Advanced ontology visualization with OWLGrEd

Jānis Bārzdiņš, Kārlis Čerāns, Renārs Liepiņš, Artūrs Sproģis

Institute of Mathematics and Computer Science, University of Latvia,
Raina blvd. 29, LV-1459, Riga, Latvia
Janis.Barzdins@lumii.lv, Karlis.Cerans@lumii.lv,
Renars.Liepins@lumii.lv, Arturs.Sprogis@lumii.lv

Abstract. Intuitive ontology visualization is a key for their learning, exchange, as well as their usage in conceptual modeling. OWLGrEd is a visual tool for compact graphical UML-style rendering and editing of OWL 2.0 ontologies. Here we introduce into OWLGrEd visualization annotations that allow defining custom visual presentations of ontology entities on the basis of annotations attached to these entities. The introduced mechanism of attaching visual annotations to annotation properties used in the ontology, besides being convenient for attaching graphical shape to user-defined annotations, appears to be powerful enough to cover visualization of e.g. UML stereotypes and other UML constructs that do not have direct counterpart in the "logic" part of OWL.

Keywords: OWL ontologies, visualization, annotations, OWLGrEd.

1 Introduction

Intuitive ontology visualization is a key for their learning, exchange, as well as their usage in conceptual modeling. There are a number of tools (e.g. [1,2,3,4]) for rendering and/or editing OWL ontologies in a graphical form. ODM [1] and OWLGrEd [4] are oriented on visualization of OWL ontologies in the style of UML [5] class diagrams. The main idea of OWLGrEd has been to provide compact graphical notation for OWL ontologies, by combining the graphical visualization facilities of UML class diagrams with textual facilities of OWL Manchester encoding [6].

Although OWLGrEd can be successfully used for ontology presentation and editing in a UML-style graphical form (cf. [7]), the practical task of ontology-based modeling suggests further facilities that would be welcome in a graphical ontology editor. The ontology designer may attach certain specific meaning to some ontology entities by user-defined annotation properties and then may wish to create specific visualization patterns for the introduced properties. For instance, if an annotation assertion relates an OWL class to a database table expression, a database icon besides the table expression field may be appropriate in rendering this annotation assertion.

There are also natural modeling constructs in UML that do not have direct counterpart in OWL, such as stereotypes, composition relation and derived union of

properties (or “abstract” property A, not having other subject-object pairs than those of union of B and C). The visual annotation framework that we introduce in this paper allows both visual effect specification for user-defined annotations and visual modeling of the mentioned “advanced” UML concepts (these concepts may be said to correspond to “user-defined” annotations from OWL perspective, as well).

In what follows, after brief reviewing of basic OWLGrEd ontology modeling facilities we proceed to the main point of outlining the visual annotation framework and some its usage examples, including advanced UML construct modeling.

2 Basic OWLGrEd constructs

OWLGrEd provides graphical notation for OWL, based on UML class diagrams. We show OWL classes as UML classes, data properties as class attributes, object properties as associations, individuals as objects, etc. We enrich the UML class diagrams with the following new extension notations (cf. [4, 8]):

- fields in classes for *equivalent class*, *superclass* and *disjoint class* expressions written in Manchester OWL syntax [6];
- fields in associations and attributes for *equivalent*, *disjoint* and *super* properties and fields for 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 object property restrictions *some*, *only*, *exactly*, etc.

OWLGrEd provides option to specify class expressions in compact textual form rather than using separate graphical element for each logical item within class expression. If an expression is referenced in multiple places, it can optionally be shown as an anonymous class. An anonymous class is also used as a base for property domain/range specification, if this domain/range is not a class.

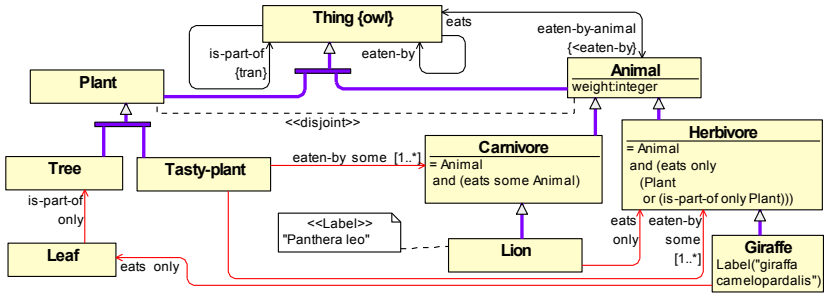

Fig. 1. Example: OWLGrEd notation for a variation of African Wildlife ontology

Figure 1 depicts a variation of the popular African wildlife ontology [9] using OWLGrEd notation. More detailed explanation of the notation used in the example and its correspondence to OWL Manchester encoding is available in [4]. We note also that OWLGrEd allows ontology interoperability with Protégé ontology editor [10].

3 Annotation Visualization Framework

The OWLGrEd's annotation visualization framework allows mapping the built-in and user-defined annotation properties onto editor's visual styles and behavior patterns, thus allowing customizable annotation assertion visualization.

An annotation assertion for an OWL entity (e.g. OWL class or OWL property) can be visualized in OWLGrEd either textually within the visual representation object for the entity (*Inside* mode, cf. *Giraffe* class label in Fig.1), or by a separate box connected to the box/line with entity representation (*Outside* mode, cf. *Lion* class label).

In order to determine the visualization mode for an annotation property, say *P*, we introduce a (visual) annotation property *aShowMode* in *og* namespace denoting *http://owlgred.lumii.lv/2011/visual/1.0/*, and annotate *P* itself with this property, as in *AA(og:aShowMode :P og:modeInside)*[1]. We introduce also annotation properties *og:aClassShowMode* and *og:aPropertyShowMode* to describe visual modes for *P*-assertion visualization for subjects that are OWL classes and properties respectively.

The advanced annotation visualization framework allows adding specific visual style and behavior annotations to *aShowMode*-assertions (cf. Fig. 2) that are interpreted by OWLGrEd. The style/behavior annotations in the case of *Inside* showing mode may set the *visual style* of the annotation *P* field (*og:displayStyle*-annotation)[2] and of the diagram element (box or line) containing the annotation field (*og:displayElemStyle*). The style annotation value is a string with encoded values of style items from visual style metamodel of [11]. It may define e.g. shape and color for a box; dash length and start/end shape for a line; font name/style, placement and visibility for a compartment (field); a box or field may also have a picture (icon) attached to it.

The *type* of showing annotation's assertions may be set by *og:prSheetType*-annotation. Its value *og:Text* (the default) denotes using text field for representing the annotation value; *og:Item* denotes representing the annotation via placing its property name (URI) in a designated single-valued compartment (only one *og:Item*-type annotation can be shown visually for any subject).

The prefix/suffix for annotation assertion presentation in the diagram can be specified by *og:displayValuePrefix* and *og:displayValueSuffix* annotations.

```
AA(     A(og:prSheetType og:Item)
        A(og:displayElemStyle "bkgColor='blue'")
        og:aClassShowMode :ClassDB og:Inside)
AA(     A(og:aDependency :ClassDB)
        A(og:displayStyle "picture='db.jpg'")
        og:aClassShowMode :DBExpr og:Inside)
```

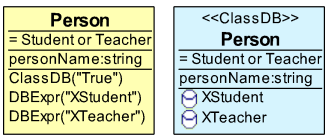

Fig. 2. Visual annotation specification and application example

A(og:aDependency :Q) annotation of *P*'s *aShowMode*-assertion states that the specified *P* visualization effects apply only to subjects that are annotated also by *Q*, and *A(og:aDomainDependency :Q)* states the dependence of *P*-assertion visualization features from its subject's domain annotation by *Q*. Fig. 2 shows OWL class *Person*, with *ClassDB* and *DBExpr* annotations, before and after visual annotation application.

---

[1] We let here and in the following examples A to stand for Annotation and AA for AnnotationAssertion

[2] An inner-level *og:compID*-assertion may specify relation of visual properties to another compartment.

We note that *A(og:prSheetType og:Item)* annotations are sufficient to state that an annotation property (e.g. *ClassDB*) behaves like UML stereotype; *og:aDependency* provides visual framework for stereotype's tagged values (e.g. a *DBExpr*-annotation is visualized in a field with database icon only for *ClassDB*-annotated classes).

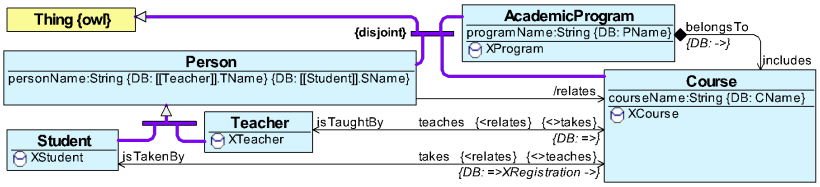

Fig. 3. Composition and property derived union in Mini-University example

Let for Mini-University ontology (Fig.3) there be assertions *AA(user:isComposition :includes "True")* and *AA(user:isDerivedUnion :relates "True")* with user-defined annotation properties *isComposition* and *isDerivedUnion* that are used for encoding UML composition and property derived union constructs, not expressible in OWL. The following visual annotations allow creating typical UML representations for these constructs in OWLGrEd (cf. representation of *:includes* and *:relates* in Fig.3):

AA(A(og:displayElemStyle "line_start='diamond'") **og:aShowMode user:isComposition og:Inside)**
AA(A(A(og:compID "name") og:displayValuePrefix "/") A(og:displayStyle "visible=False")
og:aShowMode user:isDerivedUnion og:Inside)

Fig.3 shows also different notations for *DBExpr*-annotations to OWL properties.

These examples outline the power of OWLGrEd visual annotation framework. The ability to attach visual annotations to annotation properties rather than individual ontology entities allows using user-defined annotation properties as an abstract layer for ontology visual presentation specification; this way domain-specific ontology visualization languages can be created on the basis of OWL annotation construct.

References

1. ODM UML profile for OWL, http://www.omg.org/spec/ODM/1.0/PDF/
2. TopBraid Composer, http://www.topquadrant.com/products/TB_Composer.html
3. Brockmans, S., Volz, R., Eberhart, A., Löffler, P. Visual Modeling of OWL DL Ontologies Using UML, Proc. of ISWC 2004, LNCS 3298, 2004, pp. 198-213.
4. J.Barzdins, G.Barzdins, K.Cerans, R.Liepins, A.Sprogis. OWLGrEd: a UML Style Graphical Notation and Editor for OWL 2. // K.Clark and E.Sirin (eds.), OWLED 2010: OWL: Experiences and Directions, 7th Intl. Workshop, San Francisco, CA, USA 2010.
5. Unified Modeling Language. http://www.omg.org/spec/UML/2.3/.
6. OWL 2 Manchester Syntax, http://www.w3.org/TR/owl2-manchester-syntax/
7. OWLGrEd, http://owlgred.lumii.lv/
8. J.Barzdins, G.Barzdins, K.Cerans, R.Liepins, A.Sprogis. UML Style Graphical Notation and Editor for OWL 2. // Proc. of BIR'2010, LNBIP, Springer 2010, vol. 64, p. 102-113.
9. Antoniou G., van Harmelen F. A Semantic Web Primer, Second Edition, MIT Press, 2008.
10. Protégé, http://protege.stanford.edu/
11. J.Barzdins, K.Cerans, S.Kozlovics, E.Rencis, A.Zarins. A Graph Diagram Engine for the Transformation-Driven Architecture, Proc. of 4th MDDAUI. Florida, USA, 2009, pp.29-32.