Faceted operations on composed RDF ontologies

Anastasia Analyti*

Institute of Computer Science,
FORTH-ICS, Greece
Email: analyti@ics.forth.gr
*Corresponding author

Ioannis Pachoulakis

Department of Informatics Engineering,
TEI of Crete, Greece
Email: ip@ie.teicrete.gr

Abstract: Public and private organisations increasingly release their data. The use of these open data can be supported and stimulated by providing considerable metadata, including discovery, contextual and other detailed metadata. This paper argues that faceted indexing of the RDF ontologies, published as linked open data (LOD), is an important part of their associated metadata. On the other hand, information needs to be composed from multiple sources to create new information. Through primitive faceted formulas using faceted indexing terms and a list of validity time intervals, we are able to take the union of all RDF ontologies indexed by these terms and are valid in all provided validity time intervals. The resulted RDF ontology is indexed by the provided index terms and has a list of validity time intervals derived by the validity time intervals of the unioned RDF ontologies. By combining both union and intersection operators on primitive faceted formulas, complex expressions can be formed that return a composed RDF ontology, its faceted indexing and validity time intervals. All algorithms are provided along with their complexity.

Keywords: faceted indexing of RDF ontologies; composing RDF ontologies; algorithms; complexity; web of data, web engineering.

Reference to this paper should be made as follows: Analyti, A. and Pachoulakis, I. (2016) 'Faceted operations on composed RDF ontologies', *Int. J. Web Engineering and Technology*, Vol. 11, No. 2, pp.174–194.

Biographical notes: Anastasia Analyti received her BSc in Mathematics from University of Athens, Greece and MSc and PhD in Computer Science from Michigan State University, USA. She worked as a Visiting Professor at the Department of Computer Science, University of Crete, and at the Department of Electronic and Computer Engineering, Technical University of Crete. Since 1995, she has been a principal researcher at the Information Systems Laboratory of the Institute of Computer Science, Foundation for Research and Technology – Hellas (FORTH-ICS). Her current interests include reasoning

on the semantic web, modular web rule bases, non-monotonic-reasoning, faceted metadata and semantics, conceptual modelling, contextual organisation of information, and information integration and retrieval systems for the web.

Ioannis Pachoulakis received his BSc in Physics from the University of Crete, Greece in 1988, and PhD in Astrophysics in 1996 and Masters of Science in Engineering in 1998, both from the University of Pennsylvania in the USA. In the following two years, he worked on client-server software for transportation companies that use multiple modes (trucks, trains) to move merchandise in time with minimum possible cost. From 2000 to 2003, he undertook several administrative and development tasks for a number of European and national projects at the Foundation for Research and Technology-Hellas in Heraklion, Crete. Since 2001, he has been an Assistant Professor at the Department of Informatics Engineering at TEI of Crete and Assistant Researcher at the Centre for Technological Development of Crete with mainstream interests in multimedia applications for science.

1 Introduction

During the recent years an increasing number of data providers adopted a set of best practices for publishing and connecting RDF ontologies (Cyganiak et al., 2014) on the web, leading to the creation of distributed dataspaces in the web of data (Heath and Bizer, 2011). Each RDF ontology is possibly associated with metadata information [like using the PROV-O ontology (Lebo et al., 2013), currently a W3C recommendation], expressing, for example, its quality, certainty, authority, temporal and spatial status. For the potential of metadata for linked open data (LOD) and its value to users and publishers, see Zuiderwijk et al. (2012). However, PROV-O does not allow for faceted indexing of RDF ontologies (that is it does not indexes RDF ontologies according to terms of a faceted taxonomy). Faceted indexing of research data collections is not a new idea. B2FIND (http://eudat.eu/services/b2find) is a simple, user-friendly metadata catalogue of research data collections stored in EUDAT (http://eudat.eu/what-eudat) data centres and other repositories. F2FIND supports faceted, geospatial and temporal metadata searches. CKAN (http://ckan.org/) is a powerful data management system that makes data accessible – by providing tools to streamline publishing, sharing, finding and using data. CKAN is aimed at data publishers (national and regional governments, companies and organisations) wanting to make their data open and available. It provides faceted search – drill-down via facets. Ability to consecutively narrow the search by further facets values allows the users to limit their search after they see the search results. For a survey of faceted search, see Wei et al. (201).

In this paper, we propose composing a specific set of faceted indexed RDF ontologies, through intersection and union, and automatically identify their new faceted values. All considered RDF ontologies are indexed by a set of topics and a list of validity intervals and possibly other relevant faceted values. The specific problem is how do you identify the RDF ontologies that are going to be composed based on a set of terms from a faceted taxonomy and a set of validity time intervals. Additionaly, how do you define the union and intersection of RDF ontologies. Further, how do you derive

the new faceted terms of the composed ontology. We illustrate our concepts through examples. We would like to note that there is no work in the literature that handles this problem. Some related works are reviewed in Section 5.

Consider that *topic* and *location* and *facilities* are the names of facets (see Figure 1). A *primitive faceted formula* is either

1 the name nam of a faceted indexed RDF ontology O

2 an expression like $\{(\mathit{topic}, \mathit{hotels}), (\mathit{location}, \mathit{Cephalonia}), (\mathit{facilities}, \mathit{spa}), \mathit{val\_int} = \langle[13-07-2016, 19-07-2016]\rangle\}$[1].

In case (1) the RDF ontology O will be returned together with its faceted indexing and list of validity time intervals. In case (2), the union of all considered RDF ontologies on hotels, located in Cephalonia, and offer spa for the time interval [13-07-2016, 19-07-2016] will be computed. The faceted values for the computed unioned RDF ontology will be $\{(\mathit{topic}, \mathit{hotels}), (\mathit{location}, \mathit{Cephalonia}), (\mathit{facilities}, \mathit{spa})\}$, while its list of validity time intervals will correspond to the intersection of the lists of validity time intervals of the unioned RDF ontologies. If we are interested in hotels in Cephalonia that offer not only spa but also swimming pool for the same time period, we form the expression:

$$\begin{array}{l} \{(\mathit{topic}, \mathit{hotels}), (\mathit{location}, \mathit{Cephalonia}), \\ (\mathit{facilities}, \mathit{spa}), \mathit{val\_int} = \langle[13-07-2016, 19-07-2016]\rangle\} \\ \sqcap \\ \{(\mathit{topic}, \mathit{hotels}), (\mathit{location}, \mathit{Cephalonia}), \\ (\mathit{facilities}, \mathit{swimming\ pool}), \mathit{val\_int} = \langle[13-07-2016, 19-07-2016]\rangle\} \end{array}$$

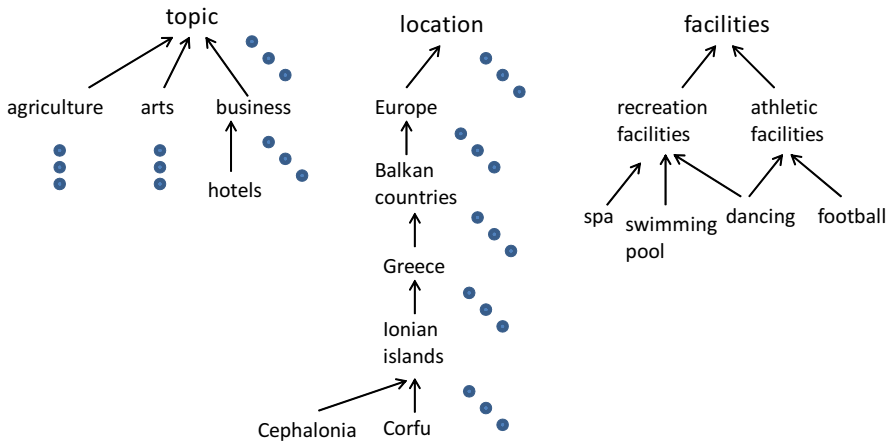
This will compute the intersection of the RDFS 1.1 closures (Hayes and Patel-Schneider, 2014) of the RDF ontologies computed by the intersected primitive faceted expressions. On the other hand the faceted indexing of the computed intersection will be $\{(\mathit{topic}, \mathit{hotels}), (\mathit{location}, \ \mathit{Cephalonia}), (\mathit{facilities}, \mathit{spa}), (\mathit{facilities}, \mathit{swimming\ pool})\}$, while the corresponding list of validity intervals will be the intersection of the lists of validity time intervals corresponding to the intersected primitive faceted expressions.

Assume that we are interested in hotels in Cephalonia or Corfu for the time interval [13-07-2016, 19-07-2016]. Then, we can form the expression:

$$\begin{array}{l} \{(\mathit{topic}, \mathit{hotels}), (\mathit{location}, \mathit{Cephalonia}), \\ \mathit{val\_int} = \langle[13-07-2016, 19-07-2016]\rangle\} \\ \sqcup \\ \{(\mathit{topic}, \mathit{hotels}), (\mathit{location}, \mathit{Corfu}), \\ \mathit{val\_int} = \langle[13-07-2016, 19-07-2016]\rangle\} \end{array}$$

This will compute the union of the RDF ontologies computed by the unioned primitive faceted expressions. On the other hand the faceted indexing of the computed union will be $\{(\mathit{topic}, \mathit{hotels}), (\mathit{location}, \mathit{Ionian\ islands})\}$, while the corresponding list of validity time intervals will the intersection of the lists of validity time intervals corresponding to the unioned primitive faceted expressions. Note that *Ionian islands* is the most specific term in the facet named *location* that subsumes both the terms *Cephalonia* and *Corfu* (see Figure 1).

Figure 1 An example faceted taxonomy (see online version for colours)



By combining both union and intersection operators on primitive faceted formulas, complex expressions can be formed that return a computed RDF ontology, its faceted indexing and list of validity time intervals. All algorithms are provided along with their complexity.

The rest of the paper is organised as folows: in Section 2, we define faceted taxonomies, named RDF ontologies, their faceted annotation and list of validity intervals, and faceted formulas. In Section 3, we define evaluation of faceted formulas leading to composed RDF ontologies. In Section 4, we formally define the faceted indexing of faceted formulas as well as their associated lists of validity time intervals. Formal definitions lead to Algorithms, where detailed description, associated examples, and complexity are provided. In Section 5, we review related work. Finally, in Section 6, we conclude the paper and provide directions for further research.

2 Faceted taxonomies and faceted formulas

In this Section, we define faceted taxonomies, named RDF ontologies, their faceted annotation and list of validity intervals, and faceted formulas.

Definition 1 (facet): A *facet* f is a triple $f = (nam_f, Terms_f, \leq)$ such that nam_f is the name of f, $Terms_f$ is the set of terms of f, and $\leq$ is a reflexive, transitive, and antisymmetric relation over $Term_f$, called *subsumption*. There is a term $nam_f \in Terms_f$ that subsumes the rest of the terms.

A *faceted taxonomy* is a set of facets (Wei et al., 201). An example faceted taxonomy is provided in Figure 1. In this paper, we consider a fixed faceted taxonomy FT that includes a facet with name *topic*. The set of names of the facets in FT is denoted by $names(FT)$.

A pair (nam_f, t), where f is a facet in FT and $t \in Terms_f$, is called *faceted indexing term*.

Definition 2 (faceted annotation): A *faceted annotation* fa is a set of faceted indexing terms.

For the faceted taxonomy of Figure 1, the set $\{(topic, hotels), (location, Cephalonia), (facilities, spa)\}$ is a faceted annotation.

The set of all possible annotations formed by the faceted taxonomy FT and include at least a pair (*topic,t*) is denoted by FA.

Below, we define a list of validity time intervals, the intersection of two lists of validity time intervals and their merging.

Definition 3 (list of validity time intervals): A *list of validity time intervals* TL is a list of sorted date intervals that do not overlap.

Definition 4 (the intersection of two lists of validity time intervals): The *intersection* of two lists of validity time intervals TL_1 and TL_2 is a list of validity time intervals that contains the intersection of the date intervals in TL_1 with the date intervals in TL_2.

Let $LT_1 = \langle[25-09-2015, 29-09-2015], [15-09-2016, 20-09-2016]\rangle$ and $LT_2 = \langle[27-09-2015, 30-09-2015], [13-09-2016, 30-09-2016]\rangle$. The intersection of LT_1 and LT_2 is $\langle[27-09-2015, 29-09-2015], [15-09-2016, 20-09-2016]\rangle$.

Definition 5 (the merging of two lists of validity time intervals): The *merging* of two lists of validity time intervals TL_1 and TL_2 is a list of validity time intervals that contains the merging of the date intervals in TL_1 with the date intervals in TL_2.

Let $LT_1 = \langle[15-09-2015, 19-09-2015], [17-09-2016, 30-09-2016]\rangle$ and $LT_2 = \langle[17-09-2015, 30-09-2015], [13-09-2016, 30-09-2016]\rangle$. The merging of LT_1 and LT_2 is $\langle[15-09-2015, 30-09-2015], [13-09-2016, 30-09-2016]\rangle$.

We denote the set of lists of validity time intervals as VL. If a validity time interval ti includes a validity time interval ti' then we write $ti' \leq ti$.

A *name* is an IRI reference or a literal (Cyganiak et al., 2014; Hayes and Patel-Schneider, 2014). A (web) *vocabulary* V is a set of names. We denote the set of names by *Nam*, the set of all IRI references by *IRI*, and the set of all literals by $\mathcal{LIT}$. We consider a set BN of blank nodes, such that the sets BN, *IRI*, $\mathcal{LIT}$ are pairwise disjoint. The vocabulary of RDF, $\mathcal{V}_{RDF}$, is a set of IRI references in the *rdf*: namespace (Hayes and Patel-Schneider, 2014), and the vocabulary of RDFS, $\mathcal{V}_{RDFS}$, is a set of IRI references in the *rdfs*: namespace (Hayes and Patel-Schneider, 2014). Let $D = \{rdf{:}langString, xsd{:}string\}$, which is the set of datatypes supported by the RDF 1.1 semantics (Hayes and Patel-Schneider, 2014).

Definition 6 (RDF triple): An RDF triple G has the form (s, p, o), where $p \in$ *IRI*, and $s, o \in$ *Nam* $\cup BN$. The symbol s is called the *subject* of the triple, the symbol p is called the *property* of the triple, and the symbol o is called the *object* of the triple. □

For example (*ex*:*Peter*, `rdf:type`, *ex*:*Person*)[2] is an RDF triple.

Our choice of allowing literals appearing in the subject position is based on our intuition that this case can naturally appear in knowledge representation. SPARQL

(Harris and Seaborne, 2013) and de Bruijn et al. (2005) also consider literals in the subject position of RDF triples. Note that the RDFS model theory (Cyganiak et al., 2014; Hayes and Patel-Schneider, 2014) does not allow literals to appear in the subject position of an RDF triple.

An RDF ontology is a set of RDF triples. Based on the notion of a RDF ontology, we define a named RDF ontology.

Definition 7 (named RDF ontology): A *named RDF ontology* is a pair $no = (nam_{no}, O_{no})$, where nam_{no} is the name of the RDF ontology O_{no}. □

Named graphs have been proposed by Carroll et al. (2005b,a) for provenance.

We also consider a fixed set of named RDF ontologies NO that do not include blank nodes. Though this is a severe limitation at the end of the concluding section we propose an approach that alleviates this problem. The set of names of the named RDF ontologies in NO is denoted by $names(NO)$.

Below we define the faceted annotation and list of validity time intervals of a named RDF ontology.

Definition 8 (faceted annotation of a named RDF ontology): The *faceted annotation of named RDF ontologies* in NO is a total function $noFA : NO \rightarrow FA$.

For example, consider a named RDF ontology no for which it holds $noFA(no) = \{(topic, hotels), (location, Cephalonia), (facilities, spa)\}$.

Definition 9 (the list of validity time intervals of a named RDF ontology): The *list of validity time intervals of named RDF ontologies* in NO is a total function $noTL : NO \rightarrow VL$.

For example, consider a named RDF ontology no for which it holds $noTL(no) = \langle[06-08-2015, 30-08-2015], [10-07-2016, 30-07-2016]\rangle$.

Below we define faceted formulas which will be used through their evaluation (see next section) for forming composed RDF ontologies.

Definition 10 (faceted formula): We define a faceted formula *ff* as follows:

- if *ff*
  1. has the form $\{(f_1, v_1), ...(f_n, v_n), val\_int = TL\}$, where $\{(f_1, v_1), ...(f_n, v_n)\}$ is a faceted annotation in FA and TL a list of validity time intervals
  2. is a name in $names(NO)$ then *ff* is called a *primitive faceted formula*
- if *ff* is an expression of the form $F_1 \sqcup ... \sqcup F_n$, where F_i is a faceted formula then *ff* is a faceted formula
- if *ff* is an expression of the form $F_1 \sqcap ... \sqcap F_n$, where F_i is a faceted formula then *ff* is a faceted formula.

Examples of faceted formulas are provided in the introductory section.

3 Evaluation of faceted formulas

Before we define evaluation of faceted formulas, we need to define the closure of an RDF ontology that provides its semantics (Hayes and Patel-Schneider, 2014). Evaluation of faceted formulas provides composition of RDF ontologies.

Let O be an RDF ontology. We define by n_O to be the maximum of 1 and the largest i such that *rdf*:_i appears in O. Recall that the *rdf*:_i properties are used in RDFS (Hayes and Patel-Schneider, 2014) to express members of containers (i.e., bags, sequences, and alternatives), which are in practice finitely limited. We define $\mathcal{V}_{RDF}^{\#n} = \mathcal{V}_{RDF} - \{rdf{:}\_i \mid i > n\}$.

We now define the closure of an RDF ontology O, denoted by *closure*(O) based on a 3-ary predicate H. We define as P_O the following definite logic set of rules.

RDF Interpretation Rules

(1) $H(?z, \texttt{type}, \texttt{Property}) \leftarrow H(?x, ?z, ?y).$
(2) $H(?x, \texttt{type}, \texttt{Resource}) \leftarrow H(?x, ?z, ?y).$
(3) $H(?y, \texttt{type}, \texttt{Resource}) \leftarrow H(?x, ?z, ?y).$

RDFS Interpretation Rules

(1) $H(?z, \texttt{type}, ?y) \leftarrow H(?x, \texttt{domain}, ?y),\ H(?z, ?x, ?w).$
(2) $H(?w, \texttt{type}, ?y) \leftarrow H(?x, \texttt{range}, ?y),\ H(?z, ?x, ?w).$
(3) $H(?x, \texttt{subClassOf}, \texttt{Resource}) \leftarrow H(?x, \texttt{type}, \texttt{Class}).$

(4) $H(?z, \texttt{type}, ?y) \leftarrow H(?x, \texttt{subClassOf}, ?y),\ H(?z, \texttt{type}, ?x).$
(6) $H(?x, \texttt{subClassOf}, ?x) \leftarrow H(?x, \texttt{type}, \texttt{Class}).$
(7) $H(?z, \texttt{subClassOf}, ?y) \leftarrow H(?x, \texttt{subClassOf}, ?y),\ H(?z, \texttt{subClassOf}, ?x).$

(8) $H(?z_1, ?y, ?z_2) \leftarrow H(?x, \texttt{subPropertyOf}, ?y),\ H(?z_1, ?x, ?z_2).$

(9) $H(?x, \texttt{subPropertyOf}, ?x) \leftarrow H(?x, \texttt{type}, \texttt{Property}).$
(10) $H(?z, \texttt{subPropertyOf}, ?y) \leftarrow H(?x, \texttt{subPropertyOf}, ?y),\ H(?z, \texttt{subPropertyOf}, ?x).$

(11) $H(?x, \texttt{subClassOf}, \texttt{Literal}) \leftarrow H(?x, \texttt{type}, \texttt{Datatype}).$
(12) $H(?x, \texttt{subPropertyOf}, \texttt{member}) \leftarrow H(?x, \texttt{type}, \texttt{ContainerMembershipProperty}).$

For $d \in D$:
(13) $H(d, \texttt{type}, \texttt{Datatype}) \leftarrow \texttt{true}.$

For each ‘‘s”^^d appearing in O, where $d \in D$:
(14) $H(\text{‘‘}s\text{”^^}d, \texttt{type}, d) \leftarrow \texttt{true}.$

For each E appearing in O, where E is not a language-tagged string:
(15) $\texttt{false} \leftarrow H(E, \texttt{type}, \texttt{rdf:langString}).$

For each s appearing in O, where s is not a well-typed string:
(16) $\texttt{false} \leftarrow H(s, \texttt{type}, \texttt{xsd:string})$.

For each RDF and RDFS axiomatic triple (s, p, o) s.t. $p, s, o \in V_{RDF}^{\#n_O} \cup \mathcal{V}_{RDFS}$:
(17) $H(s, p, o) \leftarrow \texttt{true}$.

RDF graph rules
For each $(s, p, o) \in O$:
(1) $H(s, p, o) \leftarrow \texttt{true}$.

According to RDFS 1.1. semantics (Hayes and Patel-Schneider, 2014), the semantics of an RDF ontology is $\{(s, p, o) \mid (s, p, o) \in T_{P_O}^{\uparrow\omega}(\emptyset)\}$, assuming that $\texttt{false} \notin T_{P_O}^{\uparrow\omega}(\emptyset)$.[3] In the case that $\texttt{false} \in T_{P_O}^{\uparrow\omega}(\emptyset)$, the RDF ontology O is considered inconsistent. Here, we consider only consistent RDF ontologies. Thus, we define $\mathit{closure}(O) = \{(s, p, o) \mid (s, p, o) \in T_{P_O}^{\uparrow\omega}(\emptyset)\}$.

We now define two operators $\sqcap$ and $\sqcup$ on RDF ontologies.

Definition 11: Let O_1 and O_2 be RDF ontologies. We define $O_1 \sqcap O_2 = \mathit{closure}(O_1) \cap \mathit{closure}(O_2)$.

Definition 12: Let O_1 and O_2 be RDF ontologies. We define $O_1 \sqcup O_2 = O_1 \cup O_2$.

Let O_1 and O_2 be RDF ontologies. It does not hold $closure(O_1 \sqcap O_2) = closure(O_1 \cap O_2)$. This is obvious from the following examples.

Example 1: Let $O_1 = \{(c_1, \texttt{subClassOf}, c_2), (c_2, \texttt{subClassOf}, c_3)\}$ and let $O_2 = \{(c_1, \texttt{subClassOf}, c_3)\}$ Now, $(c_1, \texttt{subClassOf}, c_3) \in closure(O_1 \sqcap O_2)$. However, $O_1 \cap O_2 = \emptyset$.

Example 2: Let $O_1 = \{(o, \texttt{type}, A), (A, \texttt{subClassOf}, B)\}$ and $O_2 = \{(o, \texttt{type}, B)\}$. It does not hold $closure(O_1 \sqcap O_2) = closure(O_1 \cap O_2)$, since $O_1 \cap O_2 = \emptyset$, while $(o, \texttt{type}, B) \in closure(O_1 \sqcap O_2)$.

However, it is easy to see that $closure(O_1 \cup O_2) = closure(closure(O_1) \cup closure(O_2))$. This is the reason that closures of ontologies are not considered in Definition 12.

Operations $\sqcap$ and $\sqcup$ are associative. The associativity of $\sqcup$ is obvious. We will also prove the associativity $\sqcap$. First we prove a Lemma.

Lemma 1: Let O_1, O_2 be RDF ontologies. It holds $closure(O_1 \sqcap O_2) = O_1 \sqcap O_2$.

Proof: We will show that $closure(O_1 \sqcap O_2) = O_1 \sqcap O_2$. Obviously, $O_1 \sqcap O_2 \subseteq closure(O_1 \sqcap O_2)$. We will show that $closure(O_1 \sqcap O_2) \subseteq O_1 \sqcap O_2$. We will prove this by induction. Let $O = O_1 \sqcap O_2$. Obviously, $\{(s, p, o) \mid H(p, s, o) \in T_{P_O}^{\uparrow 0}(\emptyset)\} \subseteq O_1 \sqcap O_2$.

Assumption: $\{(s, p, o) \mid H(s, p, o) \in T_{P_O}^{\uparrow n}(\emptyset)\} \subseteq O_1 \sqcap O_2$.

We will show that $\{(s, p, o) \mid H(s, p, o) \in T_{P_O}^{\uparrow n+1}(\emptyset)\} \subseteq O_1 \sqcap O_2$. The proof is obvious.

For example, let $(s_1, \texttt{type}, w) \in T_{P_O}^{\uparrow n+1}(\emptyset)$, according to RDFS interpretation rule (1). Then, it exists $(s_1, p_1, o_1) \in \{(s,p,o) \mid H(s,p,o) \in T_{P_O}^{\uparrow n}(\emptyset)\} \subseteq O_1 \sqcap O_2$ and $(p_1, \texttt{domain}, w) \in \{(s,p,o) \mid H(s,p,o) \in T_{P_O}^{\uparrow n}(\emptyset)\} \subseteq O_1 \sqcap O_2$.

Then, $(s_1, p_1, o_1) \in \mathit{closure}(O_1) \cap \mathit{closure}(O_2)$. $(p_1, \texttt{domain}, w) \in \mathit{closure}(O_1) \cap \mathit{closure}(O_2)$. Then, $(s_1, \texttt{type}, w) \in \mathit{closure}(O_1) \cap \mathit{closure}(O_2) = O_1 \sqcap O_2$. □

The following proposition follows now immediately from the above lemma.

Proposition 1: Let O_1, O_2, O_3 be RDF ontologies. It holds $(O_1 \sqcap O_2) \sqcap O_3 = O_1 \sqcap (O_2 \sqcap O_3)$.

The proof of the following proposition is trivial.

Proposition 2: Let O_1, O_2, O_3 be RDF ontologies. It holds $(O_1 \sqcup O_2) \sqcup O_3 = O_1 \sqcup (O_2 \sqcup O_3)$.

Unfortunately, it does dot hold the distribution property. Let O_1, O_2,and O_3 be RDF ontologies. Consider the property $(O_1 \sqcup O_2) \sqcap O_3 = (O_1 \sqcap O_3) \sqcup (O_2 \sqcap O_3)$. Additionally, consider $O_1 = \{(c_1, \texttt{subClassOf}, c_2)\}, O_2 = (c_2, \texttt{subClassOf}, c_3)$, and $O_3 = (c_1, \texttt{subClassOf}, c_3)$. It holds $(c_1, \texttt{subClassOf}, c_3) \in (O_1 \sqcup O_2) \sqcap O_3$, while $(c_1, \texttt{subClassOf}, c_3) \notin (O_1 \sqcap O_3) \sqcup (O_2 \sqcap O_3)$.

Consider now the property $(O_1 \sqcap O_2) \sqcup O_3 = (O_1 \sqcup O_3) \sqcap (O_2 \sqcup O_3)$. Additionally, consider $O_1 = \{(c_1, \texttt{subClassOf}, c_2)\}, O_2 = (c_1, \texttt{subClassOf}, c_2)$, and $O_3 = (c_2, \texttt{subClassOf}, c_3)$. It holds $(c_1, \texttt{subClassOf}, c_3) \in (O_1 \sqcup O_3) \sqcap (O_2 \sqcup O_3)$, while $(c_1, \texttt{subClassOf}, c_3) \notin (O_1 \sqcap O_2) \sqcup O_3$.

Below we define the evaluation of a primitive faceted formula which derives an RDF ontology.

Definition 13 (evaluation of a primitive faceted formula): Let *ff* be a primitive faceted formula.

1 If $\mathit{ff} = \{(f_1, v_1), ...(f_n, v_n), val\_int = TL\}$. We define
eval(*ff*) $= \bigcup \{O_{no} \mid no \in NO$ s.t. $\forall\, (f_i, v_i)\ \exists\, (f_i, t) \in noFA(no)$ and $v_i \leq t$ and $\forall ti \in TL,\ \exists ti' \in noTL(no)$ s.t. $ti \leq ti'\}$.

2 Let *ff* be name of a named RDF ontology no in NO. Then, *eval*(*ff*) $= O_{no}$.

Example 3: Let $NO = \{no_1, no_2, no_3\}$. Let $noFA(no_1) = \{(topic, hotels), (location, Cephalonia)\}$ and $noTL(no_1) = \langle[10-07-1016, 25-07-2016]\rangle$. Let $noFA(no_2) = \{(topic, business), (location, Corfu)\}$ and $noTL(no_2) = \langle[01-07-1016, 30-07-2016]\rangle$. Let $noFA(no_3) = \{(topic, arts)\}$.

Let $\mathit{ff} = \{(topic, business), (location, Ionian\ islands), val\_int = \langle[15-07-2016, 20-07-2016]\rangle$. Then, *eval*(*ff*) $= O_{no_1} \cup O_{no_2}$.

The evaluation of general faceted formulas, providing composed RDF ontologies, is provided in Algorithm 3.1.

Algorithm 3.1 $eval(ff)$

Input: a faceted formula *ff*
Output: the evaluation of the faceted formula *ff*

(1) if *ff* is a primitive faceted formula then return($eval(ff)$);
(2) if $ff = F_1 \sqcup ... \sqcup F_n$, where F_i is a faceted formula,
then return($eval(F_1) \sqcup ... \sqcup eval(F_n)$);
(3) if $ff = F_1 \sqcap ... \sqcap F_n$, where F_i is a faceted formula,
then return($eval(F_1) \sqcap ... \sqcap eval(F_n)$);
(4) return(FALSE);

Below we provide the time complexity of $eval(ff)$, for a faceted formula *ff*. Let O be an RDF ontology, we define by $CPr_O = max(|C_O|, |Pr_O|)$, where C_O are the classes of O and Pr_O are the properties of O.

Proposition 3: Let *ff* be a faceted formula. Let pr_{ff} be the number of the primitive faceted formulas in *ff*. Let ρ_{max} be the maximum number of pairs (f, u) appearing in a primitive faceted formula in *ff*. Let π_{max} be the maximum number of pairs (f, u) appearing $noFA(no)$, for $no \in NO$. Let T_{max} be the maximum number of terms of a facet in FT. Let $K = \cup\{O_{no} \mid no \in NO\}$. The time complexity of $eval(ff)$ is $\mathcal{O}(pr_{ff} * \rho_{max} * \pi_{max} * T^2_{max} + pr_{ff} * |K| + (|\sqcap| + 1) * |K|^4 * CPr^2_K + |\sqcup| * |K|^2)$.

Proof: Finding the matching named ontologies for each primitive faceted formulas in *ff*, takes $\mathcal{O}(\rho_{max} * \pi_{max} * T^2_{max})$ time. Thus, in total, $\mathcal{O}(pr_{ff} * \rho_{max} * \pi_{max} * T^2_{max})$ time. Taking the union of the matched named ontologies for each primitive faceted formula, is $\mathcal{O}(|K|)$ time. Thus, in total $O(pr_{ff} * |K|)$ time. In the proof of Proposition 15 of (Tzitzikas et al., 2014), we showed that the size of the closure of an RDF ontology O is in $\mathcal{O}(|O|^2)$, while the time for calculating the closure is in $\mathcal{O}(|O|^2 * CPr^2_O)$. In $eval(ff)$, the largest RDF ontology ontology, whose closure is to be computed is $closure(K)$. The size of $closure(K)$ is in $\mathcal{O}(|K|^2)$. Thus, the cost of all computed closures in $\mathcal{O}((|\sqcap| + 1) * |K|^4 * CPr^2_K)$. The cost of all computed intersections is in $\mathcal{O}(|\sqcap| * |K|^4)$. Additionally, the cost of all computed unions is in $\mathcal{O}(|\sqcup| * |K|^2)$. Thus, the total time complexity of $eval(ff)$ is $\mathcal{O}(pr_{ff} * \rho_{max} * \pi_{max} * T^2_{max} + pr_{ff} * |K| + (|\sqcap| + 1) * |K|^4 * CPr^2_K + |\sqcup| * |K|^2)$. □

4 Faceted indexing and list of validity intervals of evaluated faceted formulas

In this section, we formally define the faceted indexing of faceted formulas as well as their associated lists of validity time intervals. Formal definitions lead to algorithms, where detailed description, associated examples, and complexity are provided.

Let $ff = \{(f_1, v_1), ...(f_n, v_n), val\_int = TL\}$ be a primitive faceted formula. The named RDF ontologies in the set $\{no \in NO \mid \forall\ (f_i, v_i)\ \exists\ (f_i, t) \in noFA(no)$ and $v_i \leq t$ and $\forall ti \in TL,\ \exists ti' \in noTL(no)$ s.t. $ti \leq ti'\}$ are called *matching RDF ontologies* of *ff*.

Definition 14 (faceted indexing of primitive faceted formulas):

1 Let $no \in NO$. A faceted indexing term (f, t) indexes the primitive faceted formula nam_{no} iff there is a pair $(f, t') \in noFA(no)$ s.t. $t' \leq t$. The list of validity time intervals associated with nam_{no} is $noTL(no)$.

2 Let $ff = \{(f_1, v_1), ...(f_n, v_n), val\_int = TL\}$ be a primitive faceted formula. A faceted indexing term (f, t) indexes ff iff there is a pair $(f, t') \in ff$ s.t. $t' \leq t$. Let $no_1, ..., no_m$ be the matching RDF ontologies of ff. The list of validity time intervals associated with ff is formed by the intersection of $noTL(no_1), ..., noTL(no_m)$.

Definition 15 (faceted indexing of complex faceted formulas): Let ff_1 and ff_2 be faceted formulas.

1 A faceted indexing term (f, t) indexes $ff_1 \sqcap ff_2$ iff it indexes either ff_1 or ff_2. The list of validity intervals of $ff_1 \sqcap ff_2$, in the case that ff_1 and ff_2 have no exactly the same faceted indexing terms, corresponds to the intersection of the lists of validity time intervals associated with ff_1 and ff_2. On the other hand, the list of validity intervals of $ff_1 \sqcap ff_2$, in the case that ff_1 and ff_2 have exactly the same faceted indexing terms, corresponds to the merging of the lists of validity time intervals associated with ff_1 and ff_2.

2 A faceted indexing term (f, t) indexes $ff_1 \sqcup ff_2$ iff it indexes both ff_1 and ff_2. The list of validity intervals of $ff_1 \sqcup ff_2$ corresponds to the intersection of the lists validity time intervals associated with ff_1 and ff_2.

Algorithm 4.1 *indexing*(ff) takes as input a faceted formula ff and returns the faceted indexing and list of validity intervals of the evaluated faceted formula ff, according to Definitions 14 and 15. Detailed description of the algorithm, associated examples, and complexity are provided. Algorithm *indexing*(ff) calls the auxiliary Algorithms 4.2, 4.3, 4.4, and 4.5.

Algorithm 4.2 *TL_intersect*(TL, TL') takes as input two lists of validity intervals TL and TL' and returns a list of validity intervals that corresponds to the intersection of the time intervals in TL with that of TL' (intersection of TL with TL'). Algorithm 4.3 *TL_merge*(TL, TL') takes as input two lists of validity intervals TL and TL' and returns a list of validity intervals that corresponds to the merging of the time intervals in TL with that of TL' (merging of TL with TL'). Algorithm 4.4 *more_general_common_terms*(fn, t, t') takes as input the name fn of a facet $f \in FT$ and two terms t, t' of f and returns the more general common terms of t and t' in f. Algorithm 4.5 *most_specific_terms*(f, TS) takes as input the name fn of a facet $f \in FT$ and a set of terms TS of f and returns the most specific terms of TS in facet f.

Algorithm 4.1 *indexing*(ff) first checks if ff is the name of a named ontology $no \in NO$. In this case, ff is indexed by the faceted indexing terms in $noFA(no)$ and the list of validity time intervals is $noTL(no)$ (line 3). This is according to Definition 14.1.

If $ff = \{(f_1, v_1), ..., (f_n, v_n), val\_int = TL\}$ (see Definition 14.2) then ff is indexed by the faceted indexing terms in $\{(f_1, v_1), ..., (f_n, v_n)\}$ (line 11). The algorithm identifies in NO' the matching RDF ontologies of ff (line 5). Let $NO' = \{no_1, ..., no_m\}$. The intersection of $noTL(no_1), ..., noTL(no_m)$ is computed

and stored in TL' (lines 7–10). TL' is the list of validity time intervals corresponding to *ff* (line 11).

If $ff = ff_l \sqcap ff_2$ (see Definition 15.1) then *indexing*(ff_l) and *indexing*(ff_2) are recursively called and the results are the sets $fi_1 = \{(f_1^1, v_1^1), ..., (f_n^1, v_n^1), val\_int = TL_1\}$ and $fi_2 = \{(f_1^2, v_1^2), ..., (f_m^2, v_m^2), val\_int = TL_2\}$, respectively (lines 13-17). If the faceted indexing terms in fi_1 and fi_2 are the same (line 18) then *TL_merge*(TL_1, TL_2) is called and the result is considered as the list of validity time intervals associated with *ff* (line 20). *ff* is indexed by the faceted indexing terms in fi_1 (which are the same with the faceted indexing terms in fi_2) (line 20). If the faceted indexing terms in fi_1 and fi_2 are not the same then *TL_intersection*(TL_1, TL_2) (line 21) is called and the result is considered as the list of validity time intervals associated with *ff* (line 23). *ff* is indexed by the faceted indexing terms in fi_1 unioned the faceted indexing terms in fi_2 (line 23).

If $ff = ff_l \sqcup ff_2$ (see Definition 15.2) then *indexing*(ff_l) and *indexing*(ff_2) are recursively called and the results are the sets $fi_1 = \{(f_1^1, v_1^1), ..., (f_n^1, v_n^1), val\_int = TL_1\}$ and $fi_2 = \{(f_1^2, v_1^2), ..., (f_m^2, v_m^2), val\_int = TL_2\}$, respectively (lines 25-29). *TL_intersection*(TL_1, TL_2) (line 30) is called and the result is considered as the list of validity time intervals associated with *ff* (line 46). For each $f \in names(FT)$ s.t. exists $(f, v) \in fi_1$ and exists $(f, u) \in fi_2$, for each $(f, t) \in fi_1$ and $(f, t') \in fi_2$, the more general common terms of t and t' in the facet with name f are computed and all computed sets are unioned in the set TS (line 39). Then, the most specific terms of TS in the facet with name f are computed (line 42). If these are $u_1, ..., u_n$ (line 43) then the pairs (f, u_i) belong to the faceted indexing terms of *ff* (lines 44–46).

Example 4: Consider the primitive faceted formulas:

$pr_1 = \{($*topic*, *hotels*$)$, $($*location*, *Cephalonia*$)$,
($*facilities*, *spa*$)$, *val_int* $= \langle[13-07-2016, 19-07-2016]\rangle\}$

$pr_2 = \{($*topic*, *hotels*$)$, $($*location*, *Cephalonia*$)$,
($*facilities*, *dancing*$)$, *val_int* $= \langle[13-07-2016, 19-07-2016]\rangle\}$

$pr_3 = \{($*topic*, *hotels*$)$, $($*location*, *Corfu*$)$,
($*facilities*, *swimming pool*$)$, *val_int* $= \langle[13-07-2016, 19-07-2016]\rangle\}$

$pr_4 = \{($*topic*, *hotels*$)$, $($*location*, *Corfu*$)$,
($*facilities*, *football*$)$, *val_int* $= \langle[13-07-2016, 19-07-2016]\rangle\}$

Consider the faceted formula $ff = (pr_1 \sqcap pr_2) \sqcup (pr_3 \sqcap pr_4)$. Assume that *indexing*$(pr_1)$ match such named ontologies in NO that the resulting list of validity time intervals is $\langle[30-05-2016, 25-10-2016]\rangle$. Thus, *indexing*$(pr_1)$ returns [Algorithm 4.1, lines (4–11)]:

$pr_1 = \{($*topic*, *hotels*$)$, $($*location*, *Cephalonia*$)$,
($*facilities*, *spa*$)$, *val_int* $= \langle[30-05-2016, 25-10-2016]\rangle\}$

Assume that *indexing*(pr_2) match such named ontologies in NO that the resulting list of validity time intervals is $\langle[25-05-2015, 30-10-2016]\rangle$. Thus, *indexing*$(pr_2)$ returns [Algorithm 4.1, lines (4–11)]:

$\{($*topic*, *hotels*$)$, $($*location*, *Cephalonia*$)$,
($*facilities*, *dancing*$)$, *val_int* $= \langle[25-05-2016, 30-10-2016]\rangle\}$

Algorithm 4.1 *indexing(ff)*

Input: a faceted formula ff
Output: faceted indexing and list of validity time intervals of evaluated faceted formula ff

(1) Case(ff)
(2) $ff \in names(NO)$:
(3) if ff is the name of a named ontology $no \in NO$ then
return($noFA(no) \cup \{val\_int = noTL(no)\}$);
(4) $ff = \{(f_1, v_1), ..., (f_n, v_n), val\_int = TL\}$:
(5) $NO' = \{no \in NO \mid \forall\ (f_i, v_i)\ \exists\ (f_i, t) \in noFA(no)$ and $v_i \leq t$ and
$\forall ti \in TL,\ \exists ti' \in noTL(no)$ s.t. $ti \leq ti'\}$;
(6) if $NO' = \emptyset$ then return(FAILURE);
(7) Let $TL' = \{[\infty, +\infty]\}$;
(8) For $no \in NO'$ do
(9) $TL' = TL\_intersection(TL', noTL(no))$;
(10) EndFor
(11) return($\{(f_1, v_1), ..., (f_n, v_n), val\_int = TL'\}$);
(12) $ff_1 \sqcap ff_2$:
(13) $fi_1 = indexing(ff_1)$;
(14) $fi_2 = indexing(ff_2)$;
(15) if fi_1=FAILURE or fi_2=FAILURE then return(FAILURE);
(16) Let $fi_1 = \{(f_1^1, v_1^1), ..., (f_n^1, v_n^1), val\_int = TL_1\}$;
(17) Let $fi_2 = \{(f_1^2, v_1^2), ..., (f_m^2, v_m^2), val\_int = TL_2\}$;
(18) If $\{(f_1^1, v_1^1), ..., (f_n^1, v_n^1)\} = \{(f_1^2, v_1^2), ..., (f_m^2, v_m^2)\}$ then
(19) $TL = TL\_merge(TL_1, TL_2)$;
(20) return($\{(f_1^1, v_1^1), ..., (f_n^1, v_n^1), val\_int = TL\}$);
(21) $TL = TL\_intersection(TL_1, TL_2)$;
(22) if TL is the empty list then return(FAILURE);
(23) return $(\{(f_1^1, v_1^1), ..., (f_n^1, v_n^1), (f_1^2, v_1^2), ..., (f_m^2, v_m^2), val\_int = TL\}$;
(24) $ff_1 \sqcup ff_2$:
(25) $fi_1 = indexing(ff_1)$;
(26) $fi_2 = indexing(ff_2)$;
(27) if fi_1=FAILURE or fi_2=FAILURE then return(FAILURE);
(28) Let $fi_1 = \{(f_1^1, v_1^1), ..., (f_n^1, v_n^1), val\_int = TL_1\}$:
(29) Let $fi_2 = \{(f_1^2, v_1^2), ..., (f_m^2, v_m^2), val\_int = TL_2\}$:
(30) $TL = TL\_intersection(TL_1, TL_2)$;
(31) if TL is the empty list then return(FAILURE);
(32) $FI = \emptyset$;
(33) For each $f \in names(FT)$ s.t. exists $(f, v) \in fi_1$ and exists $(f, u) \in fi_2$ do
(34) $TS_1 = \{t \mid (f, t) \in fi_1\}$;
(35) $TS_2 = \{t \mid (f, t) \in fi_2\}$;
(36) $TS = \emptyset$;
(37) For $t \in TS_1$ do
(38) For $t' \in TS_2$ do
(39) $TS = TS \cup more\_general\_common\_terms(f, t, t')$;
(40) EndFor
(41) EndFor
(42) $TS = most\_specific\_terms(f, TS)$;
(43) Let $TS = \{v_1, ..., v_n\}$;
(44) $FI = FI \cup \{(f, v_1), ..., (f, v_n)\}$;
(45) EndFor
(46) return($FI \cup \{val\_int = TL\}$);
(47) return(FAILURE);

Algorithm 4.2 *TL_intersect*(TL, TL')

Input: two lists of validity time intervals TL and TL'
Output: a list of validity time intervals that corresponds to the intersection of TL with TL'

(1) if TL is the empty list then return(TL');
(2) if TL' is the empty list then return(TL);
(3) Get fist time interval ti from TL;
(4) Get fist time interval ti' from TL';
(5) if $end(ti) < start(ti')$ then
(6) Remove ti from TL;
(7) return($TL\_intersect(TL, TL')$);
(8) if $end(ti') < start(ti)$ then
(9) Remove ti' from TL';
(10) return($TL\_intersect(TL, TL')$);
(11) if $start(ti) \geq start(ti')$ then
(12) if $end(ti) = end(ti')$ then
(13) Remove ti from TL;
(14) Remove ti' from TL';
(15) return($\langle [start(ti), end(ti)] \rangle + TL\_intersect(TL, TL')$);
(16) if $end(ti) < end(ti')$ then
(17) Remove ti from TL;
(18) Remove ti' from TL';;
(19) Add at the beginning $[end(ti+1), end(ti')]$ to TL';
(20) return($\langle [start(ti), end(ti)] \rangle + TL\_intersect(TL, TL')$);
(21) if $end(ti) > end(ti')$ then
(22) Remove ti from TL;
(23) Remove ti' from TL';;
(24) Add at the beginning $[end(ti'+1), end(ti)]$ to TL;
(25) return($\langle [start(ti), end(ti')] \rangle + TL\_intersect(TL, TL')$);
(26) if $start(ti) < start(ti')$ then
(27) if $end(ti) = end(ti')$ then
(28) Remove ti from TL;
(29) Remove ti' from TL';
(30) return($\langle [start(ti'), end(ti')] \rangle + TL\_intersect(TL, TL')$);
(31) if $end(ti') < end(ti)$ then
(32) Remove ti from TL;
(33) Remove ti' from TL';;
(34) Add at the beginning $[end(ti'+1), end(ti)]$ to TL;
(35) return($\langle [start(ti'), end(ti')] \rangle + TL\_intersect(TL, TL')$);
(36) if $end(ti') > end(ti)$ then
(37) Remove ti from TL;
(38) Remove ti' from TL';
(39) Add at the beginning $[end(ti+1), end(ti')]$ to TL';
(40) return($\langle [start(ti'), end(ti)] \rangle + TL\_intersect(TL, TL')$);

Algorithm 4.3 $TL\_merge(TL, TL')$

Input: two lists of validity time intervals TL and TL'
Output: a list of validity time intervals that corresponds to the merging of TL with TL'

(1) Let $TL'' = TL + TL'$;
(2) Order interval ti in TL'' by $start(ti)$;
(3) Let L be the empty list;
(4) Get first interval ti from TL'' and remove it from TL'';
(5) $current\_start = start(ti)$;
(6) $current\_end = end(ti)$;
(7) For each time interval ti taken in sequence from list TL'' do
(8) If $start(ti) \leq current\_end + 1$ and $end(ti) > current\_end$ then
(9) $current\_end = end(ti)$;
(10) If $start(ti) > current\_end + 1$ or ti is the last interval in TL'' then
(11) Add at the end of L the time interval $[current\_start, current\_end]$;
(13) $current\_start = start(ti)$;
(14) $current\_end = end(ti)$;
(15) If $start(ti) > current\_end + 1$ and ti is the last interval in TL'' then
(16) Add at the end of L the time interval $[start(ti), end(ti)]$;
(17) EndFor
(18) return(L);

Algorithm 4.4 *more_general_common_terms*(fn, t, t')

Input: the name *fn* of a facet $f \in FT$ and two terms t, t' of f
Output: the more general common terms of t and t' in f

(1) Let $f \in FT$ be the facet with name *fn*;
(2) Let $T = \{t'' \in Terms_f \mid t'' \geq t\}$;
(3) Let $T' = \{t'' \in Terms_f \mid t'' \geq t'\}$;
(4) return($T \cap T'$);

Algorithm 4.5 *most_specific_terms*(fn, TS)

Input: the name *fn* of a facet $f \in FT$ and a set of terms TS of f
Output: the most specific terms of TS in facet f

(1) Let $f \in FT$ be the facet with name *fn*;
(2) Let $T = \emptyset$;
(3) For all $t \in TS$ do
(4) If there is no $t' \in TS - \{t\}$ s.t. $t' \leq t$ then
(5) $T = T \cup \{t\}$;
(6) EndFor
(7) return(T);

Now *indexing*($pr_1 \sqcap pr_2$) returns

$\{(topic, hotels), (location, Cephalonia), (facilities, spa)$
$(facilities, dancing), val\_int = \langle[25-05-2016, 25-10-2016]\rangle\}$

Similarly, assume that *indexing*($pr_3 \sqcap pr_4$) returns

$\{(topic, hotels), (location, Corfu), (facilities, football)$
$(facilities, swimming\ pool), val\_int = \langle[23-05-2016, 28-10-2016]\rangle\}$

Now according to the faceted taxonomy of Figure 1, *indexing*$(pr_1 \sqcap pr_2) \sqcup (pr_3 \sqcap pr_4)$, returns [Algorithm 4.1, lines (24–46)]:

$\{(topic, hotels), (location, Ionian\ islands), (facilities, recreation\ facilities)$
$(facilities, athletic\ facilities), val\_int = \langle[23-05-2016, 25-10-2016]\rangle\}$

Example 5: Consider the primitive faceted formulas:

$pr_1 = \{(topic, hotels), (location, Cephalonia),$
$(facilities, spa), val\_int = \langle[13-07-2016, 19-07-2016]\rangle\}$

$pr_2 = \{(topic, hotels), (location, Cephalonia),$
$(facilities, spa), val\_int = \langle[10-12-2016, 19-12-2016]\rangle\}$

Consider the faceted formula $ff = pr_1 \sqcap pr_2$. Assume that *indexing*(pr_1) match such named ontologies in NO that the resulting list of validity time intervals is $\langle[01-06-2016, 19-09-2016]\rangle$. Thus, *indexing*$(pr_1)$ returns [Algorithm 4.1, lines (4–11)]:

$pr_1 = \{(topic, hotels), (location, Cephalonia),$
$(facilities, spa), val\_int = \langle[01-06-2016, 19-09-2016]\rangle\}$

Assume that *indexing*(pr_2) match such named ontologies in NO that the resulting list validity time intervals is $[25-11-2016, 30-01-2017]$. Thus, *indexing*(pr_2) returns [Algorithm 4.1, lines (4–11)]:

$\{(topic, hotels), (location, Cephalonia),$
$(facilities, spa), val\_int = \langle[25-11-2016, 30-01-2017]\rangle\}$

Now *indexing*($pr_1 \sqcap pr_2$) returns [Algorithm 4.1, lines (16–20)]:

$\{(topic, hotels), (location, Cephalonia), (facilities, spa)$
$val\_int = \langle[01-06-2016, 19-09-2016], [25-11-2016, 30-01-2017]\rangle\}$

Proposition 4: Let *ff* be a faceted formula. Let pr_{ff} be the number of the primitive faceted formulas in *ff*. Let ρ_{max} be the maximum mumber of pairs (f, u) appearing in a primitive faceted formula in *ff*. Let μ_{max} be the maximum mumber of facet names appearing in faceted annotations of all primitive faceted formulas in *ff*. Let π_{max} be the maximum number of pairs (f, u) appearing $noFA(no)$, for $no \in NO$. Let T_{max} be the maximum number of terms of a facet in FT. The time complexity of $indexing(ff)$ is $\mathcal{O}(pr_{ff} * \rho_{max} * \pi_{max} * T_{max}^2 + |\sqcap| * \mu_{max} * T_{max} + |\sqcap| * \mu_{max} * T_{max}^4)$.

Proof: Finding the matching named ontologies for each primitive faceted formulas in *ff*, takes $\mathcal{O}(\rho_{max} * \pi_{max} * T^2_{max})$ time. Thus in total, $\mathcal{O}(pr_{ff} * \rho_{max} * \pi_{max} * T^2_{max})$ time. For each $\sqcap$ in *ff*, the cost is $\mathcal{O}(\mu_{max} * T_{max})$. Thus, in total, for all $\sqcap$ in *ff*, the cost is $\mathcal{O}(|\sqcap| * \mu_{max} * T_{max})$. For each $\sqcup$ in *ff*, the cost is $\mathcal{O}(\mu_{max} * T^4_{max})$.[4] Thus, in total, for all $\sqcup$ in *ff*, the cost is $\mathcal{O}(|\sqcup| * \mu_{max} * T^4_{max})$. Thus, the time complexity of $indexing(ff)$ is $\mathcal{O}(pr_{ff} * \rho_{max} * \pi_{max} * T^2_{max} + |\sqcap| * \mu_{max} * T_{max} + |\sqcap| * \mu_{max} * T^4_{max})$. □

5 Related work

In this section, we review related works.

The Dublin core metadata element set (DCMI, 2012) is a vocabulary of fifteen properties for use in the description of resources. Among them the property *dc:coverage* provides the spatial or temporal topic of the resource, the spatial applicability of the resource, or the jurisdiction under which the resource is relevant. Spatial topic and spatial applicability may be a named place or a location specified by its geographic coordinates. Temporal topic may be a named period, date, or date range. The property *dc:description* provides an account of the resource. Description may include but is not limited to: an abstract, a table of contents, a graphical representation, or a free-text account of the resource. The property *dc:subject* provides the topic of the resource. Typically, the subject will be represented using keywords, key phrases, or classification codes. Recommended best practice is to use a controlled vocabulary. VoID (Alexander et al., 2009, 2011) is an RDF schema vocabulary for expressing metadata about RDF datasets. Void uses the Dublin Core metadata elements *dc:subject* and *dc:description*. For the general case, it recommends the use of a DBpedia resource URI to categorise a dataset. The OMV – ontology metadata vocabulary (Hartmann et al., 2005) is a proposal for a metadata standard to improve accessibility and reuse of ontologies. Among their properties they include *description*, *documentation, subject* and *keywords*. MOD (Dutta et al., 2015) is also a metadata vocabulary for ontology description and publication. Among their properties they include *dc:description* and *keywords*. In our case, we use faceted terms for categorising an RDF ontology and a special characteristic of our approach is that composing RDF ontologies, new faceted indexing terms are derived for the resulting composed ontology.

In Mitra and Wiederhold (2012), an ontology O is defined as a pair (G, R), where $G = (N, E)$ is a directed labelled graph and R is a set of rules. They also consider a set of *articulations rules* between two ontologies connecting concepts of the two ontologies. The intersection between two ontologies is defined, where the nodes in the intersection ontology are those nodes that appear in the articulation rules. The edges of the intersection ontology are the edges among the nodes in the intersection ontology that were present in the source ontologies plus the edges of the articulation rules. The rules in the intersection ontology are the rules in the source ontologies that use only concepts that appear in the intersection ontology plus the articulation rules. The union of two ontologies is also defined, as the union of the elements of the source ontologies union the elements of the intersection ontology. As it is evident, our definitions are completely different, basically because we consider RDF ontologies and we do not consider articulation rules.

In Kaushik et al. (2006), an ontology O is defined based on

1 a set of primitive classes and properties
2 a set of classes and properties of finite type
3 a schema naming
4 a set of instances
5 instance naming.

The intersection of two ontologies is defined based on the commonalities of the characteristics of the two ontologies. Additionally, the union of two ontologies is defined based on the union of the characteristics of the two ontologies. Though conceptually our definitions are the same, we consider RDF ontologies and in particular in the intersection operation we take the intersection of the closures of the source RDF ontologies.

In Theodorakis et al. (2002), the authors define a context as an object associated with a set of named objects. The intersection of contexts is defined by taking the intersection of the sets of objects associated with the original contexts. An object in the intersection context has all the names it had in the original contexts. The union of contexts is defined by taking the union of the sets of objects associated with the original contexts. An object in the union context that existed in both original contexts has all the names it had in the original contexts. In Mylopoulos and Pitrik (1995), contexts, their union and intersection, are defined in a similar way. However, the authors impose a strict constraint on the naming of objects which are assigned unique names within a context. Thus, it is possible for an object in the intersection and union context to have no name. In our case, in the intersection and union, we do not consider sets of named objects but sets of RDF triples.

In Delbru et al. (2008), an ontology context is described as a named graph composed of a set of RDF triples. An ontology context can have import relationships with other ontology contexts making possible the derivation of new RDF triples through the RDFS inference triples. The *import closure* of an ontology context c consists of the union of the contexts that directly of indirectly imported in c. In the case that a context has a single import relationship with another context, the result of our union will be the same. In Analyti et al. (2015) a context c is defined a structure that

1 is associated with a name
2 contains a contextual RDF graph G_c
3 contains a set of mappings (called *references*) from IRIs to other contexts.

The authors define merging of contexts which in the case that the their references are empty, the result of our union will be the same. Yet, both in Delbru et al. (2008) and Analyti et al. (2015) intersection of contexts is not defined.

In Tzitzikas et al. (2003), the authors propose an algebra whose operators allow to specify the valid compound terms of a faceted taxonomy, in a flexible manner (by combining positive and negative statements). In Analyti et al. (2009), they treat the same problem but in a more general setting, where the facets of the faceted taxonomy are not independent but are (possibly) interrelated through narrower/broader relationships between their terms. The works (Tzitzikas et al., 2003; Analyti et al., 2009) are useful

in faceted browsing of a collection of resources. In our case, the set of faceted indexing terms in a primitive faceted formula forms always a valid combination, as it is provided by the users.

In Arenas et al. (2014), the authors provide rigorous theoretical underpinnings for faceted search in the context of RDF-based knowledge graphs enhanced with OWL 2 ontologies. They identify well-defined fragments of SPARQL that can be naturally captured using faceted search as a query paradigm, and establish the computational complexity of answering such queries. They also study the problem of updating faceted interfaces, which is critical for guiding users in the formulation of meaningful queries during exploratory search. Note that this work is useful for browsing, through faceted search, RDF-based knowledge graphs enhanced with OWL 2 ontologies. In our case, we use faceted terms for identifying the faceted indexed RDF ontologies that are going to be composed.

Note that none of the above works considers the derivation of new faceted terms for composed, faceted indexed RDF ontologies.

6 Conclusions

In this paper, we consider RDF ontologies whose content is indexed by faceted terms. The idea of faceted indexing is not new as it also appears in Joseph and Serafini (2011), where contextual RDF ontologies are organised in a partial order, based on faceted indexing terms, in order to allow transfer of knowledge among these contexts.

In this paper, we consider the composition of faceted indexed RDF ontologies. Through primitive faceted formulas using faceted terms and a list of validity intervals, we are able to take to union of all RDF ontologies indexed by these terms and are valid in all provided validity intervals. By combining both union and intersection operators on primitive faceted formulas, complex expressions can be formed that return a computed RDF ontology, its faceted indexing and validity intervals. All algorithms are provided along with their complexity. Complex expressions formed using faceted terms and a lists of validity time intervals allow the user to express his/her wishes for the characteristics of the resulting ontology. To the best of our knowledge our approach is completely novel.

A disadvantage of our approach is that we consider RDF ontologies without blank nodes. As future work we plan to alleviate this problem by considering blank node matching between two RDF ontologies, for example, using the algorithms proposed in Tzitzikas et al. (2012). In blank node matching, pairs of blank nodes of two different RDF ontologies are matched and they are replaced by a new blank node in the composed ontology. Further, in the future we plan to conduct experiments to validate our approach.

References

Alexander, K., Cyganiak, R., Hausenblas, M. and Zhao, J. (2009) 'Describing linked datasets', in *WWW, Workshop on Linked Data on the Web (LDOW)*.

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

Analyti, A., Tzitzikas, Y. and Spyratos, N. (2009) 'Specifying valid compound terms in interrelated faceted taxonomies', in *28th International Conference on Conceptual Modeling (ER)*, pp.360–373.

Analyti, A., Damásio, C.V. and Pachoulakis, I. (2015) 'Nested contextualised views in the web of data', *International Journal Web Engineering Technology*, Vol. 10, No. 1, pp.31–64.

Arenas, M., Grau, B.C., Kharlamov, E., Marciuska, S. and Zheleznyakov, D. (2014) 'Faceted search over ontology-enhanced RDF data', in *23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*, pp.939–948.

Carroll, J.J., Bizer, C., Hayes, P.J. and Stickler, P. (2005a) 'Named graphs', *Journal of Web Semantics*, Vol. 3, No. 4, pp.247–267.

Carroll, J.J., Bizer, C., Hayes, P.J. and Stickler, P. (2005b) 'Named graphs, provenance and trust', in *14th International Conference on World Wide Web (WWW)*, pp.613–622.

Cyganiak, R., Wood, D. and Lanthaler, M. (2014) *RDF 1.1 Concepts and Abstract Syntax*, W3C Recommendation, 25 February [online] https://www.w3.org/TR/rdf11-concepts/.

DCMI (2012) *Dublin Core Metadata Element Set, Version 1.1*, DCMI Recommendation, 14 June [online] http://dublincore.org/documents/dces/.

de Bruijn, J., Franconi, E. and Tessaris, S. (2005) 'Logical reconstruction of normative RDF', in *OWL: Experiences and Directions Workshop (OWLED)*, Galway, Ireland, November.

Delbru, R., Polleres, A., Tummarello, G. and Decker, S. (2008) 'Context dependent reasoning for semantic documents in sindice', in *4th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS)*.

Dutta, B., Nandini, D. and Shahi, G. (2015) 'MOD: metadata for ontology description and publication', in *DCMI International Conference on Dublin Core and Metadata Applications (DC)*, pp.1–9, 2015.

Harris, S. and Seaborne, A. (2013) *SPARQL 1.1 Query Language*, W3C Recommendation, 21 March [online] http://www.w3.org/TR/sparql11-query/.

Hartmann, J., Palma, R. and Sure, Y. (2005) 'OMV – ontology metadata vocabulary', in *ISWC Workshop on Ontology Patterns for the Semantic Web*.

Hayes, P. and Patel-Schneider, P.F. (2014) *RDF 1.1 Semantics*, W3C Recommendation, 25 February [online] http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/.

Heath, T. and Bizer, C. (2011) *Linked Data: Evolving the Web into a Global Data Space*, 1st ed., Morgan & Claypool.

Joseph, M. and Serafini, L. (2011) 'Simple reasoning for contextualized RDF knowledge', in *Fifth International Workshop on Modular Ontologies (WoMO)*, pp.79–93.

Kaushik, S., Farkas, C., Wijesekera, D. and Ammann, P. (2006) 'An algebra for composing ontologies', in *4th International Conference on Formal Ontology in Information Systems (FOIS)*, pp.265–276.

Lebo, T., Sahoo, S. and McGuinness, D. (2013) *PROV-O: The PROV Ontology*, W3C Recommendation, Consulted [online] http://www.w3.org/TR/2013/REC-prov-o-20130430/.

Lloyd, J.W. (1987) *Foundations of Logic Programming*, 2nd ed., Springer-Verlag, New York, NY, USA.

Mitra, P. and Wiederhold, G. (2012) 'An algebra for the composition of ontologies', in *ECAI, Workshop on Knowledge Transformation for the Semantic Web*.

Mylopoulos, J. and Pitrik, R.M. (1995) 'Partitioning Information Bases with Contexts', in *3rd International Conference on Cooperative Information Systems (CoopIS)*, pp.44–54.

Theodorakis, M., Analyti, A., Constantopoulos, P. and Spyratos, N. (2002) 'A theory of contexts in information bases', *Information Systems*, Vol. 27, No. 3, pp.151–191.

Tzitzikas, Y., Lantzaki, C. and Zeginis, D. (2012) 'Blank node matching and RDF/S comparison functions', in *11th International Semantic Web Conference (ISWC)*, pp.591–607.

Tzitzikas, Y., Analyti, A., Spyratos, N. and Constantopoulos, P. (2003) 'An algebraic approach for specifying compound terms in faceted taxonomies', in *13th European-Japanese Conference on Information Modelling and Knowledge Bases (EJC)*, pp.67–87.

Tzitzikas, Y., Kampouraki, M. and Analyti, A. (2014) 'Curating the specificity of ontological descriptions under ontology evolution', *Journal on Data Semantics (JODS)*, Vol. 3, No. 2, pp.75–106.

Wei, B., Liu, J., Zheng, Q., Zhang, W., Fu, X. and Feng, B. (2013) 'A survey of faceted search', *Journal of Web Engineering*, Vol. 12, Nos. 1&2, pp.41–64.

Zuiderwijk, A., Jeffery, K. and Janssen, M. (2012) 'The potential of metadata for linked open data and its value for users and publishers', *The eJournal of eDemocracy and Open Government*, Vol. 4, No. 2, pp.222–244.

Notes

1 For dates we follow the format *day-month-year*.

2 For simplification, in the rest of the paper we ignore namespaces from IRIs.

3 $T_P(.)$ is the *immediate consequence operator* on a definite logic program P and is defined in Lloyd (1987). $T_P^{\uparrow\omega}(\emptyset)$ denotes that the operator T_P is applied from the empty set until closure.

4 Note that the time complexity of *more_general_common_terms*(f, t, t') is $\mathcal{O}(T_{max}^2)$. Additionally, the time complexity of *most_specific_terms*(TS) is $\mathcal{O}(T_{max}^2)$.