OWLGrEd/S: a graphical schema editor for Stardog OWL/RDF-databases

Karlis Cerans, Guntis Barzdins, Renars Liepins, Arturs Sprogis, Julija Ovcinnikova

Institute of Mathematics and Computer Science, University of Latvia
{ Karlis.Cerans, Guntis.Barzdins, Renars.Liepins, Arturs.Sprogis,
Julija.Ovcinnikova}@lumii.lv

Abstract. The developers of Stardog OWL/RDF DBMS have pioneered a new use of OWL as a schema language for RDF databases. This is achieved through explicit splitting of the OWL ontology into "open world assumption" (OWA) inference part and "closed world assumption" (CWA) integrity constraint validation part. This presents a challenge for legacy OWL editors to support seamless ontology authoring with axioms both in OWA and CWA modes. For example, in UML-style graphic diagrams subclass relationship more likely will be included in (OWA) inference while cardinality restrictions are more likely to be interpreted as constraints (CWA). We present here OWLGrEd/S – an extension of the intuitive yet compact graphical UML-style OWL ontology editor OWLGrEd with the editing facilities for OWL ontologies together with integrity constraints within a single ontology schema.

Keywords: OWL, integrity constraints, open-world, closed-world, graphical editor, OWLGrEd, UML class diagram

1 Introduction

Web ontology language OWL [1,2,3] is defined to have "open world assumption" (OWA) semantics and its traditional uses are within modeling domains that admit partiality of the explicitly specified knowledge. In the recent years there have been also efforts to introduce "integrity constraints" (see e.g. [4,5,6]) over OWL ontology models through "closed world assumption" (CWA) semantics, meant to require certain "completeness" properties of the constructed ontology models. The "completeness" or "closedness" assertions often appear very natural in e.g. information system specifications. For instance, the assertion that a person has a telephone number, in the situation that for a person x no telephone number is specified, might well require an interpretation of raising an error, rather than inferring that x really has a telephone number but we just don't know what the number is. The integrity constraints are to be added to OWL, if OWL is to be used in the situations where the CWA interpretation of the knowledge is required.

The approach of integrity constraint specification in [5,6] is remarkable due to reuse of OWL syntax itself for integrity constraint specification. The integrity constraint semantics for OWL is developed in [5] by introducing the notion of "extended knowledge base" (or, extended ontology) as a pair <K,C>, where K is a

knowledge base itself (interpreted according to OWA) and C is integrity constraint specification (interpreted according to CWA), both expressed in OWL syntax.

The Stardog[1] OWL/RDF database brings forward the idea of OWL integrity constraint usage by offering an implementation of integrity constraints through extended knowledge bases. As the developers of Stardog put it, the Stardog database environment materializes the idea of using "the full expressivity of OWL and OWL 2 ... as a schema language for RDF"[2]. On the practical application side, this opens a possibility for a wide range of applications of (extended) OWL in information base structure (schema) specification. This, however, raises an issue of suitable notation for extended OWL notation rendering and editing. Comparing RDF/OWL databases with other database paradigms, it can be easily noticed that there are widely used visual schema development tools for relational databases; the visual UML class diagram notation [7,8] is principal schema definition language for object repositories and databases. It would therefore be of utmost importance for practical usability of RDF/OWL databases to offer a graphical modeling language for database schema authoring and visualization.

There are a number of approaches and tools (see e.g. [9], ODM [10], Top Braid Composer [11], OWLGrEd [12,13]) implementing (some variant/extension of) UML class diagram notation as visual notation for OWL ontologies[3]; these may serve as a good starting point for developing visual notation that supports also extended ontology (including the integrity constraints) authoring and visualization. There is, however, a general problem to be solved, namely that of overcoming the split of the extended ontology into OWA part (the "inference" part, used for ontology/knowledge base model construction) and CWA part (the integrity constraints, to be validated on the basis of the constructed model). A simple "solution" corresponding to modeling visually only the OWA-part of the extended ontology and leaving the CWA-part for specification with other means (e.g. some textual syntax), although contributing to the understanding of the ontology data structuring, does not allow to reap the full benefits of expressing the CWA-part in OWL syntax and graphical UML-based presentation of OWL ontologies[4].

A general observation allowing combining the extended ontology OWA (inference) part K and CWA (constraint) part C within a single schema is based on a simple identity $<K,C> \equiv <K,K \cup C>$ that is yet not well stressed in the literature, at least in the context of Stardog RDF database. This identity asserts, in terms of [5], that every model M of the knowledge base K (the model M is built from K using OWA reasoning mechanisms) that satisfies the constraints C, satisfies also the union of assertions $K \cup C$, where both K and C are viewed in the same closed-world assumption (CWA) sense. In other words, in every model of the extended knowledge base <K,C> both the assertions from K and the assertions from C are merely true

[1] http://stardog.com/

[2] cf. http://clarkparsia.com/pellet/icv/

[3] We note that Stardog database documentation http://stardog.com/docs/sdp/ also use custom UML notation to describe the structure of their example.

[4] Clearly, this depends on the existence of good graphical (UML-based) notation for full OWL 2, including the features (e.g. various kinds of restrictions) that are typically used in constraint specification. We claim that the OWLGrEd editor offers syntax for most of OWL 2 constructs that is easier comprehensible than their plain textual rendering.

regardless of “OWA” or “CWA” or any other “senses” in which this truth is meant. This joint assertion set K∪C (that is, the union of OWA and CWA parts of the extended ontology) is the one that can be visualized graphically (or managed in other editors) as the logical assertion set that is valid on every model of the extended ontology.

The issue remains, of course, how to single out the OWA-part out of the joint OWA+CWA logical schema of the extended ontology. We argue here that in many cases this can be achieved by “meta-level” splitting based just on axiom types rather than performing splitting on the concrete axiom level. The justification for this approach comes from UML class diagrams where such “intuitive” split has been successfully used for decades, where e.g. subclass notation is used in “inference” sense (an instance of a subclass is assumed to be also an instance of super-class) and the cardinality notation is used in the constraint sense (the model is assumed to have an error, if a cardinality expression is not satisfied). To extend the approach, we have defined a universal “split definition language” along with some typical splits as examples. We have implemented this meta-level splitting procedure as a complimentary part of OWLGrEd/S editor; however, it can equally be used with any other OWL editor such as Protégé [14].

The “meta-level” splitting procedures may appear sufficient for many uses of the editor in a “disciplined” ontology/database schema authoring mode, however, for visualization of an arbitrary extended ontology a finer granularity may be necessary; therefore we offer extended graphical notation in OWLGrEd/S editor allowing marking up the extended ontology axioms as belonging to the either OWA or CWA part.

In the following sections of the paper we review first the principles of UML notation usage for OWL ontology specification and the basic principles of OWLGrEd ontology editor that is based on compact notation combining UML class diagram graphical features with textual rendering of advanced OWL constructs not fitting into UML. Then we move to the main subject of the paper on introducing OWL integrity constraints into OWLGrEd, outlining two principal solutions of splitting the ontology into OWA and CWA parts – the general one (meta-level splitting) applicable equally well to OWLGrEd/S and other ontology editors (e.g. Protégé [14]), and the specific one (assertion-level splitting) that is based on marking of individual axioms within the editor as belonging to OWA or CWA part of the ontology.

2 Visual ontology modeling with OWLGrEd

Despite the semantic differences between the UML and OWL modeling approaches, there is certain similarity of concepts in UML and OWL that serves as the basis of using UML class diagrams for presenting core features of OWL ontologies. So, OWL classes can be presented as UML classes, OWL object properties can be typically presented as association roles in the UML diagram, OWL data properties can be presented as attributes in the UML class diagrams, OWL SubClassOf axiom can presented as UML class diagram generalization. There is, however, a very important difference between OWL and UML already on this very basic correspondence level,

where in OWL the object properties and data properties are independent entities, while in UML both the association roles and attributes are structured in accordance to their domain classes. The assumption that underlies the use of UML in OWL modeling is that typically there is no more than a single domain and range assertions for any data/object property in OWL, and, therefore, an appropriate place for the data/object property display in the UML diagram can be found. A suitable representation solution, of course, has to be found also for the case when there appear to be several domain/range assertions for some OWL object or data property.

2.1 Naïve UML modeling of OWL

In addition to syntactic structuring of association roles and attributes in accordance to their domain classes, UML class diagrams have an assumption that association roles/attributes with equal names attached to different domain classes denote different entities. The naïve translation of a UML model where two classes A and B both have an attribute p:string into OWL would be to introduce a single data property p with domain A∪B and range xsd:string (we call this "OR"-semantics of p domain assertions being A and B in the UML diagram). Figure 1 shows simple company ontology, adopted from Stardog documentation pages[5], informally using OWLGrEd notation with the naïve OR-semantics.

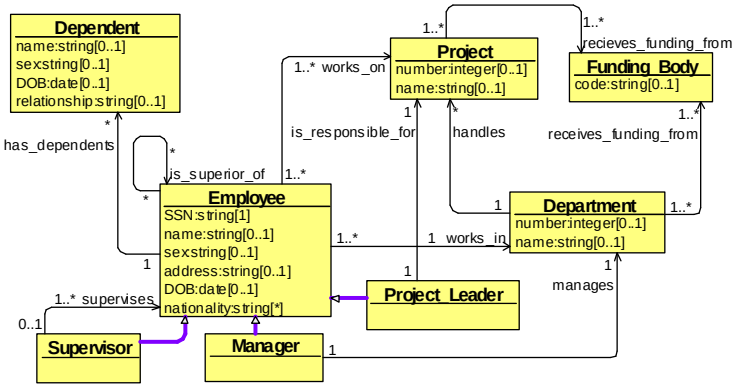

Fig.1. Company ontology in OWLGrEd with naïve OR-semantics

We note e.g. data property *name*, whose domain, in accordance to the OR-semantics would be *Dependent OR Employee OR Project OR Department*. Note also that OWL classes, object and data properties, their domain and range assertions, sub-class-of relations as well as cardinality restrictions can be modeled here in a satisfactory way.

Although intuitively appealing, the OR-semantics of UML class diagrams has certain drawbacks that prohibit its effective use in full OWL ontology modeling. Suppose, for instance, that we would like to make an assertion about the data property *name* that it is a sub-property of some more general data property *identifier* (not

[5] http://stardog.com/docs/sdp/

shown in Figure 1). There is an easy notation *name:string {<identifier}* that can be used to express the sub-property fact, however the problem appears with the place, where this assertion is to be put in the diagram: shall it be put at all places where *name* is mentioned, or in at least one place? Either of the choices would lead to consequences counterintuitive to OR-semantics, where the change of *name* description at e.g. *Dependant* class would affect its behavior at any other class, where it is mentioned (e.g. *Project* class). The problem stems from the fact that OWL properties possibly may be depicted in the UML class diagram in several "OR-related" places; the affected are possibilities of visual depiction of any characteristic that is pertinent to a property (including e.g. reflexivity, transitivity and property chains of object properties).

2.2 The OWLGrEd notation

OWLGrEd provides a complete graphical notation for OWL 2, based on UML class diagrams. We visualize OWL classes as UML classes, data properties as class attributes, object properties as associations, individuals as objects, cardinality restrictions on association domain class as UML cardinalities, etc. We enrich the UML class diagrams with the new extension notations, e.g. (cf. [12,13]):

- fields in classes for *equivalent class*, *superclass* and *disjoint class* expressions written in Manchester OWL syntax [15];
- 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 (we show graphically only those anonymous classes that need to have graphic representtation in order to be able to describe other ontology concepts in the diagram);
- connectors (as lines) for visualizing binary *disjoint*, *equivalent*, etc. axioms;
- boxes with connectors for n-ary *disjoint*, *equivalent*, etc. axioms;
- connectors (lines) for visualizing object property restrictions *some*, *only*, *exactly* (e.g. *Giraffe* < *eats only Leaf* in Figure 3), as well as cardinality restrictions.

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 named class.

Figure 2 contains re-engineering of the Figure 1 ontology in accordance to "genuine" OWLGrEd diagram semantics, using "AND"-semantics (intersection) for domain/range of properties that are mentioned in several places in the ontology diagram. In the example this means, in essence, eliminating of the multiple mentioning of a property (e.g. *name*, or *receives_funding_from*) within the ontology diagram. We demonstrate the use of anonymous classes (e.g. *=Department or Project*) for data and object property domain visualization, as well as using distinct identifiers for different properties along with introducing common super-property for the case when all these properties have to be referred jointly by the same name.

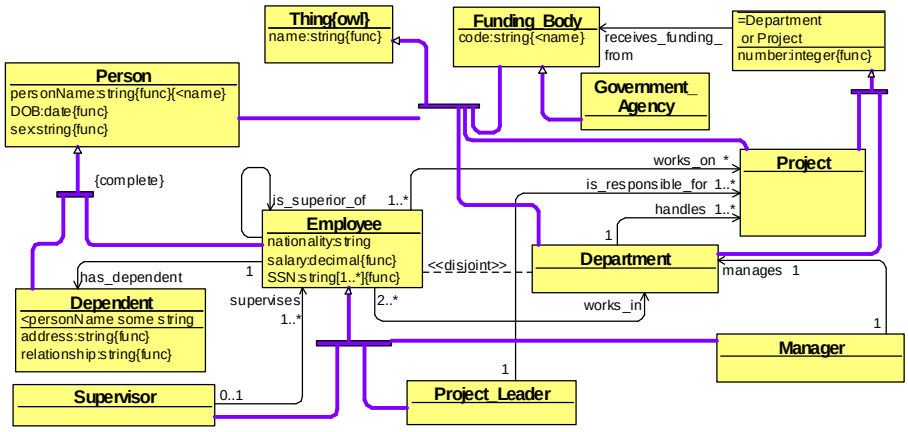

Fig.2. Company ontology re-engineered in OWLGrEd

Figure 3 [13] illustrates some further OWLGrEd notation (e.g. class restrictions in textual and graphical forms and annotation assertions) on the basis of popular African Wildlife ontology example [16].

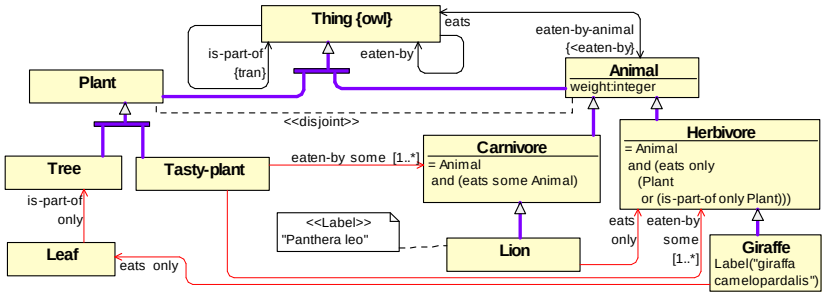

Fig. 3. Example: OWLGrEd notation for a variation of African Wildlife ontology

The OWLGrEd notation allows enriching the ontology of Figure 2 with assertions, adopted from Stardog documentation and characterizing deeper properties of the model (Figure 4). We note that we are currently just presenting the ontology model, and we are not yet discussing its OWA/CWA interpretations.

The OWGrEd editor offers the ontology interoperability (import/export) functionality with Protégé 4.1. ontology editor [14]. The two principal OWLGrEd usage tool chains are:

- ontology authoring (create and edit an ontology in OWLGrEd, then export it to Protégé, where it can be analyzed and submitted to other ontology processing tools)
- ontology visualization, where an ontology that is imported from Protégé is displayed graphically to obtain a comprehensible visual view on it.

Any combination of these two OWLGrEd usage patterns, including ontology round-trip engineering between OWLGrEd and Protégé are possible, as well.

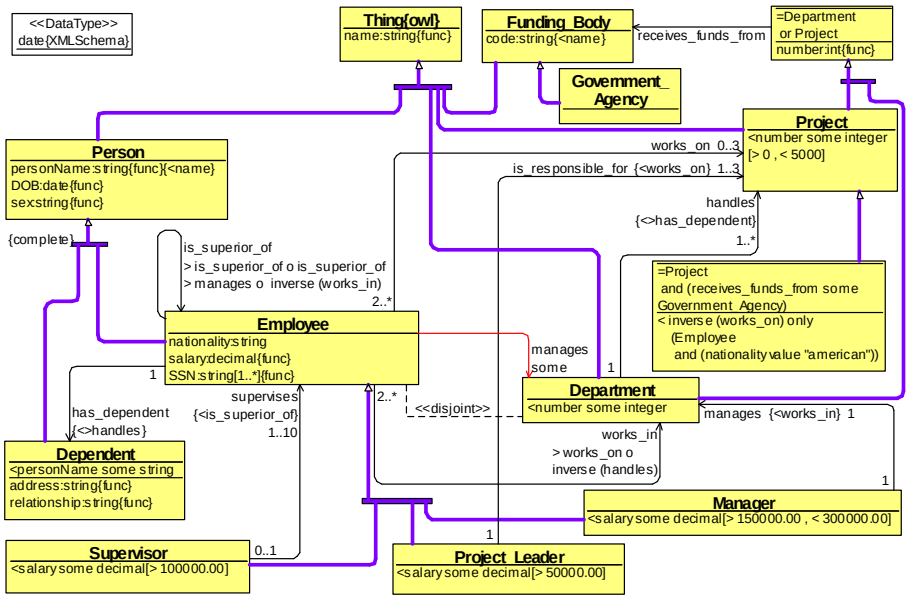

Fig.4. Company ontology in OWLGrEd, with assertions

3 OWLGrEd/S: Incorporating Integrity Constraints

The idea of integrity constraint incorporation into OWLGrEd/S is based on the understanding that the editor simultaneously visualizes the extended ontology, including both axioms that are present in OWA and CWA modes. This corresponds to the understanding that in any model of the extended ontology both OWA and CWA axioms are equally valid; with the only difference being in the contribution these axioms are bringing into the construction of the model from the explicitly given knowledge base. The task of incorporating the integrity constraints into the editor is thus transformed into the task of splitting the graphically specified extended ontology into OWA and CWA parts.

3.1 Generic Ontology Splitters

We argue that for use of OWLGrEd/S in schema authoring mode for RDF/OWL databases a typical scenario for extended ontology splitting would be to channel extended ontology axioms “of the same kind” into the same group of either OWA or CWA axioms. For instance, one may require all SubClassOf(A,B) axioms, where B is a named class, to be interpreted in OWA mode, while e.g. all cardinality restrictions to be interpreted in CWA mode. There can be various “ontology splitting disciplines”, however, often these can be stated in a generic way, independent of specific ontology

(we note that StarDog reasoning types[6] OWL 2 QL, OWL 2 EL, OWL 2 RL, RDF Schema, OWL 2 DL also describe such "splitting disciplines", where an axiom is to be interpreted in OWA mode only if it falls within the respective reasoning type).

In the light of this observation we introduce into OWLGrEd/S the construct of *semantics profile* that can be ascribed to any ontology that is created or displayed in the editor.

Formally, a semantics profile is a function that given an ontology (a set of OWL axioms) O, produces two sets of axioms OWA(O) and CWA(O), whose union has the same logical meaning as O (i.e. OWA(O) $\cup$ CWA(O) is valid on a model M if and only if O is valid).

We note that a semantics profile allows producing an extended ontology (i.e. both OWA and CWA parts) from a single syntactic OWL file; therefore, if the level of granularity provided by the semantics profile level is sufficient, no other means of integrity constraint incorporation are required in the ontology editor.

The proposed semantics profile definition language is based on describing translation of the source ontology O axioms: each axiom A is translated into its corresponding axiom sets OWA(A) and CWA(A). A typical translation behavior for an axiom A would be to "move" axiom A either into OWA(A) or CWA(A) entirely, leaving the other set empty. Meanwhile for some axioms it might be necessary to "re-factor" them into parts, and then process parts (mark as OWA or CWA, or re-factor further) separately. The possible axiom re-factoring rules are summarized in Figure 5, every semantics profile will have to specify, which of these re-factoring rules have to be applied prior to executing the semantics profile.

(i) EquivalentClasses(?X ?Y) -> {SubClassOf(?X ?Y), SubClassOf(?Y ?X)}
(ii) EquivalentClasses(?X1 .. ?Xn) -> {EquivalentClasses(?Xi ?Xj) | 1≤i<j≤n}
(iii) DisjointClasses(?X1 .. ?Xn) -> {DisjointClasses(?Xi ?Xj) | 1≤i<j≤n}
(iv) SameIndividual(?X1 .. ?Xn) -> {SameIndividual(?Xi ?Xj) | 1≤i<j≤n}
(v) DifferentIndividuals(?X1 .. ?Xn) -> {DifferentIndividuals(?Xi ?Xj) | 1≤i<j≤n}
(vi) SubClassOf(?X ObjectIntersectionOf(?Y1 .. ?Yn)) ->{ SubClassOf(?X ?Yi) | 1≤i≤n}
(vii) SubClassOf(?X ObjectExactCardinality(?Y ?Z ?W))->{ SubClassOf(?X ObjectMinCardinality(?Y ?Z ?W)), SubClassOf(?X ObjectMaxCardinality(?Y ?Z ?W))}
(viii) SubClassOf(?X DataExactCardinality(?Y ?Z ?W)) -> { SubClassOf(?X DataMinCardinality(?Y ?Z ?W)), SubClassOf(?X DataMaxCardinality(?Y ?Z ?W))}
(ix) DisjointUnion(?X ?Y1 .. ?Yn)->{DisjointClasses(?Y1 .. ?Yn), EquivalentClasses(ObjectUnionOf(?Y1 .. ?Yn) ?X)}
(x) EquivalentObjectProperties(?X1 .. ?Xn) -> {EquivalentObjectProperties(?Xi ?Xj) | 1≤i<j≤n}
(xi) EquivalentObjectProperties(?X ?Y) -> {SubObjectPropertyOf(?X ?Y), SubObjectPropertyOf(?Y ?X)}
(xii) EquivalentDataProperties(?X1 .. ?Xn) -> {EquivalentDataProperties(?Xi ?Xj) | 1≤i<j≤n}
(xiii) EquivalentDataProperties(?X ?Y) -> {SubDataPropertyOf(?X ?Y), SubDataPropertyOf(?Y ?X)}
(xiv) ClassAssertion(ObjectIntersectionOf(?X1 .. ?Xn) ?Y)-> {ClassAssertion(ObjectIntersectionOf(?Xi ?Y) | 1≤i≤n}
(xv) f(ObjectComplementOf(ObjectComplementOf(?Y))) -> {f(?Y))} for any context f
(xvi) f(ObjectComplementOf(ObjectUnionOf(?X1 .. ?Xn))) -> {f(ObjectIntersectionOf(?X1 .. ?Xn))} for any context f

Fig.5. Re-factoring rules

[6] http://stardog.com/docs/sdp/

Semantics profile definition language

The semantics profile definition consists of a sequence of rules of either the form '*Q.*' (the unconditional rules), or '*Q :- C1, .., Cn.*' (the conditional rules), where:

- *Q* is an assertion in one of the forms: *OWA(p)*, *CWA(p)*, with *p* in OWL Functional syntax, possibly with the following placeholders:
  ? – matching any OWL Functional syntax term
  ?.. – matching any list of OWL Functional syntax [2] terms
  ?X – matching any OWL Functional syntax term, additionally marking the term by the meta-variable name "X"
  exp – matching the regular expression *exp*, composed of literals; and
- *Ci* for $1 \leq i \leq n$ is the rule condition.

The processing of an ontology axiom *A* consists of finding the first rule in the sequence, where the axioms' OWL functional syntax description matches the assertion pattern p and satisfies the corresponding rule conditions. When such a rule is found, *A* is moved into OWA or CWA sets accordingly.

The rule conditions can be built over the entire axiom A (denoted within the condition by *?A*) and the terms *?X* that are marked within the rule's assertion part, using logical connectives over:

- simple syntactic predicates, such as *isEntity(?X)*,
- matching predicate -:-, allowing to match *?A* or *?X* to OWL Functional syntax term with placeholders *?*, *?..* and *exp*, as described above,
- simple semantic predicates in the form *isAsserted(?f)*, where *?f* is OWL Functional syntax expression with containing possible term markers *?X*.

Consider, for instance, the rule (*): *OWA(SubClassOf(?X ?Y)* :- *isEntity(?Y) or ?Y-:-DataHasValue(?..) or ?Y-:-AllValuesFrom(?..) or ?Y -:- ObjectComplementOf(?) or ?Y-:-DataMaxCardinality(['0'|'1'] ? ?) or ?Y-:-DataMinCardinality(['0'|'1'] ? ?).* This rule marks as OWA those *SubClassOf(?X ?Y)* axioms, where *?Y* has some of the forms, admitted for superclasses in OWL 2 RL profile [17].

The following rule marks as OWA only those subPropertyOf axioms, where the super-property is annotated by *owlgred_s:isInferred*- annotation assertion (**): *OWA(SubObjectPropertyOf(? ?X)) :-*
isAsserted(AnnotationAssertion(?X owlgred_s:isInferred True)).

Semantics profile examples

Given the described language constructions for semantic profile, it is up to the user of the ontology editor to define the semantic profile he/she is willing to use for the ontology splitting into OWA or CWA parts. We discuss here, however a few semantics profiles that the users might find reasonable to use.

First, there is a simple "open-world" semantics profile "OWA(?)." that will interpret the entire ontology according to the open-world semantics. Another simple

semantics profile would be "OWA(Declaration(?)). CWA(?)." that will interpret everything except the declarations in the closed-world sense[7].

In what follows, we offer a semantics profile example for use in information base development on the basis of Stardog OWL/RDF data store with OWL 2 RL reasoning enabled. In our example we would like to follow the guidelines of restricting the OWA reasoner from (i) inferring the existence of new individuals in the knowledge base model, and (ii) from inferring the co-incidence of two differently named individuals, since we believe that specification of individuals and their co-incidence are highly sensitive tasks that are to be handled manually (it should, however, be possible to specify explicit *SameAs*-axioms)[8].

These principles would require exclusion of cardinality restrictions and *SomeValuesFrom*-restrictions from OWA reasoner, and this corresponds to the intuition of widely advocated interpretation of cardinalities as integrity constraints (see e.g. [5]), and also used in UML semantics [7,8]. Furthermore, we would like to retain an intuitive property of model determinism/completeness, what would be violated, for instance, by OWA-interpretation of disjunctive *SubClassOf* (in superclass position) and *ClassAssertion* axioms[9]. Within our example profile we shall exclude also property domain and range axioms from OWA reasoning, although we well admit that in some situations the reverse decision might be more appropriate[10].

On the other hand, assigning OWA sense to *SubClassOf* axioms with a named class in the superclass position would be one of the most important inferences that the database should be able to make[11] [12]. We note that since we allow for different interpretations of different *SubClassOf*-axioms, it would be necessary to refactor the *EquivalentClasses*-axioms and process the resulting *SubClassOf*-axioms separately.

A situation with sub-property axioms (both for data and object properties) is somewhat trickier, as it is easy to provide examples for both OWA and CWA interpretations to be the most natural ones. For instance in Figure 4 we would like to have the assertion *personName*<*name* to be interpreted as OWA-assertion, while *works_on*<*manages* – as a constraint.

---

[7] We do not bring the annotation axioms into the discussion here since they do not affect the ontology meaning. They can always reasonably be put into the OWA-part of the ontology.

[8] We stress that we consider here development of possible ontology splitter that we find reasonable and that this is not to exclude any other preferences in splitter definition.

[9] if *ClassAssertion(ObjectUnionOf(B C) a)*, then among models of this axiom there will be those where the object corresponding to *a* will belong either to the set corresponding to *B* or *C* (the queries over the information base would list *a* as belonging to $B \cup C$, however, these will not list *a* as belonging to either *B* or *C*, since neither of these options can be inferred from the knowledge base

[10] Suppose there is a class *Person*, with possibly overlapping subclasses *Student* and *Teacher*. Only *students* are allowed to *take* a *course*. It may well be possible that by an error we have indicated that *Prof.Smith takes* a *course Programming Basics* (it should have been that he *teaches* this *course*). The OWA semantics would "cope" with the situation by inferring that *Prof.Smith* is a *student*, whilst a more natural reaction from the system would be to raise an error.

[11] We recognize that it might be possible to interpret even this kind of axioms as constraints, as demonstrated e.g. in Stardog ICV documentation pages http://stardog.com/docs/sdp/.

[12] There will be need to constrain also the class expression in subclass position (e.g. to exclude *AllValuesFrom*- expressions) for SubClassOf-axiom interpretation in OWA-sense.

In the case when the modeling methodology requires both OWA and CWA interpretations of sub-property axioms, one might resort to a finer-grained structure of axiom-level annotation, as described in Section 3.2. We argue here, however, that coarser-grained means may still be possible (and preferable), namely introducing a special annotation property, say, *owlgred_s:isInferred*, and creating a conditional rule in semantics profile definition that marks a *SubClassOf(A B)*-axiom as OWA whenever *B* is annotated by the introduced annotation (for *SubObjectPropertyOf* this is rule (**) from the last subsection)[13]. A convenient custom graphical notation for *isInferred*-annotations is easily introduced either in OWLGrEd/S, or OWLGrEd, along the lines of the approach outlined in [17] (we denote the existence of *isInferred* annotation for a property by a '/i/' suffix added to the data or object property name).

An important OWA-axiom for information bases would be also *InverseProperties*. We would refrain from including *Functional* and *InverseFunctional* property characteristics, as well as *Key*-assertions in the ontology OWA-part since these could be used to infer implicit co-incidences among entities. The *Reflexive*, *Symmetric* and *Transitive* object property characteristics could be well understood as adding new "ground knowledge" to the data model, therefore their place would rather be in the ontology OWA-part (it may have limited, although not completely void, sense having these property characteristics as constraints; we discuss some limitations later). Our proposal would be to put *DisjointClasses* and *DisjointProperties* axioms, as well as *Irreflexive*- and *Asymmetric*- property assertions into ontology OWA-part, as well[14].

Our observations regarding the OWA-part of the ontology fall, in fact, in line with OWL reasoning profile OWL RL [18] (we are willing to include in ontology OWA-part a subset of axioms allowed for OWL RL), except for *ReflexiveObjectProperty*-axiom that is excluded from the profile for efficiency reasons[15]. As per example, we exclude the *ReflexiveObjectProperty*-axiom from the OWA-part of the model, with the understanding, however, that this may need to be re-considered as soon as a serious use of this axiom in the information base appears.

```
Enabled refactorings (Fig. 5):  i, ii, iii, vi, ix, x, xi, xii, xiv
Ontology splitting rules:
OWA(SubClassOf(?X ?Y)) :- isEntity(?X) or ?X-:-ObjectOneOf(?..) , isEntity(?Y) or ?Y-:-DataHasValue(?..)  .
OWA(DisjointClasses(?X,?Y)) :- isEntity(?X) or ?X-:-ObjectOneOf(?..),isEntity(?Y) or ?Y-:-ObjectOneOf(?..).
OWA(SubObjectPropertyOf(? ?X)) :- isAsserted(AnnotationAssertion(?X owlgred_s:isInferred True)).
OWA(SubDataPropertyOf(? ?X)) :- isAsserted(AnnotationAssertion(?X owlgred_s:isInferred True)).
-- OWA(ReflexiveObjectProperty(?)).       -- Excluded
OWA(InverseObjectProperties(? ?), SymmetricObjectProperty(?), TransitiveObjectProperty(?),
                                    AsymmetricObjectProperty(?), IrreflexiveObjectProperty(?)).
OWA(ClassAssertion(?X ?)) :- isEntity(?X).
refactor (ClassAssertion(ObjectIntersectionOf(?..) ?).
OWA(SameIndividuals(?..), DifferentIndividuals(?..), OWA(ObjectPropertyAssertion(? ? ? ),
                                    DataPropertyAssertion(? ? ?), DatatypeDefinition(? ?)).
CWA(?).
```

Fig.6. UML-style semantics profile

[13] We note the parallels of the introduced *isInferred* notion for properties with *isDerivedUnion* notion for UML association roles, although these notions cannot be semantically equated.

[14] These would allow detecting certain inconsistencies already on the OWA-reasoning level.

[15] http://lists.w3.org/Archives/Public/public-owl-wg/2008Sep/0212.html

The final principle in the "UML-style" semantics profile example (Figure 6) design is adding the conditions on *SubClassOf* and *DisjointClasses* axioms, so as to conform to a proper subset of OWL RL profile, and so avoid indirect introduction of OWA-assertions violating the design principles stated here.

We recall that the axioms that are present in the ontology editor and that are not brought into its OWA-part, do not "disappear" – they are put into the CWA-part and interpreted as integrity constraints.

We have tested the example semantics profile with RL reasoning mode of Stardog and have found that it produces reasonable results. We note that there might be also an approach of defining a semantics profile that is based on OWL QL [18].

A semantics profile conceptually is an integral part of OWLGrEd/S, since changing the profile may seriously affect the ontology semantics. Note, however, that the semantic profiles are not necessarily to be tied up with OWLGrEd/S editor; these can be used in the context of other ontology editors (e.g. Protégé), as well.

3.2 Graphical Interface for Axiom-level Ontology Splitting

The "meta-level" splitting procedures may appear sufficient for many uses of the editor in a "disciplined" ontology/database schema authoring mode, however, for visualization of an arbitrary extended ontology a finer granularity may be necessary; therefore we offer extended graphical notation in OWLGrEd/S editor allowing marking up the extended ontology axioms as belonging to the either OWA or CWA part. Figure 7 contains an example of explicit marking of axioms as integrity constraints within company ontology from Figure 4.

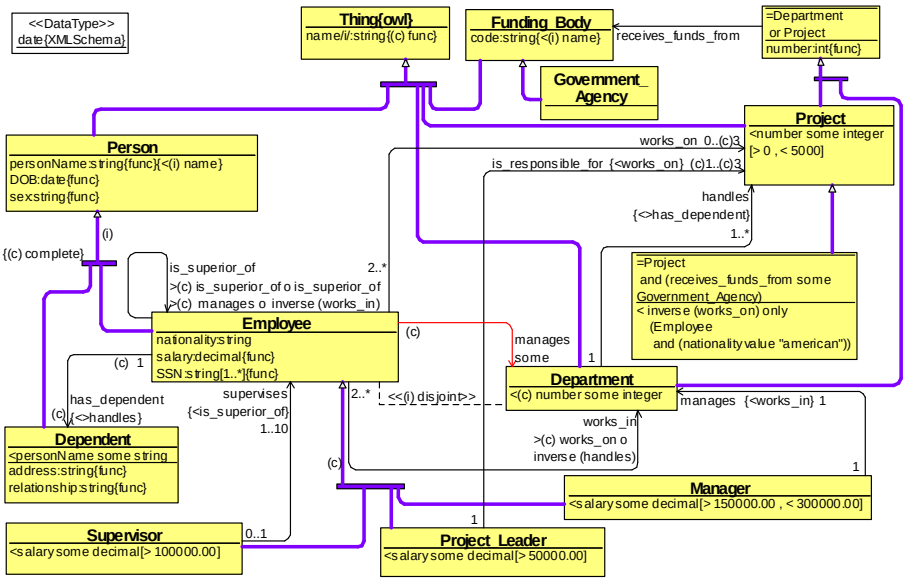

Fig.7. Company ontology, marked with axiom level CWA-specifications

The idea of the notation is to attach (i) (standing for "inference", meaning inclusion of the axiom in OWA-part of the ontology) or (c) (standing for "constraints", meaning inclusion of the axiom in CWA part of the ontology) notations to the visual representations of OWL axioms. We note that the concrete axiom markings are meant to be used as an addition to the semantics profiles, described in Section 3.1. The explicit marking of an axiom as (i)/(c) takes precedence over its processing instructions, as defined in the semantics profile. We note, however, that explicit (i)/(c) marking is not possible on axiom "part" levels, as it have been possible in the case of semantics profiles via axiom re-factoring. In the case, if an axiom is reflected in several parts of the diagram, the (i)/(c) marking of any single place of the axiom representation suffices to have the entire axiom marked as OWA/CWA, respectively.

The positions, where the (i)/(c)-markings can be introduced into OWLGrEd/S editor are, as follows:

- object property line start and end positions, reflecting domain and range assertions for the property (e.g. property *has_dependent* in Figure 7);
- data property name prefix, reflecting data property domain assertion;
- equivalent classes, disjoint classes and superclasses assertions within class nodes (e.g. assertion < *number some integer* in *Department* class); in a similar way the notation is extended also to equivalent, disjoint and super-properties (e.g. *<(i)name* assertion for *personName* property in *Person* class)
- property chain assertions
- object and data property characteristics (e.g. *{(c)func}* notation for name property in class *Thing*)
- cardinality restrictions (e.g. *(c)1..(c)3* cardinality for *is_responsible_for*)
- object property restrictions that are shown in a graphical form (e.g. *Employee < manages some Department*)
- generalization lines (*SubClassOf*-markers) in the graphical form (e.g. for subclasses of *Person* and *Employee* classes)
- disjoint/complete assertions placed at generalization set descriptors (forks), e.g. the fork joining subclasses of *Person* class.

We note also the /i/-suffix notation for the name property at Thing class that is used to annotatate the property with *owlgred_s:isInferred*-annotation as a property whose sub-property assertions on the semantics profile level can be defined to belong to the ontology OWA-part.

The conceptual tool chain for working with OWLGrEd/S[16] editor involves defining/importing the semantics profile, then editing the ontology in the editor, possibly assigning the individual axiom markers. Further on two ontologies, say, *open.owl* and *ic.owl* are exported from the editor and can be used in Stardog database environment as open schema and integrity constraints files. There is an alternative implementation, however, with independent ontology splitter, where the ontology is created in OWLGrEd or OWLGrEd/S (if extra markings for custom annotation properties, or individual axioms are required), then exported into an .owl file that is enriched with custom annotations (the individual axiom markers are implemented as axiom-level annotations); the file is further on split using the created ontology splitter.

[16] A current version of the editor can be found at owlgred.lumii.lv/s

The OWLGrEd/S editor has also a principal possibility to visualize any extended ontology with integrity constraints due to its axiom-level granularity of OWA/CWA splitting; we are working towards implementation of this mechanism, as well, that would allow involving the OWLGrEd/S editor also starting from later stages of OWL/RDF database development in Stardog environment.

4 Related Work

There are a number of approaches for visualizing and authoring OWL ontologies using graphics that is based on UML class diagram notation. Most notable of the existing approaches are [9], ODM [10] and TopBraid Composer [11]. The distinction of the OWLGrEd ontology editor [12,13] from the above lies in the possibility to use systematically OWL Manchester syntax [15] form of class and property restrictions to complement the graphic notation, thus obtaining a compact yet comprehensible representation of OWL ontologies. To the best of our knowledge of the authors, none of the abovementioned visual ontology editors offer support for integrity constraint specification, as proposed here in OWLGrEd/S editor.

The problem of notations and approaches for integrity constraint incorporation in existing OWL editors is well studied in [6], where the new vocabulary, ontology annotation, axiom annotation and rich annotations approaches are considered (the Stardog database implementation follows most closely the “ontology annotation” approach). We observe that the idea of generic axiom splitters that are defined as meta-level procedures appears to be a new integrity constraint specification way, if compared to those described in [6].

5 Conclusions

The development of integrity constraints for RDF/OWL databases may open wider the way of the application of these database systems in real information base modeling and implementation. The proposal of this paper to graphically visualize and author RDF/OWL database schemas may ease substantially the schema development, understanding and sharing for RDF/OWL databases, thus further removing obstacles for spreading of the RDF/OWL database technology.

We believe that the seemingly obvious single-schema observation that allows for creation and maintaining of a joint visual model of OWA and CWA assertions within the ontology may be followed also by other ontology editors as the way of integrity constraint implementation.

The introduction of meta-level ontology splitting notation provides a base for further discussions on natural semantics variants for joint OWA+CWA assertion specification within a single ontology schema, as well as allowing the “power users” of ontology editors to define their own semantics profiles fitting their specification purposes (we recall that e.g. simple annotation assertions to entities may be exploited in semantics profile definitions in order to define OWA/CWA axiom splitting). The axiom-level markup of integrity constraints in OWLGrEd/S can be used on top of

underlying semantics profile to achieve the finest degree needed in OWA/CWA axiom splitting both in ontology schema visualization and authoring situations.

The creation of the OWLGrEd/S editor has been possible do to an open-architecture, model-based and highly customizable implementation of the OWLGrEd editor based on TDA platform [19] and the tool definition meta-model developed over it [20]. The open and customizable tool architecture has been maintained also in OWLGrEd/S, allowing the expert users to tailor the appearance and even to some extent the functionality of the editor to the user's specific needs. As a possible future work we consider including the support in OWLGrEd and OWLGrEd/S tools for custom integrity constraints specificified e.g. in SPARQL language [21].

References

1. Smith, M. K.; Welty, C.; and McGuiness, D.: OWL Web Ontology Language Guide, 2004
2. Motik, B; Patel-Schneider P.F; Parsia B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, 2009
3. Motik, B.; Patel-Schneider, P. F.; and Grau, B. C.: OWL 2 Web Ontology Language Direct Semantics, 2009
4. Motik, B.; Horrocks, I.; and Sattler, U. Bridging the Gap between OWL and Relational Databases. In Proc. of WWW 2007, 807–816, 2007.
5. Tao, J.; Sirin, E.; Bao J; McGuinness, D.: Integrity Constraints in OWL. In Proc. of AAAI 2010, 2010.
6. Sirin, E; Smith, M; Vallace, E: Opening, Closing Worlds – On Integrity Constraints. In Proc. of OWLED 2008, 2008.
7. Unified Modeling Language: Infrastructure, version 2.1. OMG Specification ptc/06-04-03, http://www.omg.org/docs/ptc/06-04-03.pdf
8. Unified Modeling Language: Superstructure, version 2.1. OMG Specification ptc/06-04-02, http://www.omg.org/docs/ptc/06-04-02.pdf
9. Brockmans, S., Volz, R., Eberhart, A., Löffler, P. Visual Modeling of OWL DL Ontologies Using UML, Proc. of ISWC 2004, LNCS 3298, pp. 198-213, 2004.
10. ODM UML profile for OWL, http://www.omg.org/spec/ODM/1.0/PDF/
11. TopBraid Composer, http://www.topquadrant.com/products/TB_Composer.html.
12. Barzdins, J.; Barzdins, G.; Cerans, K.; Liepins, R.; Sprogis, A.: OWLGrEd: a UML Style Graphical Notation and Editor for OWL 2. In Proc. of OWLED 2010, 2010.
13. Barzdins, J.; Cerans, K.; Liepins, R.; Sprogis, A.: UML Style Graphical Notation and Editor for OWL 2. In Proc. of BIR'2010, LNBIP, Springer 2010, vol. 64, p. 102-113, 2010.
14. Protégé 4, http://protege.stanford.edu/
15. OWL 2 Manchester Syntax, http://www.w3.org/TR/owl2-manchester-syntax/
16. Antoniou G., van Harmelen F. A Semantic Web Primer, Second Edition, MIT Press, 2008.
17. Barzdins, J.; Cerans, K.; Liepins, R.; Sprogis, A.: Advanced ontology visualization with OWLGrEd. In Proc. of OWLED 2011, 2011.
18. Motik, B.; Grau, B. C.; Horrocks, I.; Wu, Z.; Fokoue, A.; Lutz, C.: OWL 2 Web Ontology Language Profiles, 2009
19. Barzdins J., Rencis E., and Kozlovics S. The Transformation-Driven Architecture, Proc. of 8th OOPSLA Workshop on Domain-Specific Modeling. Nashville, USA, pp.60-63, 2008.
20. Barzdins, J.; Cerans, K.; Kozlovics, S.; Lace, L.; Liepins, R.; Rencis, E.; Sprogis, A; Zarins, A.: An MDE-based Graphical Tool Building Framework. In Scientific Papers, University of Latvia, Vol 756, ISSN 1407-2157, pp. 121-138, 2010.
21. SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/, 2008.