Intelligently Designed Ontology Alignment: A Case Study from the Sequence Ontology

Michael Sinclair[1], Michael Bada[2], and Karen Eilbeck[1]*

[1]Department of Biomedical Informatics, University of Utah School of Medicine, Salt Lake City, UT 84108, USA
[2]University of Colorado Anschutz Medical Campus, Department of Pharmacology, Aurora, CO 80045, USA

ABSTRACT

As the number and size of biomedical ontologies continue to grow, the problem of coordinating them becomes increasingly urgent. This is particularly problematic for ontologies with semantically overlapping or analogous content but based on different underlying conceptualizations. Ontology matching methods have been developed to align existing ontologies for semantic interoperability, but this is a laborious and inefficient process compared to designing related ontologies to be aligned from the outset. We present a method for designing companion ontologies to be largely semantically and structurally aligned as they evolve. We have applied this method to the specific case of the coordinated management of two ontologies focused on the representation of analogous types of biological sequences: the Sequence Ontology and the Molecular Sequence Ontology. We derived general principles that any group can apply to their own use cases: 1) Create inter-ontology logical definitions for those portions of the ontologies to be aligned, 2) use class annotations to automatically manage those portions of the ontologies not to be aligned, and 3) use the taxonomy of one ontology to classify the other with an OWL reasoner. We propose this as an efficient way of designing companion ontologies to be aligned throughout their development life cycle.

1 INTRODUCTION

As the number and size of biomedical ontologies continue to grow, the problem of coordinating them becomes ever more urgent. This is particularly problematic for ontologies with semantically overlapping or analogous content but based on different underlying conceptualizations, which is often indicative of semantic inconsistency between the ontologies. In such a case for two independently developed ontologies, a range of techniques designed to address this, called "ontology matching", have been researched (Otero-Cerdeira, Rodríguez-Martínez, & Gómez-Rodríguez, 2015), but many of these techniques result in either direct or indirect mappings between the ontologies rather than the resolution of any underlying conceptual inconsistency.

Bringing such preexisting ontologies into alignment is often a laborious task. For the case of two ontologies that do not semantically overlap but are semantically analogous in that they directly or indirectly rely on a shared underlying conceptualization, an alternative is to design the ontologies to align with each other from the outset. However, after their initial design, this alternative requires the coordinated management of these ontologies such that they do not semantically diverge from each other.

In this paper, we present such a mechanism for our specific use case, the management of two ontologies focused on analogous types of biological sequences required for different bioinformatics tasks, specifically the Sequence Ontology (SO) (Eilbeck et al., 2005) and the Molecular Sequence Ontology (MSO) (Mungall, Batchelor, & Eilbeck, 2011). We have designed these ontologies to be conceptually consistent by formally defining classes from the former in terms of their counterparts in the latter and relying on a reasoner to keep them substantially aligned, and for those portions of the ontologies that are not to be aligned, we employ class- and property-specific annotations. In doing so, this methodology allows for the automatic generation of two closely related ontologies from one master file, thus obviating the need for laborious and error-prone manual editing of the two ontologies.

2 BACKGROUND

Macromolecular sequences have long been central to basic biomedical research and, with the advent of molecular medicine, to clinical work as well. This can be easily appreciated with the proliferation of actively maintained and utilized sequence databases, in which is stored a vast diversity of macromolecular information, including sequences, sequence variations and modifications, structures, and associated pathways and processes, diseases and pathologies, and phenotypes. The Sequence Ontology (SO), a member of the Open Biomedical Ontologies (OBO) library (Smith et al., 2007), is focused on the formal representation of these key biomedical entities, particularly on functional and structural regions, types of variations, and sequence assemblies (Eilbeck et al., 2005). The SO has been used to annotate millions of sequences in such databases, thus enabling sophisticated querying, visualization, and analysis of these sequences.

*To whom correspondence should be addressed: michael.sinclair@utah.edu.

Though these types of entities are needed for a thorough representation of biology, there has been confusion as to their nature as represented in the SO, namely whether they represent material entities, informational entities, or abstract entities of some other kind. Mungall et al. (2011) have asserted that the classes of the SO are generically dependent continuants, a fundamental type of abstract entity in the Basic Formal Ontology (BFO), an upper-level ontology used by many OBO developers to categorize at a high level the content of their ontologies in a manner consistent with those of other OBOs (Arp, Smith, & Spear, 2015); at the same time they acknowledge that the logical definitions for at least some of the classes of the SO apply rather to material sequence entities, particularly classes defined in terms of attributes such as biological conservedness, catalytic ability, and mobility. Additionally, the textual definitions of many, if not most, of the classes are written in terms of the material sequence entities. This semantic conflation of the nature of these sequence entities has been a primary motivation for what has been referred to as the refactoring of the Sequence Ontology. This effort has been additionally motivated by the fact that although the SO has a rich internal semantic structure, it is not integrated with other OBOs, either through placement under higher-level classes of external OBOs or through logical definitions of such classes.

The central feature of this refactoring work is the creation of two largely parallel ontologies: the Sequence Ontology, which remains an ontology of generically dependent continuants, and the Molecular Sequence Ontology (MSO), whose principal sequence entity classes represent material (molecular) entities (Bada & Eilbeck, 2012). Most of the classes of the current public SO have counterparts in the refactored SO and the MSO, and they are in a relationship where classes of the former are generically dependent on their counterparts in the latter. Both of these have been integrated into the wider OBO ecosystem: The classes of the SO are directly classified as BFO generically dependent continuants, and those of the MSO are asserted to be subsumed by classes of the Chemical Entities of Biological Interest ontology (ChEBI) (Degtyarenko et al., 2008) and of the BFO.

It is important to note that both of these ontologies are required, but for different bioinformatics tasks: The SO will continue to be used for its most prominent task, i.e., annotation of sequences in databases, as it contains classes that may be needed for such annotation but that represent types of sequence entities that do not intuitively have material counterparts (e.g., read (SO:0000150), contig (SO:0000149)) and therefore do not have corresponding classes in the MSO. On the other hand, the MSO, in addition to representing fundamental material sequence entities, is needed for integration with classes of external OBOs representing other types of material biological entities and biological processes whose participants are material entities. Therefore, we were presented with a problem of having to maintain two closely related ontologies sharing the same underlying conceptualization without having to rely on concurrent manual editing of both, which we assessed as being laborious and error-prone.

To overcome this challenge, we surveyed existing methodologies and tools. As mentioned above, ontology matching algorithms have been devised for mapping classes in one ontology to those in another that the algorithm determines to be semantically similar (Otero-Cerdeira et al., 2015). However, these methods only align ontologies that already exist and do not specify the generation of new ontologies so as to be aligned from inception and evolve concordantly. Moreover, they cannot prevent semantic and structural divergence as ontologies develop, they can only periodically realign them, which is laborious, inefficient, and thus not suitable for our purposes. We therefore turned to some commonly used ontology editing programs: Protégé[1], ROBOT[2], and dead-simple OWL design patterns (Osumi-Sutherland, Courtot, Balhoff, & Mungall, 2017). Each of these applications lacked one or more needed functions: Protege requires manual editing and recreates the problem of human curation, ROBOT cannot currently filter classes based on annotation values or add new property assertion axioms, and dosdps also cannot make decisions based on annotations. While we recognize the utility and value of these tools for ontology management, they do not address our use case.

In the present paper, we describe our solution to the problem of simultaneous development and maintenance of the SO and MSO. We propose that this approach can be generalized to any similar scenario, namely, the auto-generation and maintenance of a suite of companion ontologies from a common (i.e. "master") ontology file.

3 APPROACH

We present below each step of our solution, first in general so that anyone can apply it to their own case, and then as specifically applied to ours.

Create parallel inter-ontology logical definitions for classes to be aligned

A class of a given ontology can be assigned a logical definition specified in terms of one or more classes of other ontologies. Such a definition often takes the form of a genus, i.e., a directly asserted superclass, along with one or more differentiae, i.e., specified attributes that differentiate the class being defined from other kinds of the superclass, e.g.:

Ontology1:ClassA equivalentTo (Ontology1:ClassB and *has_property* Ontology2:ClassC)[3]

[1] https://protege.stanford.edu/

[2] http://robot.obolibrary.org/

[3] In this paper, Manchester OWL syntax is used to specify OWL expressions

We are designing the SO and the MSO to share an underlying conceptualization and therefore to be largely parallel to each other, with the classes of the former being in a relation of generic dependence with the corresponding classes of the latter. This relation represents how generically dependent continuants, a class of entities in the BFO that depend on some physical thing for existence, are distinguished by existing as multiple copies in physical bearers. In our case, we are dealing with types of biological sequence (SO), which depend on sequence molecules or regions thereof that are repeatedly duplicated (MSO). Thus, we created logical definitions of the former in terms of the latter, relying on generic dependence for the differentiae, e.g.:

SO:gene equivalentTo (SO:'biological sequence entity' and *generically_depends_on* MSO:gene)

Note that we are not explicitly asserting the immediate superclasses in these logical definitions as the genera; instead, we will be relying on an OWL reasoner to infer a hierarchy for the SO largely parallel to that of the MSO.

When SO classes are thus defined, they can be derived automatically from the data for MSO; that is, if there is an MSO:gene, then there must exist an SO:gene that generically depends on it. Therefore, only MSO need be included in the master file, except SO classes that do not depend on any MSO class (see below). The lesson is to exploit logical definitions to derive the data for other ontologies from a file that only contains a single ontology.

Annotate classes that are not to be represented in both ontologies

In the case of an ontology that is to be automatically generated from another, it may not be required or even correct to create counterparts for certain classes. We mentioned above that there are classes in the SO for which there are not intuitive material counterparts and therefore will not be represented in the MSO (e.g., read (SO:0000150), contig (SO:0000149)); thus, logical definitions in terms of corresponding MSO classes cannot be created for these SO classes. Using OWL annotation properties, we have attached Boolean annotations to such classes (e.g. is_represented_only_in_SO = true) and included them in the master file. Analogously, we have created such annotations for MSO classes that are not to be correspondingly represented in the SO, e.g., for attributes that are sensible only for material entities. Our code checks for the presence of these annotations and ensures that the annotated classes are created only for the appropriate ontologies. In our case, classes flagged as only in SO are assigned to the SO output file, likewise for classes flagged as only in MSO.

Use the taxonomy of one ontology to classify the other

For those portions of the ontologies that are to be aligned, we can use an OWL reasoner to classify one ontology from the other without the need to independently specify all subclass relations:

Ontology1:ClassA equivalentTo (Ontology1:ClassE and *has_property* Ontology2:ClassB)

Ontology1:ClassC equivalentTo (Ontology1:ClassE and *has_property* Ontology2:ClassD)

ClassB SubClassOf ClassD

therefore,

ClassA SubClassOf ClassC

If most SO classes generically depend on MSO classes, and the SO taxonomy should parallel the MSO taxonomy, we can classify the entire SO taxonomy without any further axioms. We need only directly import the MSO into the SO space and run a reasoner. For example:

SO:intron equivalentTo (SO:'biological sequence entity' and *generically_depends_on* MSO:intron)

SO:'transcript region' equivalentTo (SO:'biological sequence entity' and *generically_depends_on* MSO:transcript region)

MSO:intron SubClassOf MSO:transcript region

therefore,

SO:intron SubClassOf SO:'transcript region'

Thus, logical definitions can be used not only to automatically derive classes, but also the taxonomy, of two ontologies from a master file. For us, this step accomplishes the refactoring of SO to have the same taxonomy as the MSO.

Algorithm for generating SO and MSO

Putting everything above together, the following is our algorithm for a program to generate the SO and MSO from a single input, or master, ontology file. The master file contains MSO classes, and their defining axioms, on which an SO term generically depends. It also contains classes flagged with annotations in both ontologies that do not participate in generic dependence across the ontologies:

(1) Make a copy of the master file ontology (CopyA) and remove all classes that have annotation is_represented_only_in_SO = true. Output CopyA as "MSO".

(2) Make a copy of the master file ontology (CopyB) and remove all classes that have annotation is_represented_only_in_MSO = true.

(3) Change all class IRIs in CopyB to their counterparts in the SO namespace.

(4) Remove properties that are not used in SO from CopyB.

(5) Assert generically_depends_on properties between MSO and SO classes in CopyB.

(6) Import the MSO into CopyB.

(7) Classify the taxonomy in CopyB with a reasoner. Accept all inferred axioms as part of CopyB.

(8) Remove the MSO direct import.

(9) Output CopyB as "SO_refactored".

A curator need only work with the master file, for example when adding or deprecating classes, and can regenerate both ontologies after updating the master file by running this program.

4 RESULTS AND DISCUSSION

We have implemented the above approach in our own OWL API program, available at the MSO github repository[4]. The program takes a master file as input which contains all MSO classes as well as SO classes that do not depend on any MSO class, and outputs complete and independent SO and MSO ontology files. The MSO can be reasoned over in seconds with FaCT++ or JFact with no inconsistencies. We have not yet implemented reasoning of SO from within the program (steps 6-8 in the algorithm), so to classify the new SO currently, the MSO must be directly imported into an editor such as Protege and then reasoned over. The reasoning takes approximately 1 minute and 20 seconds with FaCT++, and 26 minutes and 20 seconds with JFact with no inconsistencies. We are currently incorporating classification of the refactored SO from within the program so that users do not need to do it themselves.

Our solution enables MSO to be used in cross products with neighboring ontologies such as GO and ChEBI and to be automatically dovetailed with sequence metadata in bioinformatics databases in the form of the corresponding SO classes. The time-consuming efforts currently underway to align the GO with neighboring OBO ontologies by searching these, in some cases extremely large, ontologies for potential cross-products (Hill et al., 2013) are, in one step, permanently avoided for SO and MSO. Along the way, we discovered that annotations on ontology classes are an under-utilized resource for nesting multiple related ontologies in the same file, to be sorted as part of a release pipeline.

We are also automating the addition of other axioms to SO aside from generic dependence. When generating one ontology from another solely on the basis of logical definition, axioms besides those linking the ontologies are lost. Our program needs a systematic way of capturing those that are important for the environments in which SO is used. Mereological axioms (e.g. *has_part*, *is_part_of*) are crucial for SO to create gene and protein models consisting of an assembly of functional domains, and so must be retained. Since the corresponding MSO classes participate in the same mereological axioms, we need only copy them to SO when it is being generated. Therefore, we are modifying our algorithm to include retention of mereological axioms in SO.

[4]https://github.com/The-Sequence-Ontology/MSO

5 CONCLUSIONS

We have presented a methodology for designing companion ontologies to be semantically and structurally aligned throughout their development life cycle. We present it as a plausible way of planning ahead to ensure consistency rather than be faced with the difficult work of mapping, linking, and aligning ontologies after they have grown to considerable size and significantly diverged from one another. Although we applied the method to our own specific task of coordinated evolution of the SO and MSO, it can be used by other groups designing their own ontologies and seeking ways to keep them conceptually synchronized.

ACKNOWLEDGEMENTS

We thank Alex Henrie for general programming help, Ignazio Palmisano for help with the OWL API, and the whole Protege user support team for quick and frequent responses. Michael Sinclair acknowledges the support of the National Library of Medicine Training Grant T15LM007124.

REFERENCES

Arp, R., Smith, B., & Spear, A. D. (2015). *Building Ontologies with Basic Formal Ontology*. Cambridge, US: The MIT Press.

Bada, M., & Eilbeck, K. (2012). *Efforts toward a More Consistent and Interoperable Sequence Ontology*. Paper presented at the International Conferene on Biomedical Ontology, Graz, Austria. http://ceur-ws.org/Vol-897/session3-paper13.pdf

Degtyarenko, K., de Matos, P., Ennis, M., Hastings, J., Zbinden, M., McNaught, A., . . . Ashburner, M. (2008). ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Research, 36*(Database issue), D344-D350. doi:10.1093/nar/gkm791

Eilbeck, K., Lewis, S. E., Mungall, C. J., Yandell, M., Stein, L., Durbin, R., & Ashburner, M. (2005). The Sequence Ontology: a tool for the unification of genome annotations. *Genome Biol, 6*(5), R44. doi:10.1186/gb-2005-6-5-r44

Hill, D. P., Adams, N., Bada, M., Batchelor, C., Berardini, T. Z., Dietze, H., . . . Lomax, J. (2013). Dovetailing biology and chemistry: integrating the Gene Ontology with the ChEBI chemical ontology. *BMC Genomics, 14*, 513. doi:10.1186/1471-2164-14-513

Mungall, C. J., Batchelor, C., & Eilbeck, K. (2011). Evolution of the Sequence Ontology terms and relationships. *J Biomed Inform, 44*(1), 87-93. doi:10.1016/j.jbi.2010.03.002

Osumi-Sutherland, D., Courtot, M., Balhoff, J. P., & Mungall, C. (2017). Dead simple OWL design patterns. *J Biomed Semantics, 8*(1), 18. doi:10.1186/s13326-017-0126-0

Otero-Cerdeira, L., Rodríguez-Martínez, F. J., & Gómez-Rodríguez, A. (2015). Ontology matching: A literature review. *Expert Systems with Applications, 42*(2), 949-971.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., . . . Lewis, S. (2007). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat Biotechnol, 25*(11), 1251-1255. doi:10.1038/nbt1346