# Instances of Instances Modeled via Higher-Order Classes

douglas foxvog

Digital Enterprise Research Institute (DERI), National University of Ireland, Galway, Ireland

**Abstract.** In many languages used for expressing ontologies a strict division between classes and instances of classes is enforced. Other languages permit instances of instances without end. In both cases, ontological meaning is often mistakenly assumed from these purely syntactic features. A rigorous set of definitions for different levels of classes is presented to enable concepts to be unambiguously defined.

## 1 Introduction

Ontology languages normally distinguish types of objects (also called class, concept, collection, ...) from things that are not types (individuals, instances, ...). An issue that has often been recognized is that types of classes exist, thus making instances of these meta-classes classes themselves. Some systems address this issue by disallowing meta-classes, others allow meta-classes without restriction, and others do not distinguish between classes and individuals. In order to provide for rigor in this field, an ontology of levels of meta-classes was created for the Cyc Project and is presented here.

Section 2 presents terminology used in this paper; Section 3 presents related work; Section 4 presents the system of levels of meta-classes; Section 5 describes object ontology order; and Section 6 provides some rules concerning classes of various levels.

## 2 Terminology

In this document, the word **class** is used to denote a type for which instances are permissible in an ontology, whether or not the instances are actually included in the ontology. Various ontology languages use the words "concept", "type", or "collection" for this meaning or a subset of this meaning. Outside the field of computerized ontologies the word "category" is also used. Class is distinct from group in that a class has no physical properties, while a group of objects may have mass, temporal duration, location, ownership, and other such properties[1]. Class is distinct from set in that a class may have different instances at different

[1] Note that groups may have properties in addition to those derived from their members: a group of people may own a piece of land even though none of its members individually owns even a piece of the land.

times or in different contexts[2], while a set's elements are fixed[3]. Different classes may have the same extent (including an empty extent) in a given context, but be different classes, while two sets with the same members are the same set. A class's instances typically have one or more features or attributes in common, while a set may have arbitrarily selected members - however, ontology languages typically also allow for extensionally defined classes. A class is similar to an intensionally described set (one whose members are specified by rules as opposed to by a recounting of its members) and the distinction between the two concepts is not made in some ontology languages in which the context for the description is constant.

The term **individual** is used here to represent any component of an ontology that is an element or instance of a class but is not itself a class. Certain ontology components, such as datatype values (e.g. numbers and character strings), relations, and functions are included as "individuals" (and types of them as classes) in some systems but not in others. In systems that do not permit a class to be an instance of another class, the word "instance" is frequently used for all elements of a class, even if the real-world referent is a type, such as a dish offered on a menu.

The word "instance" is also used for all instances of classes in systems that allow meta-classes but do not require every element of the ontology to be an instance of some class. In systems in which every term is an instance of some class, the word is not used in an unqualified manner. The term **instance** is used in this document only to refer to the relationship which one element of an ontology has with respect to a class in the same ontology.

The term **meta-class** is defined as the class of all classes each of whose instances is necessarily a class. Since each instance of meta-class is by definition a class, meta-class is an instance of itself. An ontology of meta-classes is presented in Section 4, with the terms being defined as they are presented.

Two classes are **disjoint** if and only if they necessarily have no elements in common.

## 3 Related work

### 3.1 Russell's Simple Theory of Types

Russell's Simple Theory of Types [13], developed in the early 20th Century, defines and describes different categories of sets, starting with sets of individuals, sets of sets of individuals, and so on. Allowing sets of more than one level to be

[2] A context is a domain of applicability within which a set of statements are true, for example the point of view of a political party, the background situation of a work of fiction, or a hypothetical situation. For a detailed description of contexts see [9].

[3] A computer variable which refers to a set may reference a different set at different times. The sets themselves do not change.

members of the same set opened the way for the paradoxical "set of all sets that do not contain themselves", and so were disallowed.

Russell and other philosophers developed numerous variations of the original Simple Theory of Types.

Russell (for obvious reasons) did not apply his various levels or orders of sets to computerized ontologies.

### 3.2 KIF

KIF [6] distinguishes class and individual and allows for classes of classes. KIF does not further distinguish different types of meta-classes. In order to avoid Russell's paradox, KIF does not permit a class to have an unbounded element, with a bounded element defined as either a non-set or a set all of whose elements are bounded. Using this definition, no class may contain itself, because any such class would be unbounded.

### 3.3 UML

UML has a four-level architecture in which each component at one level is an instance of exactly one component at the next highest level [1]. Each component at the highest level is an instance of some component (possibly itself) at this same level. As such, each level is used to define the next lower level, with the lowest level (M0) being the "user object" level, the next level (M1) being the "user model" level, the next level (M2) being the "metamodel" or ontology language level used for defining terms at the M1 level, and the highest level (M3) being the "metametamodel" level used for defining the ontology language.

Some items may appear at different levels and others may be instances of components at different levels. For example, `Class` is a component of both M2 and M3, and `HenryThoreau` from the M0 level is an instance of both `Person` at the M1 level and `Instance` at the M2 level.

Since only items at the lowest two levels are part of the user-created ontology, user ontologies can not contain concepts which are classes of classes.

### 3.4 RDFS(FA)

Pan and Horrocks argue in their proposal for RDFS(FA) (RDFS with Fixed metamodeling Architecture) that at least two levels of class are needed and relay a finding that "in practice, it has not been found useful to have more than two class primitives in the metamodeling architecture," so they find it reasonable to define only two class primitives [12]. They do not consider a possible desire by the ontology builder to use multiple levels of classes in the user ontology.

### 3.5 Other Ontology Languages

Other ontology languages either do not distinguish individuals from classes, allowing any instance of a class to have its own instances (e.g., F-Logic [2], RDF-S [8], TRIPLE [14]), or do not permit classes of classes (e.g., DAML+OIL [16], OWL [11], OXML [5], Power-Loom [3], WSML [4]).

## 4 Ontology of Levels of Meta-Classes

Although the Cyc Project [7] had distinguished meta-classes for years, in 2002, the lack of an ontology of levels of meta-classes became obvious to ontological architects at Cycorp. Since every constant in CycL is an instance of some class, the following system was designed (using the term "`Collection`" instead of "`Class`" in conformance with standard CycL terminology) to provide clean definitions and to prevent occasional misuse of meta-classes during initial ontologization of a new area.

Classes are distinguished by their "order" - the number of iterations of instantiations that are necessary in order to obtain individuals. The order of empty intentionally-defined classes can be determined from their defining rule.

### 4.1 First-Order Class

First-Order Class is defined as the meta-class of all subclasses of Individual. Each instance of First-Order Class is a class, each of whose instances is necessarily an individual. This is the most common type of class, having instances such as `Person`, `Computer`, and `Ontology`. `Hobbit` would also be an instance of First-Order Class since, although it has no instances in the real world, in the context of *The Lord of the Rings* [15] it has many instances, all of which are necessarily individuals. The class `Individual` is an instance of First-Order Class since, by definition, all of its instances are individuals.

### 4.2 Second-Order Class

Second-Order Class is defined as the meta-class of all subclasses of First-Order Class. Each instance of Second-Order Class is a class, each of whose instances is a First-Order Class. Typical Second-Order classes include `Car-Brand` (with instances such as `VolkswagenCar` and `HondaCar`), `AnimalSpecies` (with instances such as `GreyWolf` and `Dodo`), `Occupation`, and `USArmyRank`. First-Order Class is an instance of Second-Order Class since, by definition, all of its instances are First-Order classes.

### 4.3 Third-Order Class

Third-Order Class is defined as the meta-class of all subclasses of Second-Order Class. Each instance of Third-Order Class is a class, each of whose instances is a Second-Order Class. Instances of Third-Order Class are rare in OpenCyc. A few typical instances are `BiologicalTaxonType`, having instances such as `BiologicalSpecies` and `BiologicalGenus`, and `MilitaryRankType`, having instances such as `USArmyRank` and `RussianArmyRank`. Second-Order Class is an instance of Third-Order class since, by definition, all of its instances are Second-Order classes.

### 4.4 Fourth- and Higher Order Classes

Fourth-Order Class is defined as the meta-class of all subclasses of Third-Order Class. Higher order meta-classes could be similarly defined; however, the utility of implementing such classes is questionable. The only Fourth-Order Class in OpenCyc [10] is Third-Order Class (called `ThirdOrderCollection`), which is an instance since, by definition, all of its instances are Third-Order classes. Similarly, Fourth-Order Class would likely become the only instance of Fifth-Order Class, and so on, ad infinitum.

### 4.5 Fixed-Order Class

First-Order Class, Second-Order Class, Third-Order Class, and Individual are mutually disjoint classes, which each have the property that every one of their instances has the same order, i.e. it takes a fixed number of iterations of instantiation to reach an Individual. [For a formal definition of ontology element order, see Section 5.]

Thus, every instance of `Individual` is an Individual - this could be considered zero-order. No instance of First-Order Class is an Individual, but every instance of every instance of it must be. No instance or instance of an instance of Second-Order Class is an Individual, but every instance of an instance of an instance of it must be. Third- and Fourth-Order Class are similarly one and two steps more removed.

Fixed-Order Class is defined as the meta-class of all classes with this property. First-Order Class, Second-Order Class, Third-Order Class, and Fourth-Order Class are not only instances of Fixed-Order Class; they are subclasses of it as well, which means that all of their instances are transitively instances of Fixed-Order Class. Individual is an instance, but not a subclass of Fixed-Order Class.

### 4.6 Variable-Order Class

Not every class is a Fixed-Order Class. Fixed-Order Class, itself, has classes of different orders as instances. Thus, it is an instance of Variable-Order Class.

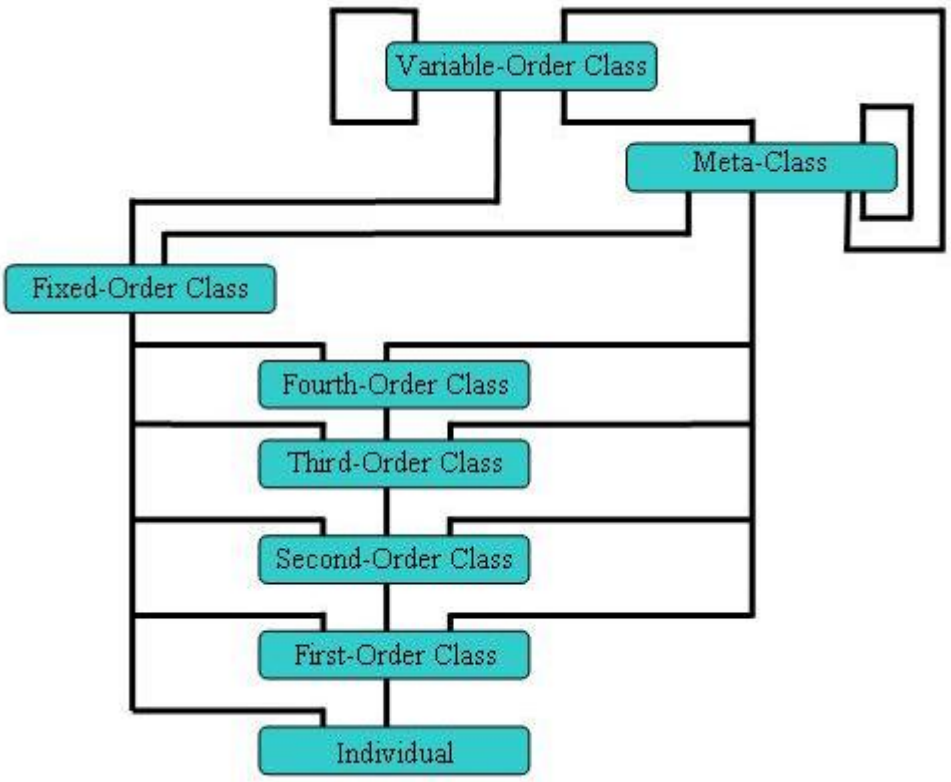


**Fig. 1.** Instance-Of Relations in Meta-Class Ontology

Variable-Order Class is defined at the class of all classes which (1) may have instances of more than one order, (2) are instances of themselves, or (3) have any instances which are variable-order classes. Every class is therefore either fixed-order or variable order, but not both.

Variable-Order Class is an instance of both itself and Meta-Class. Class and Meta-Class are instances of Variable-Order Class. The universal class (called `Thing` in CycL and RDF) is an instance of Variable-Order Class. Variable-Order Class is not a subclass of Meta-Class since some of its instances, e.g. `Thing`, have instances that are Individuals as well as instances that are Classes.

### 4.7 At Least Nth-Order Class

The class `Class` is equivalent to "at least first-order class." The class Meta-Class is equivalent to "at least second-order class." Meta-Class Type, equivalent to "at least third-order class," has been created in OpenCyc (as `CollectionTypeType`), but is little used. At-least Nth-Order class for N higher than three has not been found useful, no instances having been ontologized other than Third-Order Class and Fourth-Order Class.

### 4.8 At Most Nth-Order Class

We found no need for a class such as "at-most second-order class", although such classes were considered. Such a class would have been a subclass of Fixed-Order Class in that none of its instances would be variable order classes. It would have been an instance of Variable Order Class.

First-Order class would be equivalent to "at-most first-order class, and thus creation of such a class would be redundant. "At-most third order class" would include all classes except Fourth-Order Class, Third-Order Class, Fixed-Order Class, and those classes which include variable-order classes as instances, and was not deemed useful.

### 4.9 Self-Including Class

Although Cyc has several classes that include themselves as instances, a meta-class of such classes was not created. This class would be an instance and subclass of Variable-Order Class. Classes in OpenCyc that would be instances of this class include `Thing`, `VariableOrderCollection`, `CollectionType` (meta-class), and `CollectionTypeType` (meta-class type).

However, since the class of all classes that are not instances of this class is paradoxically both an instance of this class and not (see Russell's paradox), this class is also problematic and was not included in the ontology. [4]

### 4.10 Self-Excluding Class

This class corresponds to the paradoxical set referred to in Russell's Paradox which is neither an member of itself nor not a member of itself. It was not created as part of this ontology.

[4] Note that if this class existed, it would be an element of itself, the argument being reductio ad absurdum:

- Define Self-Including Class (`SIC`) as being the class that has as instances those, and only those, classes which have themselves as instances.
- Assume this class is not an element of itself.
- Add to that class the single class necessary to generate a class that contained itself. E.g., if `SIC` were {`Thing`, `Meta-Class`, `VariableOrderClass`, ...}, `SIC2` would be {`SIC2`, `Thing`, `Meta-Class`, `VariableOrderClass`, ...}.
- This new class would contain all the self-containing classes (including itself) and only self-containing classes.
- Thus, this new class would be the Self-Including Class and our original assumption (that Self-Including Class does not contain itself) is proved wrong.

The third step in this argument would not be permitted for sets in modern Set Theory – another reason to shun the creation of such a class, even though classes are not sets.

## 5 Ontology Element Order

Let Individuals be zeroth-order elements of an ontology. A class is defined as being of the Nth order if every one of its instances is necessarily of (N-1)st order. Any class that cannot be assigned an order under this rule is deemed a variable-order class.

Note that empty classes may still have an order. The class `Dodo` is first-order in 2005 even though there have been no instances of the class for hundreds of years. Similarly, the class `AnimalSpecies` was second-order a billion years ago even though there were no animals at the time.

## 6 Rules

The following rules relating to class order hold:

- Any instance of a First-Order Class is an Individual.
- Any instance of an Nth-Order Class (for N > 1) is an N-1st Order Class.
- Any instance of Nth-Order Class (for N > 1) is a subclass of N-1st Order Class.
- Every Class is either a Fixed-Order Class or a Variable-Order Class.
- No Fixed-Order Class has an instance of the same order as itself.
- If a Variable Order Class has an Nth-Order Class as an instance, it also has an instance that is not an Nth-Order Class (in some context, even if not in the present context).
- If N does not equal M, Nth-Order class and Mth-Order class are disjoint.
- Meta-Class is disjoint with First-Order Class.
- Meta-Class is disjoint with Individual.

## 7 Acknowledgment

This work was developed as part of the Cyc project at Cycorp of Austin, Texas.

## References

1. Álvarez, J. et al.: MML and the Metamodel Architecture. WTUML: Workshop on Transformation in UML (2001).
2. Balaban, M.: The F-Logic Approach for Description Languages. Annals of Mathematics and Artificial Intelligence **15** (1995) 19–60
3. Chalupsky, H., et al.: PowerLoom Manual. Information Sciences Institute, USC. Downloaded from http://www.isi.edu/isd/LOOM/PowerLoom/documentation/manual/manual.pdf 15/5/2005.
4. de Bruijn, J., et al.: The Web Service Modeling Language WSML. DERI Technical Report D16.1v0.2 (2005).
5. Erdmann, M.: OXML 2.0 Reference manual for users and developers of OntoEdit's XML-based Ontology Representation language, Version 0.92. (2001). Downloaded from http://www.cs.ait.ac.th/ waralak/oxml2.0.pdf 15/5/2005.

6. Genesereth, Michael R., Fikes, R. E.: Knowledge Interchange Format Volume 3.0 Reference Manual (1992).
7. Lenat, D.: Mapping Ontologies into Cyc. Ontologies and the Semantic Web, Technical Report WS-02-11. AAAI Press. (2002) 1–6.
8. Manola, F., Miller, E.: RDF Primer. (2004) Downloaded from http://www.w3.org/TR/rdf-primer 15/5/2005.
9. McCarthy, J.: Notes on Formalizing Context. IJCAI '93. (1993) 555–560.
10. OpenCyc: OpenCyc website. http://www.opencyc.org, downloaded 15/5/2005.
11. Patel-Schneider, P. F., et al., eds.: OWL Web Ontology Language: Semantics and Abstract Syntax. Downloaded from http://www.w3.org/TR/2004/REC-owl-semantics-20040210/ on 15/5/2005 (2004).
12. Pan, J. Z., Horrocks, I.: Metamodeling architecture of web ontology languages. Proceedings of the Semantic Web Working Symposium. (2001) 131–149.
13. Russell, B.: Principia Mathematica (1910).
14. Sintek, M., Decker, S.: TRIPLE - An RDF Query, Inference, and Transformation Language. DDLP'2001, Japan (2001).
15. Tolkein, J.R.R.: The Lord of the Rings (1954).
16. van Harmelen, F.; et al.: Reference description of the DAML+OIL (March 2001) ontology markup language. Downloaded from http://www.daml.org/2001/03/reference 15/5/2005 (2001).