PUC

ISSN 0103-9741

Monografia em Ciência da Computação
n° 01/11

The Role of Constraints in Linked Data

Marco A. Casanova, Karin K. Breitman, Antonio L. Furtado
Departamento de Informática – PUC-Rio

Vânia M.P. Vidal, José A. F. de Macêdo
Departamento de Computação – UFC

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO

Monografias em Ciência da Computação, No. 01/11 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena April, 2011

The Role of Constraints in Linked Data

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

[1]Department of Informatics – PUC-Rio – Rio de Janeiro, RJ – Brazil

{casanova, karin, furtado}@inf.puc-rio.br

[2]Department of Computing, Federal University of Ceará – Fortaleza, CE – Brazil

{vvidal, jose.macedo}@lia.ufc.br

Abstract. This paper investigates the role that constraints play in Linked Data in the context of a multi-step modeling process, involving three ontologies. The source ontology provides a local model of the exported data. The domain ontology provides a conceptual model of the application domain. The application ontology describes the external model of the exported data, using a subset of the vocabulary of the domain ontology. The main contributions of the paper are methods for constructing application ontology constraints and for defining the mappings between the three ontologies. The methods assume that the ontologies are written in an expressive family of languages and depend on a procedure to test logical implication, which explores the structure of sets of constraints.

Keywords: constraints, Linked Data, mediated schema.

Resumo. Este trabalho investiga o papel de restrições de integridade em *Linked Data*, no contexto de um processo de modelagem envolvendo ontologias em três níveis. A ontologia da fonte de dados oferece um modelo local para os dados exportados. A ontologia de domínio provê um modelo conceitual do domínio de aplicação. A ontologia de aplicação descreve os dados exportados usando um subconjunto do vocabulário da ontologia de domínio. A principal contribuição do trabalho consiste em métodos para construção das restrições de integridade da ontologia de aplicação e para definir os mapeamentos entre as três ontologias. Os métodos assumem que as ontologias estão escritas em uma família de linguagens com poder de expressividade adequado e dependem de um procedimento para testar implicação lógica que explora a estrutura do conjunto de restrições.

Palavras-chave: restrições de integridade, Linked Data, esquema mediado.

Summary

In charge of publications:

Rosane Teles Lins Castilho

Assessoria de Biblioteca, Documentação e Informação

PUC-Rio Departamento de Informática

Rua Marquês de São Vicente, 225 - Gávea

22451-900 Rio de Janeiro RJ Brasil

Tel. +55 21 3527-1516 Fax: +55 21 3527-1530

E-mail: bib-di@inf.puc-rio.br

Web site: http://bib-di.inf.puc-rio.br/techreports/

1 Introduction

The term *Linked Data* refers to a set of best practices for publishing and connecting structured data on the Web [3]. A Linked Data source may publish a local model of the exported data. A *model* is expressed as a *vocabulary*, that is, a collection of classes and properties, expressed in RDF, using terms from RDFS and OWL. The Linked Data source may also publish a mapping between its local vocabulary and a *reference* vocabulary (or several such vocabularies). A *mapping* may be expressed as a set of RDF triples that link classes and properties in one vocabulary to those in another, or it may be defined using a schema mapping language.

We investigate in this paper the role that constraints play in Linked Data. The motivation lies in that a Linked Data source should publish its local vocabulary together with a set of local constraints that capture the semantics of the local classes and properties. If the Linked Data source publishes a mapping between its local vocabulary and a reference vocabulary, then we have to specify the local constraints so that they are somehow consistent with those of the reference classes and properties.

More precisely, an *ontology* is a pair $O = (V_O , C_O)$, where V_O is a vocabulary and C_O is a set of constraints over V_O. A *mapping* from a *source* ontology $S = (V_S , C_S)$ into a *target* ontology $T = (V_T , C_T)$ is a set of definitions of the form $D \equiv e$, where D is a class (or property) in V_T and e is a class definition (or a property definition) that uses symbols in V_S. The *identity mapping* maps a symbol into itself (the symbol must therefore occur in both V_S and V_T). We do not require that a mapping provides definitions for all symbols in V_T.

To better formulate the local constraint specification problem, we introduce a multi-step process to model Linked Data. Briefly, the process goes as follows. First create a *source ontology* $S = (V_S , C_S)$, which models the data to be published. Then select a *domain ontology* $D = (V_D , C_D)$, which models the application domain. In fact, D may be a combination of ontologies covering distinct domains. Proceed by creating *a source-to-domain* mapping γ from S into D. Note that γ may not have definitions for all symbols in V_D. Let V_A be the subset of V_D such that D is in V_A iff γ has a definition of the form $D \equiv e$. Create a set of constraints C_A that model the semantics of the classes and properties in V_A. That is, create an *application ontology* $A=(V_A,C_A)$. We reinterpret γ as *a source-to-application* mapping γ_{SA} from S into A and introduce an *application-to-domain* identity mapping γ_{AD} from A into D that associates each symbol in V_A into itself (this is possible since V_A is a subset of V_D). The problem is how to create C_A and decide what properties $A = (V_A , C_A)$, γ_{SA} and γ_{AD} should have.

Suppose first that we require that the data exported through γ_{SA} satisfies C_A, and that C_A contains all constraints that can be derived from C_D and that use only symbols in V_A. In this case, we say that the application ontology is an *open fragment* of the domain ontology. The last step of the process is then to create C_A and perhaps adjust γ_{SA} so that these requirements are met. Sections 3.1 and 4 formulate these requirements in detail and describe methods to support this step.

Suppose now that we require that the data exported through γ_{SA} satisfies C_D, when all classes and properties in V_D, but not in V_A, are taken as the empty set (when the source data is published). In this case, we say that the application ontology is a *closed fragment* of the domain ontology. The

last step of the process now is to adjust V_A and γ_{SA} so that this requirement is met. Sections 3.2 and 4 formulate these requirements in detail and describe methods to support this step.

These requirements are justified since, if A is a closed fragment of D, then any Web application that processes data modeled according to D will also be able to process data published by the data source. Furthermore, if A is an open fragment of D, then any Web application that processes data modeled according to D and *that uses only the classes and properties in the vocabulary of* A will also be able to process the data published by the data source.

The main contributions of the paper are methods for constructing application ontology constraints and for adjusting the definition of the mappings, when the application ontology is an open or a closed fragment of the domain ontology. The methods assume that the ontologies are written in an expressive family of attributive languages and depend on a procedure to test logical implication, which explores the structure of sets of constraints, captured as a *constraint graph* [5].

The results reported in the paper cover a topic – the role that constraints play in Linked Data – that is much neglected in the literature. The question of Linked Data semantics is not new, though. Recent investigation [7,8,12] in fact questions the correct use of owl:sameAs to inter-link datasets.

They contribute to the discussion on the triplication of relational databases [6]. Indeed, triplification tools (see [17] for a list) typically do not consider constraints.

The results in this paper also contribute to improving ontology browsing tools based on the idea of *focus+context* [16], where the notion of focus would be carried out by a vocabulary selection and the notion of context would be provided by the constraints. The methods to construct fragments of the domain ontology would act as a lens through which the user would browse the (large) domain ontology. The notions of open and closed ontology fragments are akin to the classification of the mappings between the local sources and the mediated schema into sound, exact or complete [9]. The three-step process adopted in the paper is similar to that proposed in [13,15]. We reported a preliminary investigation on the generation of application ontologies in [13], but without using constraint graphs.

The paper is organized as follows. Section 2 presents the formal framework adopted in the paper. Section 3 focuses on how to construct the constraints of the application ontology. Section 4 discusses how to adjust the mapping from the source ontology into the application ontology. Finally, Section 5 contains the conclusions.

2 A Formal Framework

2.1 A Brief Review of Attributive Languages

We adopt a family of *attributive languages* [1] defined as follows. A *language* $\mathcal{L}$ in the family is characterized by an *alphabet* $\boldsymbol{A}$, consisting of a set of *atomic concepts*, a set of *atomic roles*, the *universal concept* $\top$ and the *bottom concept* $\bot$. The set of *role descriptions* and the set of *concept descriptions* of $\mathcal{L}$ (or in $\boldsymbol{A}$) are defined as follows:

- An atomic concept, and the universal and bottom concepts are concept descriptions, and an atomic role is a role description

- If e and f are concept descriptions and p is a role description, then $\neg e$ (*negation*), $e \sqcup f$ (*union*), and $(\geq n\ p)$ (*at-least restriction*) are concept descriptions, and p^- (*inverse*) is a role description.

An *interpretation* $\boldsymbol{s}$ for $\boldsymbol{A}$ consists of a nonempty set Δ^s, the *domain* of $\boldsymbol{s}$, whose elements are called *individuals*, and an *interpretation function*, also denoted $\boldsymbol{s}$, where:

- $\boldsymbol{s}(\bot) = \varnothing$ and $\boldsymbol{s}(\top) = \Delta^s$
- $\boldsymbol{s}(A) \subseteq \Delta^s$, for each atomic concept A of $\boldsymbol{A}$
- $\boldsymbol{s}(P) \subseteq \Delta^s \times \Delta^s$, for each atomic role P of $\boldsymbol{A}$

The function $\boldsymbol{s}$ is extended to role and concept descriptions of L as follows:

- $\boldsymbol{s}(\neg e) = \Delta^s - \boldsymbol{s}(e)$ (the complement of $\boldsymbol{s}(e)$ w.r.t. Δ^s)
- $\boldsymbol{s}(e \sqcup f) = \boldsymbol{s}(e) \cup \boldsymbol{s}(f)$ (the union of $\boldsymbol{s}(e)$ and $\boldsymbol{s}(f)$)
- $\boldsymbol{s}(\geq n\ p) = \{I \in \Delta^s \mid |\{J \in \Delta^s \mid (I,J) \in \boldsymbol{s}(p)\}| \geq n\}$

  (the set of individuals that $\boldsymbol{s}(p)$ relates to at least n distinct individuals)
- $\boldsymbol{s}(p^-) = \boldsymbol{s}(p)^-$ (the inverse of $\boldsymbol{s}(p)$)

A *formula* of $\mathcal{L}$ (or in $\boldsymbol{A}$) is an expression of the form $u \sqsubseteq v$, called an *inclusion*, or of the form $u \equiv v$, called an *equivalence*, where u and v are both concept descriptions or they are both role descriptions of $\mathcal{L}$. A *definition* is an equivalence of the form $D \equiv e$, where D is an atomic concept and e is a concept description, or D is an atomic role and e is a role description.

Let $\boldsymbol{s}$ be an interpretation for $\boldsymbol{A}$, σ be a formula and Σ and Γ be sets of formulas of $\mathcal{L}$. We say that

- $\boldsymbol{s}$ *satisfies* $u \sqsubseteq v$ iff $\boldsymbol{s}(u) \subseteq \boldsymbol{s}(v)$, and $\boldsymbol{s}$ *satisfies* $u \equiv v$ iff $\boldsymbol{s}(u) = \boldsymbol{s}(v)$
- $\boldsymbol{s}$ is a *model* of σ, denoted $\boldsymbol{s} \models \sigma$, iff s satisfies σ
- $\boldsymbol{s}$ is a *model* of Σ, denoted $\boldsymbol{s} \models \Sigma$, iff $\boldsymbol{s}$ satisfies all formulas in Σ
- Σ *logically implies* σ, denoted $\Sigma \models \sigma$, iff any model of Σ satisfies σ
- Σ *logically implies* Γ, denoted $\Sigma \models \Gamma$, iff any model of Σ is also a model of Γ
- Σ is *tautologically equivalent to* Γ iff Σ *logically implies* Γ, and vice-versa

If $\boldsymbol{B}$ is a subset of $\boldsymbol{A}$, then $\Sigma / \boldsymbol{B}$ denotes the set of formulas σ using only symbols in $\boldsymbol{B}$ such that Σ logically implies σ.

Finally, we use the following abbreviations: "$e \sqcap f$" for "$\neg(\neg e \sqcup \neg f)$" (*intersection*), "$\exists p$" 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*).

2.2 Extralite Ontologies and Ontology Mappings

We will work with *extralite ontologies* [5] that partially correspond to OWL Lite.

Definition 1: An *extralite ontology* is a pair $O = (V_O, C_O)$ such that

(i) V_O is a finite alphabet, called the *vocabulary* of O, whose atomic concepts and atomic roles are called the *classes* and *properties* of O, respectively.

(ii) C_O is a set of formulas in V_O, called the *constraints* of O, which must be of one the forms

- *Domain Constraint:* $\exists P \sqsubseteq D$ (property P has class D as domain)

- *Range Constraint:* $\exists P^{-} \sqsubseteq R$ (property P has class R as range)
- *minCardinality Constraint:* $C \sqsubseteq (\geq k\ P)$ or $C \sqsubseteq (\geq k\ P^{-})$

  (property P or its inverse P^{-} maps each individual in class C to at least k distinct individuals)
- *maxCardinality Constraint:* $C \sqsubseteq (\leq k\ P)$ or $C \sqsubseteq (\leq k\ P^{-})$

  (property P or its inverse P^{-} maps each individual in class C to at most k distinct individuals)
- *Subset Constraint:* $E \sqsubseteq F$ (class E is a subclass of class F)
- *Disjointness Constraint:* $E \mid F$ (classes E and F are disjoint). □

We will sometimes associate a *prefix*, such as "*o:*", to the alphabet V_O and indicate that a symbol T occurs in V_O by writing "*o:T*". We *normalize* a constraint by eliminating any abbreviated form used. For example, "$\exists P \sqsubseteq D$" is normalized as "$(\geq 1\ P) \sqsubseteq D$" and "$C \mid D$" as "$C \sqsubseteq \neg D$". A *constraint expression* is an expression that may occur on the right- or left-hand sides of a normalized constraint.

A *mapping* from a *source ontology* $\mathcal{S} = (V_S, C_S)$ into a *target ontology* $\mathcal{T} = (V_T, C_T)$ is a set γ_{ST} of definitions of the form $D \equiv e$ where D is an atomic concept (or atomic role) in V_T and e*u* is a concept description (or a role description) in V_S such that no two definitions in γ_{ST} have the same symbol on the left-hand side. Note that we do not require that γ_{ST} has a definition for each symbol in V_T. Finally, γ_{ST} induces a function $\hat{\gamma}_{ST}$ from interpretations for V_S into interpretations for V_T, as defined in [5].

2.3 Constraint Graphs

In this section, we introduce the notion of concept graphs, which captures the structure of sets of constraints and is essential to the constraint construction methods of Section 3 and to the mapping redefinition method of Section 4.

We say that the *complement* of a non-negated expression e is $\neg e$, and vice-versa; the *complement* of $\bot$ is $\top$, and vice-versa. If c is an expression, then $\bar{c}$ denotes of complement of c.

Let Σ be a set of normalized constraints and Ω be a set of constraint expressions.

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

(i) For each concept expression 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 expression e, then there must be exactly one node in γ labeled with $\bar{e}$.

(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 $\bar{f}$ and $\bar{e}$, 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 expressions, is defined from $g(\Sigma,\Omega)$ by collapsing each clique of $g(\Sigma,\Omega)$ into a single node labeled with the expressions that previously labeled the nodes in the clique. 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 an expression e, then $\overline{K}$ denotes the node labeled with $\bar{e}$ (which may be K itself). We use $K\rightarrow M$ to indicate 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 $\perp$-*node with level n*, for a non-negative integer n, iff one of the following conditions holds:

(i) K is is a $\perp$-*node with level 0* iff
  a. K is labeled with $\perp$, or
  b. there are nodes M and N, not necessarily distinct from K, and a non-negated concept expression 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 $\perp$-*node with level n+1* iff
  a. There is a $\perp$-node M of level n, distinct from K, such that $K\rightarrow M$, and M is the $\perp$-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 $\perp$-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 $\perp$-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 $\perp$-*node* iff K is a $\perp$-node with level n, for some non-negative integer n. We also say that K is a $\top$-*node* iff $\overline{K}$ is a $\perp$-node. □

Based on constraint graphs, we introduce a procedure to test logical implication for extralite ontologies, whose soundness and completeness is established in [5]:

IMPLIES(Σ, $e \sqsubseteq f$)
input: a set Σ of normalized constraints and a normalized constraint $e \sqsubseteq f$
output: "YES - Σ logically implies $e \sqsubseteq f$"
"NO - Σ does not logically imply $e \sqsubseteq f$"
begin Construct $G(\Sigma, \{e, f\})$, the representation graph for Σ and $\{e, f\}$;
if the node of $G(\Sigma, \{e, f\})$ labeled with e is a $\bot$-node, or
the node of $G(\Sigma, \{e, f\})$ labeled with f is a $\top$-node, or
there is a path in $G(\Sigma,\{e, f\})$ from the node labeled with e to the node labeled with f,
then return "YES - Σ logically implies $e \sqsubseteq f$";
else return "NO - Σ does not logically imply $e \sqsubseteq f$";
end

2.4 A Simple Example

The following example illustrates the concepts introduced thus far.

Example 1: Figure 1(a) shows the ER diagram of ontology ***PC*** = (V_{PC}, C_{PC}) (a fragment of a model for a phone company). Figure 1(b) formalizes the set of constraints C_{PC}. The first column shows the domain and range constraints; for example, `placedBy` is an atomic role modeling a binary relationship from `Call` to `Phone`. The second column depicts the cardinality constraints; for example, `number` has maxCardinality equal to 1 w.r.t. `Phone`. The third column contains the subset and disjointness constraints; for example, `MobilePhone` and `FixedPhone` are disjoint. (For simplicity, we ignore data types, such as `String`, which would be treated as classes). Figure 2 depicts the graph $G(C_{PC})$ that represents C_{PC} (using normalized constraints). □

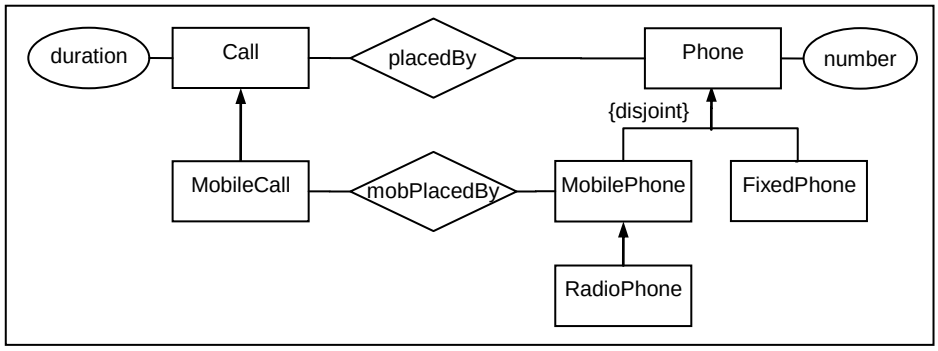

Fig. 1(a). ER diagram of ***PC*** (without cardinalities).

(≥1 number) ⊑ Phone (≥1 duration) ⊑ Call (≥1 placedBy) ⊑ Call (≥1 placedBy⁻) ⊑ Phone (≥1 mobPlacedBy) ⊑ MobileCall (≥1 mobPlacedBy⁻) ⊑ MobilePhone	Call ⊑ ¬(≥2 duration) Phone ⊑ ¬(≥2 number)	FixedPhone ⊑ Phone MobilePhone ⊑ Phone RadioPhone ⊑ MobilePhone MobilePhone ⊑ ¬FixedPhone MobileCall ⊑ Call

Fig. 1(b). Constraints of ***PC***, after normalization.

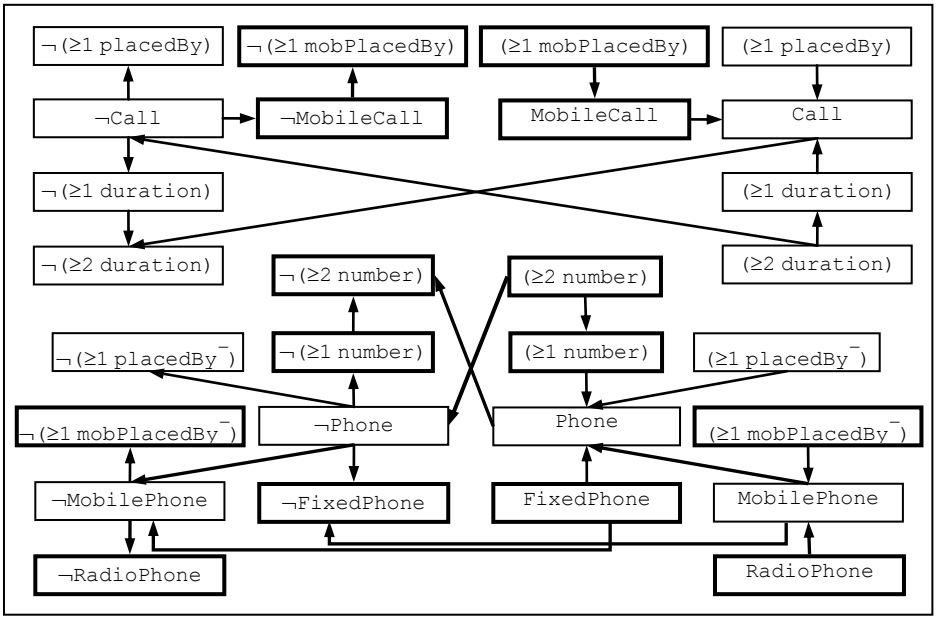

Fig. 2. The graph $G(C_{PC})$ representing the constraints of ***PC***, after normalization.

3.1 Application Ontology as an Open Fragment of the Domain Ontology

The design of the application ontology $\mathcal{A}$ and definition of the mappings depend on what requirements they must satisfy, discussed in detail in this and the next sections.

Recall that γ_{SA} induces a function $\hat{\gamma}_{SA}$ from interpretations for V_S into interpretations for V_A. Also recall that C_D/V_A denotes the set of formulas σ using only symbols in V_A such that C_D logically implies σ. Consider the following set of requirements:

R0. V_A is a subset of V_D
R1. C_A is tautologically equivalent to C_D/V_A
R2. For any model $\boldsymbol{s}$ of C_S, $\hat{\gamma}_{SA}(\boldsymbol{s})$ is a model of C_A

An application ontology $\mathcal{A}$ that satisfies R0 and R1 is called an *open fragment* of $\mathcal{D}$. Requirement R0 guarantees that the data is exported in a subset of the vocabulary of the domain ontology. Requirements R1 and R2 indicate that the data published by the data source through γ_{SA} will be consistent with all constraints that can be derived from C_D and that use only symbols in V_A. Intuitively, Requirements R0, R1 and R2 imply that any Web application that processes data modeled according to $\mathcal{D}$ and *that uses only the classes and properties in* V_A will also be able to process the data published by the data source through γ_{SA}.

Assume that the designer has already created V_A by selecting symbols from V_D so that R0 is trivially satisfied. The procedure **OpenFragment** generates C_A so that R1 is satisfied, based on the representation graph of C_D and on the vocabulary V_A:

OpenFragment(C_D , V_A ; C_A)
input: the set C_D of normalized constraints of the domain ontology
the vocabulary V_A of the application ontology
output: the set C_A of normalized constraints of the application ontology
begin Initialize $C_A = \varnothing$;
Construct $G(C_D)$, the representation graph for C_D;
Mark all nodes of $G(C_D)$ labeled with expressions that use only
atomic concepts and atomic roles in V_A;
for each pair of nodes M and N of $G(C_D)$
if M and N are marked and there is a path from M to N in $G(C_D)$
then add $e \sqsubseteq f$ to C_A where
e and f are expressions that label nodes M and N, respectively, and
e and f are expression of V_A, and
$e \sqsubseteq f$ is an allowed constraint (in the sense of Section 2.), and
$\bar{f} \sqsubseteq \bar{e}$ is not already in C_A /* to avoid redundant constraints */
return C_A;
end

We note that **OpenFragment** is non-deterministic since the set of constraints generated depends on the order that the for-loop selects pairs of nodes of $G(C_D)$, which is not unique. The correctness of **OpenFragment** follows from the correctness of **IMPLIES**, introduced at the end of Section 2.3.

We will discuss how to adjust the source-to-application mapping to guarantee R2 in Section 4.

Example 2: Assume that the domain ontology is ***PC*** = $(V_{PC}$, $C_{PC})$, introduced in Example 1, and that the designer wants to define an application ontology, called ***LPC*** (for local phone company). He starts by manually defining the vocabulary V_{LPC} by selecting symbols from V_{PC}. Assume that V_{LPC} is:

(1) V_{LPC} = {`FixedPhone`, `RadioPhone`, `Number`, `MobileCall`, `mobPlacedBy`}

Then, procedure **OpenFragment** generates the constraints C_{LPC} for ***LPC***:

(2) (≥ 1 mobPlacedBy) ⊑ MobileCall
(3) FixedPhone ⊑ ¬(≥ 2 number)
(4) RadioPhone ⊑ ¬(≥ 2 number)
(5) RadioPhone ⊑ ¬FixedPhone

Recall that Figure 2 shows $G(C_{PC})$, the representation graph for C_{PC}. To help follow this example, the thicker boxes in Figure 2 indicate the marked nodes (that contain expressions in V_{LPC}) and the thicker lines indicate the paths between marked nodes. Note that there is a path from `FixedPhone` to ¬(≥ 2 number), which implies that (3) is a logical consequence of C_{PC}. The other constraints follow likewise.

Therefore, constraints (2) to (5) use only symbols in V_{LPC} and they are logical consequences of C_{PC} (albeit not necessarily in C_{PC}). In fact, they meet Requirement R1. But, for example, **OpenFragment** will not output

(6) FixedPhone $\sqsubseteq$ ($\geq$ 1 number)
(7) ($\geq$ 1 mobPlacedBy$^-$) $\sqsubseteq \neg$($\geq$ 2 number)
(8) FixedPhone $\sqsubseteq \neg$RadioPhone

The procedure does not output (6) since there is no path $G(C_{PC})$ from FixedPhone to ($\geq$ 1 number), that is, (6) is not a logical consequence of C_{PC}. It does not output (7) since it is not an allowed constraint. Finally, it does not output (8) because (5) is already in C_{LPC}. However, since **OpenFragment** is non-deterministic, it could have returned (8) instead of (5). □

3.2 Application Ontology as a Closed Fragment of the Domain Ontology

Let C_A^+ be the set of constraints C_A extended with new constraints of the form $C \sqsubseteq \bot$ (or $P \sqsubseteq \bot$) that force each class C (or property P) in V_D, but not in V_A, to be the empty set. We now consider a different set of requirements:

R0. V_A is a subset of V_D

R1'. C_A^+ logically implies C_D

R2. For any model $\boldsymbol{s}$ of C_S, $\hat{\gamma}_{SA}(\boldsymbol{s})$ is a model of C_A

An application ontology $\mathcal{A}$ that satisfies R0 and R1' is called a *closed fragment* of $\mathcal{D}$. Requirement R0 again guarantees that the data is exported in a subset of the vocabulary of the domain ontology. Requirements R1' and R2 indicate that the data published by the data source through γ_{SA} satisfies C_D, when each class C (or property P) in V_D, but not in V_A, is taken to be the empty set. Note that R1' then implies that the data published by the data source through γ_{SA} can always be extended to a consistent state of $\mathcal{D}$. Intuitively, Requirements R0, R1' and R2 imply that any Web application that processes data modeled according to $\mathcal{D}$ will also be able to process data published by the data source through γ_{SA}, when each class C (or property P) in V_D, but not in V_A, is taken to be the empty set.

Assume that the designer has already created V_A by selecting symbols from V_D so that R0 is trivially satisfied. In this section, we introduce a procedure, **ClosedFragment**, to extend V_A and γ_{SA} and to create C_A so that R1' is satisfied. Again, we will discuss how to adjust the source-to-application mapping to guarantee R2 in Section 4.

Contrasting with the result in Section 3.1, the second method may fail, if atomic roles must be included in V_A.

The procedure **ClosedFragment** is again based on the representation graph of C_D:

ClosedFragment(C_D , V_A , γ_{SA} ; $\bar{V}_A$, $\bar{C}_A$, $\bar{\gamma}_{SA}$)

input: the set C_D of normalized constraints of the domain ontology
the vocabulary V_A of the application ontology
the source-to-application mapping γ_{SA}

output: the new vocabulary $\bar{V}_A$ of the application ontology
the set of constraints $\bar{C}_A$ of the application ontology
the new source-to-application mapping $\bar{\gamma}_{SA}$

begin Initialize $\bar{V}_A = V_A$, $\bar{C}_A = \emptyset$ and $\bar{\gamma}_{SA} = \gamma_{SA}$;
Construct $G(C_D)$, the constraint graph for C_D;
Mark all nodes of $G(C_D)$ labeled with expressions that use only atomic concepts and atomic roles in V_A;
Create a new graph G_A by deleting any node N from $G(C_D)$ such that
N is labeled with positive expressions and
N has no antecedent which is marked and labeled with a positive expression, or
N is labeled with negative expressions and
N has no descendent which is marked and labeled with a negative expression;
Let AR be the set of atomic roles that occur in expressions that label nodes in G_A but which are not in V_A;
if AR is not empty,
then stop warning that "a closed fragment cannot be automatically created";
Let AC be the set of atomic concepts that label nodes in G_A but which are not in V_A;
for each s in AC
add s to $\bar{V}_A$;
add "$s \equiv s_1 \sqcup \ldots \sqcup s_n$" to $\bar{\gamma}_{SA}$,
where s labels a node M and $s_1, \ldots, s_n$ label nodes $M_1, \ldots, M_n$, and
$M_1, \ldots, M_n$ are all nodes such that (M_k, M) is in G_A ;
Recursively replace each s_i in a new definition "$s \equiv s_1 \sqcup \ldots \sqcup s_n$" added to $\bar{\gamma}_{SA}$ until the right-hand side of the definitions in $\bar{\gamma}_{SA}$ contain only symbols in the vocabulary of the source ontology.

for each arc (M,N) of G_A
add $e \sqsubseteq f$ to $\bar{C}_A$ where
e and f are expressions that label nodes M and N, respectively, and
e and f are expression of V_A, and
$e \sqsubseteq f$ is an allowed constraint (in the sense of Section 2.), and
$\bar{f} \sqsubseteq \bar{e}$ is not already in $\bar{C}_A$ /* to avoid redundant constraints */
return $\bar{V}_A$, $\bar{C}_A$, $\bar{\gamma}_{SA}$;
end

We note that **ClosedFragment** is also non-deterministic. Furthermore, the recursive adjustment of the definitions added to $\bar{\gamma}_{SA}$ is always possible since $G(C_D)$ is acyclic, by Definition 3. The correctness of **ClosedFragment** also follows from the correctness of **IMPLIES**, introduced at the end of Section 2.3.

Example 3: Consider the same scenario as in Example 2. The goal now is to compute the constraints of ***LPC*** so that ***LPC*** is a closed fragment of ***PC***. Recall that V_{LPC} is:

(1) V_{LPC} = {FixedPhone, RadioPhone, Number, MobileCall, mobPlacedBy}

Then, **ClosedFragment** outputs the new vocabulary:

(2) $\bar{V}_{LPC}$ = {FixedPhone, RadioPhone, MobilePhone, Phone, Number, MobileCall, Call, mobPlacedBy }

and the normalized constraints shown in Figure 3. The new source-to-application mapping is defined in two stages. The first stage (in the first for-each loop of **ClosedFragment**) generates the following new definitions:

(3) a:MobilePhone ≡ a:RadioPhone
(4) a:Phone ≡ a:MobilePhone ⊔ a:FixedPhone
(5) a:Call ≡ a:MobileCall

Indeed, recall that Figure 2 shows $G(C_{PC})$, the representation graph for C_{PC}. Then, for example, the first loop generates the definition in (4) since the node labeled with Phone has two antecedents, labeled with FixedPhone and MobilePhone.

The second stage recursive replaces each symbol in $\bar{V}_A$ that occurs on the right-hand side of (3), (4) or (5) by its definition in $\bar{\gamma}_{SA}$ until the right-hand sides of all definitions in $\bar{\gamma}_{SA}$ contain only symbols in V_S, the vocabulary of the source ontology. □

(≥1 number) ⊑ Phone (≥1 mobPlacedBy) ⊑ MobileCall (≥1 mobPlacedBy⁻) ⊑ MobilePhone	Phone ⊑ ¬(≥2 number)	FixedPhone ⊑ Phone MobilePhone ⊑ Phone RadioPhone ⊑ MobilePhone MobilePhone ⊑ ¬FixedPhone MobileCall ⊑ Call

Fig. 3. Constraints of ***LPC*** when ***LPC*** is a closed fragment of ***PC***.

4 Adjusting the source-to-application mapping

In this section, we discuss how to guarantee Requirement R2, namely:

R2. For any model $\boldsymbol{s}$ of C_S, $\hat{\gamma}_{SA}(\boldsymbol{s})$ is a model of C_A

We reformulate R2 in a way that is both intuitive and amenable to mechanization. The essential problem is that C_A reflects C_D, the constraints of $\mathcal{D}$, rather than C_S, the constraints of $\mathcal{S}$. We then proceed as follows (the reader should pay attention to the subscripts used). Assume that, after the designer selects V_A, he also creates a set of constraints C_{AS}, written in V_A, that reflects C_S, in the sense that

R3. For any model $\boldsymbol{s}$ of C_S, $\hat{\gamma}_{SA}(\boldsymbol{s})$ is a model of C_{AS}

We call C_{AS} the set of *endogenous constraints* of $\mathcal{A}$ and C_A the set of *exogenous constraints* of $\mathcal{A}$. This choice of terms calls attention to the fact that C_{AS} reflects C_S whereas C_A reflects C_D. We then require that:

R4. C_{AS} logically implies C_A

Note that R3 and R4 trivially imply R2. The question now is how to compute C_{AS} and perhaps adjust γ_{SA} so that they satisfy R3 and R4. In the rest of this section, we very briefly address this question.

We first illustrate how to compute C_{AS} so that it satisfies R3, when γ_{SA} is a renaming, which is perhaps the simplest case. Let V_{SA} be the subset of V_S such that U is in V_{SA} iff γ_{SA} has a definition of the form $a{:}T \equiv s{:}U$. Let C_{SA} be the set of constraints that procedure **OpenFragment** returns when called with parameters C_S and V_{SA}. Then, we know from Section 3.1 that C_{SA} is tautologically equivalent to C_S/V_{SA}, the set of formulas σ using only symbols in V_{SA} such that C_S logically implies σ. We construct C_{AS} by applying the renaming γ_{SA} to C_{SA}, that is, in each constraint of C_{SA}, we replace U by T, whenever γ_{SA} has a definition of the form $a{:}T \equiv s{:}U$. Then, C_{AS} will satisfy R3.

With the help of an example, we now briefly sketch a method to modify γ_{SA} to guarantee R4 and yet maintain R3. The method explores the structural differences between the constraint graphs of C_{AS} and C_A.

Example 4: Assume that the application ontology is $\boldsymbol{LPC2} = (V_{LPC2}, C_{LPC2})$, where:

(1) V_{LPC2} = {`MobilePhone`, `Number`, `MobileCall`, `mobPlacedBy`}

and C_{LPC2} contains the following constraints (using the prefix "`a:`" to stress that the symbols are from V_{LPC2}, the application ontology vocabulary):

(2) (≥ 1 a:mobPlacedBy) ⊑ a:MobileCall
(3) (≥ 1 a:mobPlacedBy$^-$) ⊑ a:MobilePhone
(4) a:MobilePhone ⊑ ¬(≥ 2 a:number)

Suppose that the source ontology is $\boldsymbol{RPC} = (V_{RPC}, C_{RPC})$, where:

(5) V_{RPC} = {`Fixed`, `Nextel`, `id`, `PhoneCall`, `byFixed`, `byNextel`}

and C_{RPC} contains the following constraints (using the prefix "s:" to stress that the symbols are from V_{RPC}, the source ontology vocabulary):

(6) $(\geq 1\ \texttt{s:byFixed}) \sqsubseteq \texttt{s:PhoneCall}$
(7) $(\geq 1\ \texttt{s:byNextel}) \sqsubseteq \texttt{s:PhoneCall}$
(8) $(\geq 1\ \texttt{s:byFixed}^{-}) \sqsubseteq \texttt{s:Fixed}$
(9) $(\geq 1\ \texttt{s:byNextel}^{-}) \sqsubseteq \texttt{s:Nextel}$

Suppose also that the source-to-application mapping γ_{SA} contains the definitions:

(10) $\texttt{a:MobilePhone} \equiv \texttt{s:Nextel}$
(11) $\texttt{a:number} \equiv \texttt{s:id}$
(12) $\texttt{a:MobileCall} \equiv \texttt{s:PhoneCall} \sqcap (\geq 1\ \texttt{s:byNextel})$
(13) $\texttt{a:mobPlacedBy} \equiv \texttt{s:byNextel}$

Then, the set of endogenous constraints, $C_{LPC2, RPC}$, contains just two constraints:

(14) $(\geq 1\ \texttt{a:mobPlacedBy}) \sqsubseteq \texttt{a:MobileCall}$
(15) $(\geq 1\ \texttt{a:mobPlacedBy}^{-}) \sqsubseteq \texttt{a:MobilePhone}$

Note that (14) follows from (13), (12) and (7), and (15) follows from (13), (10) and (9). We then modify the definitions in (10) to (13) by propagating backwards the expressions that label the nodes of the constraint graph $G(C_{LPC2})$. The propagation only follows arcs in $G(C_{LPC2})$ that have no counterpart in $G(C_{LPC2, RPC})$. The modified source-to-application mapping will contain the following definitions:

(16) $\texttt{a:MobilePhone} \equiv \texttt{s:Nextel} \sqcap \neg(\geq 2\ \texttt{s:id})$
(17) $\texttt{a:number} \equiv \texttt{s:id}$
(18) $\texttt{a:MobileCall} \equiv \texttt{s:PhoneCall} \sqcap (\geq 1\ \texttt{s:byNextel})$
(19) $\texttt{a:mobPlacedBy} \equiv \texttt{s:byNextel}$

Definition (16) follows from (10). Indeed, $G(C_{LPC2})$ has an arc connecting nodes respectively labeled with a:MobilePhone and $\neg(\geq 2\ \texttt{a:number})$ (derived from (4)), which is not in $G(C_{LPC2, RPC})$. Hence, $\neg(\geq 2\ \texttt{a:number})$ has to be propagated to the definition of a:MobilePhone, as indicated in (16), with a:number replaced by its definition s:id, as indicated in (17). Under this new mapping, one may verify that R4 now holds and R3 still holds. Consequently, R2 is met.□

5 Conclusions

In this paper, we introduced automatic methods for constructing application ontology constraints and for adjusting ontology mappings, when the application ontology is an open or a closed fragment of the domain ontology. The final set of constraints and the mappings will have useful properties, as detailed in Section 3.1 and 3.2. The methods assume that the ontologies are written in an expressive family of attributive languages and depend on a procedure to test logical implication, based on constraint graphs.

The results in the paper are directly mapped to the RDF context and cover a topic – the role that constraints play in Linked Data – that is much neglected in the literature. They also contribute to the investigation of triplification strategies for relational databases that properly consider relational constraints, such as keys and foreign keys.

As for current work, we are modifying a triplification tool [14] to generate application ontology constraints, as described in the paper. We are also extending the strategy for generating endogenous constraints, described in Section 4, to account for more complex source-to-ontology mappings, using results from [10].

Acknowledgements

This work was partly supported by CNPq, under grants 473110/2008-3 and 557128/2009-9, by FAPERJ under grant E-26/170028/2008, and by CAPES under grant NF 21/2009.

References

[1] 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, Cambridge, UK (2003), pp. 43–95.

[2] Berrueta, D., Phipps, J. Best Practice Recipes for Publishing RDF Vocabularies - W3C Working Group Note. http://www.w3.org/TR/swbp-vocab-pub/ (Accessed June 14, 2009).

[3] Bizer, C., Cyganiak, R., Heath, T. How to publish Linked Data on the Web. http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/ (Accessed June 14, 2009).

[4] Bizer, C., Heath,T., Berners-Lee,T. Linked Data - The Story So Far. Int. Journal on Semantic Web and Information Systems, 5(3), pp. 1-22, 2009.

[5] 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.

[6] Das, S., Sundara, S., Cyganiak, R. R2rml: Rdb to rdf mapping language. W3C RDB2RDF working group. http://www.w3.org/TR/r2rml/ (accessed Dec. 15, 2010).

[7] Halpin, H., Hayes, P. J. When owl:sameAs isn't the same: An analysis of identity links on the semantic web. In Proc. Int'l. Workshop on Linked Data on the Web (2010).

[8] Jaffri, A., Glaser, H., Millard, I. URI disambiguation in the context of linked data. In Proc. of the 1st Int'l. Workshop on Linked Data on the Web (2008).

[9] Lenzerini, M.: "Data Integration: A Theoretical Perspective". In: Proc. of ACM Symposium on Principles of Database Systems (2002).

[10] Lauschner, T., Casanova, M. A., Vidal, V. M. P., Macedo, J. A. F. Efficient Decision Procedures for Query Containment and Related Problems. In: Proc. XXIV Brazilian Symposium on Databases (2009).

[11] Lutz, M. Ontology-based Discovery and Composition of Geographic Information Services. PhD Thesis, Institut für Geoinformatik (2006).

[12] McCusker, J., McGuinness, D. L. owl:sameAs considered harmful to provenance. In Proc. ISCB Conference on Semantics in Healthcare and Life Sciences (2010).

[13] Sacramento, E., Vidal, V. M. P., Macedo, J. A. F., Lóscio, B. F., Lopes, F. L. R., Casanova, M. A. Towards Automatic Generation of Application Ontologies. Journal of Information and Data Management, v.1, p.535 - 550, 2010.

[14] Salas, P. E., Breitman, K. K., Viterbo, J., Casanova, M. A. Interoperability by Design Using the Std-Trip Tool: an a priori approach In: Proc. 6th Int'l. Conf. on Semantic Systems (I-SEMANTICS 2010), 2010, Graz.

[15] Vidal, V.M.P., Sacramento, E. R., Macedo, J. A. F., Casanova, M. A. An Ontology-Based Framework for Geographic Data Integration. In: Proc. 3rd Int'l. Workshop on Semantic and Conceptual Issues in GIS (SeCoGIS 2009). Springer, 2009.
[16] Villegas, A., Olivè, A. A method for filtering large conceptual schemas. In Proc.29th Int'l. Conf. on Conceptual modeling (ER'10), Springer-Verlag, Berlin, Heidelberg, pp. 247-260.
[17] W3C. Semantic Web Development Tools. http://www.w3.org/2001/sw/wiki/Tools (Accessed Abril 2nd, 2011).