Web Semantics: Science, Services and Agents on the World Wide Web 39 (2016) 62–80

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Defining and computing Least Common Subsumers in RDF☆

S. Colucci [a,*], F.M. Donini [b], S. Giannini [a], E. Di Sciascio [a]

[a] Politecnico di Bari, Via Orabona 4, 70125 Bari, Italy

[b] Università della Tuscia, Via S. Carlo 32, 01100 Viterbo, Italy

ARTICLE INFO

Article history:
Received 9 November 2015
Received in revised form
25 January 2016
Accepted 29 February 2016
Available online 11 March 2016

Keywords:
Least Common Subsumer
RDF
Rooted-graph
Selection of RDF triples
RDF simple-entailment

ABSTRACT

Several application scenarios in the Web of Data share the need to identify the commonalities between a pair of **RDF** resources. Motivated by such needs, we propose the definition and the computation of Least Common Subsumers (LCSs) in **RDF**. To this aim, we provide some original and fundamental reformulations, to deal with the peculiarities of **RDF**. First, we adapt a few definitions from Graph Theory to paths and connectedness in **RDF**-graphs. Second, we define *rooted* **RDF**-graphs (*r*-graphs), in order to focus on a particular resource inside an **RDF**-graph. Third, we change the definitions of LCSs originally set up for Description Logics to *r*-graphs. According to the above reformulations, we investigate the computational properties of LCS in **RDF**, and find a polynomial-time characterization using a form of graph composition. This result remarkably distinguishes LCSs from Entailment in **RDF**, which is an NP-complete graph matching problem. We then devise algorithms for computing an LCS. A prototypical implementation works as a proof-of-concept for the whole approach in three application scenarios, and shows usefulness and feasibility of our proposal. Most of our examples are taken directly from real datasets, and are fully replicable thanks to the fact that the choice about which triples are selected for the computation is made explicit and flexible.

1. Introduction

A *Common Subsumer* (CS) of two formulas ϕ_1, ϕ_2 is another formula ϕ_3 that is logically entailed by both ϕ_1 and ϕ_2. A *Least* Common Subsumer (LCS) is, intuitively, a most specific CS of ϕ_1, ϕ_2—for logics in which the LCS is unique, it entails every other CS of ϕ_1, ϕ_2. Intuitively, a CS expresses some *logical commonalities* of ϕ_1, ϕ_2, and an LCS expresses all of them.

LCSs have been introduced in 1992 [1] in Description Logics (DLs) [2] and adopted in disparate application domains, including inductive learning, bottom-up construction of knowledge bases, information retrieval, among others. All such domains include tasks which share the need to infer and/or compute commonalities between domain items, that may be formally modeled as concepts in DLs knowledge bases.

Apparently, no work has been done so far to support the adoption of such a reasoning service in what is known to be the biggest available and addressable knowledge base: the Web of Data [3]. Although combining different datasets, the Web of Data may in fact be considered as a unique data source in which resources are all modeled in one language, **RDF** [4], whose semantics allows for logical reasoning over them. To the best of our knowledge, the definition and computation of LCSs in **RDF** has never been addressed in the literature, even though the search for commonalities between pairs of **RDF** resources turns out to be useful in several tasks, and has been studied only with statistical or algebraic tools. Hence, the development of a Knowledge Representation service computing an LCS (or even a CS) of two resources in **RDF** could be a useful tool for building Semantic Web applications.

In spite of the apparently low expressiveness of **RDF**, the language has some special features which make it not comparable to any DL. For this reason, developing an idea presented in a preliminary work [5], we propose a novel definition of LCSs, which is able to manage **RDF** assertions and extract the informative content hidden in resource descriptions. Although we focus on Simple Entailment [4], our definition of LCS can be used also for stronger entailment regimes adopted for interpretation of **RDF** assertions. Moreover, in our implementation, we show that also (non-least) Common Subsumers can be useful in several applications. In order to manage the peculiarities of LCSs in **RDF**, we make the original contributions below:

☆ This work is an almost completely rewritten version of Colucci et al. (2013) and Colucci et al. (2014). Most results are unpublished.

* Corresponding author.
E-mail addresses: simona.colucci@poliba.it (S. Colucci), donini@unitus.it (F.M. Donini), silvia.giannini@poliba.it (S. Giannini), eugenio.disciascio@poliba.it (E. Di Sciascio).

http://dx.doi.org/10.1016/j.websem.2016.02.001
1570-8268/

(1) we adapt definitions about paths and connectedness from Graph Theory to **RDF**-graphs, taking into account the fact that labels of nodes and arcs can be mixed in (what we call) **RDF**-paths
(2) we define *rooted* **RDF**-graphs (r-graphs) $\langle r, T_r \rangle$, that focus on a particular resource r inside an **RDF**-graph, and on a set of chosen triples T_r describing r in the Web of Data
(3) we define entailment between r-graphs, that is entailment in which roots must be mapped to roots
(4) we change the LCS definitions (originally set up for Description Logics) to r-graphs.

The redefinition of LCSs described above led us to a remarkable result in terms of complexity, which constitutes a main contribution of the article. In particular, we found a polynomial-time characterization of LCS in **RDF**, based on the computational properties we investigate. Such a characterization uses a form of graph composition and distinguishes the computation of an LCS from Entailment in **RDF**, which is known to be an NP-complete graph matching problem. According to this novel polynomial-time characterization, we provide algorithms to compute an LCS.

In order to support our claims and motivate our work, we identified three application scenarios in which LCSs may significantly improve information extraction in the Web of Data: (i) entity disambiguation/linking in the Semantic Web, (ii) automated clustering of collections of **RDF** resources, and (iii) automated extraction of features shared among drugs. In all the three scenarios, we show how the computation of a CS not only provides a semantic approach to similarity between two **RDF** resources, but also a description of their shared features—in other words, the CS does not only show *if* or *how much* two resources are similar, but also *why* they are. In this respect, our work makes applicable to **RDF** several similarity measures based on LCSs often called *semantic similarity* [6–8].

The article is organized as follows: in the next section, we give some background to make the paper self-contained. Least Common Subsumers are defined in Section 3 and their theoretical properties are investigated in Section 4. Their computation is addressed in Section 5 and exemplified with reference to the three application scenarios in Section 6, which also evaluates performance. The literature related to our work is analyzed in Section 7, before concluding the paper.

2. Background and notation

We denote by U the set of all URIs, by B the set of all blank nodes, and by L the set of all possible literals, which can be either plain or typed literals. An ***RDF**-graph* is a subset of the set of all possible triples in $(U \cup B) \times (U) \times (U \cup B \cup L)$ while a *generalized **RDF**-graph* – according to ter Horst [9] – adds B to the middle term of the above cartesian product—*i.e.*, generalized **RDF**-graphs allow a blank node to appear also in the predicate position of a triple.[1] When a graph G contains no blank nodes, we say that G is *ground*.

To correctly embed **RDF** triples in English text we use the notation $\ll a\ p\ b \gg$ to mean what in **RDF** documents is written as "*a p b* **.**" (with the full stop at the end of the triple).

Definition 1. Given a generalized **RDF**-graph G, we denote by $terms(G)$ the *terms* of G, *i.e.*, the subset of $U \cup B \cup L$ occurring in any position in any triple of G. Moreover, we denote by $bl(G) \subset B$ the set of blank nodes occurring in (some triple of) G.

[1] For our setting, we do not need to consider the even larger definition of generalized **RDF**-graphs in which, *e.g.*, literals in the subject or predicate position are allowed.

Recall that each blank node corresponds to an existentially quantified variable, where the scope of the quantification is the document the blank node occurs in. A blank node without a name is denoted by `[]`, while named blank nodes are prefixed by "`_:`", *e.g.*, `_:xxx`.

Definition 2 (*[9, 2.4]*)**.** Given a generalized **RDF**-graph G and a partial function $h : B \to B \cup U \cup L$, we denote by G_h the *instance* of G, obtained by replacing in every triple of G, every blank node b in the domain of h with $h(b)$.

The semantics of **RDF** is given in terms of set-theoretic interpretations I. Given two partially overlapping sets of *resources* R_I and *properties* P_I, terms in U are mapped by an interpretation function I into $R_I \cup P_I$. Intuitively, terms that occur in triples only as subjects or objects can be mapped into R_I, terms that occur only as properties can be mapped into P_I, while terms that appear *both* as properties and as subjects/objects must be mapped into $R_I \cap P_I$—which explains why R_I and P_I must overlap. Then, another interpretation function – call it I_{ext} – maps elements in P_I to subsets of $R_I \times R_I$, that is, pairs of resources. Plain literals are interpreted as themselves, while typed literals are interpreted as (a special kind of) resources. An interpretation I *satisfies* a ground triple $\ll s\ p\ o \gg$ when the pair $(I(s), I(o)) \in I_{ext}(I(p))$, and I satisfies a ground graph G when it satisfies all triples in G. Blank nodes are interpreted by another function $A : B \to U \cup L$, and we can extend it to non-ground graphs saying that $A(G)$ is G where every blank node b has been replaced by $A(b)$ in every triple. Finally, an interpretation I satisfies a non-ground graph G if there exist a function A such that I satisfies the ground graph $A(G)$.

We remark that elements interpreted in $R_I \cap P_I$ make **RDF** an essentially untyped logic. To make **RDF** well-typed, several researchers adopted *punning* [10], in which the different occurrences of a resource (as an individual, a class, or a predicate) are treated as different symbols—that is, $R_I \cap P_I = \emptyset$. However, we do not want to assume punning, for reasons that are clarified in Section 3.1, and embrace the original **RDF** semantics.

The weakest form of entailment between two **RDF**-graphs G and H is Simple Entailment, denoted by $G \models_S H$. Every statement which holds for Simple Entailment holds also for stronger entailment relations, such as **RDF**-Entailment, and **RDF-S**-Entailment. In this paper, we concentrate only on Simple Entailment, which can be characterized in three equivalent ways.

The first definition of Simple Entailment states that $G \models_S H$ if every interpretation satisfying G satisfies also H. Simple Entailment is a transitive relation, *i.e.*, $G_1 \models_S G_2$ and $G_2 \models_S G_3$ implies $G_1 \models_S G_3$.

The second characterization of Simple Entailment says that $G \models_S H$ if and only if there exists a subgraph $G' \subseteq G$ such that G' is an *instance* of H—*i.e.*, there exists a partial mapping $h : B \to B \cup U \cup L$ such that $G' = H_h$. The Interpolation Lemma [11] proves that this characterization of Simple Entailment is indeed equivalent to the one based on interpretations of **RDF**-graphs. The Interpolation Lemma has been extended to generalized **RDF**-graphs too [9, Prop. 2.12]. In the proofs of Sections 4 and 5, we use this characterization of Simple Entailment.

For sake of completeness, we mention that $G \models_S H$ if and only if there is a way of applying Rules **se1** and **se2** shown in Fig. 1 to triples in G such that the resulting graph G' contains H as a subgraph. Hence, this statement can be taken as a third, equivalent, characterization of Simple Entailment.

We stress the fact that Simple Entailment is the most general entailment relation in **RDF**, which means that $G \models_S H$ always implies $G \models_{\mathbf{RDF}} H$, which in turn implies $G \models_{\mathbf{RDF\text{-}S}} H$—where the subscript refers to the entailment regime. Hence in the sections about properties of Least Common Subsumers, we discuss also how to extend proofs based on Simple Entailment to stronger entailment relations.