Deconstruction, Reconstruction, and Ontogenesis for Large, Monolithic, Legacy Ontologies in Semantic Web Service Applications

Damian DG Gessler[1], Cliff A Joslyn[2], Karin M Verspoor[2], and Stefan E Schmidt[3]

(1) National Center for Genome Resources, Santa Fe, NM 87505 USA

(2) Computer, Computational and Statistical Sciences Division, Los Alamos National Laboratory, Los Alamos, NM, 87545, USA

(3) Technische Universitat, Dresden, Germany

Abstract

We present a distributed approach to ontology representation suited for semantic web services that is robust to extension and re-use. The representation encompasses, but is not limited to, the subsumption relationships included by the ontology creator. We discuss a refactoring of a large biological ontology into a property-centric model that allows improved consistency checking and easier cross-ontology integration, and yet maintains the original subsumption relationships without requiring explicit attribution of subsumption semantics. Upon the use of Formal Concept Analysis (FCA), the approach becomes ontogenic: it facilitates automatic, dynamic, data-driven ontology and knowledge base creation through the analysis of individuals' shared properties.

Key words: semantics, web services, ontologies, formal concept analysis, ontogenic

1 Introduction

Traditional representations of ontologies utilize subsumption in the form of nested subclass relationships to organize concepts hierarchically. In the world of semantic web services, in which information-providing services are distributed while their properties, capabilities, interfaces, and effects are encoded in an unambiguous, machine-understandable form [1], this type of representation leads to ontologies that are difficult to access dynamically and that are brittle with respect to extensibility and re-use.

These problems are exacerbated for very large ontologies, such as those used in biomedicine. There is an effort to standardize the syntax for biomedical ontologies through the Open Biomedical Ontologies project (OBO: obo.sourceforge.net). Yet the ontologies tend to be monolithic in conception, and reside as one ontology per file in their representation. Such representations

create difficulties for dynamic web services environments that seek access across ontologies on the per-term level. In this paper, we explore an alternative, property-centric representation for ontologies that maintains the information content of hierarchical ontologies and yet is lighter and more flexible for use in semantic web services.

1.1 The Large Ontology Problem

Ontologies in biology tend to be large. For example, the Gene Ontology (GO: www.geneontology.org) is comprised of three separate ontologies (Biological Process, Molecular Function, and Cellular Component) containing nearly 20,000 terms. This is dwarfed by the almost 23,000 descriptors and 151,000 supplemental headings in the Medical Subject Headings (MeSH: www.nlm.nih.gov/pubs/factsheets/mesh.html) and the one million biological concept and vocabulary terms in the National Library of Medicine's Unified Medical Language System Metathesaurus (UMLS: www.nlm.nih.gov/research/umls/about_umls.html#Metathesaurus).

Large ontologies, *per se*, do not present noteworthy problems. For example, standard ETL (extract, translate, and load) operations on compressed files of many MB are easily handled by today's PCs with access to broadband internet connections. But these same ontologies can present a formidable challenge to distributed semantic web services architectures that require dynamic access to specific, generally small, portions of the ontologies in response to particular information requests.

To address this issue, the Virtual Plant Information Network (VPIN: vpin.ncgr.org; www.semanticmoby.org), an open-world semantic web service architecture, has been designed such that service discovery, service requests, and service responses reference semantic terms *atomically*. That is, terms from disparate ontologies are referenced via term-specific URLs (Uniform Resource Locators) much in the same way as hyperlinks are used on a web page. Just as it would be unreasonably burdensome to require that web pages dereference the content behind a URL before including a reference to it on a page (or even worse, downloading the entire web site simply to reference the URL), it is unreasonably burdensome on semantic web service architectures such as the VPIN to require that full ontologies be downloaded by clients and servers simply for the purpose of referencing one or two terms contained therein. For the VPIN, we seek a way to allow rapid, on-demand access across ontologies on a term-by-term basis at transaction time. We need a way to access ontological terms atomically, yet without losing the ability to identify the topological relationships of any given term in the ontology. As we will see later, this is not sufficiently solved by simply using the fragment ('#') token in a URL.

1.2 The Static Subsumption Problem

In a well controlled domain space, one can use nested subclass relationships to organize concepts hierarchically. Indeed, many ontologies, such as the Gene Ontology, are essentially static subsumption hierarchies. By static subsumption hierarchies, we mean that they explicitly assert subclass relationships axiomatically (for example, by using the `owl:subClassOf` predicate), instead of deriving subsumption dynamically (for example, by inferring classes and subclasses from sets of individuals based on shared properties)

If the problem at hand is well defined, and if ontology maintenance, extension, and deprecation are centrally controlled, then axiomatic subsumption can yield substantial value in semantically organizing information. But static subsumption hierarchies suffer from two problems in an open world of semantic web services. The first is that they tend to be built from the top down, so concepts near the root—for which changes necessarily alter the definition of every subclass—are established early in the ontology's creation when there is the least amount of experience and real-world feedback on the utility of the ontological model. Changes to those terms established earliest in the project are unfortunately likely to cause the most extensive repercussions later on, thereby working against evolvability. The second problem is the antagonism between static subsumption and loose-coupling, or the need to allow individual nodes in a distributed system to change without affecting or requiring change in other parts of the system [2]. Because static subsumption classes are *defined* in transitive, cascading subclass definitions, it is difficult for third-parties to extend concepts unless the entire subsumption tree matches the concepts relevant for their problem at hand; *i.e.*, the users' problem space has to match the ontology creator's world view up the chain of the hierarchy. This is particularly acute across ontologies where static subsumption limits cross-ontology modularization due to lack of extensibility and re-use.

Thus static subsumption hierarchies fuel both rigidity (the inability to change to meet new demands) and fragility (the propensity to fail, often in multiple and seemingly unrelated places, under changes). We seek an ontological model that is robust to extension and re-use in an open world, and specifically one that encompasses, but does not limit, the pre-conceived subsumption relationships of the ontology creator.

2 Deconstruction

Despite the above limitations, there still is high value in preserving the information content of legacy ontologies (such as the Gene Ontology and others of the OBO), even if this information is organized primarily as a static subsumption hierarchy. So the problem is: How can we refactor legacy ontologies so that we can apply their value while addressing the limitations of monolithicity and static subsumption assertions? In the sections that follow, we

reference a partial refactoring of OBO and GO available at ontologies.ncgr.org. We use the W3C standard of OWL as serialized by RDF/XML to encode the ontologies.

2.1 Using Full URLs instead of '#' Fragments

A straight-forward approach to decomposing a large ontology into component terms is to reference each term via a URL. A common practice in OWL RDF/XML ontologies is to use the URL fragment identifier '#' to "hash into" a large file (*e.g.*, www.myWebsite.org/largeOntology#term100). But the W3C standard on '#' clearly states that it is interpretable solely by the actor dereferencing the URL (see [3]). This is primarily a client-side activity; indeed, network caches and proxies may preclude a server from even seeing the '#' in the URL on a HTTP GET or POST. In other words, an HTTP GET on www.myWebsite.org./largeOntology#term100 may invoke the server to return the entire document at www.myWebsite.org/largeOntology, independent of the '#term100' visible at the client end.

In our decomposition of GO (ontologies.ncgr.org/GeneOntology), we represent each GO term as a separately dereferencable URL without the use of #; *e.g.*, ontologies.ncgr.org/GeneOntology/BiologicalProcess/GO_0000001. Each term is only a dozen or so lines of RDF/XML, guaranteeing scalability such that semantic web service applications can execute term-by-term, on-demand, random access in a high-throughput manner, independent of the size of the ontology.

2.2 Removing Static Subsumption Assertions

Simply decomposing a large file into numerous smaller files[1] each dereferencable by a unique URL is hardly modularization. For the mix-and-match capabilities of a semantic web services architecture, we also seek a representation in each file that is essentially self-contained; *i.e.*, a valid OWL (or specifically OWL-DL) definition of the term that can be used in a variety of circumstances independent of any *a priori* class hierarchy. This is similar to how one can choose and use words from a dictionary without claiming adherence to the global model of all words as defined in the language. In aggregation across terms, we seek a lossless representation, such that all subsumption information in the monolithic formulation is recoverable. We do this by emphasizing each term's *properties*, instead of its class inheritance.

[1] There is, of course, no implementation requirement that these be physical files in a file system. Thousands of distinct URLs can be handled by a single Java servlet that generates a distinct document for each URL on demand.

For ontologies such as GO, the problem is made more difficult because there is no readily available set of properties that can be used to infer the class subsumption hierarchy of the ontology. Essentially the entire topology of the subsumption hierarchy is built on the axiomatic assertions that one class is a subclass of another, without explicitly listing those properties that would allow a reasoner to so infer. For example, 'Cell Adhesion' is a subclass of 'Cellular Process' which is a subclass of 'Biological Process,' but there is no explicit list of properties that would allow a reasoner to group individuals into these classes based on those individuals' properties. Individuals—for example, gene entries in databases—get *assigned* to classes: a procedure called annotation. This may be done manually by a curator. Automatic (algorithmic) annotation, where it exists, is rule based, referring to external data to assign matches between individuals and GO classes [4, *inter alia*].

To address the static subsumption limitation of property-poor legacy ontologies such as GO, we note that the transitive relationship `C subClassOf B`, `B subClassOf A` for classes `A`, `B`, and `C` implies by definition that all individuals of class `C` have all the properties required for individuals to be of class `B` (and maybe more), and similarly all individuals of class `B` have all the properties required for individuals to be of class `A` (and maybe more). The triple '`C subClassOf B`' is purposefully analogous to the use of triples in RDF specifying a subject-predicate-object relationship. As one traverses up a subsumption hierarchy from the subclass to the superclass, one goes from the specific (more properties) to the general (fewer properties).

We first redefine each legacy class in GO (each GO term) to be an *individual* (an instance, as distinct from a class) at a URL. We then introduce three new properties called `superProperty`, `subProperty`, and `rootProperty`. For each individual (corresponding to a GO term in the new formulation), we add a `superProperty` predicate for each of its legacy superclasses, with the superclasses' instance representation as the object of the predicate: *e.g.*, `C owl:subClassOf B` becomes `{ C superProperty B, C superProperty A }`. Thus we replace static *owl:subClassOf* assertions with properties suitable for dynamic class inference—since without this subsumption information GO is little more than a controlled vocabulary.

For each term's direct legacy children (but not grand children, etc.), we add a `subProperty` statement; *e.g.*, `A subProperty B`. Thus from any GO term as a URL, one can traverse a sequence of hyperlinks both up and down the legacy subsumption tree, and therefore have a mechanism to dereference the entire ontology, including preserving its subsumption topology, from starting at any node.

For implementation issues, we note some minor refinements. We add the convenience property `rootProperty` to every individual. This points directly to the root of the ontology; *e.g.*, `C rootProperty A`. We also note that the

minimum information needed to recreate the subsumption topology from the `superProperty` predicate is for each individual to preserve its relationship solely to its direct legacy parent(s) (since one could successively dereference definitions up the `superProperty` chain, just as we do down on the `subProperty` chain), yet for any given term we include `superProperty` statements up to the root. This has the advantage of allowing a term's single file definition to comprehensively include all statements about it on a single URL dereference. It has the disadvantage that term maintenance requires checking that all `superProperty` statements remain true over time. This is the same maintenance issue that all web masters must address in ensuring that URLs embedded in their pages continue to reference relevant content. Different implementations will assess this cost/performance issue differently. We also note that there is no distinction in the use of `superProperty` as to the claimed subsumption relationships of the objects; *e.g.*, `{ C superProperty B, C superProperty A }` does not tell one if `B superProperty A` or `A superProperty B`. This is important as it encapsulates statements between subjects and objects, so that, for example, `C` is only making statements about itself as the subject, and is not making assertions about other terms within its definition. Other terms' `superProperty` statements should appear, and only appear, in their definition documents at their URLs.

The final result is a deconstruction of a large, legacy, monolithic ontology built almost entirely on static subsumption statements into distributed, atomic, components; each separately dereferencable by its own URL, and each devoid of the legacy `owl:subClassOf` static assertions. The deep hierarchy topology is broken. Yet while no component makes an explicit subsumption statement about its legacy subsumption relationships, any part of the entire subsumption topology can be recreated by noting that all components that share the same objects of `superProperty` triples can be inferred to belong to the same class marginal on those objects.

3 Reconstruction

The advantage of replacing static subsumption assertions—particularly cascading assertions in deep hierarchies—with a property-centric model is that it brings the benefits of extensibility and loose-coupling more directly to ontology modularization. These benefits include improved consistency checking, easier third-party extension and re-use, easier cross-ontology integration.

3.1 `superProperty` creates principal filters which allow for a lossless recreation of the subsumption assertions

Our use of `superProperty` creates a partially ordered set (a poset) of all individuals from any given node to the root. The set theoretic *principal filter* for

some individual *X* is the set of all individuals, from *X* to the root, that are objects of `superProperty` relationships with *X* as the subject; in other words, the single file definition of a term as described above. The transitive reduction [5] on the union over all principal filters is guaranteed to recreate the poset (graphically a Hasse diagram) guaranteeing that this modularization of subsumption information is lossless [6].

It is trivial that if we replace all `owl:subClassOf` predicates with `superProperty` predicates that one could recreate the subsumption assertions by simply stating semantic equivalency between the two. Yet a stronger statement can be made, in that the poset and principal filter approach [6] shows that subsumption information can be recovered purely on a set theoretic basis based on the relational information present within the formal context, whatever the source of that information. In other words, even when subsumption information is not explicitly encoded, the relational properties in the context allow induction of subsumption relations on a set theoretical basis, without requiring that `superProperty` be overloaded with subsumption semantics.

3.2 Formal Concept Analysis drives the Dedekind-MacNeille Completion

One problem with manually created, deep subsumption hierarchies is the possibility of failing to explicitly identify all relevant classes. For example, Figure 1 shows how a manually asserted subsumption hierarchy may fail to explicitly delineate the class of "individuals with properties in common to `A` and `B`, but without all properties of `E`". In Figure 1, `Q` is called the least common subsumer (LCS) of `D` and `E`. Least common subsumers arise often in the solutions to answering questions using ontologies (*e.g.*, what is minimally common to `D` and `E`?) [7].

Figure 1. Subsumption hierarchies (Hasse diagrams) where arrows indicate subclass relationships (*e.g.*, *owl:subClassOf*). (Left) Original, manually asserted hierarchy. (Right) The Dedekind-MacNeille completion makes it clear that there exists a *least common subsumer* (LCS) `Q` that is the minimal class of all individuals with properties shared by `D` and `E`.

Least common subsumers fall out naturally from the Dedekind-MacNeille completion [8] over a bounded poset. Consider a matrix where the rows and columns refer to the individuals, or nodes, in the poset. We add a virtual top *R* (unique superclass of all classes) and virtual bottom Z (unique subclass of all classes) if these nodes do not already exist. Let element e_{ij} be unity for the RDF triple `i superProperty j` or `i = j`, and zero otherwise. Thus the matrix is a summary of all static subsumption statements over the legacy ontology:

	R	A	B	C	D	E	Z
R	1	0	0	0	0	0	0
A	1	1	0	0	0	0	0
B	1	0	1	0	0	0	0
C	1	1	1	1	0	0	0
D	1	1	1	1	1	0	0
E	1	1	1	0	0	1	0
Z	1	1	1	1	1	1	1

It can be shown [6,8] that by using Formal Concept Analysis (FCA), the Dedekind-MacNeille completion over this matrix is a lattice as shown graphically in the right hand side of Fig. 1. The lattice explicitly includes `Q`, the LCS of `D` and `E`. In theory, FCA will construct missing LCSs for all nodes in legacy ontologies modularized by the `superProperty` approach, thereby adding an important automated consistency and completeness step to the otherwise manual process of ontology construction. In practice, we have not implemented FCA on large ontologies, so performance and feasibility issues remain to be addressed.

4 Data-driven Ontology Induction

The ontogenic properties of FCA can be used in an even stronger manner in conjunction with a property-centric data model. Traditionally, we separate ontology creation (ontogenesis) from annotation (the process of associating individuals, such as gene records, with ontological concepts, such as "Cellular Adhesion"). But this is a laborious and error-prone task. What we seek is a way that ontologies can be created automatically, and ideally in a manner that reflects those concepts most applicable to the problem at hand.

One can use FCA for data-driven, automatic, *de novo* ontology creation. To do so, construct a $n \times m$ matrix of n individuals and m binary properties. These

properties, or attributes, can come from a variety of sources: legacy ontologies, experimental data, computational results, etc. One can then apply FCA to the matrix to create a subsumption lattice which integrates the properties across these distinct sources.

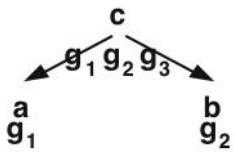

Figure 2. Annotated portion of an ontology where individuals G are annotated to (ordered) concepts {a, b, c}

To illustrate, consider a (hypothetical) fragment of an ontology, with individuals g_1 to g_3 annotated to concepts {a, b, c} as in Figure 2, coupled with keyword associations derived from texts between individuals g_2 to g_4 and keywords k_1 to k_3, as indicated in Figure 3(a).

	k_1	k_2	k_3
g_2	1	0	1
g_3	0	1	0
g_4	1	1	0

Figure 3a. Keyword property matrix

	a	b	c	k_1	k_2	k_3
g_1	0	1	1	0	0	0
g_2	1	0	1	1	0	1
g_3	0	0	1	0	1	0
g_4	0	0	0	1	1	0

Figure 3b: Merged property matrix

The merged matrix of individual/property associations is then shown in Figure 3(b). The application of FCA to this matrix results in the concept lattice in Figure 4. While the ontological information of the ontology fragment is preserved and respected in the merged subsumption hierarchy, there is no explicit underlying hierarchical structure on the keywords *K* in this example.

Rather, the concept lattice process derives the hierarchical relations among the keywords implicit in the structure of the properties. For example, k_1 is a more general keyword than k_3, while k_2 is of effectively indeterminate generality. So there is not a complete poset relation among the properties as shown in both the merged matrix and the lattice.

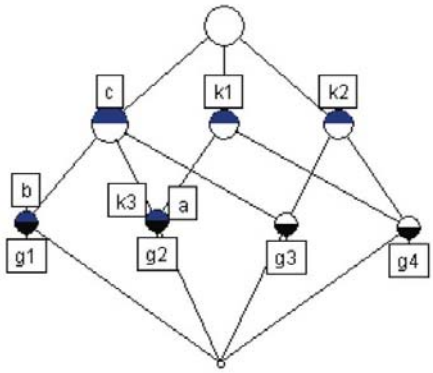

Figure 4. Joint concept lattice for the merged property matrix.

This subsumption lattice is an on-demand knowledge base that classifies individuals according to their hierarchically related common properties. The subsumption classes essentially create a data-driven, *de novo* ontology. In this manner, one can construct data-driven ontology induction to create ontologies from disparate sources customized to the problem at hand. So for biology, instead of thinking about biological concepts *ex situ* and organizing them in a static, deep subsumption hierarchy; then discovering genes and laboriously annotating associated gene records to the ontological concepts—all without the guarantee that the classification is consistent and complete—one would enter gene properties, concepts, literature records, etc., into an "FCA engine" which would generate a data-driven, guaranteed consistent, annotated knowledge base automatically. The semantic web services support allows this FCA engine to operate over disparate, distributed resources. Such knowledge bases would be more dynamic, ephemeral, and suited to the problem at hand, rather than the more static and encyclopedic conception of knowledge bases currently popular. The quality of the knowledge base would be critically dependent on the quality of the property matrix. But because that matrix is built from considering property-centric statements instead of global subsumption models, it is substantially easier to verify on a statement-by-statement basis.

5 Conclusions

We have presented a property-centric approach to capturing subsumption relations as expressed in traditional ontologies, and discussed how this refactoring enables both more dynamic access to the ontology in a semantic web services environment. We have also shown how it could be use to drive ontology induction from properties associated with individuals in the property-centric representation.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0516487, as well as work supported by the Department of Energy under contract with Los Alamos National Security, LLC.

References

1. McIlraith, S., Son, T.C. and Zeng, H. "Semantic Web Services", IEEE Intelligent Systems. Special Issue on the Semantic Web. 16(2):46--53, March/April, 2001.
2. See http://www.mcdowall.com/webservices/2002_11_19_archive.html
3. See http://www.w3.org/TR/webarch/#media-type-fragid.
4. Verspoor, KM; Cohn, JD; Mniszewski, SM and Joslyn, CA: (2006) "A Categorization Approach to Automated Ontological Protein Function Annotation", *Protein Science*, v. 15, pp. 1544-1549.
5. Aho, AV; Garey, MR; and Ullman, JD: (1972) "The Transitive Reduction of a Directed Graph", *SIAM Journal of Computing*, v. **1**:2, pp. 131-137.
6. Joslyn, CA; Gessler, DDG; Schmidt, SE; and Verspoor, KM: (2006) "Distributed Representations of Bio-Ontologies for Semantic Web Services", Joint BioLINK and 9th Bio-Ontologies Meeting, 14th Annual International Conference on Intelligence Systems for Molecular Biology (ISMB 06).
7. Baader, Franz; Sertkaya, Baris; and Turham, Anni-Yasmi: (2004) "Computing the Least Common Subsumer w.r.t. a Background Terminology", in: *Proc. JELIA 2004*, *Lecture Notes in AI*, v. **3229**, pp. 400-412.
8. Ganter, Bernhard and Wille, Rudolf: (1999) *Formal Concept Analysis*, Springer-Verlag.