Auditing Redundant Import in Reuse of a Top Level Ontology for the Drug Discovery Investigations Ontology

Zhe He[1,*], Christopher Ochs[1], Larisa Soldatova[2], Yehoshua Perl[1], Sivaram Arabandi[3] and James Geller[1]

[1]NJIT, Newark, NJ, USA, [2] Brunel University, London, UK, [3]Ontopro LLC, Houston, TX, USA

ABSTRACT

The use of a top-level ontology, e.g. the Basic Formal Ontology (BFO), as a template for a domain ontology is considered a best practice. This saves design efforts and supports multi-disciplinary research. The Drug Discovery Investigations ontology (DDI) for automated drug discovery investigations followed the best practices and imported BFO. However not all BFO classes were used. Quality assurance is an important process in the development of ontologies. One methodology proven to support quality assurance is based on automatic derivation of abstraction networks (ANs) from the original ontologies. An AN of an ontology is a compact secondary network summarizing the ontology. ANs were shown to support the identification of sets of concepts with higher concentrations of errors than control sets. In this paper, an AN is derived for the DDI, based on object properties. The top node of this AN represents a set of 81 classes without any object properties. Nodes of an AN representing many classes tend to indicate modeling errors. Upon reviewing these 81 classes, we discovered that among them are most of the classes imported from BFO, and that most of these classes are irrelevant for DDI. An algorithm for hiding such irrelevant classes from a specified ontology is described. As many as 18 (56%) of the 32 BFO classes represented by the top node of the AN were hidden from DDI by the algorithm. We conclude that ontologies reusing a top-level ontology should employ this AN-based approach.

1 INTRODUCTION

The use of a top-level ontology as a template for a domain ontology is considered a best practice in ontology engineering. A well-developed top-level ontology eases the development of domain ontologies and reduces possible errors and inconsistencies. The Open Biomedical Ontologies (OBO) Foundry (www.obofoundry.org) recommends the use of the Basic Formal Ontology (BFO) (www.ifomis.org/bfo) as a top-level ontology for biomedical ontologies. The ontology of Drug Discovery Investigations (DDI) has been developed to support automated drug discovery investigations run by a Robot Scientist "Eve" (Qi et al. 2010). DDI defines the essential entities for the recording and reasoning with data about the biological activity of compounds. A logically consistent description of the data and knowledge in the project is essential for the automation of scientific discovery (King et al. 2009).

DDI has been designed to be easily extendible to and compatible with other applications. Consequently DDI uses BFO and the RO (Relations Ontology) as design templates, and extends BFO by classes for drug design. For example the class *role* has been extended by the subclasses *drug role*, *agonist role*, etc. Some imported BFO classes were left unused, e.g., the class *connected_temporal_region*. "Unused" means that there are no child classes introduced, and no object properties added to an imported class.

Since ontologies intend to facilitate the representation of semantics for humans and computers, it is important that the navigation through an ontology by humans and during automated reasoning be efficient. Therefore unused classes diminish the usability of the ontology, because they unnecessarily complicate it. The simplification of the DDI by the removal of even a few classes has a considerable impact on efficiency. Eve runs thousands of parallel experiments and records millions of data items. Therefore any unused classes should be hidden from the data recording procedures as soon as the domain ontology has reached a stable state. At the current state-of-the-art this is not standard practice, because there is no easy way to identify and hide such classes. Reasoners do not report them as problematic. Ontology evaluation criteria are not explicit in regard to unused classes.

To be reliably usable, ontologies need to go through a Quality Assurance (QA) process, e.g. as a part of a larger software system. QA may involve an auditing regimen for discovering modeling errors and inconsistencies in the ontology. Without such an auditing process, the errors in an ontology may cause a malfunction of an information system using the ontology. A part of QA should be the hiding of imported classes that are not used by the domain ontology. We refer to this process as "hiding the redundant imports."

One of the methodologies proven to support QA is based on the automatic derivation of an Abstraction Network (AN) from an original ontology (Wang et al. 2007). An AN of an ontology is a compact secondary network summarizing the structure and content of the ontology. ANs were shown to support the identification of sets of concepts with higher concentrations of errors than randomly chosen control sets (Halper et al. 2007). Focusing QA efforts on such sets increases the yield of QA personnel, measured in number of problems found and corrected per unit of time.

In this paper, an AN is derived for the DDI, based on domain-defined and restriction-defined object properties,

[*] To whom correspondence should be addressed: zh5@njit.edu

referred to as a *partial area taxonomy* (Ochs et al. 2013). The top node of this taxonomy is the *Entity* node, representing 81 classes (12.5% of all DDI classes) that do not have any object properties. Large groups of classes represented by a single node in an AN, especially by the top node, tend to have a high concentration of modeling errors (Ochs et al. 2012; Ochs et al. 2013). Upon reviewing the classes summarized by the top node, we discovered that they contain most BFO classes.

BFO has no object properties and most classes imported from BFO were not used in the DDI and should be hidden from the DDI. An algorithm for hiding this "redundant portion" of BFO from DDI will be discussed. As many as 18 (56%) of the 32 BFO classes in the top node will be hidden from a subsequent release of DDI. All ontologies importing a top-level ontology should employ this AN-based approach to QA, for hiding redundant top level classes.

2 METHODS

In previous research, two kinds of ANs, called *area taxonomies* and *partial area taxonomies* have been developed to support QA for Systematized Nomenclature of Medicine – Clinical Terms (SNOMED CT) (Wang et al. 2007). We have also derived taxonomies for OWL-based ontologies that use similar but not identical definitional elements for orientation and QA. The AN derivation and QA methodologies were successfully applied to the Ontology of Clinical Research (OCRe) (Ochs et al. 2012) and the Sleep Domain Ontology (SDO) (Ochs et al. 2013). An *area* is defined as the set of all classes that are explicitly defined or inferred as being exactly in the domains of a given set of object properties. The list of names of the object properties is used to name the area. In general, the object properties that define an area can all be "domain-defined," or all be "restriction-defined" or there may be a mix of both.

Areas are connected by child-of links derived from the underlying ontology's subclass links. A *root* of an area is defined as a class that has no parents in the same area (i.e. none of its parents share its set of object properties). An area may have more than one root. Every root of an area defines a *partial area*: a set of classes that includes this root and all its descendants in the same area. Just as areas, partial areas are connected by child-of links derived from underlying subclass links. Partial area A is a child-of partial area B if a parent (superclass) of A's root class resides in B.

Fig. 1 (a) provides an excerpt of 13 classes taken from the DDI, along with five object properties. Two object properties *is_conretized_as* and *is about* have explicit domains (in red), while three object properties *has_participant*, *has_specified_input*, and *has_specified_output* are used in class restrictions (in black). Classes that are within the domain of a particular set of object properties are shown in a dashed bubble, e.g., the class *generally_dependent_continuant* is in the domain of the object property *is_concretized_as*. *Information content entity* is explicitly defined as the domain of *is about*, but it also inherits *is_concretized_as* from *generally_dependent_continuant*. *Conformation* and *contact information* are both implicitly in the domain of *is about* and *is_concretized_as* due to inheritance from *Information content entity*.

Fig. 1(b) shows the area taxonomy for the excerpt of DDI in Fig. 1(a). *Generally_dependent_continuant*, within the domain of *is_concretized_as*, is represented by the area with the name "is_concretized_as." Child-of links are shown as lines connecting the areas. Areas are organized into color-coded levels based-on their numbers of object properties. Areas with more object properties are lower down in the diagram.

Fig. 1(c) shows the partial area taxonomy for Fig. 1(a). Partial areas are represented by white boxes within area boxes. Each partial area is named by its root. The number of classes (including the root) in a partial area is shown in parentheses. For example, in the area named "has_specified_input, has_specified_output", there are two partial areas "data item extraction from journal article" and "documenting," each containing one class. We derived the partial area taxonomy for the DDI *Entity* hierarchy (Fig. 2).

Our methodology hides all the classes that were imported from a top-level ontology T into a domain ontology

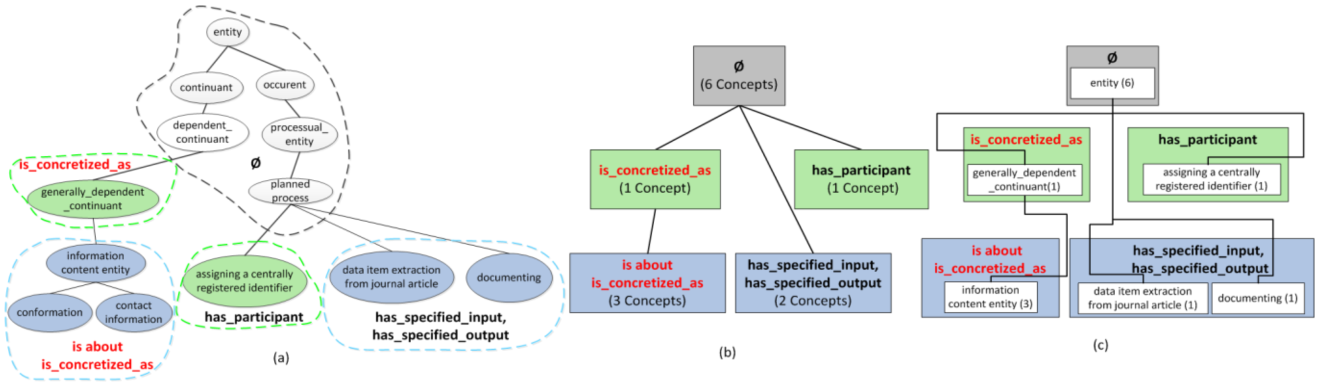

entity
continuant
occurent
dependent_continuant
processual_entity
Ø
planned process
is_concretized_as
generally_dependent_continuant
information content entity
conformation
contact information
is about
is_concretized_as
assigning a centrally registered identifier
has_participant
data item extraction from journal article
documenting
has_specified_input, has_specified_output
(a)
Ø
(6 Concepts)
is_concretized_as
(1 Concept)
has_participant
(1 Concept)
is about
is_concretized_as
(3 Concepts)
has_specified_input, has_specified_output
(2 Concepts)
(b)
Ø
entity (6)
is_concretized_as
generally_dependent_continuant(1)
has_participant
assigning a centrally registered identifier (1)
is about
is_concretized_as
information content entity (3)
has_specified_input, has_specified_output
data item extraction from journal article (1)
documenting (1)
(c)

Fig. 1. (a) An excerpt of 13 classes and 5 object properties taken from the Ontology for Drug Discovery Investigations. **(b)** The area taxonomy derived from the classes in (a). **(c)** The partial area taxonomy derived from the classes in (a).

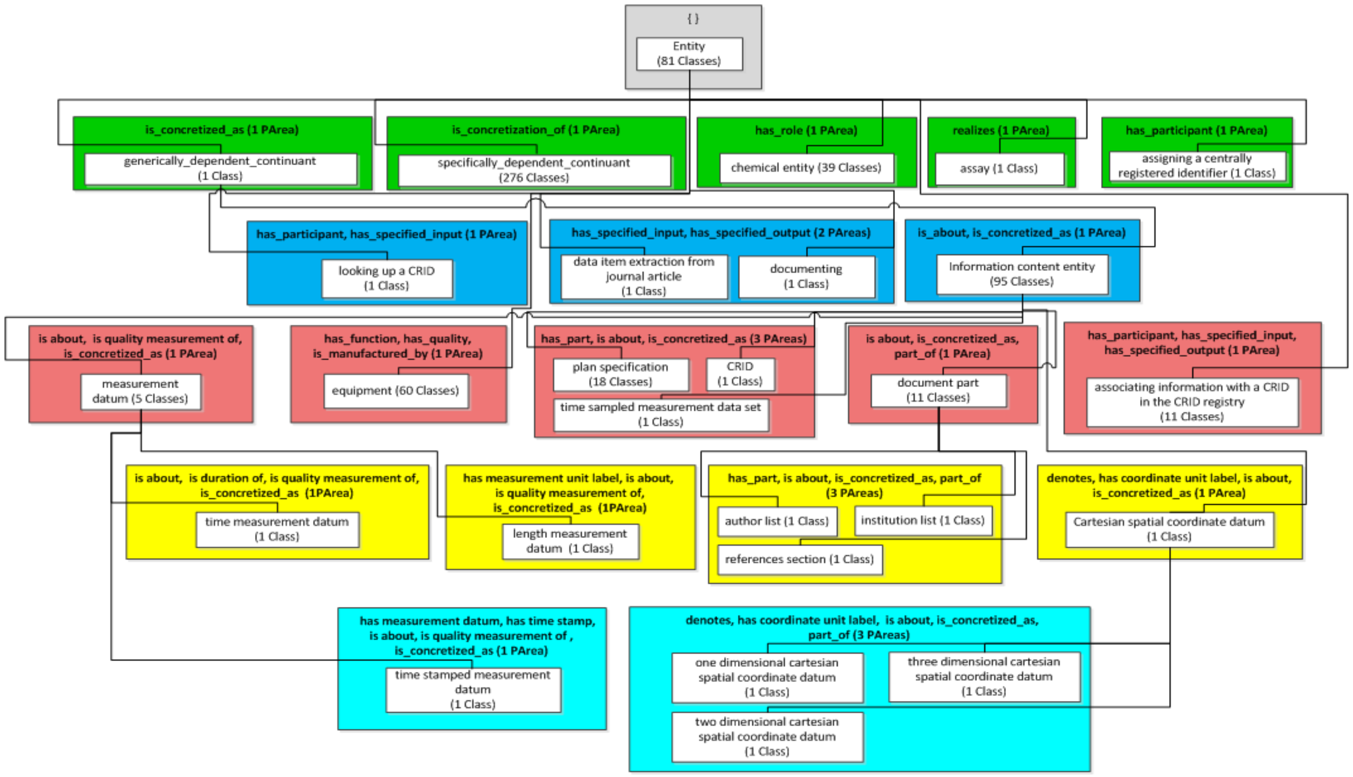

{ }
Entity (81 Classes)
is_concretized_as (1 PArea)
generically_dependent_continuant (1 Class)
is_concretization_of (1 PArea)
specifically_dependent_continuant (276 Classes)
has_role (1 PArea)
chemical entity (39 Classes)
realizes (1 PArea)
assay (1 Class)
has_participant (1 PArea)
assigning a centrally registered identifier (1 Class)
has_participant, has_specified_input (1 PArea)
looking up a CRID (1 Class)
has_specified_input, has_specified_output (2 PAreas)
data item extraction from journal article (1 Class)
documenting (1 Class)
is_about, is_concretized_as (1 PArea)
Information content entity (95 Classes)
is about, is quality measurement of, is_concretized_as (1 PArea)
measurement datum (5 Classes)
has_function, has_quality, is_manufactured_by (1 PArea)
equipment (60 Classes)
has_part, is about, is_concretized_as (3 PAreas)
plan specification (18 Classes)
CRID (1 Class)
time sampled measurement data set (1 Class)
is about, is_concretized_as, part_of (1 PArea)
document part (11 Classes)
has_participant, has_specified_input, has_specified_output (1 PArea)
associating information with a CRID in the CRID registry (11 Classes)
is about, is duration of, is quality measurement of, is_concretized_as (1PArea)
time measurement datum (1 Class)
has measurement unit label, is about, is quality measurement of, is_concretized_as (1PArea)
length measurement datum (1 Class)
has_part, is about, is_concretized_as, part_of (3 PAreas)
author list (1 Class)
institution list (1 Class)
references section (1 Class)
denotes, has coordinate unit label, is about, is_concretized_as (1 PArea)
Cartesian spatial coordinate datum (1 Class)
has measurement datum, has time stamp, is about, is quality measurement of , is_concretized_as (1 PArea)
time stamped measurement datum (1 Class)
denotes, has coordinate unit label, is about, is_concretized_as, part_of (3 PAreas)
one dimensional cartesian spatial coordinate datum (1 Class)
three dimensional cartesian spatial coordinate datum (1 Class)
two dimensional cartesian spatial coordinate datum (1 Class)

Fig. 2. Domain-defined and restriction-defined partial area taxonomy for the DDI's *entity* hierarchy

O, but were not utilized in O. In this paper, we are only considering top-level ontologies without object properties. Hence, a class C of T can be reused when constructing O by either associating object properties in O with C, or by modeling new classes of O as children or descendants of C. By limiting ourselves to classes of O in the root partial area R in the taxonomy of O, we are automatically limiting our attention to classes imported from T without any object properties in O. If a class of T has an object property in O (there are a seven such classes of BFO in DDI) then such a class cannot exist in the root partial area R of the taxonomy.

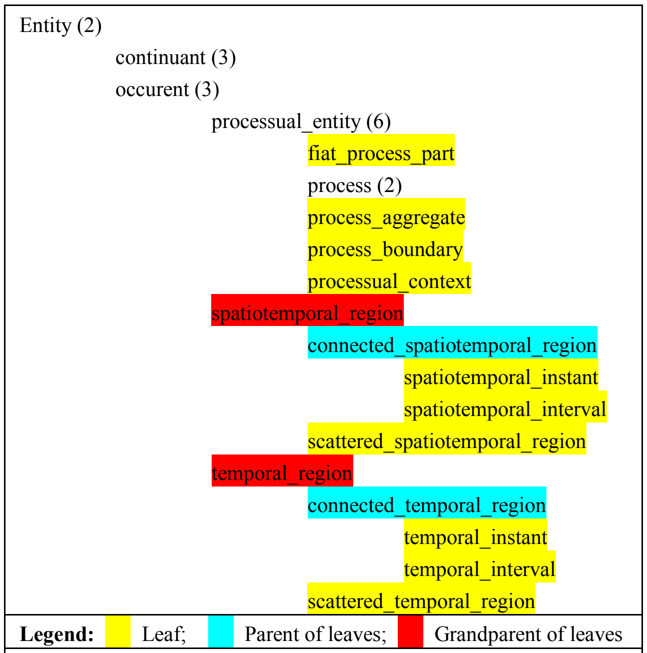
In addition, if a class in R was imported from T and is a leaf in O then, it was not utilized as the parent of any domain-related classes in O. Hence, such classes of O (T-imported classes, for short) are irrelevant for O and should be hidden. Following BFO, DDI and many BioPortal ontologies, we furthermore assume, in this paper, that no classes of T and O have multiple parents. In other words, the hierarchies of T and O are trees, which simplifies algorithmic processing.

We describe (but do *not* show) a recursive algorithm *Hide*(R, O, T, v) to automate the process of hiding unused T-imported classes from O. The algorithm uses three different contexts O, R and T, and operates in these contexts. *Hide* performs a post-order traversal of the classes in R. First, *Hide* is recursively applied to all subclasses w of the argument class v in R, initiating at R's root. When the traversal backtracks to v, the algorithm checks whether v is a leaf in O. Note that v may have been a leaf in O before, or may have become a leaf due to the removal of all its subclasses in O. Furthermore, if v was T-imported into O and is not in the range of an object property of O, then it is not used at all in O and needs to be hidden from O. By limiting the traversal to R, *Hide* is efficient with O(|R|) complexity, where R is only a small part of O, since the post-order traversal of a tree hierarchy is linear in the number of its nodes. However, the test whether a class is internal is done in O and not in R, because a leaf in R with subclasses in O was reused in O's modeling and should not be hidden from O.

3 RESULTS

DDI's *Entity* hierarchy contains 614 classes, which is 97.8% of DDI. There are 24 object properties, 14 of which are used within restrictions, 13 are given explicitly defined domains, and three have both.

First, we utilized the partial area taxonomy derivation methodology described by Ochs (Ochs et al. 2013) to derive the domain-defined or restriction-defined taxonomy of DDI's *Entity* hierarchy (614 classes) (Fig. 2). It contains 27 partial areas in 20 areas including five large partial areas (over 20 classes) and four medium size partial areas (of 5-20 classes). The other 18 partial areas include one class each. By reviewing the nine large and medium partial areas, e.g. *chemical entity* (39), *information content entity* (95), and *document part* (11), the user of DDI obtains a summary of the nature of classes in DDI. The names and sizes of these nine partial areas communicate knowledge about their content, supporting user orientation into DDI. The taxonomy also displays the interaction among 18 partial areas of one class each, for sophisticated users, e.g. DDI curator, who are interested in orientation into the fine details of DDI's content and structure.



- Entity (2)
  - continuant (3)
  - occurent (3)
    - processual_entity (6)
      - fiat_process_part
      - process (2)
      - process_aggregate
      - process_boundary
      - processual_context
    - spatiotemporal_region
      - connected_spatiotemporal_region
        - spatiotemporal_instant
        - spatiotemporal_interval
      - scattered_spatiotemporal_region
    - temporal_region
      - connected_temporal_region
        - temporal_instant
        - temporal_interval
      - scattered_temporal_region

Legend: 🟨 Leaf; 🟦 Parent of leaves; 🟥 Grandparent of leaves

Fig. 3. Excerpt of BFO classes in *Entity* partial area before hiding

Finally, we executed the algorithm *Hide*(*Entity*, DDI, BFO, *entity*), where *entity* is the root class of the root partial area *Entity*, which is the partial area the algorithm traverses.

Out of 81 classes of the partial area *Entity*, 32 are BFO-imported. An excerpt of these BFO classes, focusing on descendants of *occurent*, is shown in an indented format in Fig. 3**.** Seven other BFO-imported classes have object properties added in DDI and thus should not be hidden. Four external subclasses of *spatial region* appear in ranges of object properties, and should also not be hidden. A number in parentheses in Fig. 3 indicates the number of children in O.

The 10 BFO-imported classes highlighted in yellow do not have any added DDI class descendants and are not used in range specifications of object properties. After these classes are hidden, the two classes in blue will appear as leaves and are also hidden. Then, the classes in red are hidden when they will appear as leaves. The curator of DDI (co-author LS) will hide the 18 unused BFO classes in a new release of DDI once the hiding mechanism is supported by BioPortal. There are only 14 reused classes from BFO in the *Entity* partial area that remain after hiding the 18 unused classes**.**

4 DISCUSSION

Reuse of a top-level ontology is quite common in BioPortal, i.e., at least 36 ontologies contain BFO classes. Since an ontology designer often does not know which top-level ontology classes will later be used, the common practice is to import the whole ontology. However, once a domain ontology is mature, there is no efficient way to remove unused classes. Our methodology hides the unused imports assuming that both ontologies have a tree hierarchy. We use a partial area taxonomy to limit the input size of the algorithm.

Following (www.imbi.uni-freiburg.de/ontology/), we distinguish between top-level, top-domain and domain ontologies. Reuse is practiced at two levels. Some top-domain ontologies such as BioTop (Beisswanger et al. 2008) and OGMS (bioportal.bioontology.org/ontologies/1414) are reusing BFO, while SDO (Arabandi 2010) reuses OGMS and thus indirectly also reuses BFO. There is a need to avoid a proliferation of unused imported classes.

In this paper, both the top level and the domain ontology have a tree hierarchy, but this is not always the case. OGMS has a tree hierarchy but SDO and the BioTop top-domain ontology do not. Hence, future research needs to consider cases where both ontologies do not have tree hierarchies. Furthermore, some top-domain ontologies such as BioTop, have object properties, another research consideration.

5 CONCLUSIONS

We described a recursive linear algorithm for hiding unused imported top-level ontology classes of an OWL-based ontology. The algorithm was demonstrated, hiding BFO-imported classes from the DDI.

ACKNOWLEDGMENTS

This work has been partially supported by the BRIEF award, Brunel University, London. We thank Natasha Noy of NCBO for her help.

REFERENCES

Arabandi, S. (2010). Developing a Sleep Domain Ontology. AMIA TBI/CRI Summt. San Francisco, CA.

Beisswanger, E., S. Schulz, et al. (2008). "BioTop: An Upper Domain Ontology for the Life Sciences." Appl Ontology **3**(4): 205-212.

Halper, M., Y. Wang, et al. (2007). "Analysis of error concentrations in SNOMED." AMIA Annu Symp Proc: 314-318.

King, R. D., J. Rowland, et al. (2009). "The automation of science." Science **324**(5923): 85-89.

Ochs, C., A. Agrawal, et al. (2012). "Deriving an abstraction network to support quality assurance in OCRe." AMIA Annu Symp Proc: 681-689.

Ochs, C., Z. He, et al. (2013). "Choosing the Granularity of Abstraction Networks for Orientation and Quality Assurance of the Sleep Domain Ontology". The 4th International Conference on Biomedical Ontology. Montreal, QC, Canada.

Qi, D., R. D. King, et al. (2010). "An ontology for description of drug discovery investigations." J Integr Bioinform **7**(3).

Wang, Y., M. Halper, et al. (2007). "Structural methodologies for auditing SNOMED." J Biomed Inform **40**(5): 561-581.