OWLGrED: a UML Style Graphical Editor for OWL

Jānis Bārzdiņš, Guntis Bārzdiņš, Kārlis Čerāns, Renārs Liepiņš and Artūrs Sproģis
Institute of Mathematics and Computer Science, University of Latvia

Introduction

OWL is gradually becoming the most widely used knowledge representation language that has been successfully deployed in a number of applications. Due to formal semantics and availability of reasoners for OWL, it is gaining popularity also in the software engineering community so far largely dominated by UML. Many newcomers have a background in software engineering where UML diagrams are the prevalent form of data modeling and they share many characteristics with OWL ontologies. Although the two languages are similar and it would be natural to reuse the existing familiarity, the UML notation cannot be used as is, because some OWL constructs have no equivalents in UML. We have designed an extended UML notation for OWL that has additional symbols and textual expressions for missing constructs. To make the notation usable in practice we have built a graphical tool that can be used to edit ontologies in a WYSIWYG fashion as well as visualize existing ontologies using a number of layout algorithms.

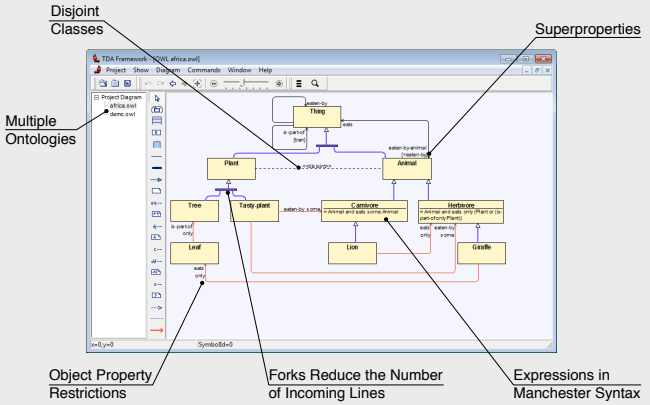
Extended UML Notation for OWL

The proposed graphical notation is based on UML class diagrams. For most features there is one to one mapping from OWL to UML concepts, e.g. ontologies to packages, OWL classes to UML classes, data properties to class attributes, object properties to associations, individuals to objects, etc. Meanwhile for OWL concepts not having a good UML equivalent, the following new extension notations are added:

- a field in classes for equivalent class, superclass and disjoint class expressions written in Manchester OWL syntax
- a field in associations and attributes for specifying equivalent, disjoint and super properties as well as a field for specifying property characteristics, e.g., functional, transitive, etc.
- anonymous classes containing equivalent class expression but no name
- connectors for visualizing disjoint, equivalent, etc. axioms
- boxes with connectors for n-ary disjoint, equivalent, etc. axioms
- connectors for visualizing data property restrictions some, only, exactly, etc.

The main advantage of these extensions is the option to specify class expressions in compact textual form rather than using separate graphical element for each logical class, constructor (and, or, not) and restriction. If the expression is referenced multiple times, it can optionally be shown as an anonymous class.

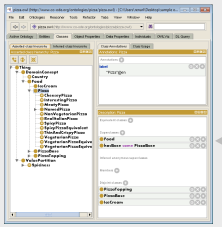
OWLGrEd



The editor has a number of features to ease ontology development:

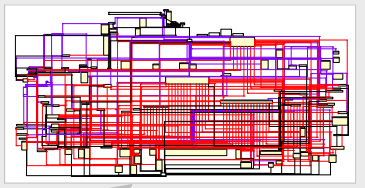
- a number of layout algorithms
- undo/redo
- search
- zoom
- change visual appearance of graphical elements, e.g. fill and stroke color, hide or show textual fields
- splitting of ontologies into sub-diagrams
- working simultaneously with multiple ontologies

Interoperation with Protégé

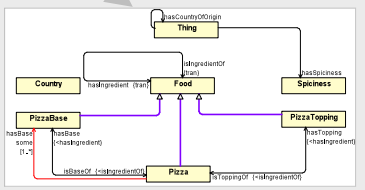


Pizza ontology in Protégé

Full pizza ontology



Chose what to export



Partial pizza ontology (only object properties)

The editor has support for interoperation with Protégé ontology editor, i.e. active ontology can be sent to Protégé for reasoning and serialization and ontologies can be received from Protégé for visualization. The interoperation is implemented as a Protégé plugin that communicates with OWLGrEd through TCP/IP.

When exporting ontology from Protégé to OWLGrEd for visualization one can select:

- what kinds of axioms to export, e.g. only object properties with referenced classes
- what kinds of reductions to perform, e.g. merge subclass-off lines with forks

Implementation

The editor is implemented using transformation driven architecture (TDA) technology. TDA stores its information in the form of MDA-style models that are connected by model transformations. The user interface in TDA is implemented by means of universal engines (e.g. a graph diagramming engine, a property editor engine, etc.). Each individual tool (e.g. OWLGrEd) is created through a tool definition configurator that creates instances of Tool Definition Model storing all meta information about an individual tool – element types, element styles, constraints and relationships among element types.

The Tool Definition Model instances are then interpreted by a universal interpreter that in cooperation with other TDA engines processes all end-user's actions. Furthermore, for OWLGrEd, as for other tools, specific transformations can be created to support domain specific needs. In our case, only transformations supporting interoperability with Protégé, and specific attribute and annotation parsers had to be created.

Future Work

We are planning to add a number of features:

- an option to store graphic layout information inside ontologies (we consider adding it as a special kind of annotations)
- improve integration with Protégé, in particular, synchronize ontologies in both tools after every editing step - current implementation allows exchanging only whole ontologies
- an option to save export preferences

More Information

http://OWLGrEd.lumii.lv

ESF EIROPAS SOCIĀLAIS FONDS
LATVIJAS UNIVERSITĀTE ANNO 1919