ReTAX++: a Tool for Browsing and Revising Ontologies

Sik Chun (Joey) Lam, Derek H. Sleeman, Wamberto Vasconcelos
Dept. of Computing Science
University of Aberdeen
Aberdeen, Scotland, UK
{slam, sleeman, wvasconc}@csd.abdn.ac.uk

Abstract

Many existing ontology tools provide an integrated environment to browse and edit ontologies as well as inconsistency checking facilities. However, their visualization facilities are limited and guidance on how to correct the detected errors is not usually provided. We present our ontology editor, ReTAX++, a tool that facilitates browsing and revision of ontologies.

1 Introduction

Many exiting ontology tools such as Protégé[1], WedODE[2], Oiled[3] and OntoEdit[4] provide list-based representations for selection of classes or pop-up windows for manipulating classes. However, users usually find it difficult to browse the logical structure of ontologies graphically, and only OWLViz[5], the plugin of Protégé, provides graphical visualization facilities which are static. These tools are also able to check for errors and inconsistencies by connecting with an external reasoner. However, only Protégé-OWL[1], OWLDebugger[6] is able to help users track down the reasons for OWL classes being inconsistent; SWOOP[7] presents errors in ontologies to users in simple natural language. Other tools provide no explanation and functionalities for users to correct the detected defects and errors.

We propose a graph-based approach implemented in ReTAX++ (see Figure 1) to help knowledge engineers browse ontologies and resolve the inconsistencies. When one wants to reuse an ontology by importing it onto the system, the ontology is displayed in a graphical format. With the help of a reasoner, the system detects and highlights the inconsistent concepts. We propose graph-based algorithms to detect which relationships among concepts cause the inconsistencies, and provide options for the user to correct them. If an incomplete or inconsistent ontology is imported, a number of ontological fragments may still be formed. In this case, we aim to suggest to the user with the best concept candidate to integrate the fragments.

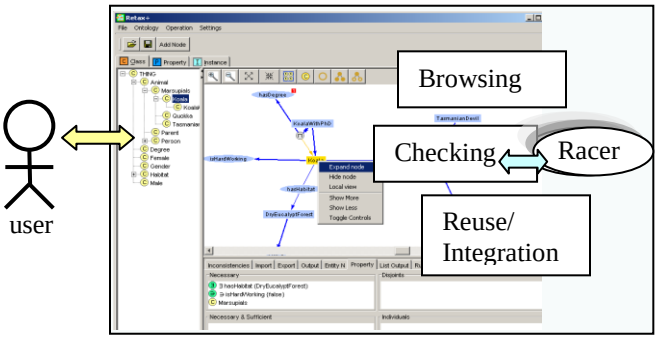


Figure 1. The structure of ReTAX++

2 Browsing

A screenshot of the ReTAX++ is shown in Figure 1. An ontology is displayed in a graphical format using TouchGraph[8]. The user can choose to browse the subsumption relationships, all concept relationships, individual relationships or a specific concept with detailed information. ReTAX++ also provides animated expansion and contraction of class hierarchy, zooming, panning, as well as mouse-enabled editing features.

3 Contradiction Checking

We formally represent an ontology as a directed graph with a finite set of nodes and edges. There are three types of nodes, *viz.* the concepts and relations of an ontology are represented as nodes in the graph; the intersection, union and complement operations between concepts are represented as $\sqcup$, $\sqcap$, $\neg$ nodes respectively. The edges represent the relationships among the nodes. In this paper we only illustrate how a complement contradiction is

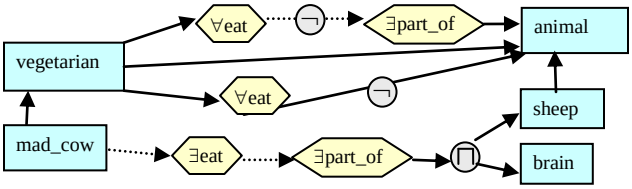
[1] http://protege.stanford.edu/
[2] http://delicias.dia.fi.upm.es/webODE/
[3] http://oiled.man.ac.uk/
[4] http://www.ontoprise.de/home
[5] http://www.co-ode.org/downloads/owlviz/
[6] http://protege.stanford.edu/conference/2005/submissions/posters/poster-drummond.pdf
[7] http://www.mindswap.org/2004/SWOOP/
[8] http://www.touchgraph.com/

checked and resolved, therefore, instances, enumerated concepts and cardinality restrictions etc. are not considered. Paths in a graph, represented as Π, are alternating sequences of nodes and edges such that each edge in the sequence is preceded by its source node. We assume that there is at most one edge between any two nodes, therefore, an edge in a graph can always be determined by its source and target nodes. Hence, a path can be is abbreviated by just enumerating the nodes $\langle n_0, n_1, \ldots, n_m \rangle$. For brevity, the nodes are written as the concept names, relation names and the concept operation symbols.

Complement contradiction occurs if a concept is defined to have a relationship with a concept but the concept must not have a relationship with that concept simultaneously. Given an ontology and a concept, we can find a set of concept paths $\Phi=\{\Pi_1, \Pi_2, \ldots, \Pi_n\}$, the first node of the paths is always the concept. Each path Π_i with a $\neg$ node is compared to each path Π_j without $\neg$ node. By comparing these paths, and converting them to negated normal form, if necessary, to make the relation nodes with the same restriction, we can infer which relationships among concepts may cause the contradiction. The two paths Π_j, Π_i are complement contradictory if the following conditions hold:

1. No $\sqcup$ node exists in both paths Π_j and Π_i, otherwise, they may be alternative.
2. The relation nodes preceding the $\neg$ node in Π_i must have the same name but different restrictions with the corresponding nodes in Π_j.
3. The relation nodes after the $\neg$ node in Π_i must have the same name and restriction with the corresponding nodes in Π_j.

The following example shows how we check which relationships cause the concept mad_cow inconsistent. mad_cow is defined that its individuals have at least one eat relationship to an individual which is at least a part of animal and of brain. vegetarian is defined that its individuals only have the eat relationship to individuals which are not part of animal, and mad_cow is defined as being vegetarian. We now illustrate how the contradiction is checked using graph-based algorithms.



vegetarian $\sqsubseteq \forall$eat.($\neg$animal) $\sqcap \forall$eat($\neg\exists$part_of.animal)
mad cow $\sqsubseteq \exists$eat.($\exists$part of.(brain $\sqcap$ sheep))

Figure 2. mad_cow example

Firstly, a set of paths are found in the graph,
$\Pi_1 = \langle$mad_cow, animal$\rangle$
$\Pi_2 = \langle$mad_cow, vegetarian, $\forall$eats, $\neg$, animal$\rangle$
$\Pi_3 = \langle$mad_cow, vegetarian, $\forall$eats, $\neg$, $\exists$part_of, animal$\rangle$
$\Pi_4 = \langle$mad_cow, $\exists$eats, $\exists$part_of, $\sqcap$, sheep$\rangle$
$\Pi_5 = \langle$mad_cow, $\exists$eats, $\exists$part_of, $\sqcap$, brain$\rangle$

A relation node in a path can be converted to negated normal form in order to make the restriction of relation nodes to be the same, $\Pi_3' = \langle$mad_cow, vegetarian, $\neg$, $\exists$eats, $\exists$part_of, animal$\rangle$. As sheep is a subconcept of animal, it is substituted by animal in Π_4, therefore, $\Pi_4'=\langle$mad_cow, $\exists$eats, $\exists$part_of, $\sqcap$, animal$\rangle$. This identifies a complement contradiction located in paths Π_3' and Π_4', and the three conditions are held by them.

Finally, the system provides the following options to resolve the complement contradiction:

1. Remove the complement axiom by eliminating the $\neg$ node in the graph.
2. Change the restriction of any relation node in one of paths. In this example, either the restriction of eat relation of mad_cow can be changed to $\forall$, or the restriction of eat relation of vegetarian can be changed to $\exists$.
3. Change the $\sqcap$ node to $\sqcup$, the path is then alternative.
4. Remove one of the relations of the concept. The eat relation of either mad_cow or vegetarian can be removed.

4 Reuse & Integration

If only portions of the ontology can be read by the system (or some concepts cannot be linked by any of the existing relations), then fragments of the ontology may be formed using the relationships which have been detected. We will conduct an empirical study with a view to capturing the heuristics used by ontology engineers when facing ill-formed ontologies which were broken into a number of fragments. We then incorporate the heuristics into the system which provides facilities and suggests the best concept candidate to integrate the fragments via *concept-subconcept* relationships, or merging two similar concepts. The resulting ontology could be either a single consistent ontology or a number of consistent fragments.

5 Discussion & Future Work

This paper outlines our system ReTAX++, a graph-based ontology browsing and revision tool, which is still being developed. The user is provided with options to resolve the inconsistencies in the ontology; however, some of the proposed solutions require a more efficient implementation. For example, an ontology may contain numerous inconsistencies which are propagated from an inconsistent concept, and hence we require a more efficient strategy to discover the root causes of inconsistency. Future work will involve performing an empirical study to evaluate the suggestions offered by the system with respect to resolving inconsistencies. The functionality of suggesting the integration of fragments will be implemented as well.