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

Zhe He\textsuperscript{1}
Christopher Ochs\textsuperscript{1}
Larisa Soldatova\textsuperscript{2}
Yehoshua Perl\textsuperscript{1}
Sivaram Arabandi\textsuperscript{3}
James Geller\textsuperscript{1}

\textsuperscript{1}New Jersey Institute of Technology, \textsuperscript{2}Brunel University, \textsuperscript{3}Ontopro LLC.
Outline

• Introduction
  – Environment
  – Motivation
  – Ontology for Drug Discovery Investigations (DDI)
  – Abstraction Networks & Partial Area Taxonomy
• Algorithm $Hide$
  – Hiding Redundant BFO (Basic Formal Ontology) classes from DDI
• Future work
• Conclusions
Environment

- BioPortal: a large repository of over 340 biomedical ontologies covering a wide range of domains.
- Many ontologies in BioPortal are released in OWL or OBO format.
- OWL (Web Ontology Language): based on Description Logic, maintained by a working group of W3C.
- OBO (Open Biological and Biomedical Ontologies ) Foundry: a collaborative experiment involving developers of ontologies who are establishing a set of principles for ontology development.
Motivation

• Use a top-level ontology as a template for a domain ontology is recommended.
• OBO Foundry recommends importing BFO (Basic Formal Ontology).
• The top-domain ontologies OGMS (Ontology for General Medical Science) and BioTop (Beisswanger et al. 2008) reuse BFO.
• Some domain ontologies reuse OGMS, thereby indirectly reusing BFO.
Motivation (cont.)

- Ontologies need to go through Quality Assurance before being put to use.
  - Discovering modeling errors and inconsistencies in the design
  - Unused imported top-level classes diminish the usability of the ontology.
  - Currently, there is no mechanism to remove unused imported classes.
  - Redundant imported top-level classes should be hidden.
Ontology for Drug Discovery Investigations

- DDI was developed to support automatic drug discovery investigations run by a Robot Scientist “Eve” (Qi et al. 2010).
- DDI is used for reasoning with data about the biological activity of compounds in regards to various drug targets.
- DDI uses BFO (Basic Formal Ontology) and RO (Relations Ontology) as design templates and extends BFO and OBI (Ontology for Biomedical Investigations).
- Some imported BFO classes were left unused in DDI.
  - `connected_temporal_region`
  - `temporal_instant`
  - `temporal_interval`
Abstraction Networks

- An abstraction network is a secondary network that provides a compact view of the structure and content of the primary ontology.
- Abstraction of an ontology is the process by which subsets of classes are each replaced by a higher-level conceptual entity (node).
Partial Area Taxonomy

• Partial area taxonomy is an abstraction network developed by our research group that summarizes sets of structurally and semantically similar classes.

• Partial area taxonomies have been derived for
  – SNOMED CT (Wang et al. 2007)
  – Ontology of Clinical Research (OCRe) (Ochs et al. 2012)
  – Sleep Domain Ontology (SDO) (Ochs et al. 2013)
  – Cancer Chemoprevention Ontology (CanCo) (He et al. 2013)
  – etc.
**Area Taxonomy**

**Area**: Set of all classes that are explicitly defined or inferred as being in exactly the domain of a given set of object properties.
Partial Area Taxonomy

**Root:** Class with no superclasses in area

**Partial area:** Root + all descendants in area
Algorithm *Hide*

- *Hide* is a post order recursive algorithm requiring linear time.
- *Hide* identifies imported classes that are not used in the domain ontology.
- **Applicability:**
  - Ontologies in OWL or OBO format
  - Both domain ontology and top-level ontology are trees.
  - Top-level ontology does not have object properties.

- A Class is redundant if:
  - Imported from the top-level ontology **AND**
  - In Root partial area of the taxonomy **AND**
  - A leaf in the domain ontology (at some stage of the algorithm) **AND**
  - Not used as range of an object property
81 classes in *Entity* root partial area of DDI taxonomy
BFO has 38 classes.
32 out of 81 classes are imported from BFO.
6 BFO classes are used as domains of object properties.
Hence, we reviewed 32 classes for redundancy.
<table>
<thead>
<tr>
<th>Class</th>
<th>Number of Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>2</td>
</tr>
<tr>
<td>continuant</td>
<td>3</td>
</tr>
<tr>
<td>dependent_continuant</td>
<td>2</td>
</tr>
<tr>
<td>independent_continuant</td>
<td>3</td>
</tr>
<tr>
<td>material_entity</td>
<td>10</td>
</tr>
<tr>
<td>fiat_object_part</td>
<td></td>
</tr>
<tr>
<td>object</td>
<td></td>
</tr>
<tr>
<td>object_aggregate</td>
<td></td>
</tr>
<tr>
<td>object_boundary</td>
<td></td>
</tr>
<tr>
<td>site</td>
<td>3</td>
</tr>
<tr>
<td>spatial_region</td>
<td>4</td>
</tr>
<tr>
<td>one_dimentional_region</td>
<td></td>
</tr>
<tr>
<td>two_dimentional_region</td>
<td></td>
</tr>
<tr>
<td>three_dimentional_region</td>
<td></td>
</tr>
<tr>
<td>zero_dimentional_region</td>
<td></td>
</tr>
<tr>
<td>occurent</td>
<td>3</td>
</tr>
<tr>
<td>processual_entity</td>
<td>6</td>
</tr>
<tr>
<td>fiat_process_part</td>
<td></td>
</tr>
<tr>
<td>process</td>
<td>2</td>
</tr>
<tr>
<td>process_aggregate</td>
<td></td>
</tr>
<tr>
<td>process_boundary</td>
<td></td>
</tr>
<tr>
<td>processual_context</td>
<td></td>
</tr>
<tr>
<td>spatiotemporal_region</td>
<td>2</td>
</tr>
<tr>
<td>connected_spatiotemporal_region</td>
<td>2</td>
</tr>
<tr>
<td>spatiotemporal_instant</td>
<td></td>
</tr>
<tr>
<td>spatiotemporal_interval</td>
<td></td>
</tr>
<tr>
<td>scattered_spatiotemporal_region</td>
<td></td>
</tr>
<tr>
<td>temporal_region</td>
<td>2</td>
</tr>
<tr>
<td>connected_temporal_region</td>
<td>2</td>
</tr>
<tr>
<td>temporal_instant</td>
<td></td>
</tr>
<tr>
<td>temporal_interval</td>
<td></td>
</tr>
<tr>
<td>scattered_temporal_region</td>
<td></td>
</tr>
</tbody>
</table>

**Legend**
- **Leaf**
- Parent of classes that are all leaves
- Grandparent of grandchildren that are all leaves
BFO Classes in *Entity* Partial Area After Hiding

- 18 unused BFO classes are hidden.
- Meaning 18/32 = 56% BFO classes in *Entity* partial area are hidden.
Future Work

• As many as 35 out of 186 ontologies we investigated in BioPortal reuse BFO classes.
• Some ontologies have a Directed Acyclic Graph (DAG) hierarchy, e.g. SDO (Sleep Domain Ontology) (Arabandi 2010).
• Need to consider cases where both top-level and domain ontologies are DAG hierarchies.
• Some top-domain ontologies have object properties, e.g. BioTop.
• Need to design algorithm to deal with issues regarding redundant import of relationships in the reuse of top-domain ontologies.
Conclusions

• We described a recursive linear algorithm for hiding unused imported top-level ontology classes of an OWL-based ontology.

• The algorithm was demonstrated by hiding 18 (56%) BFO imported classes from the DDI.

• Hiding of unused imported top-level classes should be part of the Quality Assurance process of OWL-based ontologies.
References


Thank you!

Any Questions?
Algorithm of *Hide*

- **Algorithm** `Hide(R, O, T, v)`
  - IF `isInternal(O, v)` THEN
    - FOR EACH Class `w` IN `subclasses(R, v)` {
      - `Hide(R, O, T, w)`
    } END IF
  - IF NOT(`isInternal(O,v)`)) THEN
    - IF `isClassFrom(v, O, T)` AND NOT(`in_op_range(v, O)`)
      - `hide(v, O)`
    END IF
  END IF
- RETURN

**Main Program**
- // Initially, call *Hide* on the root class `r` of the root partial area `R`.
  - `Hide(R, O, T, r)`

**Function Description**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isInternal(O, v)</code></td>
<td>Boolean function that returns true if class <code>v</code> has any subclasses in ontology <code>O</code>.</td>
</tr>
<tr>
<td><code>subclasses(R, v)</code></td>
<td>Returns iterator to the set of subclasses of class <code>v</code> in root partial area <code>R</code>.</td>
</tr>
<tr>
<td><code>isClassFrom(v, O, T)</code></td>
<td>Boolean function that returns true if the class <code>v</code> in ontology <code>O</code> is imported from Top-Level ontology <code>T</code>.</td>
</tr>
<tr>
<td><code>in_op_range(v, O)</code></td>
<td>Boolean function that returns true if class <code>v</code> is in the range of an object property of ontology <code>O</code>.</td>
</tr>
<tr>
<td><code>hide(v, O)</code></td>
<td>Hides class <code>v</code> from ontology <code>O</code> and therefore also removes all subclass relationships from <code>v</code>.</td>
</tr>
</tbody>
</table>