Constraint Reuse in DL-Lite Core with Arbitrary Number Restrictions

(Extended Abstract)

Marco A. Casanova[1], Karin K. Breitman[1], Antonio L. Furtado[1], Vânia M.P. Vidal[2], José A. F. de Macêdo[2], Eveline R. Sacramento[1,3]

[1]Department of Informatics – PUC-Rio – Rio de Janeiro, RJ – Brazil
{casanova, karin, furtado, esacramento}@inf.puc-rio.br
[2]Department of Computing, Federal University of Ceará – Fortaleza, CE – Brazil
{vvidal, jose.macedo}@lia.ufc.br
[3]Ceará State Foundation of Meteorology and Water Resources – Fortaleza, CE – Brazil
eveline@funceme.br

Abstract. The process of reusing an ontology involves two issues: (1) selecting a set of terms from the ontology vocabulary; and (2) using the ontology constraints to derive those that apply to such terms. The first issue corresponds to the familiar practice of importing namespaces. This paper proposes to address the second issue by introducing a set of operations over ontologies, treated as theories and not just as vocabularies. It discusses how to compute the operations for a class of ontologies built upon DL-Lite core with arbitrary number restrictions. Finally, for this class of ontologies, the paper addresses the question of minimizing the set of constraints which results from an operation.

Keywords: constraints, DL-Lite Core, Description Logics.

1 Introduction

We introduce a set of concepts and procedures that promote constraint reuse in ontology design. We start by introducing a set of operations on ontologies that create new ontologies, including their constraints, out of other ontologies. Such operations extend the idea of namespaces to take into account constraints and treat ontologies as theories. Then, we concretely show how to compute the operations when the ontologies are built upon DL-Lite core with arbitrary number restrictions [3]. We refer to such ontologies as *lightweight ontologies*. Finally, for lightweight ontologies, we show how to minimize the set of constraints.

The development in the paper adopts the machinery to handle constraints developed in [6][7][8], which uses DL-Lite core with arbitrary number restrictions [3]. Previous work by the authors [7] introduced the equivalent of the projection operation, but considered neither the other operations nor the question of optimizing the representation of the resulting ontologies.

The paper is organized as follows. Section 2 presents the formal definition of the operations. Section 3 shows how to compute the operations for lightweight ontologies. Section 4 addresses the question of minimizing the set of constraints that result from an operation. Finally, Section 5 contains the conclusions.

2 A Formal Framework

2.1 A Brief Review of Basic Concepts

The definition of the operations depends only on the notion of theory, which we introduce in the context of Description Logic (DL) [4].

A DL *language* $\mathcal{L}$ is characterized by a *vocabulary* V, consisting of a set of *atomic concepts*, a set of *atomic roles*, and the *bottom concept* $\bot$. The sets of *concept descriptions* and *role descriptions* of V depend on the specific description logic. An *inclusion* of V is an expression of the form $u \sqsubseteq v$, where u and v both are concept descriptions or both are role descriptions.

An *interpretation* $\boldsymbol{s}$ of V consists of a nonempty set $\Delta^{\boldsymbol{s}}$, the *domain* of $\boldsymbol{s}$, and an *interpretation function*, also denoted $\boldsymbol{s}$, with the usual definition [4]. We use $\boldsymbol{s}(u)$ to indicate the value that $\boldsymbol{s}$ assigns to an expression u of V.

Let σ be an inclusion of V and Σ be a set of inclusions of V. We say that: $\boldsymbol{s}$ *satisfies* σ or $\boldsymbol{s}$ is a *model* of σ, denoted $\boldsymbol{s} \vDash \sigma$, iff $\boldsymbol{s}(u) \subseteq \boldsymbol{s}(v)$; $\boldsymbol{s}$ *satisfies* Σ or $\boldsymbol{s}$ is a *model* of Σ, denoted $\boldsymbol{s} \vDash \Sigma$, iff $\boldsymbol{s}$ satisfies all inclusions in Σ; Σ *logically implies* σ, denoted $\Sigma \vDash \sigma$, iff any model of Σ satisfies σ; Σ is *satisfiable* or *consistent* iff there is a model of Σ; Σ is *strongly consistent* iff Σ is *consistent* and Σ does not logically imply $A \sqsubseteq \bot$, for some atomic concept A.

The *theory* of a set Σ of inclusions of V, denoted $\tau[\Sigma]$, is the set of all inclusions of V which are logical consequences of Σ.

We are specially interest in *DL-Lite core with arbitrary number restrictions* [3], denoted $DL\text{-}Lite_{core}^{\mathcal{N}}$. The sets of *basic concept descriptions*, *concept descriptions* and *role descriptions* in this case are defined as follows:

- If P is an atomic role, then P and P^{-} (*inverse role*) are role descriptions
- If u is an atomic concept or the bottom concept, and p is a role description, then u and $(\geq n\ p)$ (*at-least restriction*, where n is a positive integer) are basic concept descriptions and also concept descriptions
- If u is a concept description, then $\neg u$ (*negated concept*) *is* a concept description

For $DL\text{-}Lite_{core}^{\mathcal{N}}$, an *inclusion* of V is an expression of one of the forms $u \sqsubseteq v$ or $u \sqsubseteq \neg v$, where u and v both are basic concept descriptions. That is, an inclusion may contain a negated concept only on the right-hand side and it may not involve role descriptions.

We use the following abbreviations: "$\top$" for "$\neg\bot$" (*universal concept*), "$\exists p$" for "$(\geq 1\ p)$" (*existential quantification*), "$(\leq n\ p)$" for "$\neg(\geq n+1\ p)$" (*at-most restriction*)

and "$u \mid v$" for "$u \sqsubseteq \neg v$" (*disjunction*). By an *unabbreviated* expression we mean an expression that does not use such abbreviations.

Finally, an *ontology* is a pair $\boldsymbol{O}=(V,\Sigma)$ such that V is a finite vocabulary, whose atomic concepts and atomic roles are called the *classes* and *properties* of $\boldsymbol{O}$, respectively, and Σ is a finite set of inclusions of V, called the *constraints* of $\boldsymbol{O}$. A *lightweight ontology* is an ontology whose constraints are inclusions of $DL\text{-}Lite_{core}^{\mathcal{N}}$.

From this point on, we will use the terms class, property and constraint instead of atomic concept, atomic role and inclusion, whenever reasonable.

2.2 Definition of the Operations

We define the following operations over ontologies.

Definition 1: Let $O_1 = (V_1,\Sigma_1)$ and $O_2 = (V_2,\Sigma_2)$ be two ontologies, W be a subset of V_1 and Φ be a set of constraints over V_1.

(i) The *selection* of $O_1 = (V_1,\Sigma_1)$ for Φ, denoted $\sigma[\Phi](O_1)$, returns the ontology $O_S = (V_S,\Sigma_S)$, where $V_S = V_1$ and $\Sigma_S = \Sigma_1 \cup \Phi$.

(ii) The *projection* of $O_1 = (V_1,\Sigma_1)$ over W, denoted $\pi[W](O_1)$, returns the ontology $O_P = (V_P,\Sigma_P)$, where $V_P = W$ and Σ_P is the set of constraints in $\tau[\Sigma_1]$ that use only symbols in W.

(iii) The *union* of $O_1 = (V_1,\Sigma_1)$ and $O_2 = (V_2,\Sigma_2)$, denoted $O_1 \cup O_2$, returns the ontology $O_U = (V_U,\Sigma_U)$, where $V_U = V_1 \cup V_2$ and $\Sigma_U = \Sigma_1 \cup \Sigma_2$.

(iv) The *intersection* of $O_1 = (V_1,\Sigma_1)$ and $O_2 = (V_2,\Sigma_2)$, denoted $O_1 \cap O_2$, returns the ontology $O_N = (V_N,\Sigma_N)$, where $V_N = V_1 \cap V_2$ and $\Sigma_N = \tau[\Sigma_1] \cap \tau[\Sigma_2]$.

(v) The *difference* of $O_1 = (V_1,\Sigma_1)$ and $O_2 = (V_2,\Sigma_2)$, denoted $O_1 - O_2$, returns the ontology $O_D = (V_D,\Sigma_D)$, where $V_D = V_1$ and $\Sigma_D = \tau[\Sigma_1] - \tau[\Sigma_2]$.

Note that selections can be defined as unions. We also observe that the ontology that results from each operation is unique. However, the set of constraints of the resulting ontology is not necessarily minimal, a point elaborated in Section 4.

2.3 An Example

The *Music Ontology* (MO) [11] provides concepts and properties to describe artists, albums, tracks, performances, arrangements, etc. on the Semantic Web. It is used by several Linked Data sources, including MusicBrainz and BBC Music. The *Music Ontology* RDF schema uses terms from the *Friend of a Friend* (FOAF) [5] and the XML Schema (XSD) vocabularies, among others. We adopt the prefixes "mo:", "foaf:" and "xsd:" to respectively refer to these vocabularies.

Figure 1 shows the class hierarchies of MO rooted at classes foaf:Agent and foaf:Person. Let us focus on this fragment of MO.

We first recall that FOAF has two constraints, informally formulated as follows:

- each person has at most one name
- foaf:Person and foaf:Organization are disjoint classes

Let V_1 be the following set of terms from the FOAF and the XSD vocabularies:

V_1={ foaf:Agent, foaf:Person, foaf:Group, foaf:Organization, foaf:name, xsd:string }

Let V_2 contains the rest of the terms that appear in Figure 1:

V_2={ mo:MusicArtist, mo:CorporateBody, mo:SoloMusicArtist, mo:MusicGroup, mo:Label, mo:member_of }

Let $\boldsymbol{O_1}=(V_1,\Sigma_1)$ be the projection $\pi[V_1](FOAF)$ of FOAF over V_1. Let $\boldsymbol{O_2}=(V_2,\Sigma_2)$ be such that Σ_2 contains just the inclusions over V_2 shown in Figure 1:

Σ_2={ mo:SoloMusicArtist ⊑ mo:MusicArtist, mo:MusicGroup ⊑ mo:MusicArtist, mo:Label ⊑ mo:CorporateBody }

Then, most of Figure 1 is captured by the union $\boldsymbol{O_3}=(V_3,\Sigma_3)$ of $\boldsymbol{O_1}$ and $\boldsymbol{O_2}$. The rest of Figure 1 is obtained by the selection $\sigma[\Sigma_5](\boldsymbol{O_3})$ of $\boldsymbol{O_3}$, where

Σ_5={ mo:SoloMusicArtist ⊑ foaf:Person, mo:MusicGroup ⊑ foaf:Group, mo:CorporateBody ⊑ foaf:Organization, (≥1 mo:member_of) ⊑ foaf:Person, (≥1 mo:member_of$^-$) ⊑ foaf:Group }

where the last two constraints indicate that the domain and range of mo:member_of are foaf:Person and foaf:Group, respectively.

Finally, we construct $\boldsymbol{O_0}=(V_0,\Sigma_0)$, the ontology that corresponds to Figure 1, in two different, albeit equivalent ways:

(1) $\boldsymbol{O_0} = \sigma[\Sigma_5](\pi[V_1](FOAF) \cup \boldsymbol{O_2})$, using the selection operation

(2) $\boldsymbol{O_0} = ((\pi[V_1](FOAF) \cup \boldsymbol{O_2}) \cup \boldsymbol{O_5})$, eliminating the selection operator

We stress that the expression of the right-hand side of Eq. (1) (or Eq. (2)) provides an explanation of how $\boldsymbol{O_0}$ is constructed from FOAF and additional terms and constraints.

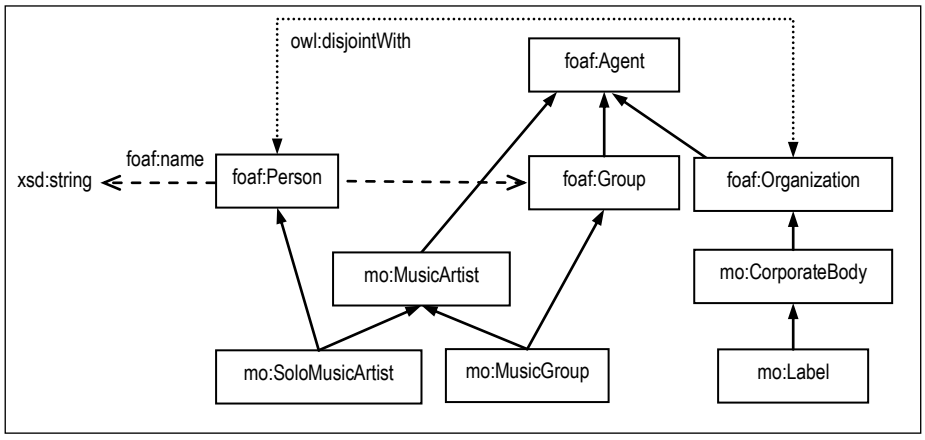

Fig.1. The class hierarchies of *MO* rooted at classes foaf:Agent and foaf:Person.

3. Computing the Operations over Lightweight Ontologies

3.1 Constraint Graphs

This section introduces the concept of constraint graphs, which leads to procedures to compute the operations over lightweight ontologies.

We say that the *complement* of a basic concept description e is $\neg e$, and vice-versa. If c is a concept description, then $\overline{c}$ denotes the complement of c.

Let Σ be a set of (unabbreviated) inclusions and Ω be a set of (unabbreviated) concept descriptions. The notion of constraint graphs [6] captures the structure of sets of constraints.

Definition 2: The labeled graph $g(\Sigma,\Omega)=(\gamma,\delta,\kappa)$ that *captures* Σ and Ω, where κ labels each node with a concept description, is defined as follows:

(i) For each concept description e that occurs on the right- or left-hand side of an inclusion in Σ, or that occurs in Ω, there is exactly one node in γ labeled with e. If necessary, the set of nodes is augmented with new nodes so that:
  (a) For each atomic concept C, there is one node in γ labeled with C.
  (b) For each atomic role P, there is one node in γ labeled with $(\geq 1\ P)$ and one node labeled with $(\geq 1\ P^{-})$.

(ii) If there is a node in γ labeled with a concept description e, then there must be exactly one node in γ labeled with $\overline{e}$.

(iii) For each inclusion $e \sqsubseteq f$ in Σ, there is an arc (M,N) in δ, where M and N are the nodes labeled with e and f, respectively.

(iv) If there are nodes M and N in γ labeled with $(\geq m\ p)$ and $(\geq n\ p)$, where p is either P or P^{-} and $m<n$, then there is an arc (N,M) in δ.

(v) If there is an arc (M,N) in δ, where M and N are the nodes labeled with e and f respectively, then there is an arc (K,L) in δ, where K and L are the nodes labeled with $\overline{f}$ and $\overline{e}$, respectively.

(vi) These are the only nodes and arcs of $g(\Sigma)$. □

Definition 3: The labeled graph $G(\Sigma,\Omega)=(\eta,\varepsilon,\lambda)$ that *represents* Σ and Ω, where λ labels each node with a set of concept descriptions, is defined from $g(\Sigma,\Omega)$ by collapsing each strongly connected component of $g(\Sigma,\Omega)$ into a single node labeled with the descriptions that previously labeled the nodes in the strongly connected component. When Ω is the empty set, we simply write $G(\Sigma)$ and say that the graph *represents* Σ. □

If a node K of $G(\Sigma,\Omega)$ is labeled with e, then $\overline{K}$ denotes the node labeled with $\overline{e}$, and $K \rightarrow M$ indicates that there is a path in $G(\Sigma,\Omega)$ from K to M.

Definition 4: Let $G(\Sigma,\Omega)=(\eta,\varepsilon,\lambda)$ be the labeled graph that represents Σ and Ω. We say that a node K of $G(\Sigma,\Omega)$ is a $\bot$-*node with level n*, for a non-negative integer n, iff one of the following conditions holds:

(i) K is a $\bot$-*node with level 0* iff
  a. K is labeled with $\bot$, or

b. there are nodes M and N, not necessarily distinct from K, and a basic concept description h such that M and N are labeled with h and $\neg h$, and $K \rightarrow M$ and $K \rightarrow N$.

(ii) K is is a $\bot$-*node with level* $n+1$ iff

a. There is a $\bot$-node M of level n, distinct from K, such that $K \rightarrow M$, and M is the $\bot$-node with the smallest level such that $K \rightarrow M$, or

b. K is labeled with a minCardinality constraint of the form $(\geq 1\ P)$ (or of the form $(\geq 1\ P^{-})$) and there is a $\bot$-node M of level n, distinct from K, such that M is labeled with $(\geq 1\ P^{-})$ (or with $(\geq 1\ P)$), and M is the $\bot$-node with the smallest level labeled with $(\geq 1\ P^{-})$ or $(\geq 1\ P)$. □

Definition 5: Let $G(\Sigma,\Omega)=(\eta,\varepsilon,\lambda)$ be the labeled graph that represents Σ and Ω. Let K be a node of $G(\Sigma,\Omega)$. We say that K is a $\bot$-*node* iff K is a $\bot$-node with level n, for some non-negative integer n. We also say that K is a $\top$-*node* iff $\overline{K}$ is a $\bot$-node. □

Theorem 1 shows how to test logical implication for $DL\text{-}Lite_{core}^{\mathcal{N}}$ inclusions [6].

Theorem 1. Let Σ be a set of $DL\text{-}Lite_{core}^{\mathcal{N}}$ inclusions and σ be a $DL\text{-}Lite_{core}^{\mathcal{N}}$ inclusion. Assume that σ is of the form $e \sqsubseteq f$ and let $\Omega = \{e, f\}$. Then, $\Sigma \models \sigma$ iff one of the following conditions holds:

(i) The node of $G(\Sigma,\Omega)$ labeled with e is a $\bot$-node; or

(ii) The node of $G(\Sigma,\Omega)$ labeled with f is a $\top$-node; or

(iii) There is a path in $G(\Sigma,\Omega)$ from the node labeled with e to the node labeled with f.

Example 1: The fragment of the Music Ontology shown in Figure 1 is formalized as the ontology $\boldsymbol{O_0}=(V_0,\Sigma_0)$, where

$V_0 = \{$ foaf:Agent, foaf:Person, foaf:Group, foaf:Organization, mo:MusicArtist, mo:CorporateBody, mo:SoloMusicArtist, mo:MusicGroup, mo:Label, mo:member_of, foaf:name, xsd:string $\}$

and the set of constraints Σ_0 shown in Table 2. Figure 3 depicts the graph $G(\Sigma_0)$ that represents Σ_0 (using unabbreviated constraints).

Table 2. Constraints of the ontology in Figure 1.

(≥1 foaf:name) ⊑ foaf:Person (≥1 foaf:name⁻) ⊑ xsd:string	(≥1 mo:member_of) ⊑ foaf:Person (≥1 mo:member_of⁻) ⊑ foaf:Group
mo:MusicArtist ⊑ foaf:Agent foaf:Group ⊑ foaf:Agent foaf:Organization ⊑ foaf:Agent mo:SoloMusicArtist ⊑ foaf:Person mo:SoloMusicArtist ⊑ mo:MusicArtist	mo:MusicGroup ⊑ mo:MusicArtist mo:MusicGroup ⊑ foaf:Group mo:CorporateBody ⊑ foaf:Organization mo:Label ⊑ mo:CorporateBody
foaf:Person ⊑ ¬foaf:Organization	foaf:Person ⊑ ¬(≥2 foaf:name)

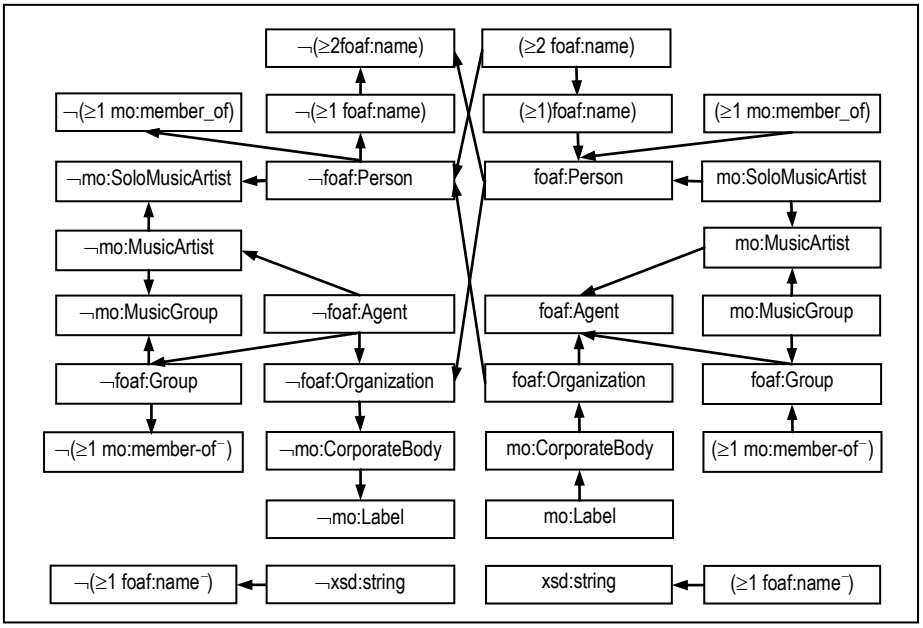

Fig. 3. The graph $G(\Sigma_{APO})$ representing the constraints of ***APO***.

3.2 Computing the Operations

Let $O_1=(V_1,\Sigma_1)$ and $O_2=(V_2,\Sigma_2)$ be two lightweight ontologies and W be a subset of V_1. We say that $O_1=(V_1,\Sigma_1)$ and $O_2=(V_2,\Sigma_2)$ are *equivalent* iff $V_1=V_2$ and $\tau[\Sigma_1]=\tau[\Sigma_2]$.

Given $O_1=(V_1,\Sigma_1)$ and W, procedure **Projection** (in Figure 4) generates Σ_2, based on the representation graph of Σ_1, so that $O_2=(W,\Sigma_2)$ is equivalent to the projection of $O_1=(V_1,\Sigma_1)$ over W. Based on Theorem 1, **Projection** generates all constraints that involve only symbols in W and that are logical consequences of Σ_1. However, it avoids generating both $e \sqsubseteq f$ and $\bar{f} \sqsubseteq \bar{e}$, which are equivalent. We note that **Projection** is non-deterministic since the set of constraints generated depends on the order that the for-loop selects pairs of nodes of $G(\Sigma_1)$, which is not unique.

The above argument can be generalized into a correctness proof of the **Projection** procedure, in the following sense. Let Σ_1/W denote the set of formulas that use only classes and properties in W and that are logically implied by Σ_1.

Theorem 2: Let $O_1=(V_1,\Sigma_1)$ be an ontology and W be a subset of V_1. Let Σ_2 be the set of constraints which **Projection** outputs for Σ_1 and W. Then, Σ_2 is logically equivalent to Σ_1/W. □

Procedures to compute the selection, union, intersection and difference of ontologies can be likewise defined.

```
Projection(V1 , Σ1 , W ; Σ2)
input:    a vocabulary V1
          a set Σ1 of normalized constraints over V1
          a subset W of V1
output:   a set of constraints Σ2
begin  Initialize Σ2 = ∅ ;
       Construct G(Σ1 ), the representation graph for Σ1;
       Mark all nodes of G(Σ1) labeled with at least one expression
            that uses only atomic concepts and atomic roles in W;
       for each node M of G(Σ1) such that M is marked and M is a ⊥-node
            for each concept description e
            such that e labels M and e uses only classes and properties in W
                 add e ⊑ ⊥ to Σ2;
       drop all ⊥-nodes and ⊤-nodes from G(Σ1);
       for each node M of G(Σ1) such that M is marked
            for each pair of concept descriptions e and f
            such that e and f label M and e and f use only classes and properties in W
                 add e ≡ f to Σ2;
       for each pair of nodes M and N of G(Σ1)      /* compute the transitive closure */
            if   M and N are marked and there is a path from M to N in G(Σ1)
            then add e ⊑ f to Σ2 where
                      e and f are expressions that label nodes M and N, respectively, and
                      e and f use only classes and properties in W, and
                      e ⊑ f is an allowed DL-Lite core inclusion (in the sense of Section 2), and
                       f ⊑ ē  is not already in Σ2      /* to avoid redundant constraints */
       return Σ2
end
```

Fig. 4. Procedure **Projection**.

4. Optimizing the Representation of Lightweight Ontologies

The procedures that implement each of the operations return a set of constraints that may contain redundancies, since they work with the transitive closure of the constraint graph (c.f. the procedure in Figure 4). Therefore, an interesting question immediately arises: How to minimize the set of constraints of a lightweight ontology (output by one such procedure)? We argue that this question is equivalent to finding the *minimum equivalent graph (MEG)* of a graph G, defined as the graph G' with the minimum set of edges such that the transitive closures of G and G' are equal. This problem has a polynomial solution for acyclic graphs and is known to be NP-hard for strongly connected graphs [1][9][10].

Let $O_1=(V_1,\Sigma_1)$ be a lightweight ontology and $G(\Sigma_1)=(\eta,\varepsilon,\lambda)$ be the constraint graph for Σ_1. In what follows, we outline how to construct a new set of constraints, Σ_2, such that Σ_2 is a minimal set and Σ_1 and Σ_2 are tautologically equivalent.

Recall that $G(\Sigma_1)$ is acyclic and that each node of $G(\Sigma_1)$ is labeled with a set of expressions that are equivalent.

Since $G(\Sigma_1)$ is acyclic, we may construct a MEG $G'=(\eta,\varepsilon',\lambda)$ of $G(\Sigma_1)=(\eta,\varepsilon,\lambda)$ in polynomial time [1][9]. The nodes of G' are those of G, with the same labels as in G.

The procedure adopted to construct a MEG of a graph has to be modified to also drop an arc *(M,N)*, if the arc $(\overline{N},\overline{M})$ connecting the dual nodes of *M* and *N* is dropped. This modification is necessary to preserve the characteristics of constraint graphs (see Definitions 2 and 3). Then, for each arc *(M,N)* in ε', select an expression *e* that labels *M* and an expression *f* that labels *N*, and add the constraint $e \sqsubseteq f$ to Σ_2.

The apparent problem lies in the expressions that label each node of $G(\Sigma_1)$. Indeed, by Definition 3, $G(\Sigma_1)$ is defined from $g(\Sigma_1)$ by collapsing each strongly connected component of $g(\Sigma_1)$ into a single node labeled with the expressions that previously labeled the nodes in the strongly connected component. Thus, we would have to construct a MEG, *G"*, of each strongly connected component of $g(\Sigma_1)$ and generate a set of constraints from *G"* as before. However, constructing a MEG of a strongly connected graph is NP-hard, as indicated before.

Recall that, for any two expressions *e* and *f* that label the same node of $G(\Sigma_1)$, we have that Σ_1 logically implies $e \equiv f$. From the point of view of an economical ontology description, for each strongly connected component of $g(\Sigma_1)$, we suggest to construct a minimal set of equivalences which are tautologically equivalent to the inclusions that correspond to the arcs of $g(\Sigma_1)$. But this new problem is equivalent to constructing a spanning tree *T* of each strongly connected component of $g(\Sigma_1)$, which can be done in polynomial time. Then, for each edge *{M,N}* of *T*, add $e \equiv f$ to Σ_2, where *e* labels *M* and *f* labels *N* in $g(\Sigma_1)$. If *T* has *k* nodes, we generate *k*-1 equivalences.

The final set of constraints, Σ_2, contains a minimal set of inclusions, which correspond to the arcs of *G'*, and a minimal set of equivalences, which correspond to the edges of spanning trees of the strongly connected components of $g(\Sigma_1)$. Finally, by construction, Σ_1 and Σ_2 will be tautologically equivalent.

5 Conclusions

In this paper, we introduced a set of concepts and procedures that promote constraint reuse in ontology design. We defined a set of operations that extend the idea of namespaces to take into account constraints and that treat ontologies as theories. We showed how to compute the operations when the ontologies are built upon DL-Lite core with arbitrary number restrictions. For such ontologies, we also showed how to minimize the set of constraints.

As for current work, from a formal perspective, we are extending the results to a more expressive variant of DL-Lite core, considered in [8], which supports a restricted form of role hierarchy. From a practical perspective, we have implemented a tool that accepts lightweight ontologies, described in OWL, and offers an interface to apply the operations to create new ontologies. We are also designing a tool to extract constraints from a Linked Data source ***S*** by combining the information in the VoiD document [2] of the source, if any, with a probing technique. The tool explores the idea of the operations introduced in this paper both to compute the constraints that apply to ***S*** and to document them in an extension of the VoiD vocabulary.

Selected References

[1] Aho, A. V., Garey, M. R., Ullman, J. D. (1972). The Transitive Reduction of a Directed Graph. SIAM J. Comp., 1(2), 131-137.

[2] Alexander, K., Cyganiak, R., Hausenblas, M., Jun Zhao, J. Describing Linked Datasets with the VoID Vocabulary. W3C Interest Group Note (03 March 2011). Available at: http://www.w3.org/TR/void/

[3] Artale, A.; Calvanese, D.; Kontchakov, R.; Zakharyaschev, M. The DL-Lite family and relations. *J. of Artificial Intelligence Research* 36, 1–69.

[4] Baader, F., Nutt, W. Basic Description Logics. In: F. Baader, D. Calvanese, D.L. McGuiness, D. Nardi, P.F. Patel-Schneider (eds), *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge U. Press, UK (2003), pp. 43–95.

[5] Brickley, D., Miller, L. FOAF Vocabulary Specification 0.98. Namespace Document 9 August 2010 - *Marco Polo Edition*.

[6] Casanova, M.A., Lauschner, T., Leme, L. A. P. P., Breitman, K. K., Furtado, A. L., Vidal, V. M. P. Revising the Constraints of Lightweight Mediated Schemas. Data & Knowledge Engineering, v.69, pp.1274 - 1301, 2010.

[7] Casanova, M. A., Breitman, K. K., Furtado, A. L., Vidal, V. M. P., Macêdo, J. A. F. The Role of Constraints in Linked Data. Proceedings of the Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011, Part II. Lecture Notes in Computer Science. Heidelberg: Springer, 2011. v.7045. p.781 - 799.

[8] Casanova, M.A., Breitman,K.K., Furtado, A.L., Vidal, V.M.P., Macêdo, J.A.F. An Efficient Proof Procedure for a Family of Lightweight Database Schemas. In: Michael G. Hinchey and Lorcan Coyle (eds.), *Conquering Complexity*. Springer, London, 2012.

[9] Hsu, H. T. (1975). An Algorithm for Finding a Minimal Equivalent Graph of a Digraph. Journal of the Association for Computing Machinery, 22(1), 11-16.

[10] Khuller, S., Raghavachari, B., Young, N. (1995). Approximating the Minimum Equivalent Digraph. SIAM Journal on Computing, 24(4), 859-872.

[11] Raimond, Y., Giasson, F. Music Ontology Specification. Specification Document - 28 November 2010. Latest version: http://purl.org/ontology/mo/ (RDF/XML, Turtle).