Real Time Summarization and Visualization of Ontology Change in Protégé

Christopher Ochs[1], James Geller[1], Mark A. Musen[2], and Yehoshua Perl[1]

[1] NJIT, Newark NJ 07102, USA
[2] Stanford University, Stanford, CA 94305, USA

Abstract. Property restrictions play an important definitional role in an ontology. The correct introduction and inheritance of restrictions is important for ensuring the correct modeling of a domain. Many ontologies have large, complex class hierarchies, and many classes are defined with restrictions. When editing such an ontology, it is often difficult to determine the global impact of a local change. For example, removing a subclass link (axiom) can result in the unintentional loss of inherited restrictions over many levels. In this paper, we introduce a dynamic summarization and visualization methodology, called a Live Difference Taxonomy (LDT), to succinctly display a summary of the effects of changes to the ontology. LDTs are created on-the-fly as the ontology is edited, allowing an ontology developer to view the overall impact of their changes in a compact, visual display. We introduce an open-source plugin for the Protégé ontology editor that implements LDTs. It lets the users choose from several kinds of LDT summaries and lets them control the degree of summarization. The LDT Plugin supports heuristics-based alerts that inform the user of changes to sets of classes that are modeled with the same types of restrictions.

Keywords: Ontology change visualization, ontology summarization, Protégé plugin, Live Difference Taxonomy, dynamic ontology visualization

1 Introduction

In Web Ontology Language (OWL) ontologies, property restrictions are used to place constraints on class descriptions [1]. Property restrictions express semantic relationships between classes and they play a critical role in the reasoning process. A restriction consists of a property (e.g., an object property) and a range (e.g., a class), along with other constraints (e.g., *someValuesFrom*). In a previous analysis of the biomedical ontologies hosted on the NCBO BioPortal we found that restrictions are widely used in class definitions [2] (i.e., 279/373=74.8% ontologies included object property restrictions and 123/373=33.0% included data property restrictions).

Many ontologies have thousands of classes, each of which may have several property restrictions. While editing a large and complex ontology, it can be difficult to identify the global impact that one local change, or a sequence of changes, has on the ontology's structure. For example, removing an asserted superclass can affect the inheritance of restrictions and removing a restriction may affect the inferred class hierarchy. Identifying incorrect and unwanted changes is an important step in ensur-

ing the quality of an ontology. The goal of this research is to develop a principle-based software tool that displays the global effect of local ontology editing operations.

Various *diff* techniques, which identify changes ("differences") between two releases of an ontology, have been developed (e.g., PromptDiff [3] and OWLDiff [4]). These tools typically only identify individual changes (e.g., additions of asserted restrictions). They do not identify the overall impact of a set of changes and do not provide details about the implicit changes that occur at other classes (e.g., inferred descendants). Ochs et al. [5] identified these and other deficiencies in previous ontology diff techniques and they introduced *diff partial-area taxonomies* to address them.

Definition: A difference partial-area taxonomy ("diff taxonomy" for short) visually summarizes changes to subhierarchies of structurally similar ontology classes.

We have used diff taxonomies [5-7] to analyze changes in several ontologies (e.g., NCIt [8], SNOMED CT [9]). We were able to identify incorrect and unintended changes in these ontologies. One major deficiency of diff taxonomies, however, is that they need to be applied in an *a posteriori* change analysis process. Previously, diff taxonomies could only be created from two *fixed* versions of an ontology.

In this paper, we introduce *Live Difference Taxonomies ("LDTs")*, which are diff taxonomies created on-the-fly while an ontology is edited. LDTs dynamically summarize and visualize changes to the introduction and inheritance of restrictions in an ontology. To integrate LDTs into the ontology development workflow, we have developed a Plugin for Protégé [10]. Using this plugin, an ontology developer is provided with a visual summary that captures the overall impact of a set of local changes.

2 Background

Various diff techniques have been developed to identify changes between two ontology releases. Examples include PromptDiff [3], OWLDiff [4], and Ecco [11]. These tools display a list of individual changes or display changes in the indented hierarchy view of an ontology. Lambrix et al. [12] describe a set of functional requirements for tools that visualize ontology evolution. As described below, the LDT technique addresses several of the requirements identified by Lambrix et al. (e.g., summarization of changes, identification of dependent changes, and change metrics).

Ochs et al. [5] introduced a graphical, summary-based ontology diff technique called a difference partial-area taxonomy (or ***diff taxonomy***), which visually summarizes changes to sets of structurally and semantically similar classes (i.e., those modeled with similar restrictions). We will now briefly describe, using Fig 1, how diff taxonomies are derived using an example based on the Pizza Ontology [13]. The complete derivation algorithm for diff taxonomies was described by Ochs et al. [5].

Definition: A ***diff area*** is a summary of changes to a set of classes that are all modeled with restrictions that utilize the same property types.

For example, we are starting with the ontology in Fig 1(a) and apply several changes, resulting in the ontology in Fig 1(b). Fig 1(c) shows a succinct summary, using a diff taxonomy, of these changes. The black box (in the middle) in Fig 1(c) summarizes two classes that had no changes to their restrictions.

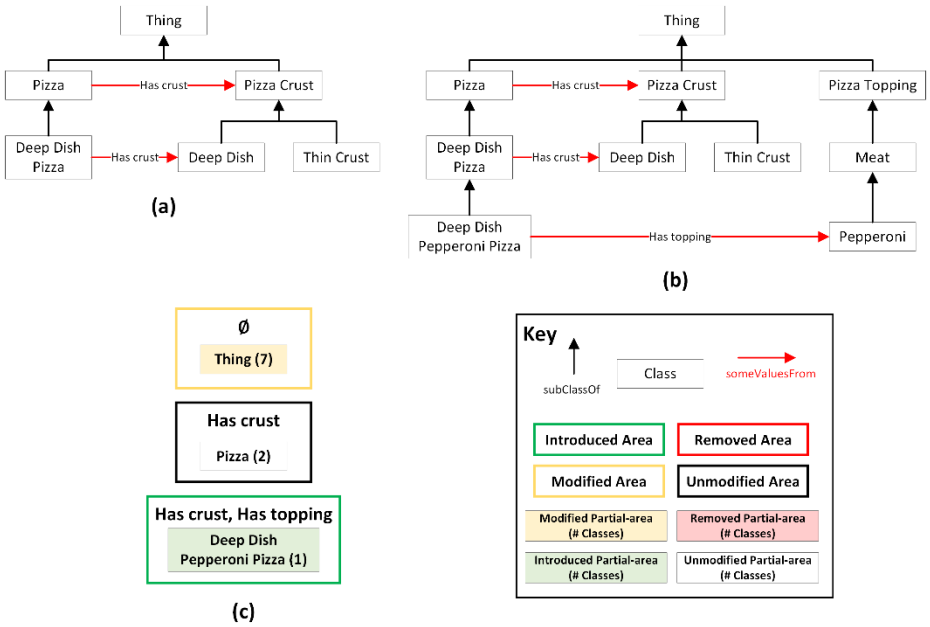

Fig 1. (a) A small ontology with six classes. *Pizza* and *Deep Dish Pizza* are modeled with *Has crust* someValuesFrom restrictions. **(b)** The ontology from (a) after several changes were applied. A subhierarchy of pizza toppings, modeled without restrictions, was added. *Deep Dish Pepperoni Pizza* was added as a subclass of *Deep Dish Pizza* and with a *Has topping* someValuesFrom restriction with a range of *Pepperoni*. **(c)** The diff taxonomy for going from (a) to (b).

The green (bottom) box in Fig 1(c) "summarizes" one class that was added to the ontology when going from Fig 1(a) to Fig 1(b), and it is modeled using restrictions with two property types. The three "outside" boxes in Fig 1(c) are diff areas. Seven classes without restrictions (without red arrows, and without ancestors with red arrows) in Fig 1(b) are summarized by *Thing* (7) in Fig 1(c).

Definition: The ***state*** of a diff area is one of {*unmodified*, *modified*, *introduced*, *removed*}, depending on the differences in the set of classes of this diff area between two ontology releases.

For example, consider the set of classes in Fig 1 without restrictions (i.e., they have no "outgoing" red arrows and none of their ancestor classes has an outgoing red arrow). This set changed when three pizza topping classes were added to the ontology (and the pizza topping classes have no restrictions). Thus, the diff area for the set of classes with no restrictions is *modified*, since the set of classes is different in Fig 1(a) and Fig 1(b). A modified area is color-coded with a yellow border.

The set of classes modeled with only *Has crust* restrictions did not change between the two releases, thus, the diff area {*Has crust*} is *unmodified* (black box). In Fig 1(b) there is a new class *Deep Dish Pepperoni Pizza*, which inherits a *Has crust* restriction from its superclass and introduces a new *Has topping* restriction. In Fig 1(a) there are no classes with both *Has crust* and *Has topping* restrictions. Thus, the diff area con-

taining *Deep Dish Pepperoni Pizza* is *introduced* (border color: green). A diff area is *removed* when there are no longer any classes with a given set of properties used in restrictions (examples will be shown in Fig 2 and Fig 3).

Within each diff area there may be one or more subhierarchies of classes that are (or were, in the case of *removed* areas) modeled with the same set of property types. In Fig 1(c), each diff area only has **one** such subhierarchy. However, there may be several subhierarchies in an area. In such a case, there would be several "inner boxes" in an "outer box." We will now formalize this idea.

Definition: A ***root class*** of a diff area is a class where none of its superclasses exists in the same diff area.

In Fig 1, *Pizza* is a root in {*Has crust*} and *Deep Dish Pepperoni Pizza* is a root in the introduced area {*Has crust, Has topping*}. The classes in a diff area are structurally similar, since they all have restrictions with the same types of properties. Classes in a diff area are semantically similar when they are descendants of the same *root class*.

Subhierarchies in a given diff area will change, in terms of the set of classes in these subhierarchies, when an ontology is modified. For example, adding another subclass of *Pizza*, *Thin Crust Pizza*, modeled with no additional restrictions, will result in the set of *Pizza* classes with only *Has crust* restrictions changing. We capture this idea in the following definition.

Definition: A ***diff partial-area*** is a summary of changes to the set of classes in a specific subhierarchy inside of a diff area.

In Fig 1(c), *Thing* (7), *Pizza* (2), and *Deep Dish Pepperoni Pizza* (1) are the diff partial-areas derived from the differences between Fig 1(a) and Fig 1(b). Fig 4 below will show examples of several diff areas that contain multiple diff partial-areas.

Definition: The *state* of a diff partial-area is one of {*unmodified, modified, introduced, removed*}, according to the difference in the set of classes in the subhierarchy between two releases (just like for a diff area).

Thing (7) is a modified diff partial-area, as the subhierarchy of classes with no restrictions under *Thing* had three additional pizza topping classes added. *Pizza* (2) is unmodified, as there were no additional descendants of *Pizza* with only *Has crust* restrictions added, and *Deep Dish Pepperoni Pizza* (1) is an introduced diff partial-area, since there was previously no *Deep Dish Pepperoni Pizza* class and there were no subhierarchies of classes with *Has crust* and *Has topping* restrictions in Fig 1(a).

Diff partial-areas highlight changes to the *introduction* of additional property types used in restrictions. The root class of a diff partial-area has at least one additional type of property in a restriction in comparison to all of its superclasses. Diff partial-areas also highlight changes to the inheritance of restrictions, as all of the classes in a diff partial-area inherit or refine the restrictions of the root class (e.g., the *Deep Dish Pizza*'s *Has Crust* restriction has a range of *Deep Dish*, more refined then *Pizza Crust*).

Definition: A ***diff taxonomy*** is a diagram of all of the *diff areas* and *diff partial-areas* derived from two releases of the same ontology. It summarizes the structural and semantic changes that occurred when going from the old to the new release.

Diff taxonomies provide a *summary of changes* that highlights significant differences in the introduction and inheritance of restrictions, while hiding other information (e.g., lists of changes to individual classes).

Diff taxonomies are represented as graphical diagrams as follows. Diff areas are shown as boxes with a bold outline, labeled with the set of properties used in restrictions. Diff areas are organized into levels according the number of property types. The box of a diff area is black for *unmodified*, yellow for *modified*, green for *introduced*, and red for *removed*. For example, in Fig 1(c), the diff area for the set of classes with no restrictions is outlined in yellow, since it is modified.

Diff partial-areas are shown embedded in their respective diff areas. Each diff partial-area is labeled with the name of the subhierarchy's root class and the number of classes in the subhierarchy (in parenthesis) in the current ontology release. The fill color of diff partial-areas is white for *unmodified*, and the same as for areas otherwise.

Thus, diff taxonomies visually identify changes to the introduction and inheritance of restrictions (according to the types of properties in the restrictions; Fig 1(c)). For example, the introduced partial-area *Deep Dish Pepperoni Pizza* (1), with a green background in a green bordered diff area, identifies a new combination of properties used in restrictions in the ontology. The diff partial-area of *Pizza* classes in the diff area *Has crust* did not change, as it has a white fill in a diff area with a black outline.

3 Methods

Given an ontology O, we define O_{before} as O prior to a sequence of changes being applied. Similarly, we define O_{after} as O after one or more changes have been applied. For the example, Fig 1(a) is O_{before} and Fig 1(b) is O_{after}. Diff taxonomies summarize the changes between O_{before} and O_{after}. In our previous research, both O_{before} and O_{after} had to be completely known. This requirement severely limited the utility of diff taxonomies, as they could not be applied dynamically during the ontology development process. In this section, we describe how we overcame this limiting restriction.

We define a ***Live Difference Taxonomy (LDT)*** as a diff taxonomy that is updated dynamically, as changes are applied to its underlying ontology. Given a sequence of n changes $C = \{c_1, c_2, c_3, \ldots, c_n\}$ an LDT highlights the effect of each change immediately in a visual manner (following the graphical scheme described in the Key of Fig 1). The development of LDTs required significant enhancements to the diff taxonomy methodology, which we will now describe in detail.

3.1 Fixed-point LDT and Progressive LDT

As changes are applied, an ontology developer may be interested in viewing the overall effects of a sequence of changes or the intermediate effects of every individual change. To accommodate these different needs we define two types of LDTs:

A *Progressive LDT* is an LDT that shows the effects of exactly one change. Given a change c, in a Progressive LDT, O_{before} is the ontology before c was applied and O_{after} is the ontology after c was applied. Using a Progressive LDT will show the intermediate effects of each change. Fig 2 illustrates a series of six changes and the six corresponding Progressive LDTs, starting with the class *Thing* only, and no restrictions.

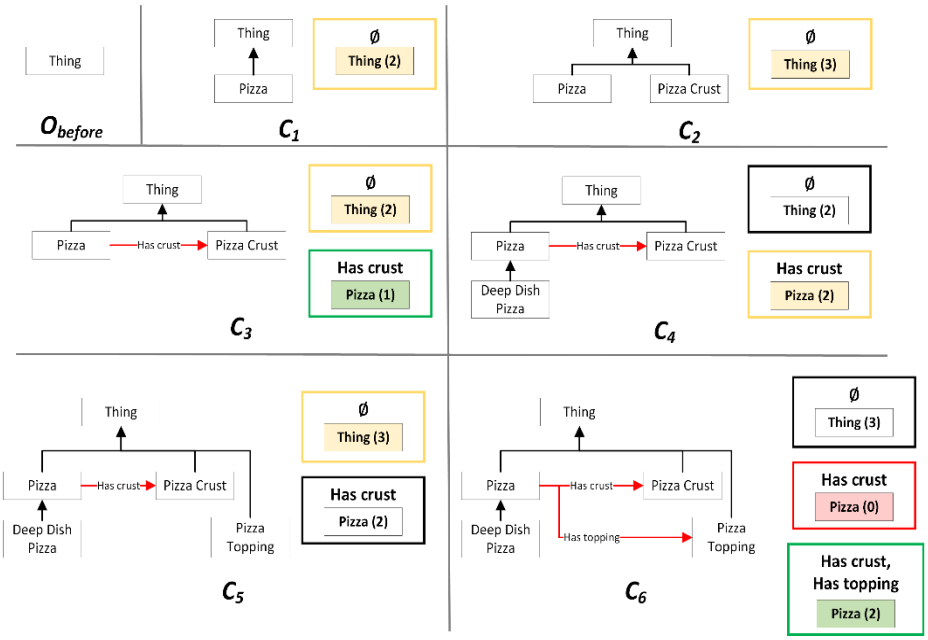

Fig 2. An example of a Progressive LDT, where, for each change $\{c_1, c_2, c_3, \ldots, c_6\}$, the ontology is shown to the left and the corresponding Progressive LDT to the right.

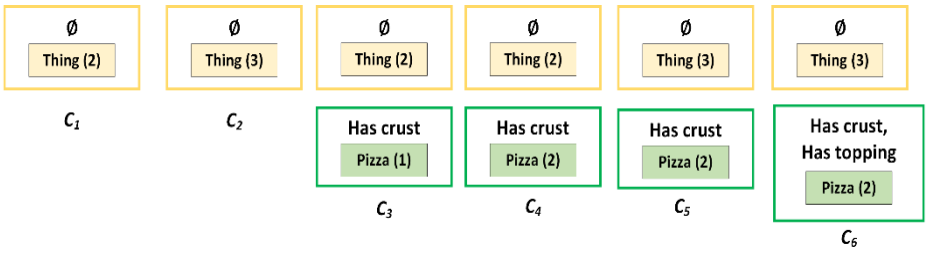

Fig 3. An example of a Fixed-Point LDT, using O_{before} in Fig 2 as the starting point. For each change in Fig 2, the Fixed-Point LDT is shown.

A *Fixed-Point LDT* is an LDT generated according to a selected "starting point" *s*, which defines O_{before}. All changes prior to *s* are ignored. As each change is applied, the LDT is updated to capture the effects of the whole sequence of changes up to the current change. When deriving a Fixed-Point LDT O_{after} is the ontology resulting from the sequence of changes up to the current change. A Fixed-Point LDT, thus, summarizes the overall effects of several changes. Fig 3 illustrates the sequence of Fixed-Point LDTs for the same sequence of six changes, again starting with *Thing* only.

3.2 Subsets of Live Difference Taxonomies

For large ontologies it may be impractical, or undesirable, to view an LDT for the complete ontology. An ontology developer may only be interested in specific parts of the ontology or in specific change operations. We have developed four such specialized LDTs which we will now describe.

Subhierarchy LDT: An ontology developer may want to focus on changes in a specific subhierarchy of classes (e.g., only the *Pizza Topping* subhierarchy). In a *Subhierarchy LDT* a class *c* is selected and the subhierarchy of classes rooted at *c* is used when deriving the LDT.

Property Type and Use LDT: An ontology developer may only be interested in the changes of a specific type of restriction (e.g., a restriction with an object property or a restriction with a data property). Restrictions may be used in different ways. Uses include *restrictions that appear as superclasses* and *restrictions that appear in class equivalences* (i.e., as necessary conditions or as necessary and sufficient conditions). Further refinements (e.g., only *someValuesFrom* or *allValuesFrom*) are also possible. In a *Property Type and Use LDT* a type of property and a use of that property, are selected. The LDT is derived using only the restrictions that meet the selected criteria; all other restrictions are ignored.

Property Subset LDT: A developer may only be interested in restrictions that utilize a certain subset of property types (e.g., only restrictions with the *Has crust* property). In a *Property Subset LDT* a subset of properties is selected and the LDT is derived using restrictions that utilize these selected properties.

Combinations of the above three LDTs can be expressed by a developer to specify subset(s) of an ontology, from which an LDT is derived and displayed. This will allow the developer to focus in on small, specific areas of the ontology that are of interest. Given the large size of ontologies such as NCIt and SNOMED CT, such focusing mechanisms are essential. Lastly, when deriving an LDT, an ontology developer may only be interested in specific kinds of changes.

Change Type LDT: In a *Change Type LDT* a subset of diff areas and diff partial-areas is selected according to their *state*. An ontology developer can create a display containing only *introduced* areas. A Change Type LDT allows an ontology developer to focus on specific types of changes in the LDT. For example, in Fig 3, only green (i.e., introduced) diff areas and green diff partial-areas would appear in this case.

3.3 Asserted LDTs and Inferred LDTs

In prior research we derived diff taxonomies using an ontology's inferred class hierarchy, as we were interested in how classes changed from the perspective of an end user. However, ontology developers modify the asserted axioms of an ontology. Thus, it is necessary to provide an LDT view that can also capture changes to the asserted class hierarchy. For each of the above-described LDT types, the asserted class hierarchy or the inferred class hierarchy can be utilized as the starting point.

3.4 Protégé LDT Plugin

Protégé [10] is a widely used software tool for editing OWL ontologies. To integrate LDTs into the ontology development workflow, we have designed and implemented a plugin for Desktop Protégé. The **LDT Plugin** provides an interactive, graphical display for LDTs. Whenever a user makes a change to the ontology in Protégé, the LDT Plugin immediately displays the corresponding LDT. Using the LDT Plugin, developers can immediately see the impact of the changes as they affect the ontology. The plugin was developed using components from the Ontology Abstraction Framework (OAF) [14], our software framework for creating visual summaries of ontologies, the OWL API [15], and Protégé's APIs.

Unlike the Ontology Abstraction Network (OAF) software tool, which shows ontology summaries in a full screen display, the Live Difference Taxonomy Plugin was designed to be included anywhere in the Protégé user interface. A user may even include multiple LDT displays on one Protégé tab, e.g., one for an Asserted LDT and one for an Inferred LDT, or one for a Subhierarchy LDT and another one for a Property Type and Use LDT.

The guiding design principle for the LDT Plugin was the minimization of information displayed to avoid information overload of the user. The options menus were also designed to enable one-click functionality (e.g., for resetting the starting point of a Fixed-Point LDT, switching between different types of LDTs, and switching between asserted and inferred axioms).

LDT-based Alerts: According to the type(s) of changes applied to an ontology, and the resulting LDT, we can define various heuristics that can be used to alert ontology developers to potentially unintended consequences of their changes. These heuristics are partially based on findings from our previous change analysis [5-7] of NCIt, SNOMED CT, etc.

For example, in NCIt we found [6] that when a class moves to the special diff area for classes with zero restrictions in an Inferred LDT, this may be unintended, as the class previously had at least one restriction (either an asserted restriction or an inherited restriction) and now it has none. If an ontology developer did not explicitly remove the restriction from the class, the LDT plugin can display an alert that identifies the details of the change. It is up to the developer to undo or keep the change.

Source code: The Live Difference Taxonomy plugin is provided as open source software on the GitHub web site (https://github.com/NJITSABOC/oaf-protege) and reference implementations of the algorithms for the above LDT types are available in the source code.

In SNOMED CT we identified [7] significant (unintended) changes in the ontology's inferred hierarchy after a subset of classes had their asserted modeling changed. These unintended changes were captured as over 100 introduced partial-areas that contained only classes with no changes in their asserted axioms. When *introduced* and *removed* diff partial-areas appear in an Inferred LDT, and none of the classes in the diff partial-areas were edited by the developer, the changes exposed by the LDT may have been unintentional, and the ontology developer should be alerted.

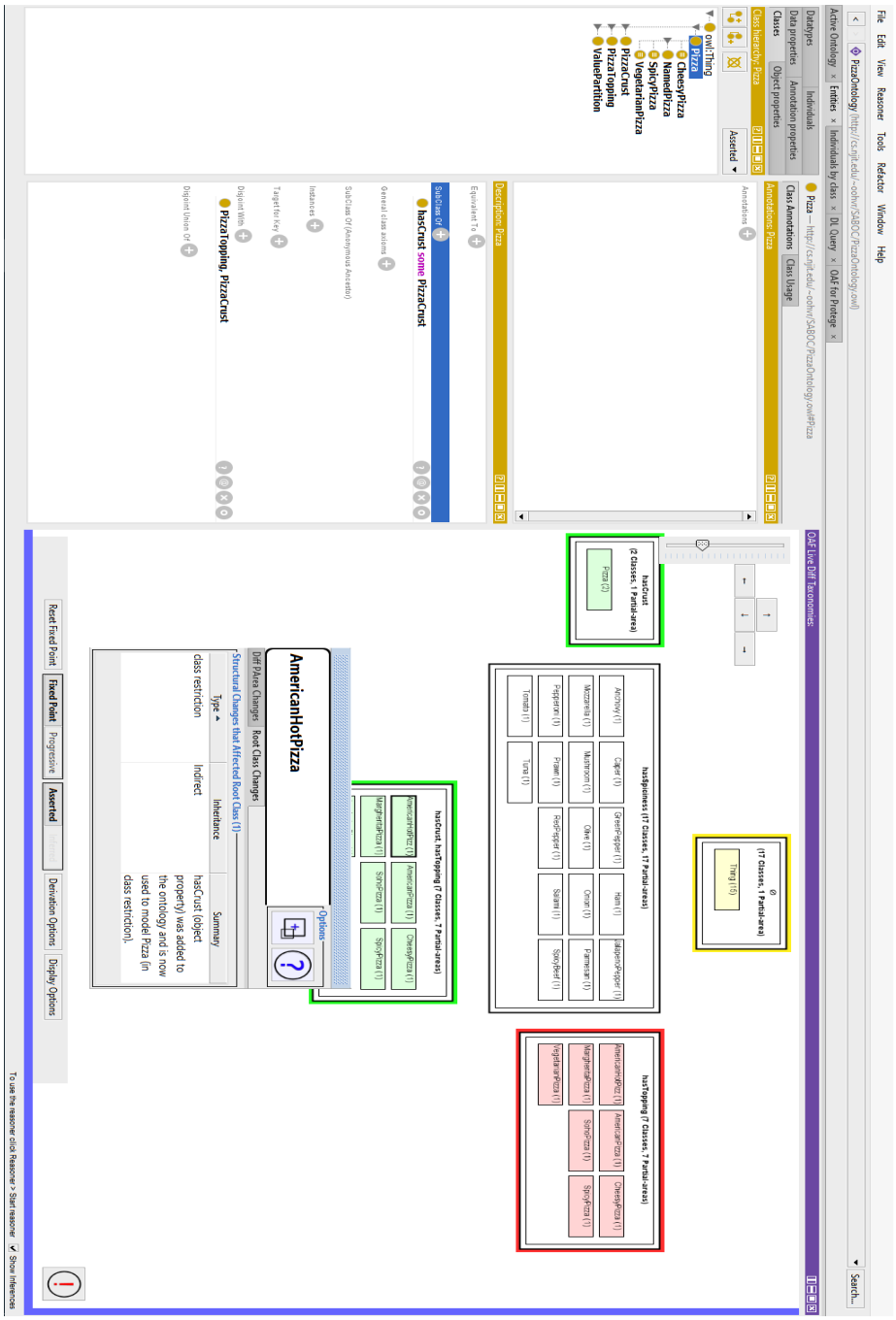

Fig 4. The Protégé Live Diff Taxonomy Plugin (right side) included on Protégé's *Entities* tab, displaying a Fixed-Point Asserted LDT for the Pizza Ontology after a *hasCrust* restriction was added to *Pizza*. The AmericanHotPizza (1) introduced partial-area has been selected.

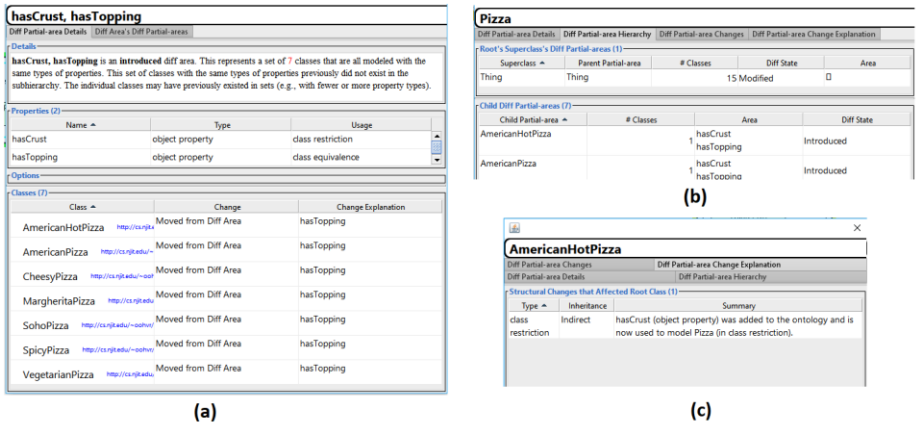

Fig 5. (a) LDT change details for the introduced area {*hasCrust, hasTopping*}. **(b)** LDT change details for *Pizza* introduced partial-area, showing LDT state of more/less refined diff partial-areas. **(c)** Change explanation for AmericanHotPizza(1) introduced partial-area.

4 Results

Live Difference Taxonomies, and Subset LDTs, have been implemented in the Protégé LDT Plugin. The Protégé LDT Plugin is available as open source software. A beta version of the plugin is available at http://saboc.njit.edu/software.php for Protégé 5.

Fig 4 shows the user interface of the LDT Plugin with the Pizza Ontology. The LDT displayed in this example is an Asserted Fixed-Point LDT after *hasCrust* restrictions were added to *Pizza* and its descendants. At the bottom left of the plugin the LDT options panel is displayed. From this options panel a user can reset the starting point for a Fixed-Point LDT, choose between Fixed-Point LDTs and Progressive LDTs (Section 3.1), toggle the use of inferred axioms (after a reasoner had been applied; Section 3.3), and choose to view a subset of an LDT (Section 3.2).

Clicking on any *diff area* (“outer box”) or *diff partial-area* (“inner box”) provides a menu for obtaining information about it and more details about what it represents. Fig 5(a) shows details for the *diff area* {*hasCrust, hasTopping*}. Fig 5(b) shows details for the set of classes with *hasCrust* and *hasTopping* restrictions.

When selecting an LDT diff area or diff partial-area, a list of ontology changes that affected its classes can also be displayed. For example, for each class in the {*hasCrust, hasTopping*} diff area, the addition of the *hasCrust* restriction caused the diff area, and its diff partial-areas, to be *introduced*. The plugin identifies the addition of the *hasCrust* restriction on *Pizza*, and its inheritance by its descendants, as the cause for the introduction of the diff area and diff partial-areas (Fig 5(c)).

Selecting a class from within Protégé will center the LDT Plugin view on the diff area that contains the selected class. Similarly, selecting a class from within the LDT Plugin display will display the selected class in the Protégé editor. This allows a user to quickly transition between the Protégé view and the LDT Plugin view.

5 Discussion and Future Work

As part of a long-term research project, we have developed various types of ontology summaries called *abstraction networks* [16] to compactly visualize the structure and semantics of an ontology. In this paper, we have extended this notion from abstraction networks of ontologies to **dynamic** abstraction networks of *changes* in ontologies. While we have previously investigated *diff taxonomy* abstractions networks, those were always static. The algorithms for deriving such abstractions networks had to be applied *a posteriori* to two fixed versions of an ontology.

Live Difference Taxonomies, and the development of the Protégé LDT Plugin, are important steps toward **(1)** creating dynamic, visual summaries of an ontology's structure and how that structure changes, **(2)** integrating abstraction networks (and abstraction-network-based methodologies) into the ontology development workflow, and **(3)** developing software tools for dynamic ontology summarization.

LDTs allow an ontology developer to zero in on the effects of changes immediately after making them and let her/him identify incorrect and inconsistent changes in the asserted and inferred relationships in the ontology and, ideally, correct them. This study has covered the underlying theory and implementation of the LDT Plugin. We are planning evaluation studies to investigate the usability of the Plugin user interface and its efficacy for enabling the correction of errors during ontology development.

One issue we are currently investigating is the scalability of the LDT user interface and derivation methodology to large ontologies. In terms of the user interface, the *subset LDTs* described in this paper can control the amount of information displayed on screen. We are investigating a heuristic-based system for automatically creating subset LDTs that focus-in on the most important changes in a large ontology.

The current LDT derivation methodology requires two copies of the ontology (O_{before} and O_{after}). For very large ontologies, with tens of thousands of classes, this causes slowdowns in the LDT Plugin, (i.e., when creating inferred LDTs, as the entire ontology has to be reasoned before the inferred LDT is derived). Optimizations for large ontologies will be investigated during the continued development of the LDT Plugin.

6 Conclusions

Live Difference Taxonomies dynamically summarize, in a visual way, changes to the introduction and inheritance of restrictions in an ontology. Several types of LDTs can be derived using both asserted and inferred axioms. An LDT Plugin for Protégé was described and is available for download. Any changes of the ontology in Protégé are dynamically reflected in the Live Difference Taxonomy displayed in its own tab.

Acknowledgements

We thank Hao Liu and Kevyn Jaremko for their contributions to the LDT Plugin. Research reported in this publication was supported by the National Cancer Institute of the National Institutes of Health under Award Number R01 CA190779. The con-

tent is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

1. Motik, B., Patel-Schneider, P. F., Parsia, B. OWL 2 Web Ontology Language Structural Specification and Functional Style Syntax. W3C -- World Wide Web Consortium, 2009.
2. Ochs, C., He, Z., Zheng, L., et al. Utilizing a structural meta-ontology for family-based quality assurance of the BioPortal ontologies. J Biomed Inform. 2016;61:63-76.
3. Noy, N. F., Musen, M. Promptdiff: A fixed-point algorithm for comparing ontology versions. AAAI/IAAI 2002. 2002:744-50.
4. Kremen, P., Smid, M., Kouba, Z. OWLDiff: A Practical Tool for Comparison and Merge of OWL ontologies. 22nd International Workshop on Database and Expert Systems Applications. 2011:229-33.
5. Ochs, C., Perl, Y., Geller, J., et al. Summarizing and Visualizing Structural Changes during the Evolution of Biomedical Ontologies Using a Diff Abstraction Network. Journal Of Biomedical Informatics. 2015;56:127-44.
6. Perl, Y., Ochs, C., Coronado, S. d., et al. Visualizing the "Big Picture" of Change in NCIt's Biological Processes. ICBO. 2016.
7. Ochs, C., Case, J. T., Perl, Y. Analyzing Structural Changes in SNOMED CT's Bacterial Infectious Diseases Using a Visual Semantic Delta. J Biomed Inform. 2017:101-16.
8. Fragoso, G., de Coronado, S., Haber, M., et al. Overview and utilization of the NCI thesaurus. Comp Funct Genomics. 2004;5(8):648-54.
9. Stearns, M. Q., Price, C., Spackman, K. A., et al. SNOMED clinical terms: overview of the development process and project status. Proc AMIA Annu Symp. 2001:662-6.
10. Musen, M. A. The protégé project: a look back and a look forward. AI Matters. 2015;1(4):4-12.
11. Gonçalves, R. S., Parsia, B., Sattler, U. Ecco: A Hybrid Diff Tool for OWL 2 ontologies. OWLED. 2012.
12. Lambrix, P., Dragisic, Z., Ivanova, V., et al. Visualization for Ontology Evolution. VOILA. 2016:54-67.
13. Horridge, M. A Practical Guide to Building OWL Ontologies Using Protege 4 and CO-ODE Tools: University of Manchester; 2011.
14. Ochs, C., Geller, J., Perl, Y., et al. A Unified Software Framework for Deriving, Visualizing, and Exploring Abstraction Networks for Ontologies. J Biomed Inform. 2016;62:90-105.
15. Horridge, M., Bechhofer, S. The OWL API: A Java API for Working with OWL 2 Ontologies. OWLED. 2009;529:11-21.
16. Halper, M., Gu, H., Perl, Y., et al. Abstraction Networks for Terminologies: Supporting Management of "Big Knowledge". Artificial intelligence in medicine. 2015;64(1):1-16.