EVALUATING ONTOLOGICAL DECISIONS *with* ONTOCLEAN

Exposing common misuses of the subsumption relationship and the formal basis for why they are wrong.

THE PROCESS OF BUILDING OR engineering ontologies[1] for use in information systems remains an arcane art form, which must become a rigorous engineering discipline. One of the principal requirements for developing a true ontology engineering practice is a general, domain-independent methodology that provides guidance not only on what kinds of ontological decisions need to be made, but on how these decisions can be evaluated.

The OntoClean methodology has been under development for a few years, and seeks to provide this kind of guidance based on highly general ontological notions drawn from philosophical ontology, especially what is now called "analytic metaphysics." Among other things, we use our methodology to validate taxonomies by exposing inappropriate and inconsistent modeling choices.

The OntoClean methodology is based on formal notions, which are general enough to be used in any ontology effort, independently of a particular domain. We use these notions to define a set of metaproperties which, in turn, are used to characterize relevant aspects of the intended meaning of the properties, classes, and relations that make up an ontology. In addition, the metaproperties impose several constraints on the taxonomic structure of an ontology, which help in evaluating the choices made.

Essence and Rigidity

The first formal notion we will discuss is *essence*. A property of an entity is essential to that entity if it must hold for it. This is a stronger notion than one of permanence, that is, a property of an entity is not essential if it just happens to be true of it, accidentally, for all time. For example, consider the property of *being hard*. We may say that it is an essential property of hammers, but not of sponges. Some sponges (dry ones) are hard, and some

[1] The computer science use of the term "ontology" has undergone some evolution since it was borrowed from philosophy by John McCarthy in the late 1970s, and today—as a subject area—it is normally taken as nearly synonymous with knowledge engineering in AI, conceptual modeling in databases, and domain modeling in OO design. We believe it is important, in light of this shift in meaning, to maintain that "ontology" is not simply a new word for something computer scientists have been doing for 20–30 years; ontology is hundreds, if not thousands, of years old, and there are many lessons learned in those centuries that we may borrow from philosophy along with the term.

NICOLA GUARINO AND CHRISTOPHER WELTY

particular sponge may be hard for its entire existence, however this does not make being hard an essential property of that sponge. The fact is that it could have been soft at some time, it just happened that it never was.

A special form of essence is rigidity: a property is rigid if it is essential to all its instances. For example, *being a person* is usually conceptualized as rigid, while we have seen that *being hard* is not. Rigidity is a subtle notion: every entity that can exhibit the property must exhibit it. So, every entity that is a person must be a person, and there are no entities that can be a person but aren't. Notice that these definitions are restricted to meaningful properties (not necessarily true nor necessarily false), so trivial cases are excluded.

Obviously there are also properties that are not essential to all their instances. Of these we distinguish properties that are essential to some entities and not essential to others (semi-rigid) from properties that are never essential (anti-rigid). For example, the property *being a student* is typically anti-rigid—every instance of student is not essentially a student (may also be a non-student), whereas the property *being hard* is semi-rigid, since there are instances (hammers) that must be hard and instances (sponges) that may be hard but also may not.

Rigidity is an important notion, every property in an ontology should be labeled as rigid, non-rigid, or anti-rigid. In addition to providing more information about what a property is intended to mean, these metaproperties impose constraints on the subsumption relation, which can be used to check the ontological consistency of taxonomic links. One of these constraints is that anti-rigid properties cannot subsume rigid properties. For example, the class *student* cannot subsume the class *person* if the former is anti-rigid and the latter is rigid. To see this, consider that every instance of student may cease being a student, while no instance of person can cease being a person. If all persons are necessarily students (the meaning of subsumption), this would create an inconsistency.

Identity and Unity

Although very subtle and difficult to explain without experience, identity and unity are the most important philosophical notions we use in our methodology. They are different notions, although strictly related and often confused with each other. In general, identity refers to the problem of being able to recognize individual entities in the world as being the same (or different), and unity refers to being able to recognize all the parts that form an individual entity.

What are identity criteria? They are difficult for even experienced conceptual modelers to appreciate because they are typically not part of the implemented system and are overlooked. In point of fact, identity criteria are conditions used to determine equality (sufficient conditions) and that are entailed by equality (necessary conditions).

It is perhaps simplest to think of identity criteria over time. For example, how do we recognize a person we know as the same person even though they may have changed? It is also very informative, however, to think of identity criteria at a single point in time. This may, at first glance, seem bizarre. How can you ask, "are these two entities the same entity?" If they are the same then there is one entity, it does not even make sense to ask the question.

The answer is not that difficult. One of the most common decisions that must be made in ontological analysis concerns identifying circumstances in which something that is apparently seen as one entity is actually two (or more). Consider the following example, drawn from actual experience: a proposed class *time duration* whose instances are things like "one hour" and "two hours," and a class *time interval* referring to specific intervals of time, such as "1:00–2:00 next Tuesday" and "2:00–3:00 next Wednesday." One proposal was to make *time interval* a kind of (subclass of) *time duration*, since all time intervals were seen as time durations. Seems to make intuitive sense, but how can we evaluate this decision?

In this case, an analysis based on the notion of identity can be very informative. According to the identity critera for time durations, two durations of the same length are the same duration. In other words, all one-hour time durations are identical—they are the same duration and therefore there is only one "one hour" time duration. On the other hand, according to the identity criteria for time intervals, two intervals occurring at the same time are the same, but two intervals occurring at different times, even if they are the same length, are different. Therefore, the two example intervals given would be different intervals, but the same duration. This creates a contradiction: if all instances of time interval are also instances of time duration (as implied by the subclass relationship), how can they be two instances under one class and a single instance under another?

This is one of the common confusions of natural language when used in describing the world. When we

CAREN ROSENBLATT

say "all time intervals are time durations" we really mean "all time intervals have a time duration"; the duration is a component of an interval, but it is not the interval itself. Therefore, we cannot model the relationship as subclass. More examples of such confusions are provided later.

A second notion extremely useful in ontological analysis is unity. Unity refers to the problem of describing the way the parts of an object are bound together, such that we know in general what is part of the object, what is not, and under what conditions the object is a whole.

Unity can tell us a lot about the intended meaning of properties or classes based on whether their instances are wholes. For some classes, all their instances are wholes, for others none of their instances are wholes. For example, the class "water," found in some commonsense ontologies, does not represent whole objects. An instance of this class is an amount of water, but it is not a whole, since it is not recognizable as an isolated entity. On the other hand, "ocean" may be a class that does represent whole objects, since an instance of this class, such as "the Atlantic Ocean," is recognizable as a single entity.

This leads us again to interesting problems with subsumption. It may make sense to say that "ocean" is a subclass of "water," since all oceans are water. However, if we claim that instances of the latter are not wholes, and instances of the former always are, then we have a contradiction. Problems like this again stem from the ambiguity of natural language, oceans are not "kinds of" water, they are composed of water.

In addition to specifying that a class represents wholes, it is also useful to analyze the specific conditions that must hold among the parts of a certain entity in order to consider it a whole. We call these conditions unity criteria, and distinguish with suitable metaproperties the classes that carry a common unity criterion for all their instances (such as "ocean") from those that do not (like "water").

The Benefits of Ontological Analysis

While the OntoClean methodology is still under development, the core is quite stable and is being used in several industrial and academic settings to validate taxonomies. The OntoClean core is based on attaching to each property (class) in an ontology suitable metaproperties that describe its behavior with respect to the ontological notions previously described. After analyzing each property, the modeler gains a number of important insights into the structure of the ontology and the nature of what is being represented.

One of the first benefits of this analysis is the identification of a backbone taxonomy. The backbone taxonomy consists of all the rigid properties in the ontology, organized according to their subsumption relationships, and represents a view of the ontology showing all the most important properties—those that cover the entire domain (or universe of discourse). This is because we assume every entity must have identity criteria and must have a rigid property that describes those criteria. The backbone taxonomy gives a jump start to the integration process, since every entity must instantiate at least one property in the backbone. Backbone properties are the most important to analyze first—those that represent the invariant, essential aspects of the domain. Moreover, if we have to integrate two different ontologies, we can start comparing their rigid properties, trying to establish a common backbone that will constitute a basic set of stable properties within the merged domain.

Another benefit of this analysis, and the subject of the rest of this article, is the discovery of inconsistent uses of subsumption in the taxonomy. Deciding whether one property should subsume another is one of the most important ontological decisions a modeler must make in building an ontology, and providing a formal foundation for evaluating these decisions has proved an important milestone in the practice of conceptual modeling.

Subsumption Misused

The subsumption (or subclass, is-a, and so forth) relation that is the basis of a taxonomy is an extremely useful tool for imparting structure on an ontology. It is by far the most commonly used structuring primitve, and in many cases, such as thesauri and object-oriented languages, it is the only one. It is, however, easily and often misused. Formally, we take it to mean that all instances of the subclass are necessarily instances of the superclass.

The OntoClean methodology provides a formal, consistent and straightforward way to explain some of the most common misunderstandings in conceptual modeling regarding the taxonomic or subsumption relation.

Instantiation

Perhaps the most confusing of all distinctions is between the two relations subsumption and instantiation. We have often found the subsumption relationship misused when instantiation was actually intended. The canonical example of this is species/animal. While most introductory courses teach the difference between classes, such as *Mammal* or *Human*, and instances, such as *Chris*, they stop short of explaining how second-order classes, such as *Species*, would fit into the picture. *Human* is a subclass

of *Mammal*, and *Chris* is a *Human* and therefore a *Mammal*. Is *Human* also a subclass of *Species*?

When we perform the analysis described on all these classes, we find that the identity criteria of *Species* are quite different from that of *Human*. Intuitively, species seem to be identified by their position in a biological taxonomy (for example, genus and differentia). On the other hand, we can assume that instances of *Human* are identified, in the simplest case, through the location in space/time of their bodies; two humans are different if they are at different places at the same time.

If *Human* was a subclass of *Species*, it would inherit its identity criteria. This can't be the case, since genus and differentia do not help in distinguishing one human from another. Therefore, *Species* cannot subsume *Human*. In fact, *Human* turns out to be an instance of *Species*, and *subsumption is not instantiation*.

Meta is Betta

The prefix "meta" acutally means "after." As with the term "ontology," its meaning in computer science is not quite correct and fairly vague. Meta seems to have something to do with levels of representation, as in a metalanguage, which is typically understood to be a language in which languages are specified. When we refer to a notion like "rigidity," we consider it a metaproperty in that rigidity is a property of properties, and not a property of objects in the world. In this sense, the notion of meta is very similar to the notion of instantiation, refering to a level of predication beyond the ground instances in a model.

We often find subsumption misused to represent metalevel relationships of this kind. For example, it may be tempting to create a class called "rigid class" and have it subsume all classes that are rigid, such as *Human*.

As with the *Species* example, a quick look at identity criteria reveals that this relationship cannot be. Instances of "rigid class" are classes, which can be identified in various ways (intensionally, in terms of the properties that define the class, or extensionally, in terms of their members). In any case, these identity criteria cannot be applied to the instances of *Human*, so being rigid is a metaproperty of the class *Human* and *subsumption is not meta*.

Part/Whole

It is often difficult for beginners in ontological analysis to distinguish between the part-of and the subclass relation. This is due to the fact that subclass is analogous to subset, and a subset of a set is a part of it. This confusion can be overcome when we realize the difference between the parts of a set and the parts of its members. Understanding the proper meaning of the part-of relation is often what ontological analysis is all about.

However, even people who understand the distinction, often misuse subsumption to represent part-of; they are both partial ordering relations after all, and when we draw diagrams showing subclass relationships we get a picture that looks the same as a diagram representing the decomposition of something.

To understand these problems, again we can apply our analysis. Take two classes like *Car* and *Engine*: is *Engine* a subclass of *Car*? We can approach this one in many ways since it is so obviously wrong; a useful strategy is to focus on the essential properties involved: among the essential properties of a car there are some functional properties, like *being able to accommodate people*. An engine has also certain functional properties as essential properties, like *being able to crank and generate a rotational force*. Since, however, the essential properties of cars do not apply to engines, one cannot subsume the other. The proper relationship here is part and *subsumption is not part*.

Disjunction/Type Restriction

An often-used "work-around" to the problem in the example here with car parts is creating artificial classes representing different levels of decomposition. For example, rather than claiming that engines are a subclass of cars, we have frequently found a class like *car part* subsuming *engine*, and a restriction or axiom requiring that all the parts of cars be car parts. This brings us to another common misuse of subsumption to represent a disjunction of classes for a type restriction.

To see how this is incorrect, rigidity analysis can be most useful. No instance of a car part is necessarily a car part (we could take an engine from a car and put it in a boat, making it no longer a car part but a boat part), so we have to make that class anti-rigid. The class engine itself is rigid, however, since we can't imagine an entity that is an engine becoming a non-engine. *Being an engine* is essential to it. An anti-rigid class, such as car part, can not subsume a rigid one, and so we have a conflict.

This type of mistake is particularly common in object-oriented and frame-based models, where value restrictions are an important part of modeling. These anti-rigid classes are created to satisfy a modeling need to represent disjunction, for instance, "any car part is either an engine, or a wheel, or a seat, …" It should be clear that this is different from saying "all engines are car parts," since, in fact, they are not. Of course, most modeling systems do not provide for disjunction, so modelers believe they are justified using these tricks, but if the intention is to make meaning as clear as possible then *subsumption is not disjunction*.

Polysemy

The most common misuse of subsumption in linguistics is to represent the multiple meanings (polysemy) of a term. For example, "book" is a polysemous term with at least two meanings: a bound volume with a size, weight, position, and so forth; an abstract entity with an author, title, and possibly many manifestations. When we say, "this book is heavy" we typically refer to the bound volume, whereas when we say, "I read this book" we are typically not talking about a particular bound volume, but about the abstract entity that the bound volume is a manifestation of.

In some linguistic ontologies, such as Mikrokosmos [7], we often find subsumption misused to represent this polysemy; the polysemous class is placed below all its possible meanings in a taxonomy. This may have some linguistic motivations, but is incorrect from the ontological point of view.

To see how this is incorrect, we can usefully employ identity or unity analysis. Bound volumes are identified, simply, by their location in space/time, so that two bound volumes cannot occupy the same space at the same time. The abstract notion of book is independent of space and time, being identified by author, title, date, and other criteria. Clearly no instance can meet both of these identity criteria; they belong to two different classes of entity, though there is a close relationship between them. No book is both a bound volume and an abstract entity; *subsumption is not polysemy*.

Constitution

The final common misuse of subsumption is to represent the fact that one thing is constituted of another. In the example given during the discussion of unity, we talked about *ocean* being subsumed or not by *water*. We examined the unity conditions and found that instances of the class *water* are not wholes, whereas instances of the class *ocean* are. Examples of this abound, as between the class *human* and *living-matter*, or between *company* and *group-of-people*. In each case, one class of entities is constituted of entities in the other class, but not subsumed by it; *subsumption is not constitution*.

Conclusion

The OntoClean methodology is in use in several places. OntologyWorks (www.ontologyworks.com) has designed a system that automates checking consistency of ontologies once their formal metaproperties have been expressed. They use the OntoClean methodology to perform a variety of commercial database integration tasks. Document Development Corporation (www.docdev.com) uses OntoClean as part of the knowledge engineering process for capturing and reusing knowledge assets in corporate documents such as contracts and proposals. At IBM Research, the methodology core was used in the SOALAR project to build ontologies of business engagements. At the Technical University of Madrid, OntoClean is being integrated with Methontology, a methodology for building ontologies based on software engineering principles. At the Italian National Research Council Laboratories (LADSEB-CNR and ITBM-CNR), in Pradova and Rome, OntoClean is in use in several projects including the development of an upper-level ontology based on a restructuring of WordNet, and the development of a core ontology for financial knowledge interchange. C

This work was partly supported by the Eureka Project IKF (E!2235, Information and Knowledge Fusion), the Italian National project TICCA (Tecnologie Cognitive per l'Interazione e la Cooperazione con Agenti Artificiali), and a Research Committee grant from Vassar College.

References

1. Guarino, N. and Welty, C. Identity and subsumption. In, R., Green, Ed. *Semantic Relations*. Kluwer, 2001.
2. Guarino, N. and Welty, C. Identity, unity, and individuality: Towards a formal toolkit for ontological analysis. In *Proceedings of ECAI-2000: The European Conference on Artificial Intelligence*. IOS Press, Berlin, Germany, 2000.
3. Guarino, N. and Welty, C. A formal ontology of properties. In R. Dieng, Ed., *Proceedings of 12th Int. Conf. on Knowledge Engineering and Knowledge Management*, Springer Verlag, 2000.
4. Hayes, P.J. Naive physics I: Ontology for liquids. In J.R. Hobbs and R. C. Moore, Eds. *Formal Theories of the Commonsense World.* Ablex, Norwood, New Jersey, 1985, pp. 71–108).
5. Miller, G.A. WordNet: A lexical database for English. *Commun. ACM, 38*, 11 (Nov. 1995), 39–41.
6. McCarthy, J. Circumscription—A form of non-monotonic reasoning. *Art. Int.* 5, 13 (1980), 27–39.
7. Nirenburg, S., Carbonell, J., Tomita, M., and Goodman, K. *Machine Translation: A Knowledge-Based Approach*. Morgan Kaufmann Publishers, Sant Mateo, NJ, 1992.
8. Quine, W.V.O. *Ontological Relativity and Other Essays*. Columbia University Press, New York, London, 1969.
9. Smith, B. and Welty, C. *Formal Ontology and Information Systems*. ACM Press, 2001.
10. Simons, P. *Parts: A Study in Ontology*. Clarendon Press, Oxford,1987.
11. Sowa, J.F. *Conceptual Structures. Information Processing in Mind and Machine, Reading*. Addison-Wesley, MA, 1984.
12. Storey, V.C. Understanding semantic relationships. Very *Large Databases J.* 2 (1993), 455–488.

Nicloa Guarino (guarino@ladseb..pd.cnr.it) is a senior research scientist at the Institute for System Theory and Biomedical Engineering of the Italian National Research Council (LADSEB-CNR) in Padova, Italy.

Christopher Welty (welty@cs.vassar.edu) is an assistant professor of computer science at Vassar College in Poughkeepie, NY.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 2002 ACM 0002-0782/02/0200 $5.00