space and time.
In other words, they are things that are observable at some level of granularity and causal by some scientifically-based measure.
They exclude imaginary and fictitious things, which cannot be seen, touched or observed in the real world.
They exclude abstract things, like mathematical objects.
They exclude what are called `mental substances', what some consider intangible things that persistent in time (e.g., soul).
They also exclude hypothetical or future processes or events.
For example Sherlock Holmes, Bigfoot, a full eye transplant, a rectangle, and a procedure planned for the future are excluded things.
If mecury in a thermometerfalls under any of the descriptions of given exclusions, you must select none of the above

Figure H.3: Categorize Example Screen for Element

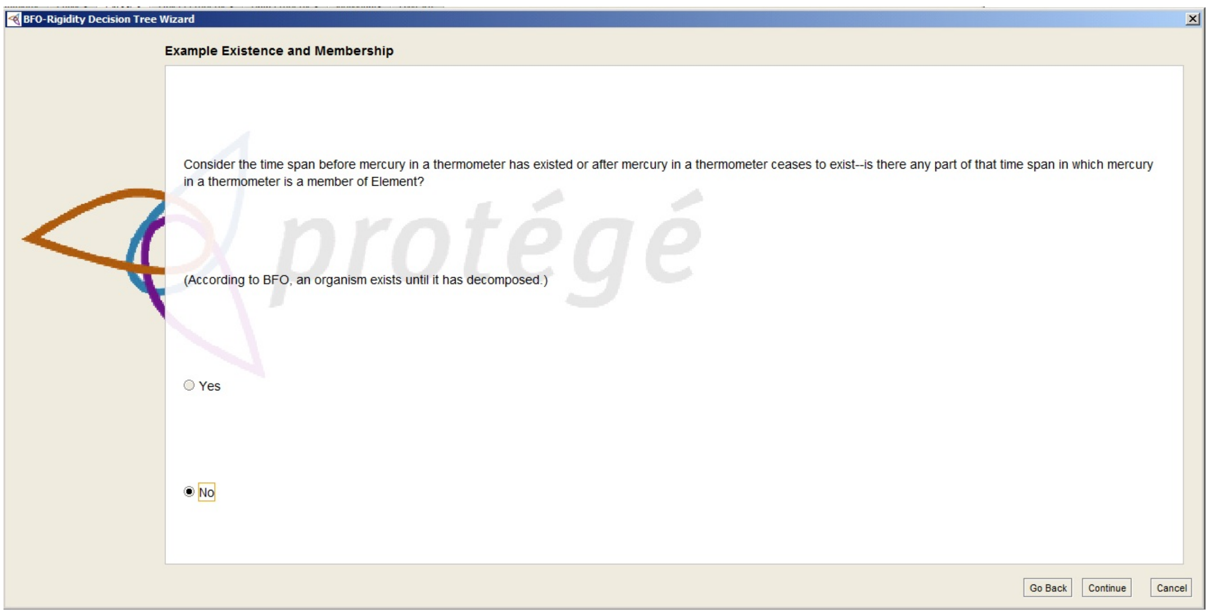

BFO-Rigidity Decision Tree Wizard

Homogeneity

Aside from mecury in a thermometer, is every other member of the class Element a physical thing, something that can be seen or touched?

[✓ Yes] No

Figure H.4: Homogeneity Screen for Element



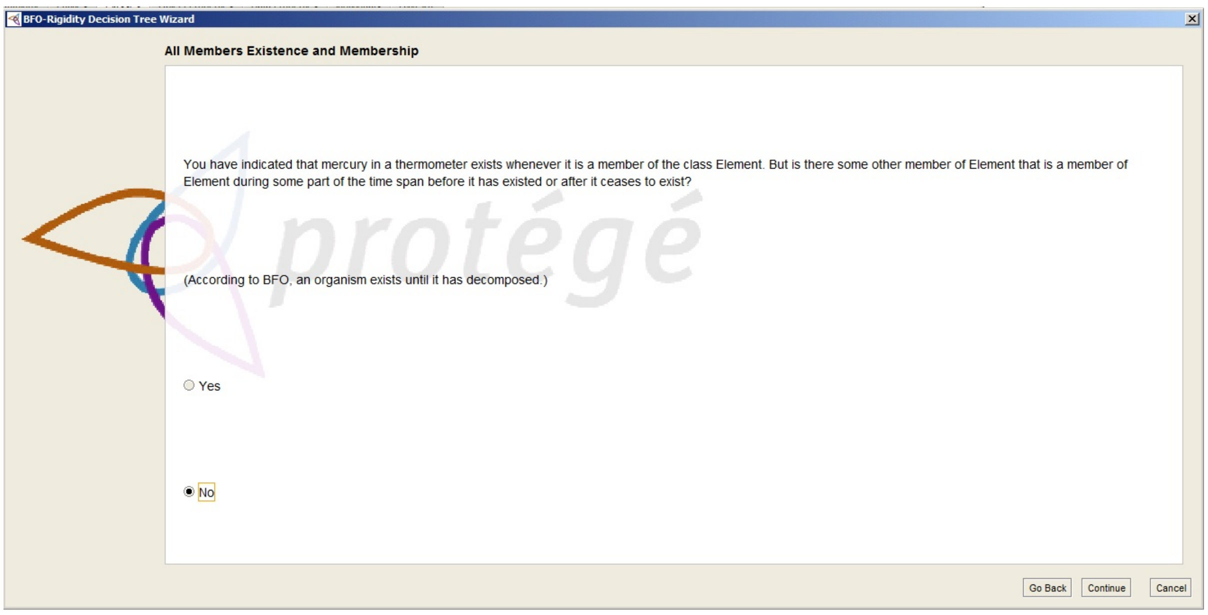
Example Existence and Membership

Consider the time span before mercury in a thermometer has existed or after mercury in a thermometer ceases to exist--is there any part of that time span in which mercury in a thermometer is a member of Element?

(According to BFO, an organism exists until it has decomposed.)

Yes [✓ No]

Figure H.5: Example Existence Screen for Element



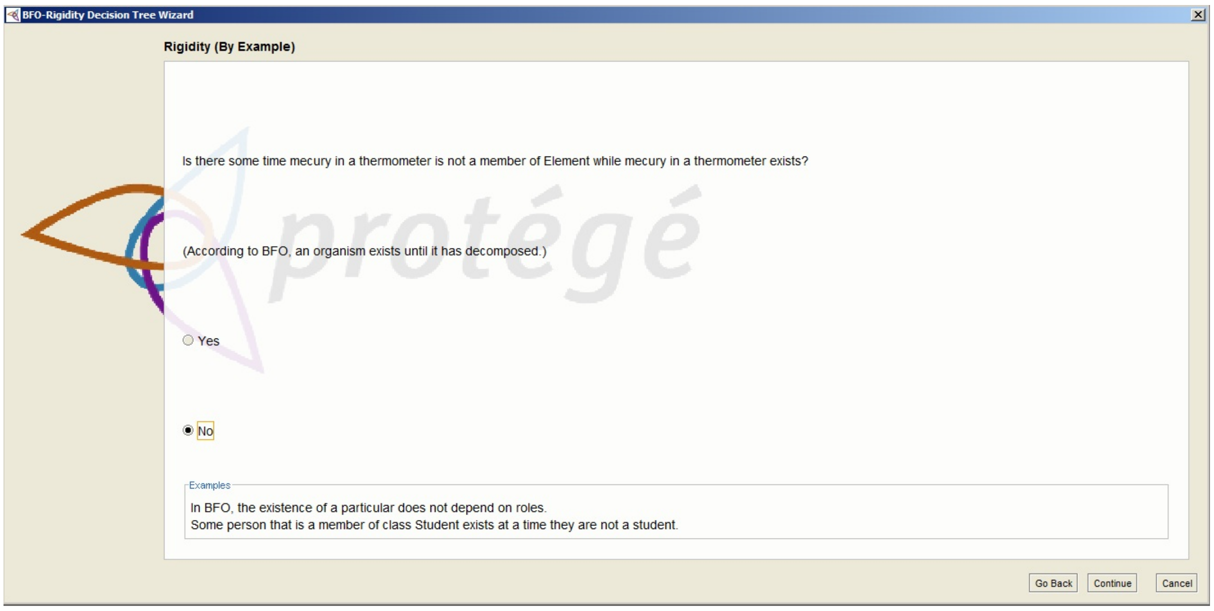
All Members Existence and Membership

You have indicated that mercury in a thermometer exists whenever it is a member of the class Element. But is there some other member of Element that is a member of Element during some part of the time span before it has existed or after it ceases to exist?

(According to BFO, an organism exists until it has decomposed.)

Yes [✓ No]

Figure H.6: All Members Existence Screen for Element



Rigidity (By Example)

Is there some time mecury in a thermometer is not a member of Element while mecury in a thermometer exists?

(According to BFO, an organism exists until it has decomposed.)

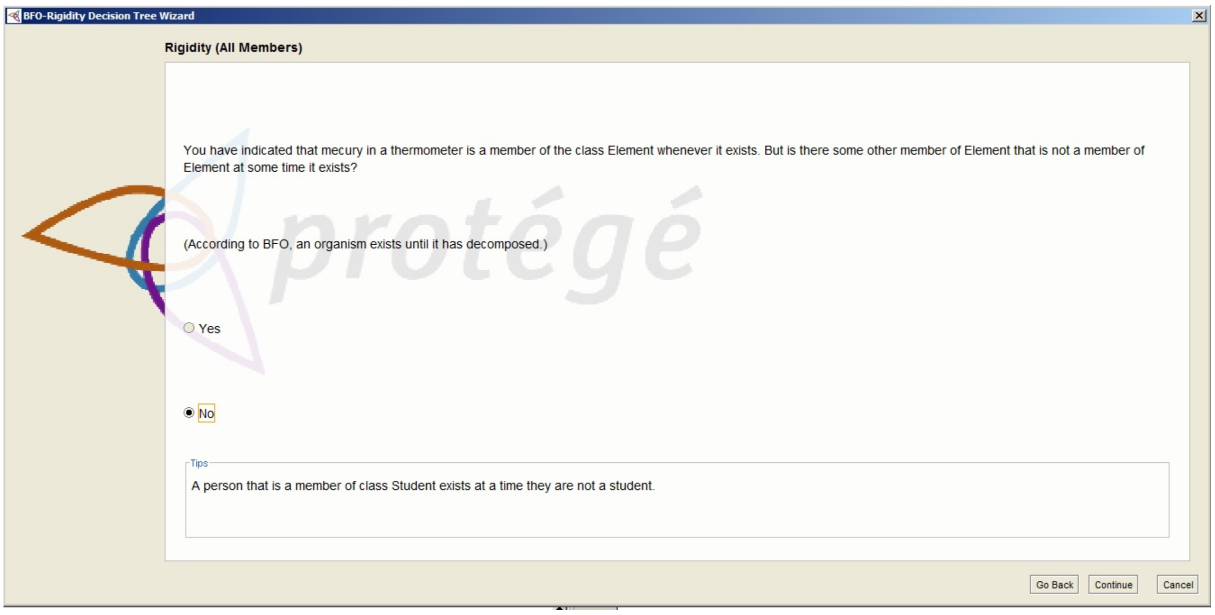
○ Yes

● No

Examples

In BFO, the existence of a particular does not depend on roles.
Some person that is a member of class Student exists at a time they are not a student.

Figure H.7: Rigidity (By Example) Screen for Element



Rigidity (All Members)

You have indicated that mecury in a thermometer is a member of the class Element whenever it exists. But is there some other member of Element that is not a member of Element at some time it exists?

(According to BFO, an organism exists until it has decomposed.)

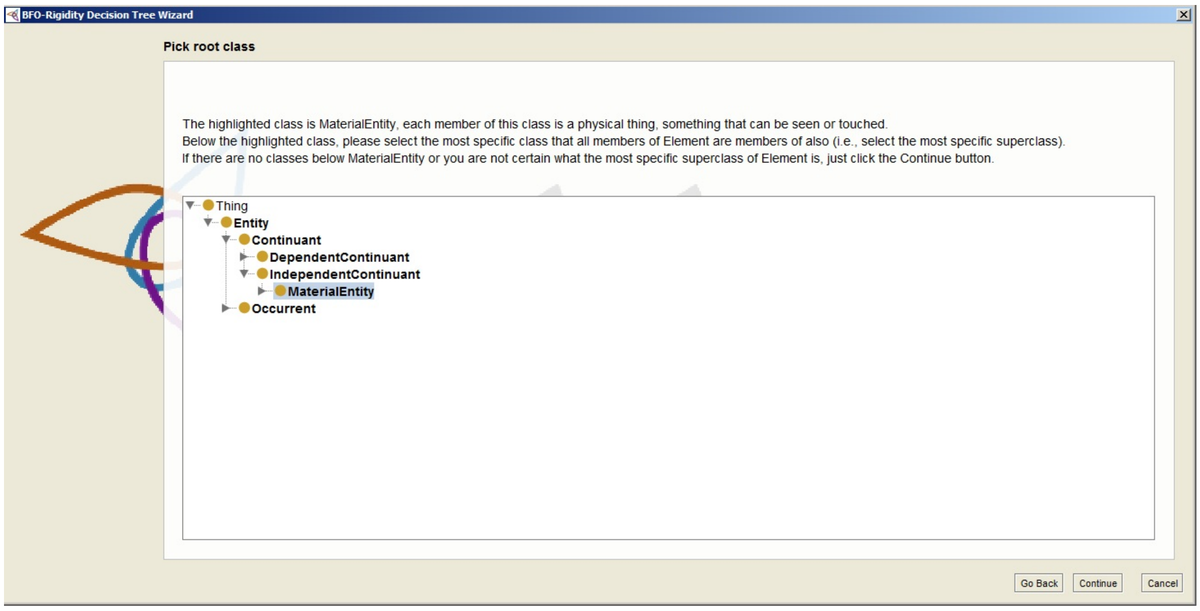
○ Yes

● No

Tips

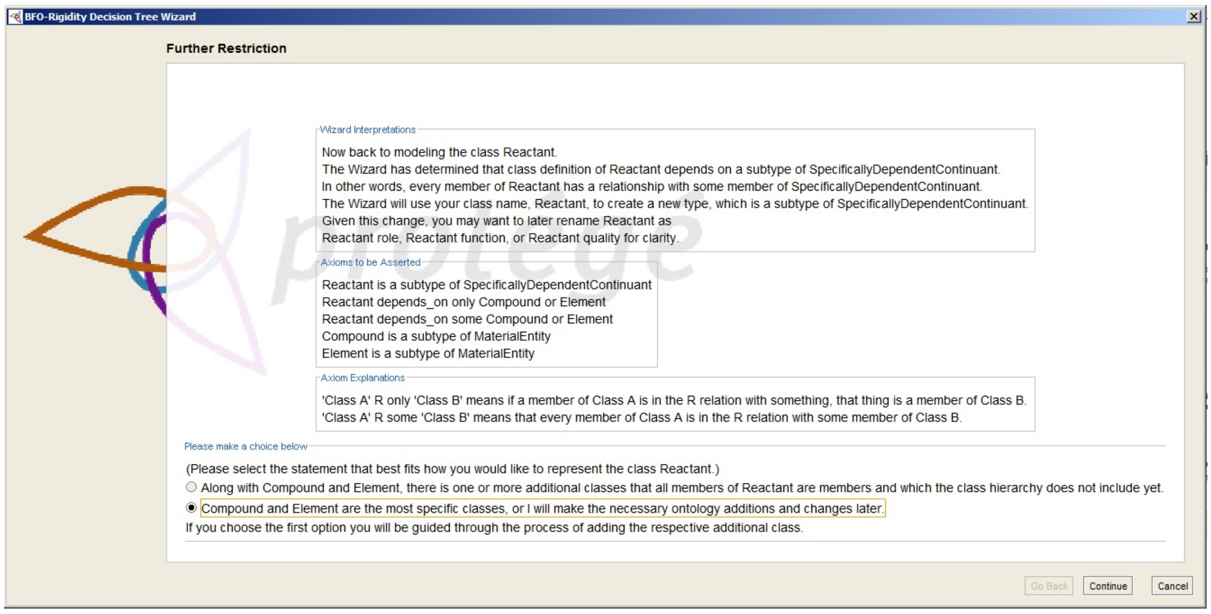
A person that is a member of class Student exists at a time they are not a student.

Figure H.8: Rigidity (All Members) Screen for Element



The highlighted class is MaterialEntity, each member of this class is a physical thing, something that can be seen or touched.
Below the highlighted class, please select the most specific class that all members of Element are members of also (i.e., select the most specific superclass).
If there are no classes below MaterialEntity or you are not certain what the most specific superclass of Element is, just click the Continue button.

Figure H.9: Pick Root Class Screen for Element



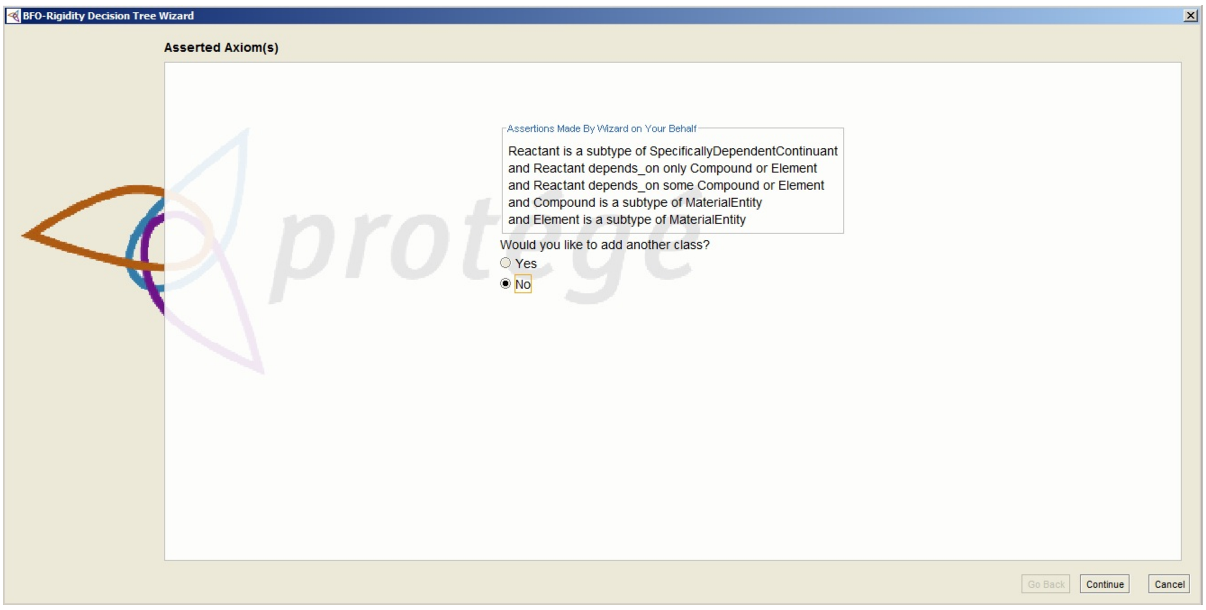
Now back to modeling the class Reactant.
The Wizard has determined that class definition of Reactant depends on a subtype of SpecificallyDependentContinuant.
In other words, every member of Reactant has a relationship with some member of SpecificallyDependentContinuant.
The Wizard will use your class name, Reactant, to create a new type, which is a subtype of SpecificallyDependentContinuant.
Given this change, you may want to later rename Reactant as
Reactant role, Reactant function, or Reactant quality for clarity.

Reactant is a subtype of SpecificallyDependentContinuant
Reactant depends_on only Compound or Element
Reactant depends_on some Compound or Element
Compound is a subtype of MaterialEntity
Element is a subtype of MaterialEntity

'Class A' R only 'Class B' means if a member of Class A is in the R relation with something, that thing is a member of Class B.
'Class A' R some 'Class B' means that every member of Class A is in the R relation with some member of Class B.

(Please select the statement that best fits how you would like to represent the class Reactant.)
○ Along with Compound and Element, there is one or more additional classes that all members of Reactant are members and which the class hierarchy does not include yet.
[✓ Compound and Element are the most specific classes, or I will make the necessary ontology additions and changes later.]
If you choose the first option you will be guided through the process of adding the respective additional class.

Figure H.10: Further Restriction Screen (3) for Reactant



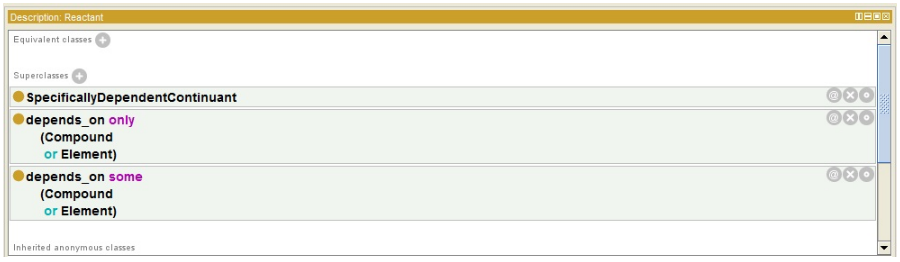
Asserted Axiom(s)

Assertions Made By Wizard on Your Behalf

Reactant is a subtype of SpecificallyDependentContinuant
and Reactant depends_on only Compound or Element
and Reactant depends_on some Compound or Element
and Compound is a subtype of MaterialEntity
and Element is a subtype of MaterialEntity

Would you like to add another class?
Yes [✓ No]

Figure H.11: Asserted Axiom(s) Screen for Reactant



SpecificallyDependentContinuant

depends_on only (Compound or Element)

depends_on some (Compound or Element)

Figure H.12: Description Screen for Reactant

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

Andersen, W. and Menzel, C. (2004). Modal Rigidity in the OntoClean Methodology. In Varzi, A. C. and Vieu, L., editors, *Formal Ontology in Information Systems*, pages 119–127, Amsterdam. IOS Press.

Armstrong, D. (1980). *A Theory of Universals: Volume 2: Universals and Scientific Realism*. Cambridge University Press, Cambridge; New York.

Arp, R. and Smith, B. (2008). *Nature Proceedings*, <http://hdl.handle.net/10101/npre.2008.1941.1>.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, Cambridge ; New York.

Barnett, D. (2004). Some Stuffs Are Not Sums of Stuff. *The Philosophical Review*, 113(1):89.

Bittner, T. and Donnelly, M. (2004). The Mereology of Stages and Persistent Entities. In de Mdmaras, R. L. and Saitta, L., editors, *In Proceedings of the 16th European Conference on Artificial Intelligence*, pages 283–287, Valencia. IOS Press.

Bittner, T. and Donnelly, M. (2005). Computational ontologies of parthood, componenthood, and containment. In Kaelbling, L. P. and Saffiotti, A., editors, *International Joint Conference on Artificial Intelligence*, volume 19, page 382.

Bittner, T. and Donnelly, M. (2007). A Temporal Mereology for Distinguishing between Integral Objects and Portions of Stuff. In Holte, R. and Howe, A., editors, *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 287–292, Vancouver. AAAI Press.

Brachman, R. J., McGuinness, D. L., Patel-Schneider, P. F., Resnick, L. A., and Borgida, A. (1991). Living with CLASSIC: When and how to use a KL-ONE-like language. In Sowa, J., editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann.

Brand, M. (1977). Identity Conditions for Events. *American Philosophical Quarterly*, 14(4):329–337.

Bunge, M. (1979). *A World of Systems.* D. Reidel, Dordrecht.

Campbell, N. A. and Reece, J. B. (2007). *Biology: AP Edition.* Pearson, Saddle River, New Jersey, 8th edition.

Carrara, M. (2004). Identity and Modality in OntoClean. *Applied Ontology*, 1(1):128–139.

Cocchiarella, N. B. (2001). Logic and Ontology. *Axiomathes*, 12:117–150.

Corcho, O., Fernández-López, M., Pérez, A. d. M. G., and Vicente, O. (2002). WebODE: An Integrated Workbench for Ontology Representation, Reasoning, and Exchange. In Gómez-Pérez, A. and Benjamins, V. R., editors, *EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, pages 138–153, London, UK. Springer-Verlag.

Dumontier, M. and Hoehndorf, R. (2010). Realism for Scientific Ontologies. In Galton, A. and Mizoguchi, R., editors, *International Conference on Formal Ontology in Information Systems*, pages 387–399, Toronto. IOS Press.

Fernandez, M., Gómez-Pérez, A., and Juristo, N. (1997). Methontology: from Ontological Art towards Ontological Engineering. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, pages 33–40, Stanford, CA. American Association for Artificial Intelligence.

Fernandez-Lopez, M., Gómez-Pérez, A., and Informatica, F. (2002). The integration of OntoClean in WebODE. In Gmez-Prez, A. and Benjamins, V. R., editors, *In Proc. of the EON2002 Workshop at 13th EKAW*, Siguenza, Spain.

Frege, G. (1950). *The Foundations of Arithmetic.* Blackwell, Oxford.

Gangemi, A., Guarino, N., Masolo, C., and Oltramari, A. (2001). Understanding top-level ontological distinctions. In A. Gómez Pérez, M Gruninger, H. S. and Uschold, M., editors, *Proceedings of the 2001 IJCAI Workshop on Ontologies and Information Sharing.*

Gómez-Pérez, A. (1995). Some ideas and examples to evaluate ontologies. In *Proceedings of the 11th Conference on Artificial Intelligence for Applications*, pages 299–305, Los Angeles. IEEE.

Gómez-Pérez, A., Corcho, O., and Fernandez-Lopez, M. (2004). *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition (Advanced Information and Knowledge Processing).* Springer, London.

Grenon, P. (2003a). BFO in a Nutshell: A Bi-categorial Axiomatization of BFO and Comparision with DOLCE. Technical report, Universitat Leipzig, Faculty of Medicine, Institute for Formal Ontology and Medical Information Science (IFOMIS).

Grenon, P. (2003b). Nuts in BFO's Nutshell: Revisions to the Bi-categorical Axiomatization of BFO. Technical report, Universitat Leipzig, Faculty of Medicine, Institute for Formal Ontology and Medical Information Science (IFOMIS).

Grenon, P. (2003c). Spatio-temporality in Basic Formal Ontology: SNAP and SPAN, Upper-Level Ontology, and Formalization. Technical report, Universitat Leipzig, Faculty of Medicine, Institute for Formal Ontology and Medical Information Science (IFOMIS).

Guarino, N. (1998). Formal Ontology and Information Systems. In Angele, J. and Sure, Y., editors, *International Conference on Formal Ontology in Information Systems*, Trento, Italy. IOS Press.

Guarino, N. and Welty, C. (2000a). A Formal Ontology of Properties. In Dieng, R. and Corby, O., editors, *Proceedings of 12th International Conference on Knowledge Engineering and Knowledge Management*, Berlin. Springer Verlag.

Guarino, N. and Welty, C. (2000b). Identity, Unity, and Individuality: Towards a Formal Toolkit for Ontological Analysis. In Horn, W., editor, *Proceedings of the European Conference on Artificial Intelligence*, pages 219–223, Amsterdam. IOS Press.

Guarino, N. and Welty, C. (2001). Identity and Subsumption. In Green, R., Bean, C. A., and Hyon, editors, *The Semantics of Relationships: an Interdisciplinary Perspective*, pages 111–126, Dordrecht, The Netherlands. Kluwer.

Guarino, N. and Welty, C. (2002). Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65.

Guarino, N. and Welty, C. (2009). An overview of OntoClean. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 201–220. Springer, Berlin, 2nd edition.

Guarino, N. and Welty, C. A. (2004). An overview of OntoClean. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, pages 151–159, Berlin. Springer Verlag.

Hirst, G. (1991). Existence assumptions in knowledge representation. *Artificial Intelligence*, 49(1-3):199–242.

Hughes, G. E. and Cresswell, M. J. (1996). *A New Introduction to Modal Logic*. Routledge, London; New York.

Jacobs, W. (1994). Caulerpa. *Scientific American*, 271(6):100–105.

Kant, I. (1933). *Critique of Pure Reason, trans.* Macmillan, London.

Kaplan, A. N. (2001). Towards a Consistent Logical Framework for Ontological analysis. In Welty, C. and Smith, B., editors, *International Conference on Formal Ontology in Information Systems*, pages 244–255, New York, NY. ACM.

Kifer, M., Lausen, G., and Wu, J. (1990). Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42:741–843.

Knublauch, H., Fergerson, R. W., Noy, N. F., and Musen, M. A. (2004). The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In Sheila A. McIlraith, Dimitris Plexousakis, F. v. H., editor, *3rd International Semantic Web Conference*, pages 229–243, New York. Springer Verlag.

Lakoff, G. (1990). *Women, Fire, and Dangerous Things*. Univ. of Chicago Press, Chicago; London.

Little, E. and Vizenor, L. (2006). Principles for the development of upper ontologies in higher-level information fusion applications. In Bennett, B. and Fellbaum, C., editors, *Formal Ontology in Information Systems*, pages 309–320, Amsterdam.

Lowe, E. J. (1989a). *Kinds of Being: A Study of Individuation, Identity, and the Logic of Sortal Terms*. Blackwell Publishers, Oxford.

Lowe, E. J. (1989b). What is a Criterion of Identity? *The Philosophical Quarterly*, 39(154):1–21.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. (2003). WonderWeb deliverable D18 ontology library. Technical report, IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web.

Meinong, A. (1904). Über Gegenstandstheorie. *Alexius Meinong Gesamtausgabe*, 2:481–535.

Mungall, C. (2007). OBO Relation Ontology. `http://www.obofoundry.org/ro/`.

Neuhaus, F. and Smith, B. (2008). Modelling Principles and Methodologies–Relations in Anatomical Ontologies. In Albert Burger, D. D. and Baldock, R., editors, *Anatomy Ontologies for Bioinformatics*, pages 289–306, London. Springer.

Oltramari, A., Oltramari, R., Gangemi, A., Guarino, N., and Masolo, C. (2002). Restructuring WordNet's top-level: The OntoClean approach. In Rodrguez, M. G. and Araujo, C. P. S., editors, *Third International Conference on Language Resources and Evaluation*, pages 17–26, Las Palmas, Canary Islands. European Language Resources Association.

Oltramari, A. and Vetere, G. (2008). Lexicon and Ontology Interplay in Senso Comune. In *OntoLex 2008 of the 6th International Conference on Language Resources and Evaluation*, page 24, Morocco. European Language Resources Association.

Parsons, T. (1980). *Non-Existent Objects*. Yale University Press, New Haven.

Phillips, G. J. (2001). Green fluorescent protein–a bright idea for the study of bacterial protein localization. *FEMS Microbiol Lett*, 204(1):9–18.

Quine, W. V. (1981). What Price Bivalence. *Journal of Philosophy*, 78(2):90–95.

Quinton, A. (1957). Properties and classes. In *Proceedings of the Aristotelian Society*, volume 58, pages 33–58, London. Harrison & Sons, Ltd.

Randell, D., Cui, Z., and Cohn, A. (1992). A Spatial Logic based on Regions and Connection. In *Third International Conference on Knowledge Representation and Reasoning*, volume 92, pages 165–176, Cambridge. Morgan Kaufmann.

Rapaport, W. (1978). Meinongian theories and a Russellian Paradox. *Noûs*, 12(2):153–180.

Rector, A. (2003). Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL. In *Third International Conference on Knowledge Capture*, volume 23, page 25, Banff.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., and Wroe, C. (2004). OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. pages 63–81. Springer.

Rector, A., Rogers, J., and Bittner, T. (2006). Granularity, scale and collectivity: When size does and does not matter. *Journal of Biomedical Informatics*, 39(3):333–349.

Russell, B. (1905). On Denoting. *Mind*, 14(56):479–493.

Ruttenberg, A. (2009). From Basic Formal Ontology to the Information Artifact Ontology. Presented at the International Conference on Biomedical Ontologies, Buffalo, NY.

Saiki, R. K., Gelfand, D. H., Stoffel, S., Scharf, S. J., Higuchi, R., Horn, G. T., Mullis, K. B., and Erlich, H. A. (1988). Primer-directed enzymatic amplification of dna with a thermostable dna polymerase. *Science*, 239(4839):487–91.

Schlick, M. (1965). About the concept of wholeness. *Reprinted in Logic of the Social Sciences*, pages 213–224.

Schulz, S., Hanser, S., Hahn, U., and Rogers, J. (2006). The Semantics of Procedures and Diseases in SNOMED CT. *Methods of Information in Medicine*, 45(4):354–8.

Simon, J., Dos Santos, M., Fielding, J., and Smith, B. (2006). Formal ontology for natural language processing and the integration of biomedical databases. *International Journal of Medical Informatics*, 75(3-4):224–231.

Simon, J. and Smith, B. (2004). Using philosophy to improve the coherence and interoperability of applications ontologies: a field report on the collaboration of ifomis and l&c.

Simons, P. (1987). *Parts: A Study in Ontology.* Clarendon Press, Oxford.

Smith, B. (1998). The basic tools of formal ontology. In *Formal Ontology in Information Systems (Frontiers in Artificial Intelligence and Applications*, pages 19–28, Amsterdam, Oxford, Tokyo, Washington, DC. IOS Press.

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

Smith, B. (2005). Against Fantology. *Experience and Analysis*, 34:153–172.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., Scheuermann, R. H., Shah, N., Whetzel, P. L., and Lewis, S. (2007). The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. volume 25, pages 1251–1255. Nature Publishing Group.

Smith, B. and Ceusters, W. (2010). Ontological realism: A methodology for coordinated evolution of scientific ontologies. *Applied Ontology*, 5(3):139–188.

Smith, B., Ceusters, W., Klagges, B., Köhler, J., Kumar, A., Lomax, J., Mungall, C., Neuhaus, F., Rector, A. L., and Rosse, C. (2005). Relations in biomedical ontologies. *Genome Biol*, 6(5).

Smith, B., Kusnierczyk, W., Schober, D., and Ceusters, W. (2006). Towards a reference terminology for ontology research and development in the biomedical domain. In *KR-MED 2006, Formal Biomedical Knowledge Representation, Proceedings of the Second International Workshop on Formal Biomedical Knowledge Representation: "Biomedical Ontology in Action"*, volume 222 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Smith, B. and Rosse, C. (2004). The role of foundational relations in the alignment of biomedical ontologies. *Studies in Health Technology and Informatics*, 107(Pt 1):444–8.

Spear, A. (2007). Ontology for the Twenty First Century: An Introduction with Recommendations. Technical report, University at Buffalo.

Strawson, P. F. (1959). *Individuals: An Essay in Descriptive Metaphysics*. Methuen, London.

Sure, Y., Angele, J., and Staab, S. (2003). OntoEdit: multifaceted inferencing for ontology engineering. *Journal on Data Semantics*, 2800:2003.

The Gene Ontology Consortium (2008). The gene ontology project in 2008. *Nucleic acids research*, 36(Database issue).

Welty, C. (2006). OntOWLClean: Cleaning owl ontologies with OWL. In *Proceedings of FOIS-2006*, pages 347–359. IOS Press.

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

Welty, C. and Guarino, N. (2001). Supporting ontological analysis of taxonomic relationships. *Data Knowledge Engineering*, 39(1):51–74.

Whitehead, A. N. and Russell, B. (1957). *Principia Mathematica*, volume 3. Cambridge University Press, Cambridge.

Williamson, T. (1990). *Identity and Discrimination*. Blackwell, Oxford.

Zemach, E. (1970). Four ontologies. *The Journal of Philosophy*, 67(8):231–247.

Zimmerman, D. (1995). Theories of masses and problems of constitution. *The philosophical review*, 104(1):53–110.